



School of Information Technology and Communications  
Department of Digital Systems

---

**Master of Science**  
**Information Systems & Services / Big Data and Analytics**

---

# **Machine learning pipelines in Serverless environments**

**Master Thesis in Computer Science**

**Author:**

Paraskevoulakou Efterpi  
Registration Number: ME1819

**Supervisor:**

Associate Professor Dimosthenis Kyriazis

February 2020



---

© University of Piraeus 2019 – 2020. All rights reserved



---

# Abstract

---

The 21st Century is the reflection of the Big Data Era since 90% of the available data have been created over the recent years<sup>1</sup>. Several frameworks and approaches are made available towards facilitating analysis of the aforementioned datasets in order to extract useful information, decode hidden patterns, discover trends and insights in various domains such as medical science, economics, biology, physics, and social sciences. To this end, data scientists initially aim at problem comprehension / formulation, which is one of the most important steps because being able to understand the business logic is part of the key to ensure the success. Following the problem formulation step, data preprocessing is performed, referring to data collection and organization, as well as the selection of the most suitable strategy to achieve the optimal transformation, and adopt useful knowledge from the statistics field to extract insights as a first overall view. In the real world, data are generally inadequate, noisy, and inconsistent, so for reasons mentioned above, analysts spend a lot of time using various techniques to manipulate, reformatting, and finally merging them in order to be consumed for analysis. The data preparation includes methods such as cleaning, integration, transformation, and reduction. The preprocessing part is vital as it has an effective role in the production of accurate and not misleading results. The main target of preprocessing is to define a well-modified dataset that is ready to be processed to enhance its reliability. More concretely, methods such as elimination of noisy data, handling missing values, label encoding in categorical data, normalization, standardization, dimensionality reduction are widely used from data scientists to get a step closer to the problem solution. As the preprocessing procedure concludes, the selection of the most suitable machine learning algorithm is performed. Machine learning algorithms enable the actual transformation of the datasets into valuable information. Subsequently to the problem (regression, classification, etc.), the available computing resources and the nature of the data, the analysts choose typically from a range of different machine learning algorithms to address the given problem, apply each one of them in their data and ultimately select the one that will bring the best and most reliable result. Traditional machine learning algorithms such as logistic regression, decision trees, support vector machines, and neural networks, are common choices for classification problems in view of the fact that the community of data analysts make wide usage of them and perform evaluation metrics (e.g., accuracy, loss, precision, recall, AUC-ROC, etc.) for the final selection.

The above steps are the baseline in order to gain the first level of experience in data science, but how this valuable knowledge can be consolidated with the giant state of the art technologies? Cloud computing offers core services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), and extended ones based on this SPI model such as Storage as a service (STaaS), Security as a service (SECaaS), Data as a service (DaaS). As a core offering and with the goal to further abstract the offerings and serve the users needs (including data scientists), serverless computing has emerged. Serverless computing is a cloud computing model that aims to abstract server management and low-level infrastructure decisions. In this model, the allocation of resources is managed by the cloud provider instead of the application architect.

---

<sup>1</sup> B. Marr, "How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read," Forbes Business Magazine, 21 May 2018 [Online]. Available: <https://www.forbes.com/sites/bernardmarr/2018/05/21>.

Serverless allows applications to be developed without anxieties for implementing, tweaking, or scaling a server from the user's perspective. This innovation brings to light the concept of Function as a Service (FaaS). FaaS is a novel paradigm that is fully adopted by innovative companies like Amazon, Google, IBM, providing a means to realize serverless offerings. The developers are able to develop their applications as microservices without dealing with the administration of the resources since the latter is performed by the cloud providers. FaaS can also be offered over edge computing resources, addressing cases such as low-latency applications or data exhaust challenges. Initial FaaS implementations have been made available, including AWS Lambda<sup>2</sup>, Google Functions<sup>3</sup>, IBM Cloud Functions<sup>4</sup>, and Microsoft Azure Functions<sup>5</sup>. Based on the above, the goal of this thesis is to deliver an innovative, beyond the state of the art solution that combines data analytics, machine learning, and serverless computing in order to build a set of microservices (as a pipeline of Functions as Services) that turns a baseline data analysis logic to a fully innovative application. The solution has been realized through the utilization of the Apache OpenWhisk serverless platform<sup>6</sup> to build the pipeline of Functions as Services, while it has been evaluated through a specific analytics use case to predict fraudulent e-commerce transactions. Notwithstanding the idea mentioned above, the proposed logic can be applied more generally to different kinds of business tasks depending on the target to be implemented. Regarding the approach mentioned above, we split the business goal into two main phases: (i) the offline phase (source code creation and implementation in local machine) that includes the data collection, preprocessing, transformation, training, and final machine learning model selection, and (ii) the online phase that addresses the implementation of the pipeline of functions as services through Apache OpenWhisk. The concluding result returns a pipeline of microservices in which new data concerning an e-commerce transaction, are obtained as input, utilize the appropriate transformation and supply a pre-trained custom machine learning model, and finally deliver the outcome if the prediction constitutes a fraudulent or a legitimate transaction. In the scope of the offline phase, three sub-phases have been envisioned. The first one concerns the collection of a real-world dataset – for the specific use case, it refers to e-commerce transactions provided by the world's leading payment service company Vesta Corporation<sup>7</sup>. The second sub-phase of the offline phase, address the data preprocessing and the preparation aspects, and consequently, the third sub-phase includes the selection of the most suitable machine learning classifier for the dataset. Regarding the online phase, it consists of two sub-phases. In the first one, deployment of Apache OpenWhisk serverless platform under the control of the Docker Compose Orchestrator has been performed, while in the second sub-phase the offline phase outcomes have been ported in the serverless platform. Moreover, in the second sub-phase of the online phase, we created three functions that are interconnected. Each one of these is dedicated to a specific task and follows a predetermined workflow. Each function receives an input, executes a procedure and produces an output, which is the input to the next function until the last one produces the final result. The first function acquires new data (e.g. from the consumer), runs specific processes in order to transform the data into a suitable shape and attaches new features according to the implemented feature engineering technique in the Offline phase. The second function applies -on testing data- the label encoding and the normalization procedure.

---

<sup>2</sup> "AWS Lambda Run code without thinking about servers. Pay only for the compute time you consume.," Amazon.com, Inc., [Online]. Available: <https://aws.amazon.com/lambda/>.

<sup>3</sup> "Cloud Functions," Google Inc., [Online]. Available: <https://cloud.google.com/functions>.

<sup>4</sup> "IBM Cloud Functions," IBM Computer hardware company, [Online]. Available: <https://www.ibm.com/cloud/functions>.

<sup>5</sup> "Azure Functions," Microsoft Corporation, [Online]. Available: <https://azure.microsoft.com/en-us/services/functions/>.

<sup>6</sup> "Apache Openwhisk: An Open Source Serverless Cloud Platform," Apache Foundation, [Online]. Available: <https://openwhisk.apache.org/>.

<sup>7</sup> <https://trustvesta.com/>

The last one performs the invocation of the pre-trained model - built during the offline phase - to make the final prediction. As selected model has emerged an artificial neural network that was implemented with the open-source high-level API Keras<sup>8</sup> and achieved 84% sensitivity in fraudulent transaction detection. Our proposed approach is strong evidence that the serverless computing model provides a lightweight solution to build innovative artificial intelligent applications executed only in a few milliseconds; thereby, we can focus on our business goal without managing the allocation resources.

### **Keywords**

Serverless Computing, Function as a Service -FaaS-, provisioning, framework, Apache OpenWhisk, structured data analysis, Machine Learning, imbalanced dataset classification, e-commerce transactional data

---

<sup>8</sup> "Keras: The Python Deep Learning library," [Online]. Available: <https://keras.io/>.





# Acknowledgements

At the end of this extraordinary experience, I would first like to thank my thesis advisor, Associate Professor Mr. Dimosthenis Kyriazis who assigned me with this remarkably interesting research subject to accomplish; furthermore he was the one who believed in my abilities and my skills and appreciated my courage, patience, and my desire to dive deep in the knowledge around this Master Graduate Programme.

I am truly grateful to Mr. Georgios Kousiouris, who was always available whenever I ran into a trouble spot or had a question about my research, giving me his point of view and his advice to approach potential problems that I found during the implementation of this thesis.

I would also like to thank all my master's Professors who introduced me to a completely different environment with their lectures and made me appreciate the valuable contribution of Computer Science in every scientific field with a special mention to Associate Professor, Mr. Michael Filippakis, whose office was always open whenever I confronted problems concerning master classes, but the most important, he reinforced me to maintain my strength and tenacity in all this knowledge journey along with my office duties.

During my master semesters, I enjoyed working with kind and talented people - my classmates that share the same outstanding enthusiasm, having inspiring discussions, and sleepless nights which we were working together before deadlines, and for all the fun we have had the last year.

Finally, I would like to express my sincere appreciation and uniquely dedication of the complete thesis work to my section head department in Hellenic Cybercrime Prosecution Division Mr. Anastasios Papathanasiou, for his kindness, his entire trust in me and his clemency concerning my office duties during my master semesters by always being patronized facilitating with the attendance days of courses, reminded me every single time that every accomplishment starts with the decision to try. His attitude mentioned above inflicted me with the freedom to fulfill my purposes without any limitations and provided me with the chance to get in touch with so many fascinating and important people in the academic area. Without his guidance and encouragement, I would not have been able to think of this significant improvement in my knowledge.



# Table of Contents

<b>Abstract</b> .....	<b>5</b>
<b>Acknowledgements</b> .....	<b>9</b>
<b>List of Figures</b> .....	<b>13</b>
<b>List of Tables</b> .....	<b>14</b>
<b>List of abbreviations</b> .....	<b>15</b>
<b>Chapter 1: Data Analysis and Machine Learning Methodology- A typical approach</b> .....	<b>17</b>
1.1 An Overview to Machine Learning scientific field.....	17
1.2 Supervised Learning .....	18
1.3 Unsupervised Learning .....	20
1.4 The affection of Data Preparation in Machine Learning Operations.....	20
1.4.1 Data preparation procedures .....	21
1.5 An overview strategy to approach a classification problem .....	25
<b>Chapter 2: Serverless Computing</b> .....	<b>27</b>
2.1 Overview .....	27
2.2 An introduction to Monolithic Application Architecture.....	27
2.3 Microservices .....	29
2.3.1 Definition –Architecture – Characteristics .....	29
2.4 Serverless computing and its impact to microservices architecture.....	30
2.5 Function as a Service – FaaS .....	33
2.5.1 The Available Serverless Frameworks and the selection of Openwhisk platform .....	34
2.5.2 Why Openwhisk?.....	34
2.5.3 Apache Openwhisk .....	34
2.5.3.1 Openwhisk main components .....	35
2.5.3.2 Openwhisk Architecture.....	36
2.5.3.3 Openwhisk Installation and Configuration .....	40
2.5.3.4 Openwhisk CLI and native commands .....	42
2.5.3.5 Additional information.....	43
2.5.3.6 Serverless vulnerabilities .....	44
<b>Chapter 3: A Proposed ML-FaaS Business-Intelligence model</b> .....	<b>45</b>
3.1 Overview .....	45
3.2 The proposed approach .....	46
3.3 Offline Phase.....	49
3.3.1 Definition of the problem.....	49
3.3.2 Dataset Collection.....	50
3.3.3 Dataset explanation .....	50

3.3.4 Fundamental instructions for data manipulation.....	54
3.3.5 Exploratory Data Analysis - EDA .....	54
3.3.6 Dimensionality Reduction – PCA.....	65
3.3.7 Feature Engineering .....	70
3.3.8 Data Cleaning.....	72
3.3.9 Label encoding for categorical variables .....	73
3.3.10 Dealing with imbalanced dataset .....	74
3.3.10.1 Proposed strategies for data transformation to test machine learning models.....	75
3.3.11 Machine Learning chosen models for card fraud detection.....	78
3.3.11.1 Dataset’s separation & the chosen Validation method .....	79
3.3.11.2 Chosen ML models for the classifier’s construction.....	80
3.4 Online Phase.....	87
3.4.1 Fundamental instructions-guidance for Openwhisk functions .....	87
3.4.2 Defining the usage of supported Openwhisk Docker containers.....	88
3.4.3 Openwhisk functions (actions) .....	90
First Openwhisk action: .....	90
Second Openwhisk action: .....	91
Third Openwhisk action:.....	91
Action chaining .....	92
<b>Chapter 4: Evaluation Performance.....</b>	<b>95</b>
4.1 Overview .....	95
4.2 Offline Phase evaluation performance .....	96
4.2.1 Evaluation measures description.....	96
4.2.2 Chosen evaluation measures for the fraud detection classifier.....	97
4.3 Online Phase.....	102
4.3.1 System’s time execution performance .....	102
<b>Chapter 5: Experience acquired from implementation.....</b>	<b>105</b>
5.1 Framework combability problem.....	105
5.2 The proposed solutions to defeat the problem .....	106
<b>Appendix_A .....</b>	<b>109</b>
<b>Bibliography .....</b>	<b>111</b>

---

# List of Figures

---

Supervised Learning Procedure.....	18
Unsupervised Learning Procedure .....	20
Our proposed business logic to approach a problem solvation .....	25
The difference between the monolithic and microservices architecture .....	30
The evolution of Serverless Computing.....	32
The growth of Serverless.....	33
Openwhisk’s entire procedure for an action execution .....	37
Openwhisk installation and configuration in a local machine.....	41
Proposed ML-FaaS Approach .....	48
Imbalance Dataset – Fraud/Authorized Transactions.....	55
Percentage of Missing Values in datasets train and identity .....	56
Countplots from dataset features concerning the fraud transactions.....	58
Distribution plots for features card1, card2, card3, card5, addr1, addr2 .....	59
Distribution of both transactional status (features card1, card2, card3, card5).....	60
Time-series fraud visualization .....	61
Transactional components for the first day of March .....	62
Scatterplot - Amount of transaction by hour .....	63
Correlation Heatmaps for various features.....	64
Dataset’s Feature importance .....	69
New features for both transaction status .....	71
SMOTE representation in 2D feature space with 3 nearest neighbor parameter .....	76
Implementation of TOMER-LINK technique for majority class sample reduction .....	77
Testing Openwhisk constructed pipeline via the REST API call.....	94
Barplots depicting Recall and AUC measures for the tested ML models.....	99
Boxplots for each action’s execution time (Duration in seconds).....	102
Time-plots for Openwhisk actions (Duration in seconds).....	103

---

# List of Tables

---

Popular Supervised ML Algorithms.....	19
Openwhisk’s action constraints.....	39
Creation-Update-Invocation-Removal commands for an action in Openwhisk .....	42
Retrieval of activation logs for each action/or sequence of actions .....	42
REST API creation command to invoke an action/sequence of actions .....	42
Creation-Update-Invocation-Removal commands for a sequence in Openwhisk .....	43
Main Features of transaction dataset .....	51
Subsample of the file transaction.csv .....	52
Subsample of the file identity.csv .....	53
Subsample of the calculation of covariance matrix.....	67
Subsample of the calculation of Eigenvectors from covariance matrix .....	68
Subsample of the calculation of Eigenvalues from covariance matrix .....	68
Custom-made dictionary for implementation of feature engineering technique.....	70
Dictionary for implementation of label encoding technique.....	73
Implemented ANN architecture .....	85
Dockerfile contents.....	89
First action construction .....	90
Second action construction.....	91
Third action construction.....	92
Sequence of functions construction.....	93
REST API construction .....	93
Evaluation Measures results .....	98
Confusion Matrix for Logistic Regression – all cases.....	100
Confusion Matrix for Random Forest – all cases.....	100
Confusion Matrix ExtraTrees – all cases.....	101
Confusion Matrix for ANN – all cases.....	101

---

# List of abbreviations

---

ML	Machine Learning
SVM	Support Vector Machines
MLP	Multilayer Perceptron
DBSCAN	Density-based spatial clustering of applications with noise
EDA	Exploratory Data Analysis
PCA	Principal Component Analysis
HTML	Hypertext Markup Language
XML	Extensible Markup Language
CSS	Cascading Style Sheets
API	Application programming interface
DevOps	Developer Operations
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
FaaS	Function as a Service
JSON	JavaScript Object Notation
VM	Virtual Machine
CAGR	Compound Annual Growth Rate
AWS	Amazon Web Services
CLI	Command Line Interface
REST	Representational state transfer
HTTP	HyperText Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
OS	Operation Systems
JDK	Java Development Kit
NoSQL	Not Only Structured Query Language
DB	Database
NAT	Network Address Translation
YML	YAML Ain't Markup Language
wsk	Openwhisk

SOA	Service Oriented Architecture
AUC	Area Under the Curve
SMOTE	Synthetic Minority Oversampling technique
CNP	Card Not Present fraud
ATM	Automatic teller machine
IP	Internet Protocol
ISP	Internet Service Provider
UA	User Application
USD	United States Dollar
N/A	Not Available
MCAR	Missing completely at random
MAR	Missing at random
MNAR	Missing not at random
ANN	Artificial Neural Networks
MSE	Mean square error
RELU	Rectified Linear Units
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
CM	Confusion Matrix
ms	Milliseconds
h5	Hierarchical Data Format 5
GPU	Graphics Processing Unit
TF	Tensorflow
AMI	Amazon Machine Image
EC2	Elastic Computer Cloud
CUDA	Compute Unified Device Architecture
cuDNN	NVIDIA CUDA Deep Neural Network
intel mkl-dnn	Intel(R) Math Kernel Library for Deep Neural Networks
pb	protobuf format
Amazon S3	Amazon Simple Storage Service
STaaS	Storage as a Service
DaaS	Data as a Service
SPI	Software Platform Infrastructure
SECaaS	Security as a Service



# Chapter 1

---

## Data Analysis and Machine Learning Methodology- A typical approach

---

### 1.1 An Overview to Machine Learning scientific field

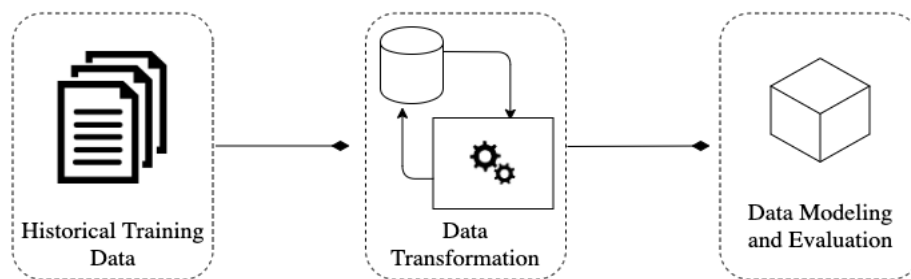
As the technology is evolving rapidly, the definition of data obtained a priceless valuation and became the trend of the century due to the fact that social networking has an enormous community of supporters, Internet of things gained a significant reputation because of innovated systems like smart homes and smart cities and the high percentage of online e-commerce transactions. The worldwide web has a powerful impact on our lives. It is a sophisticated multidisciplinary decentralized mechanism enabling individuals to communicate with one another, annihilating the geographical distance and bringing users together in topic-based communities that are not tied down to any specific place. We are living in an all the way networked, globalized society connected by new technologies. We use the Internet to interact with one another, which accordingly poses new challenges to privacy and security. The Internet is the most vital source for data generation. A remarkable volume of new data is transmitted unintentionally and invisibly to the most significant percentage of the users directly as a result of Internet-connected devices communicating with each other once they are powered up. People interact beyond the Internet, generate data, giving as a result of new trends, covering necessities, and as a consequence, new products, and services are created. The last three years, the definition of data obtained a priceless valuation and became the trend of the century due to the fact that social networking has an enormous community of supporters, Internet of things gained a significant reputation because of innovated systems like smart homes and smart cities, the percentage of online transactions has been remarkably raised, smartphones and mobile devices become the lifeline of our society, new web-based applications are launching day by day improving users life. This giant data diffusion caused the necessity for utilization and manipulation of them in order to extract useful information, discover trends, make predictions, and -at a higher level- aid business decision making.

Major companies and organizations use data analytics to build business strategies, create new challenges and opportunities, provide new advertisement campaigns, design new applications, and elevate the marketing section. Data analytics have become, without any doubt, the most notorious scientific field finding semantic support from Academic researchers. Data Analytics is based on the framework, which is called Machine Learning.

Machine Learning has a leading role in data extraction knowledge; it is a research field considered as a crossroads of statistics, artificial intelligence, and computer science and is also known as predictive analytics or statistical learning. The application of machine learning methods has become a part of users' life. Known artificial intelligence applications such as recommendation systems for Products and Services, even many modern applications concerning face, speech and hand-written recognition, sentiment-analysis applications, spam filtering, are built under the guidance of machine learning scientific field. The most attractive part is that Machine Learning makes computers learn nearly as well as people, and as a result, by using the power and the utilities of computer resources, designs automate decision-making processes. According to the type of the problem, Machine Learning introduces two definitions of learning a) Supervised Learning and b) Unsupervised Learning.

## 1.2 Supervised Learning

Supervised Learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. A supervised learning algorithm learns from already labeled data and helps to make decisions and predictions for unforeseen data. Supervised Learning allows us to collect data, or produce a data output from the previous experience, optimize performance criteria using experience, aids in solving a wide range of real-world computation problems. The below scheme depicts the entire procedure that a supervised Machine Learning Algorithm follows to fulfill its purpose:



**Figure 1 Supervised Learning Procedure**

According to the above figure, the procedure begins with the collection of the appropriate historical data. Eventually, the data analyst performs transformation techniques in order to reshape the structure of the data to feed the algorithm. The final step includes the determination of the suitable machine learning algorithm to be trained with the mentioned data.

As the training has been finished, the analyst evaluates its performance. If the evaluation metrics have high quality, formerly the model is ready to make predictions in new unseen data; otherwise, endeavors to find the correct algorithm which will bring the best results to achieve the best performance.

Supervised Learning according to the business problem can be divided into two subtypes which are: a) Regression and b) Classification.

Regression is a statistical approach to identify the correlation between variables obtained from the dataset. For regression tasks, the goal is to predict a continuous number (a real number). Regression task is -for instance- the prediction of a person's annual income according to his education, experience, and age. Another example is the prediction of the price in stock exchange in line with financial circumstances. Even the prediction of weather temperatures or the price of a house according to its location and the year of construction constitute regression problems.

On the other hand, in classification, the main target is the prediction of the class label, which is a choice from a prebuilt set of possibilities. Classification sometimes is separated into binary classification and multiclass classification. Binary classification is the particular case of distinguishing between exactly two classes in contrast with multiclass, which is the classification between more than two categories (classes). According to the type of supervised Learning, many machine learning algorithms can approach the problem.

The below table presents the most popular machine learning algorithms in supervised Learning:

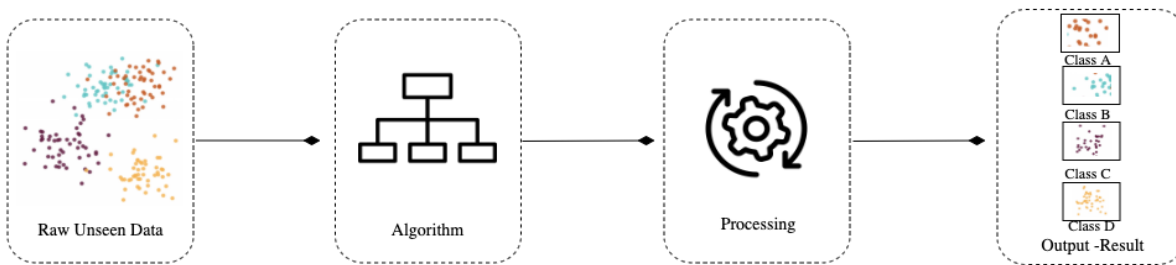
**Table 1 Popular Supervised ML Algorithms**

• Simple Linear Regression (approaches Regression Problems).
• Decision Trees (approaches Classification Problems).
• Support Vector Machines – SVM (approaches both Classification and Regression Problems).
• Multilayer Perceptron Neural Network – MLP (approaches both Classification and Regression Problems).
• Convolution Neural Networks (approaches Classification Problems, enlisted in Deep Learning Techniques).
• Logistic Regression (approaches Classification Problems).
• Naïve Bayes (approaches Classification Problems).
• Regressor Trees (approaches Regression Problems).
• K-Nearest Neighbours (approaches both Classification and Regression Problems).
• Random Forest (approaches Classification Problems).

### 1.3 Unsupervised Learning

The second section of the Machine Learning field is described as Unsupervised Learning. Unsupervised learning encompasses all kinds of machine learning where there is no known output, no guidance to the learning algorithm. It is unsupervised because we do not provide to the model any labels. In unsupervised learning, the learning algorithm is just shown the input data and required to extract knowledge from these data.

A well-known type of unsupervised learning is Clustering. Cluster analysis is a procedure of grouping data in clusters based their common characteristics. Clustering has a plethora of uses in ordinary life. Real world application including social networking analysis, customer segmentation, image segmentation anomaly detection. The beneath table depicts the clustering procedure:



**Figure 2 Unsupervised Learning Procedure**

The above diagram depicts a group of raw scatter data, which supplies a clustering algorithm, performs the processing, and eventually are separated in clusters based on their common characteristics. There are different types of clustering methods, such as Hierarchical Clustering, Density-Based Clustering, and Model-based clustering.

The most popular machine learning algorithms for Clustering with wide usage from Data Scientists are K-means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

### 1.4 The affection of Data Preparation in Machine Learning Operations

Beyond doubt, Machine Learning is a powerful weapon that helps in several cases to overwhelm the given problem. Before the utilization of Machine learning methods, a data scientist aspires to understand the data adequately rather than as a component of a more comprehensive automatic system. Data preprocessing is an indispensable step in Machine Learning as the quality of data - and the useful information that can be derived from it directly- affects the ability of the selected model. One of the most essential and crucial procedures in Data Analysis is the selection of the most effective strategy to clean, modify, fill, and transform the given -under construction- dataset. A well-defined preprocessing action ensures the quality of data providing a more robust structure for them in order to achieve a successful result for the project. Unluckily, a wide range of datasets do not always have the expected structure; usually, these are incomplete, inconsistent, without the correct format, having

useless information, and they cannot be sent without the proper modification into a machine learning model. In this section will dive into and expand on the typical steps performed during an analysis.

### 1.4.1 Data preparation procedures

After the problem has been defined, the goal is determined, and the dataset is selected, the analyst must execute a set of procedures that constitutes the data preparation. The below table presents the typical and most famous data preparation steps, which will be discussed more in detail.

**Table 2 Typical Data Preparation procedures**

• Exploratory Data Analysis – EDA.
• Dimensionality reduction (i.e. Principal Component Analysis).
• Data cleaning – (Imputation of missing values, removing outliers).
• Feature engineering.
• Encoding categorical data.
• Feature Scaling

#### 1.4.1.1 Exploratory Data Analysis – EDA

Before starting to apply various methods and techniques concerning the data preprocessing procedure, the data analyst follows a typical approach to analyze the dataset, extract useful information of it, discover insights and summarize its characteristics with the support of visual statistics tools. The method mentioned above is introducing as Exploratory Data Analysis (EDA). Every machine learning problem starts with EDA. Commonly, datasets are provided in the form of tables, and when these have a vast size, it is impossible to comprehend data. According to the above situation, EDA methods come to answer essential questions by plotting various graphs to understand the hidden patterns. Some of the typical plots used for Exploratory Data Analysis are Histograms, Distribution Plots, Scatter plots, Box plots, and Pie charts.

- **Histograms:** Histograms are providing a sharp understanding of density of underlying distribution of data more accurately probability distribution of data.
- **Distribution Plots:** These plots are utilized to identify distribution of attributes whether they are normally or skewed distributed.
- **Scatter Plots:** A scatter plot is a type of plot which uses X and Y coordinates on a two-dimensional space to illustrate points. It is a popular plotting technique to examine the interdependence of one variable over other.

- **Pair plots:** When the data is more than two – dimension, Pair Plots investigate the behavior of all variables along every other variable.
- **Box Plots:** Box plots are based on percentiles and provide a quick way to visualize the distribution of data. They also have lines extending vertically from the boxes indicating the variance outside the upper and lower quartile. The Space between different parts of the box indicate the variance (spread) of the data. It also helps in detection of outliers.
- **Pie charts:** Are commonly used to give an overview glimpse in categorical data, to show differences within groups based on one variable.

### 1.4.1.2 Dimensionality Reduction

In several cases, datasets have a plethora of features. As the number of features increase, the dataset becomes more complicated. Some features are advantageous, providing semantic information for the analysis overlapping the greatest part of the dataset’s variance, and some of them are entirely useless. A high-dimensional dataset in classification problems can cause overfitting problems. When a machine learning model is trained on a large number of features, gets increasingly dependent on the data which was trained on, becomes overfitted giving; as a result, a poor performance on new unseen data, beating the generalization purpose. The application of the dimensionality reduction technique ensures the reliability of the information, cutting its worthless components, reducing the size of the dataset becoming more eligible for manipulation. Also, the most common motivation for dimensionality reduction is visualization, which achieves the compression of the data giving them a representation that is more informative for further processing.

One of the simplest and most widely applied algorithms for dimensionality reduction is Principal Component Analysis (PCA). PCA is a statistical tool that tries to obtain a possible correlation between variables, with the support of orthogonal transformation. The result that PCA introduces is a subset of new features, in a new sub-space according to how important they are for explaining the data. The methodology of PCA is based on a compound of linear algebra and matrix operations concerning matrix multiplication, matrix transposition, inverses, decomposition, calculation of eigenvectors and eigenvalues and statistical definitions such as variance, covariance.

In data analysis language, a basic strategy of PCA begins with the calculation of a square symmetric matrix, the covariance matrix; which portrays how the particular dataset’s variables all relate to one another. Consequently, the procedure proceeds with the computation of eigenvectors and eigenvalues from the covariance matrix, distributing eigenvalues in descending order and the selection of the top k Eigenvectors that correspond to the k largest eigenvalues. The eigenvectors and eigenvalues of a covariance matrix represent the “essence” of a PCA due to the fact that Eigenvectors (principal components) define the directions of the new feature space, and the eigenvalues determine their magnitude. In other words, the eigenvalues describe the variance of the data onward the new feature

axes. It means the corresponding eigenvalue informs how much variance is included in that new modified feature.

Subsequently, the projection matrix  $W$  from the chosen  $k$  Eigenvectors is assembled, and as the last step, it remains the transform the original data set  $X$  via  $W$  to obtain the new  $k$ -dimensional feature subspace  $Y$ .

This top final top- $k$  feature subspace corresponds to those  $k$  feature space with the largest percentage of variance, which is enough to describe the data set.

### 1.4.1.3 Preprocessing procedure

After the PCA technique is completed by keeping the most important features, the analyst proceeds to the next step, which is the preprocessing procedure. As was mentioned, one of the most arduous procedures is to transform the given data having a comprehensible appearance. The preprocessing phase follows a variety of strategies like a) Imputation of missing variables, b) Label Encoding, c) Feature scaling, d) Feature Engineering.

Data Science confronts obstacles in the analysis procedure because of the presence of missing values. It has been observed that various datasets, mainly structured datasets, have a generous percentage of undefined values, commonly named missing values. Missing data arise in nearly all severe statistical analyses. Usually, appear as NaN values, and it may create problems for the selected machine learning algorithm leading to misleading results or even cause errors to the chosen model functionality. Moreover, the cumulative impact of missing data in several variables usually heads to the exclusion of a substantial proportion of the initial sample, which in turn creates a significant loss of precision and mastery. In a given structure dataset, missing values can be replaced with: a) the most common value according to the values of its feature, b) the zero value, c) the mean value, d) the median and e) even to delete the entire feature if it consists more from missing values rather than typical values.

While EDA visual techniques are performed, it is usual to observe data points that are quite different from the rest. In respect to the Statistics field, an outlier is defined as the observation point that is distant from other observations. Those irregular data points can lead to trouble for scaling techniques and affect the performance of the machine learning model. The outliers can be a consequence of a mistake throughout data collection, or it may simply be evidence of variance in the given data. Analysts can ignore outliers or detect and remove them. Visualization method, which was described in the above section with detail like boxplots and scatterplots, affords an easy way to recognize outliers, removing, or replace them with different values.

Another chapter in the data preprocessing procedure is the Feature Engineering technique. This technique constitutes the process of mining specialty information of the data to produce new features that improve the performance of the selected machine learning algorithms. Feature engineering is crucial to the application of the machine learning field and is both challenging and expensive. Discovering new valuable information from the already existing features leads to the most precise

results, representing the underlying structure of the data and therefore performing the best model. Features can be masterminded by decomposing or splitting features, from external data sources, or aggregating or combining features to create new features. Feature engineering techniques can be beneficial when the analyst has to deal with datasets having various decoded features.

It has been observed that structure datasets consist of a wide range of categorical qualitative data. These types of data are not providing numerical values, are measures of 'types,' and maybe represented by a name, symbol, or a number code. A real-world dataset would have a combination of continuous and categorical variables. Many Machine Learning algorithms, like tree-based methods, can naturally deal with categorical variables. Nevertheless, algebra-based algorithms such as linear/logistic regression, Support Vector Machines, K-Nearest Neighbours accept only numerical features as input. As a result, it is required to transform the categorical variables present on the given dataset into numbers. It is not appropriate to drop them from the dataset as they are known to hide much interesting information. The most well-known encoding methods, such as Label Encoding and One-hot Encoding, are widely used from data analysts to transform their data. The label encoding method converts non-numerical labels into numerical labels. Each category is assigned in a unique label starting from 0 and going on till N categories – 1 per feature. Label encoders are suitable for encoding variables where alphabetical alignment or numerical value of labels is essential. On the other hand, One-hot encoding works by building a column for each category present in the feature and assigning a 1 or 0 to indicate the appearance of a category in the data.

Normalizing data encourage many models to perform more beneficial after this is done. Especially those that depend on a distance metric to determine similarity. In numerous cases, a standardizing process is applied to the collected data so that it has a mean value of zero and a standard deviation of one. The purpose of normalization is to modify the values of numeric columns in the dataset to a standard scale, without changing differences in the ranges of values. As a note, Tree algorithms treat each feature on their own without needing the normalization procedure. On the other hand, neural network models have been proved that are working with normalized data.



### 1.5 An overview strategy to approach a classification problem

According to the methods as mentioned above concerning data transformation in combination with the essence of Machine Learning techniques, we are capable to design an overall business workflow logic to approach the given problem.

This thesis approaches a classification problem concerning structured data (Supervised Learning) by using the fundamental knowledge of the Machine Learning field in combination with data analytical skills. Our proposed workflow strategy of the entire procedure is introduced in the below scheme and will be discussed in the next chapter:

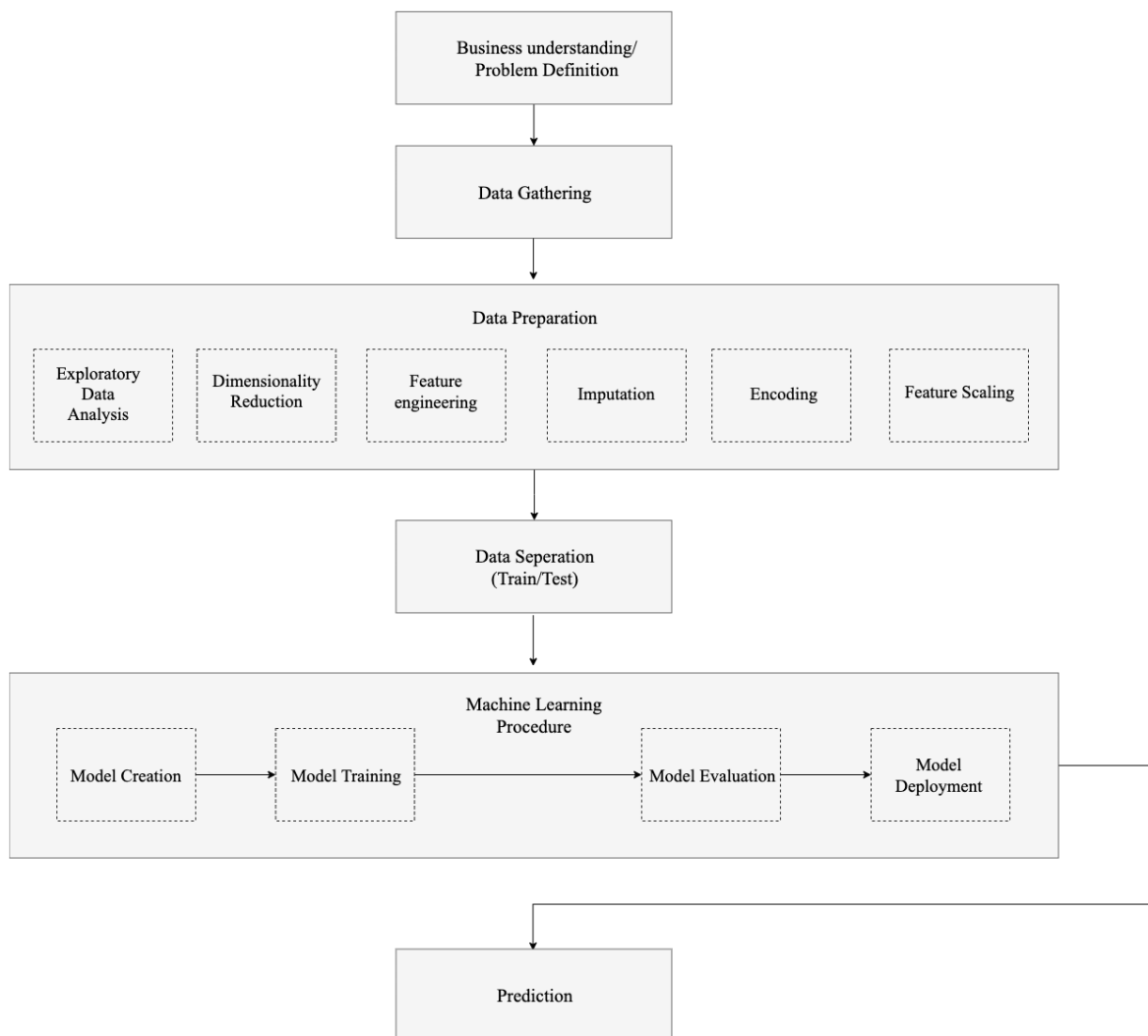


Figure 3 Our proposed business logic to approach a problem solvation



# Chapter 2

---

## Serverless Computing

---

### 2.1 Overview

In this chapter, we are working on emphasizing the valuation of cloud-based tools with respect to those, which propose an entire solution in order to perform our custom web-services without an extreme effort to control their vital components. Mainly, we are expanding in definitions of Serverless computing, its characteristics, and the concept Function as a Service (FaaS). At the end of this chapter, we are introducing Apache Openwhisk, a serverless distributed event-driven platform, since the entire business logic -concerning a classification problem- in this thesis- has been split and transferred into it. Before we begin to dive into core definitions, we are going to present the evolution of the creation and implementation of web-based applications. Starting from monolithic web application architectures, proceeding with cloud microservices, and conclude with serverless models and the deployment and configuration of Openwhisk (in a local machine).

### 2.2 An introduction to Monolithic Application Architecture

A web application acts as a special kind of client-server logic, where a large percentage of the functionality is launched back to the server-side even though the Web does not define what is behind the server. The Web relies heavily on the client-server model, and it uses popular languages such as HTML and XML, to transfer and represent data/content. Under this representation, there are various programming and scripting languages that can explosively process, transform, and produce data or provide a user's interface. In this manner, the development of Web applications can be implemented under the guidance of software engineering, which also needs to be extended. Web applications are versatile because they combine knowledge concerning software engineering, network computing, database modeling techniques, and conclusive interface design. Web applications are built in a continuously changing environment, confronting combability obstacles, where requirements are unstable, and the user community is more extensive than before. Web applications manage information from various sources dealing with structuring, processing, storing, and presenting this information. In particular, a typical web application consists of layers, in other words, three tiers including a) the User Interface (Presentation) level, b) the Business Logic level, and c) the Data Layer.

- **The presentation layer:** concerns what is needed to be seen from the user's perspective and gives the mechanisms for interaction. The specific kind of interaction depends on the custom application. Not all the applications have the same scope; one web app can be created in order to show just information without any user's interaction. In that case, not an advanced and sophisticated architecture is needed. The most well-used practical applications (have the logic to give data as an input, send it for processing, and then receive feedback, which can be the final result or a step for further progress. This layer combines user interface development processes (HTML scripts, CSS stylesheet language) with client-side scripts (JavaScript, Python -recently-). All the knowledge above can provide a productive ecosystem for user interaction and content performance. Definitely, it can be assumed that server-side scripts can be utilized to generate content, but at the ultimate level these scripts generate the HTML that will be shown to the final user through the browser, so this role of the development can be partitioned in two phases: first, the content generation which is created by the business logic layer and second, the presentation content to the view layer, sustaining the logical division of the application. As an overall, this layer interacts with the business logic layer beneath, transferring the information from the user and controlling it, then outcomes back any response it generates without leaving any decisions of the application's logic to be resolved by the user interface.
- **The business logic layer** is the essential tier in the application since it is the skeleton of the entire program. This level receives data from the "higher" level (UI) and modifies it, applying internal application logic. It also reclaims data from the most profound data level and uses it to the logics. Moreover, by integrating these two processes, it can perform modifications in both levels as well. The business logic layer is practically the leading manager of the entire workflow. The obscurest level in the layered architecture is the data layer (data tier), which deals with data retrieval from its sources.
- **The data layer** contains the database/data storage system and data access layer.

Considering those mentioned above, we proceeded to the conclusion that the construction and deployment of a monolithic application forms the necessity of a full-stack development background (front and back end decisions). The architecture, as mentioned above, indubitably, is extremely useful in many cases providing various benefits. Still, on the other hand, it gives limits to scalability, high rate of expense, and deep complexity. Moreover, the construction of a monolithic application may cause problems in the performance because in case it has a big size, it is difficult to make changes fast and correctly. Another obstacle with monolithic applications is the reliability; a confusion in any module can probably overthrow the complete process.

Furthermore, since all instances of the application are indistinguishable, that confusion will affect the availability of the entire application. Also, the adoption of new technologies, due to the fact that changes in frameworks or languages will affect an entire application, it is costly in both time and price.

## 2.3 Microservices

### 2.3.1 Definition –Architecture – Characteristics

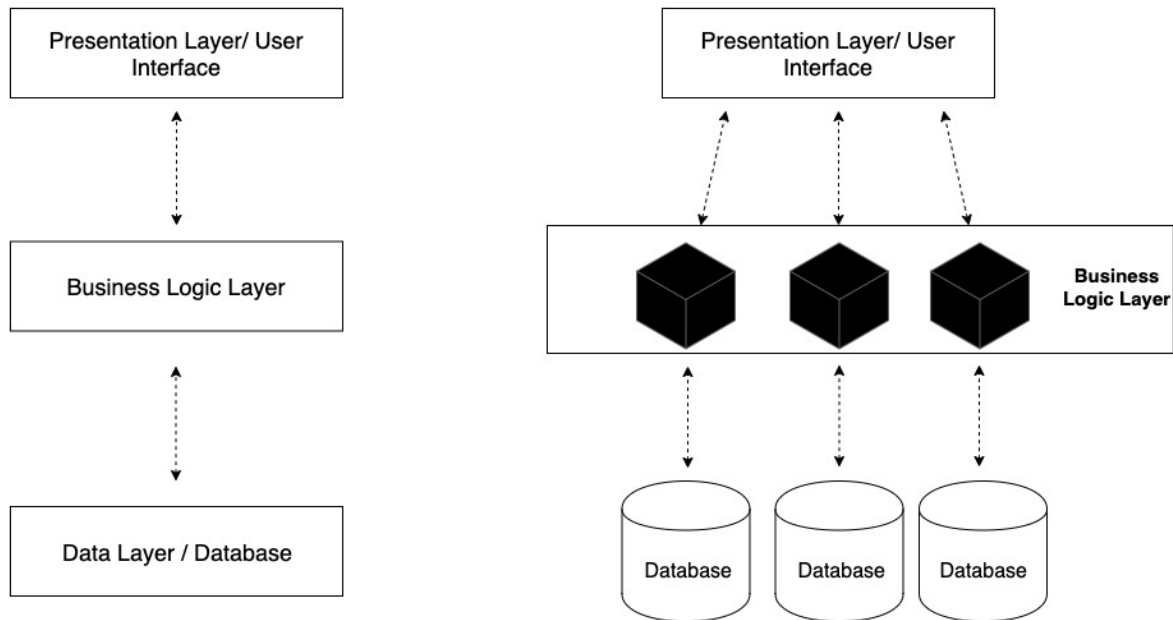
Because we are living in the Cloud epoch, the necessity for agility and scalability begat virtual machines, and the demand for technology tools, services, and platforms brought in the scenery the concept of microservices. Microservices described as an architectural approach to optimize resources that give compute, storage, and networking for at-scale services and software on modern, fast, distributed infrastructure; more formally, microservice architectural style is an approach to developing an entire application as a group of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies (James Lewis and Martin Fowler -2014 -)<sup>9</sup>. Microservices constitutes the logic of an extensive application, which is formed as a series of modular services; this logic groups application parts into nodes of logically related elements according to the business concern. Microservices architecture is described by the below characteristics:

- Each service is autonomous.
- It is independent as it concerns the code, managed, and developed by a small team.
- Each service is adjustable as it concerns the technological framework. It can choose the best technology stack for its use cases.
- Each service is responsible for a particular part of the functionality (business capability), ensuring its reliability and correctness.
- Each service has its DevOp program (testing, release, deployment, scaling, integration, and control)
- Each service is deployed in a self-contained (isolate) environment, without affecting the rest services.
- Vices parts communicate with each other by applying well-defined APIs (smart endpoints) and simple protocols like REST over HTTP.

---

<sup>9</sup> <https://martinfowler.com/microservices/>

- Each service is responsible for persisting its data and keeping external state (Only if multiple services consume the same data, such situations are handled in a common data layer).



**Figure 4** The difference between the monolithic and microservices architecture

According to the previous figure and the characteristics mentioned above, we can make a synopsis that brings as a result that in the microservices architecture, each service owns its business logic task in contrast with monolithic architecture in which the business logic tier is constructed as a unique solid component. Additionally, instead of sharing a single database like in Monolithic application, each microservice has its database. The owning of a database by service is essential because it guarantees the interconnection of service components. Furthermore, a service can use a type of database that is best suited to its needs. Microservice architecture ensures, better testability due to the fact that services are smaller and faster to test; better deployability because each service can be deployed independently, furthermore since its size is small (at least smaller than monolithic architecture component) significant changes to an existing service is more comfortable procedure, and it can be rewritten it using an all the way new technology stack.

## 2.4 Serverless computing and its impact to microservices architecture

Although the cloud providers offered -via virtualization and containerization- resources for building microservices and “self-service” services (IaaS, PaaS, SaaS, etc.), developers desired to make their life more comfortable by building their applications - in the concept of microservice architecture- concentrating undividedly on business logic and the source code without care about the

allocation of resources providing horizontal scalability, and the ability for the real-time consumption of services. This necessity introduced the definition of Serverless computing. First, was introduced -in 2014- from Amazon, as “An introduction to Lambda architecture,” and afterward, vendors like Google and Microsoft developed their serverless models (Google Cloud Functions, or Microsoft Azure Functions, respectively).

**Definition of Serverless Computing<sup>10</sup>:**

*Serverless architectures are application designs that incorporate third-party “Backend as a Service” (BaaS) services, and/or that include custom code run in managed, ephemeral containers on a “Functions as a Service” (FaaS) platform. By using these ideas, and related ones like single-page applications, such architectures remove much of the need for a traditional always-on server component. Serverless architectures may benefit from significantly reduced operational cost, complexity, and engineering lead time, at a cost of increased reliance on vendor dependencies and comparatively immature supporting services.*

The serverless computing model assigns tasks, both developers and cloud providers. As it concerns developers, serverless computing offers a full concentration in building code and business logic, and as it concerns the provider, it assigns the entire management of the resources.

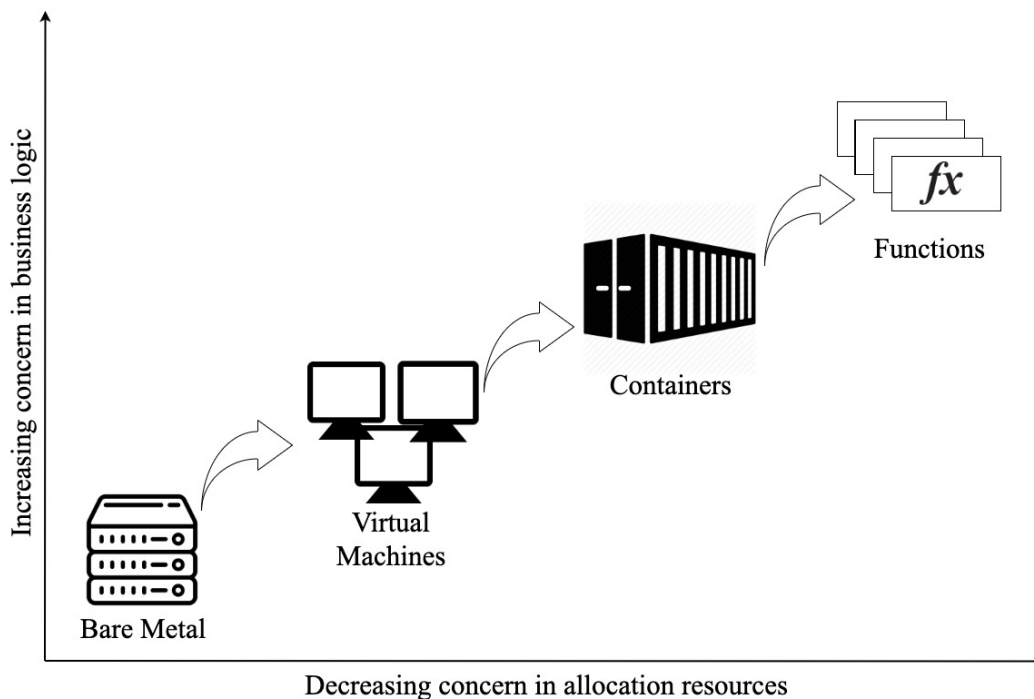
Serverless does not mean that servers are canceled; but eliminates the server management from the user/developer perspective; In the serverless framework, the user never needs to take care of, or even be aware of, any single machines — the infrastructure is entirely abstracted away. In serverless models, the cloud provider is managing the allocation resources and takes care of the infrastructure by debugging, maintaining, and monitoring it. Below, are introduced the most important characteristics of serverless computing:

- a) The serverless environment is most proper for applications needing processing in the cloud because it allows splitting the application into multiple, more straightforward services (microservices architecture),
- b) The serverless model offers the “pay-as-it-is requested” method (price is based mostly on the on-demand instantaneously scalability); this indicates that the cost for using the service is calculated according to original consumption rather than pre-purchased services based on guesswork. It is one of the significant advantages since, for years, the cost of provisioning servers and maintaining was "twenty-four seven (24/7)" causing high pricing,
- c) Serverless computing focused on building Functions (Function as a Service -FaaS),

---

<sup>10</sup> <https://martinfowler.com/articles/serverless.html>

- d)** Serverless constitutes a multilanguage environment by supporting a wide variety of programming languages including Node.js, Java, Python, JavaScript, .NET, Ruby and Swift. Furthermore, some serverless platforms can use out-of-the-box programming languages with the support of Docker pre-build images.
- e)** Presently, serverless platforms typically execute a single main function that takes a dictionary (a JSON object) as input and produces a dictionary as a result.
- f)** Provides the capability to build a chain of connected microservices (functions).
- g)** Every serverless platform has its custom debugging way by using print statements that are recorded in the execution logs. Also provides capabilities to help developers find bottlenecks, trace errors, and better understand the circumstances of function execution.



**Figure 5** The evolution of Serverless Computing

Serverless technology is not an entirely new model in the cloud world because it is based on virtual machines and containers. Several serverless platforms live under the hood of the cloud, and their core components approach the philosophy of containerization. Serverless upgraded the level of the building and deploying an application (a service).

First, there were Physical servers that offered excellent performance but provided the slightest flexibility for consolidation of workloads; Bare metal servers cannot separate the applications from the underlying hardware. The appearance of virtualization allowed bare-metal resources to be partitioned into multiple operating system instances. Virtual machines enabled to decouple multiple workloads from a physical machine, thereby decreasing the concern for infrastructure implementation.

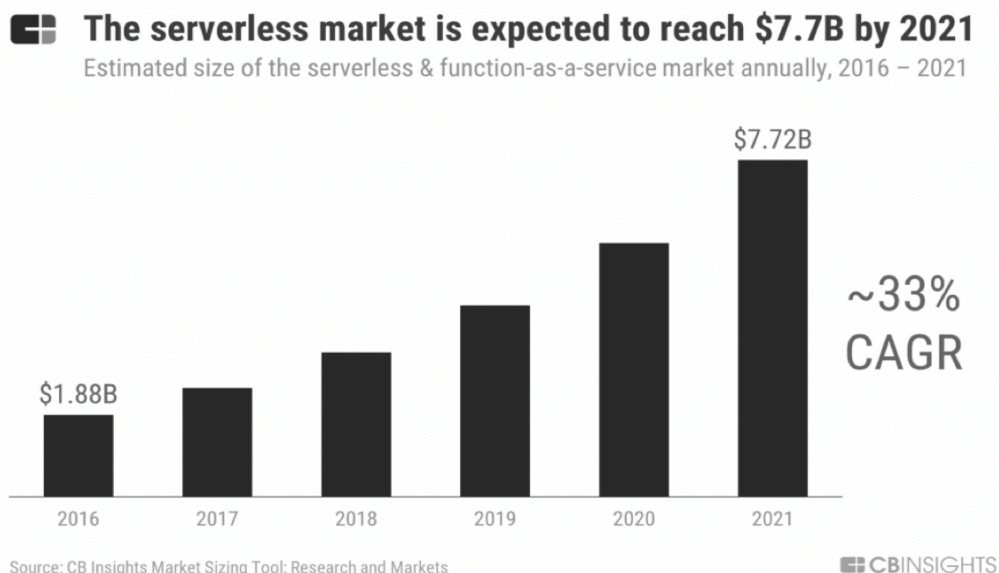
Next, Docker containers provided a light-weight alternative to VMs by abstracting out the operating system in addition to the bare metal hardware. Therefore, containers encapsulated only the



application and its dependencies and presented as a light-weight alternative to virtual machines. With the support of Docker containers, developers could bundle their artifacts (code + app and their dependencies) effectively and be confident that what runs on their machine would run in dev/test and production. Functions reduce the work required from the developer to get the code running even more than containers.

## 2.5 Function as a Service – FaaS

The serverless computing brought in the prospect of the Function concept; A function is the most important part of the serverless model due to the fact that the entire business logic of an application is stored in pieces inside functions (a chain of functions). Its task concerning the execution of a specific and custom computation, which is defined by the developer according to his business logic. Typically, a function is a piece of code that receives data as input, makes the appropriate computation, and returns an output as a result, nothing more and nothing less. Functions are strongly connected with the definition of Function as a Service since the last one provides the ability to deploy (onto the cloud) a single function or part of an application or even series of functions as an entire application. Furthermore, an essential feature of the serverless model is that its functions are stateless; this means that after each function process is completed, the lifecycle of the specific task that the function executes is destroyed. Stateful functions - in contrast with stateless functions - are expensive because they need external storage to save their data and also limit the scalability.



**Figure 6 The growth of Serverless<sup>11</sup>**

<sup>11</sup> CB Insights Market Sizing Tool-Research and Markets; <https://www.cbinsights.com/research/serverless-cloud-computing/>

According to a recent report CB Insights Market Sizing Tool, the serverless computing industry expected to have a tremendous grow up from (\$1.8B (2016) to (\$7.72B (2021) with 33% Compound annual growth rate (CAGR). Serverless computing can serve all types of applications, regardless of the industry or its custom case. Furthermore, it can improve the growth of the process overall and not only the functionality encountered by the end-user.

### **2.5.1 The Available Serverless Frameworks and the selection of Openwhisk platform**

As we mentioned, in 2014, the pioneer company -cloud vendor Amazon released its serverless model called Lambda. After this introduction, in 2017, Microsoft Azure<sup>12</sup>, Google Cloud<sup>13</sup>, IBM Bluemix introduced their own Functions as a Service serverless models to run event-driven code written in many programming languages. The first three services are provided by its own cloud provider except the Apache's serverless model called Openwhisk. Apache OpenWhisk was initially developed by IBM but later was donated to the Apache Software Foundation and released under a commercially robust open source license, the Apache License 2.0; this makes it open source, independent from limitations, friendly to commercial ventures (or users/researchers/developers) adopting it, and maintained in the long run.

### **2.5.2 Why Openwhisk?**

We chose Openwhisk because any Apache project has a working code base as well as an active community, adopters of Apache software can trust they will not be alone in using it. Also, Apache Foundation supported software, has a detailed list of compliance rules allowing everyone to use the released software without risking getting caught in the non-commercial traps that some other open-source licenses have. Additionally, to those benefits of adopting an Apache project, Openwhisk also is used in production and powers via IBM's BLUEMIX Cloud Services.

### **2.5.3 Apache Openwhisk**

According to the official Apache's Foundation website<sup>14</sup>, Openwhisk is an open-source, distributed Serverless platform that executes functions (fx) in response to events at any scale. In this subchapter, we are focused on explaining the Openwhisk user's visible components in its architecture and its main core components to understand its powers and weaknesses and the platform's limitations.

---

<sup>12</sup> <https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview>

<sup>13</sup> <https://cloud.google.com/functions/docs/release-notes>

<sup>14</sup> <https://openwhisk.apache.org/>

The serverless platform Openwhisk operates by executing stateless functions (which in its language are called **actions**) in response to events; this means that the entire procedure utilized when data from various sources (databases, message queues, IoT devices, websites) are passing into functions. That is why we called Openwhisk an event-driven platform. OpenWhisk supports source code, which is given as input that provisions executing a single command with its Unix-based command-line interface (CLI) and then delivers services through the web to multiple consumers, such as other websites, mobile applications, or services based on REST APIs.

### 2.5.3.1 Openwhisk main components

As we mentioned before, functions that the serverless platform executes must be stateless due to the fact the being stateful creates the necessity for external storage, decrease the limit of scalability and demands reliable synchronization between invocations which increases the load, the state-keeping infrastructure and as a result restricts the ability to grow. It is essential to know that stateless status is maintained only in the level of function construction, not in the entire Openwhisk environment. Openwhisk provides the management in infrastructure, and the developer writes the source code to create the actions that responds correctly when an event is coming.

Openwhisk provides a variety of components which are can be handled by the developer but the most well used are the below:

- **Actions**
- **Triggers**
- **Rules**
- **Packages**
- **Namespaces**
- **Activations**
- **APIs**

**Action** component corresponding to the construction of a function; Openwhisk support various programming languages to write the functions; these programming languages are Python, Node.js, Java,.NET, Ruby, PHP, Swift, also gives the capability to use out-of-the-box languages (i.e Scala, Go, C/C++) with the support of docker container as prebuild image. Each action receives data in JSON format, completes its tasks, and produces a JSON string similarly as a result (i.e., if the developer writes his source code in Python, he must provide as input a dictionary structured object and receive likewise, as a result, a dictionary). Furthermore, each action can be performed as web action in order to build web-based applications; this feature allows the developer to program backend logic in which his custom web application can be accessed anonymously by anyone user without requiring authentication. In addition, the action can be a zip including the file with the source code (according

to the chosen runtime i.e., Java, Python), and the necessary files (or virtual environments including necessary libraries) which, are important for the execution. Action can also be performed and as a ready-made docker container, but it must implement the specific action's interface. Openwhisk also gives the vital capability to connect each action with another. Pipeline actions use as input the output of the previous actions. Initially, the first action of a sequence will accept the parameters (in JSON format), and the last action of the sequence will provide the final result as a JSON string.

**Triggers and Rules** components add event-driven capabilities to the serverless platform. Events from outside and inside event sources are forwarding through a trigger, and rules allow the actions to react to these events. Triggers can be fired either by a user either by an external event source. Triggers and rules are strongly connected; A trigger is just an Openwhisk component without producing nothing by itself; in order to be activated (fired), it needs to be associated with a rule. The rule is the component that associates a trigger with an action, with every activation of the trigger forces the corresponding action to be invoked along with the trigger event as input.

**Packages** provides the ability to gather a set of related actions and share them with others; Packages including actions and feeds; Feeds is the mechanism which connects the data source with the trigger (and the rule) to be fired.

**Namespaces** also concludes Openwhisk entities (actions, triggers, rules)

**Activation** component heads into the platform's log data. Each Openwhisk action -when completes its task- produces log data; each action's log data are available via an activation ID. This component provides the ability to the developer to view the execution time for each function and detect analytical his possible errors.

**API** corresponds to the automated construction of a REST API endpoints, which will be associated with specific actions or a chain of actions. It is common to access the service through a REST API, which is the request's first responder for testing purposes. The preceding components are visible and manageable from the user side in order to create an application or a part of it; actually, the entire process is all straightforward from the user's point of view, but in reality, Openwhisk follows a series of procedures for an action to be executed.

### 2.5.3.2 Openwhisk Architecture

Openwhisk has been constructed with the support of popular open-source tools in combinations with its own custom core components, which are packaged as containers. These components are: NGIX server as a high-performance webserver and reverse proxy, CouchDB a document-oriented

NoSQL Database, Kafka a distributed, high-performing publish/subscribe messaging system and Openwhisk's custom components Controller, Load Balancer, Invoker, and action execution container.

The above picture describes precisely the entire procedure for an action execution which will be discussed in detail:

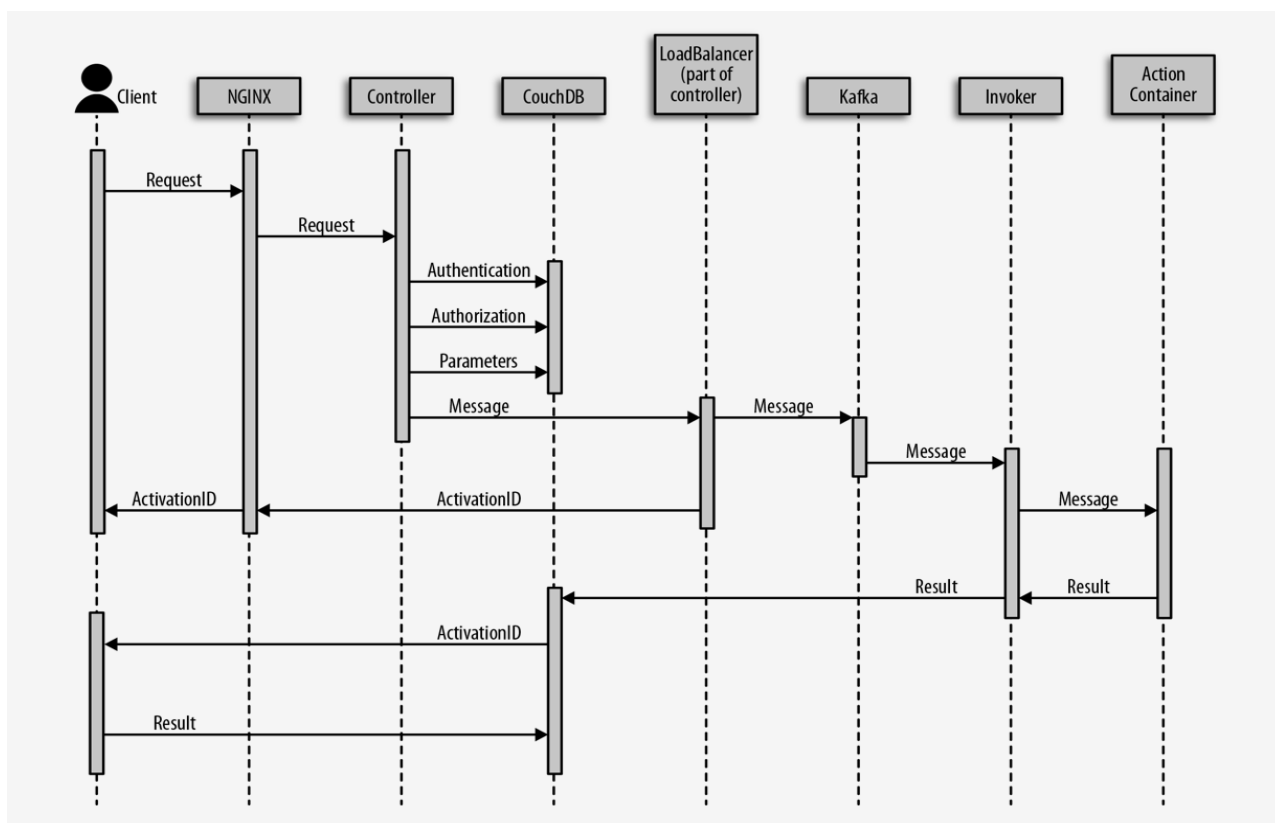


Figure 7 Openwhisk's entire procedure for an action execution<sup>15</sup>

Each of the components mentioned above completes a task in order to fulfill the process for the action execution. It is essential to mention that the user can make his request (bypassing his data parameters) into an action -usually- via Web when the action has been performed as a web action, via Openwhisk's custom CLI, through a REST API and, when a trigger is fired (with a rule) and initializes the action which is associated with. The first responder of the request is the NGINX server; every user's request is translated into an HTTP request and hits the webserver NGINX. The main target of NGINX is to perform support for the HTTPS secure web protocol by enriching the request with all the certificates required for secure processing. After this task is completed, the request is passing to the next component, which is Controller; the Controller checks if the request is capable of being executed; an action is executed if only if the request is authenticated. Once the request is authorized, it must be enriched with additional parameters as part of the action's configuration. This step is performed along

<sup>15</sup> Learning Apache OpenWhisk by Michele Sciabarrà (O'Reilly). Copyright 2019 Michele Sciabarrà, 978-1-492-04616-5, p. 38.

with the support of Couch DB, which stores the parameters and metadata for the specific request. After this step is completed, the request is passing to the load balancer, which checks if there are available instances to execute the action; in case that there are not available actions, then create new ones.

Afterward, the request must be pass to the invoker in order to initialize the action to be executed, but sometimes, invokers may be busy by executing another action, or maybe an invoker is not responding or the entire system having a problem and restarting. For that reason, Openwhisk uses Kafka to control and adjust the invocations. Kafka is a high performing publish and subscribe messaging system that the requests can be stored until they are ready to be executed. The initial HTTP request transformed into a Kafka message addressed for the action execution. Each Kafka message (each request) is sent to the invoker in order to take an ActivationID which is unique and corresponds to the specific request, then the invoker passes the request to the action container -which supports specific programming language runtime-, completes the action execution and stores the result to the CouchDB plus it returns it to the user. The action container, in reality, is a docker container that is initialized, execute its custom computation in an isolated environment, and it is terminated (stateless status). Openwhisk provides both asynchronous and synchronous processing. In the case of the asynchronous procedure, the user can make his request without expecting the answer directly. Openwhisk for each invocation is providing -as we mentioned- an Activation ID which is associated with and stores it in CouchDB along with the result, so the user can retrieve the specific result by checking the specific activation log. From the other side, in the synchronous procedure, the connection stays open until the user gets the result.

The above-mentioned components concerning Openwhisk architecture, are not visible for the user/developer due to the fact that he is entirely concentrated in writing source code to build actions. On the other hand, Openwhisk actions have some constraints and limitations which must be taken in mind when the developer builds his business logic. The below table represents the most important action execution restrictions. All constraints have some significance in terms of time or space (timeout, frequency, memory); some of them are configurable, and others are difficult to be modified. Typical constraints are:

**Table 2 Openwhisk's action constraints<sup>16</sup>**

limit	Detail/explanation	configurable	unit	default value
timeout	a container is not allowed to run longer than N milliseconds	per action	milliseconds	60000
memory	a container is not allowed to allocate more than N MB of memory	per action	MB	256
logs	a container is not allowed to write more than N MB to stdout	per action	MB	10
concurrent	no more than N activations may be submitted per namespace either executing or queued for execution	per namespace	number	100
minuteRate	no more than N activations may be submitted per namespace per minute	per namespace	number	120
codeSize	the maximum size of the action code	configurable, limit per action	MB	48
parameters	the maximum size of the parameters that can be attached	not configurable, limit per action/package/trigger	MB	1
result	the maximum size of the action result	not configurable, limit per action	MB	1

According to the above table, an action must complete its process in a range of sixty seconds. Although the execution time can be modified by the developer of the action using annotations. The same holds for the memory size. An important action limitation is concerning the code size; the developer must provide his source code without exceeding the size of 48 MB. An action can be a zip file that includes the primary function (including its source code), which will be executed from the Openwhisk action container along with other files which are essential for the execution, but the developer has to remember that the size of the entire function cannot be up to 48 MB. As it concerns actions implemented in Python, popular libraries like requests and BeautifulSoup are available as global packages. Additional packages can be imported using virtualenv during invocations. Python external packages can be used by building virtual environments. Developers install the packages locally and include the virtual folder in the archive for deployment. Openwhisk also gives the ability

<sup>16</sup> <https://github.com/apache/openwhisk/blob/master/docs/reference.md>

to execute one hundred and twenty action invocations per minute, but from the other side, limits the size of the given input parameters which it must not exceed 1 MB. As it concerns the other constraints, they cannot be configured by the user, but the system administrator of Openwhisk can configure them.

### 2.5.3.3 Openwhisk Installation and Configuration

In this subchapter, we are explaining the entire procedure for the installation and deployment of the Openwhisk platform in our local machine to test our custom services.

Openwhisk community offers - via its official repository in GitHub<sup>17</sup> - various ways to install the entire platform in a local machine. The users can follow the instructions for a native development in order to deploy the platform in their local machine (MacOS, Ubuntu), as a standalone jar file, or with the support of Kubernetes (Deployment in Kubernetes cluster), or with the support of Docker-Compose<sup>18</sup>.

For this thesis, we chose to deploy Openwhisk with the support of Docker-Compose. Before starting to describe the entire procedure, we must refer that we selected to install the VMWARE Fusion (edition 11.5 professional) workstation, a tool for virtualization, and ran an Ubuntu (16.04 LTS Xenial) image with 4GB RAM. The entire installation was held into Ubuntu OS. In all over the procedure of the installation, we captured snapshots (copies) in each step in order to avoid failure or possible system errors. VMWARE allow customizing settings through a friendly web interface. For that reason, we selected (through UI) to adjust networking settings in NAT mode because we wanted our virtual machine to share the external network with our OS.

Openwhisk (for the installation in a ubuntu machine) requires various tools before building and deploying it; these are a) the programming language JAVA<sup>19</sup> (version 8) along with JDK, b) Docker<sup>20</sup> (edition 1.13 +), and c) Docker Compose<sup>21</sup> (edition 1.6 +). After the installation of the tools as mentioned above, we cloned the target repository into a new directory of our OS including all Openwhisk core components (NGIX, controller, invoker, Kafka, etc.) as docker containers that are orchestrated by Docker Compose. Docker Compose is a mechanism for establishing and managing multi-container applications; actually, acts like an orchestrator for all the components that synthesize an application, it builds a stack of applications to run a complete service by using a YML configuration file that is broken into sections, each section represents a single container which, when combined with the other containers, create the service.

Also, the directory folder includes a Makefile, which defines a set of tasks to be executed. The Makefile includes a bash script which the OS must execute in order to install all the appropriate

---

<sup>17</sup> <https://github.com/apache/openwhisk>

<sup>18</sup> <https://github.com/apache/openwhisk-devtools/blob/master/docker-compose/README.md>

<sup>19</sup> <https://www.java.com/en/download/>

<sup>20</sup> <https://docs.docker.com/v17.09/engine/installation/linux/docker-ce/ubuntu/>

<sup>21</sup> <https://docs.docker.com/compose/install/>



Openwhisk components. Serverless development is performed initially at the terminal using a CLI. The analytical steps for Apache Openwhisk's installation is described in [Appendix A](#). After the installation is completed, we had to download and compile Openwhisk's CLI in order to interact with the entire serverless platform and its components. The executable file for the CLI's installation is in a subdirectory of the folder mentioned above, which must be established as a global variable in the OS. Openwhisk is initialized with the command `wsk` as it follows:

```

Terminal
devops@ubuntu-openwhisk: ~
devops@ubuntu-openwhisk:~$ wsk
  _____
 /_ _ _ _ _ \
|  _ \| | | | | |
| |_) | |_| |
|  __/|  _  |
|_|   |_| |_|

Usage:
wsk [command]

Available Commands:
action      work with actions
activation  work with activations
package     work with packages
rule        work with rules
trigger     work with triggers
sdk         work with the sdk
property    work with whisk properties
namespace   work with namespaces
list        list entities in the current namespace
api         work with APIs
project     The OpenWhisk Project Management Tool

Flags:
--apibhost HOST           whisk API HOST
--apiversion VERSION      whisk API VERSION
-u, --auth KEY            authorization KEY
--cert string             client cert
-d, --debug               debug level output
-h, --help                help for wsk
-i, --insecure            bypass certificate checking
--key string              client key
-v, --verbose             verbose output

Use "wsk [command] --help" for more information about a command.
devops@ubuntu-openwhisk:~$

```

**Figure 8 Openwhisk installation and configuration in a local machine**

As we observe from the above figure, currently Openwhisk does not provide a graphical interface; all the interactions implemented via its custom Unix-Based CLI; therefore, the developer/user has the ability to manipulate all Openwhisk components by using native commands built by the Openwhisk creators. As we already mentioned, Openwhisk supports various programming languages as runtimes, but for this thesis, we selected the programming language Python (version 3.6.1) to build our custom actions. The supported Openwhisk runtimes are provided through the official Docker repository<sup>22</sup> that includes pre-build images for Openwhisk. In order to integrate the python runtime into Openwhisk, we had to pull the specific docker container, including the python environment, to execute actions.

<sup>22</sup> <https://hub.docker.com/r/openwhisk/python3action>

### 2.5.3.4 Openwhisk CLI and native commands

Generally, the developer can interact with the user's visible components (actions, triggers, API, etc.) through the abbreviation **wsk**. The **wsk** command is composed of many commands, each with many subcommands. The general format is this:

**wsk <COMMAND> <SUBCOMMAND> <PARAMETERS> <FLAGS>**<sup>23</sup>

There are various examples for implementing actions in Openwhisk serverless environment are described in the official Openwhisk repository in GitHub<sup>24</sup>, but we will present the general structure in order to create and invoke an action or a sequence of functions in python and trace the activation logs.

**Table 3 Creation-Update-Invocation-Removal commands for an action in Openwhisk**

a/a	abbreviation	flag	command	subcommand	parameter	source	Global flag
1	wsk	-i	action	create	action name	Written code-file	--kind python3, --web true
2	wsk	-i	action	update	action name	--	--web true
3	wsk	-i	action	delete	action name	--	--
4	wsk	-i, -r	action	invoke	action name	--	--param name

**Table 4 Retrieval of activation logs for each action/or sequence of actions**

a/a	abbreviation	flag	command	subcommand	parameter	Global flag
1	wsk	-i	activation	list	--	--
2	wsk	-i	activation	get	activationID	--logs

**Table 5 REST API creation command to invoke an action/sequence of actions**

abbreviation	flag	command	subcommand	parameter	subcommand	parameter	Additional flags
wsk	-i	api	create	/api name	get	action/sequence name	--response-type json

<sup>23</sup> <PARAMETERS> and <FLAGS> are different for each <SUBCOMMAND>, and for each <COMMAND> there are various subcommands

<sup>24</sup> <https://github.com/apache/openwhisk/blob/master/docs/samples.md>

Table 6 Creation-Update-Invocation-Removal commands for a sequence in Openwhisk

a/a	abbreviation	flag	command	subcommand	parameter	source	Global flag
1	wsk	-i	actionA	create/update	action name	Written code-file	--web true
2	wsk	-i	actionB	create/update	action name	Written code-file	--web true
3	wsk	-i	actionC	create/update	action name	Written code-file	--web true
4	wsk	-i	action	create/update	action (sequence) name	--	--sequence actionA, actionB, actionC, --web true
5	wsk	-i, -r	action	invoke	action (sequence) name	--	--param name
6	wsk	-i	action	delete	action (sequence) name	--	--

### 2.5.3.5 Additional information

According to the below tables, we provide some guidance coming from the Openwhisk inventors to describe some of the previous components. If the developer uses a local OpenWhisk deployment with a self-signed SSL certificate, he can use the `--insecure (-i)` flag to bypass certificate validation. The `-r` flag represents the direct appearance of the result/response according to the request. The global flag `--web true`, makes the action as web action that is accessible from any user without requesting an authentication. Furthermore, each invocation produces an ActivationID, which practically is a number and it is stored in CouchDB, therefore in order to access the logs from the specific action invocation, we follow the second command from Table 5. After the OpenWhisk environment is enabled, the developer can use OpenWhisk with his web apps or mobile apps with REST API calls. All the abilities in the Openwhisk's system are accessible via a REST API. There are collection and entity endpoints for actions, triggers, rules, activations, and namespaces.

- [https://\\$APIHOST/api/v1/namespaces](https://$APIHOST/api/v1/namespaces)
- [https://\\$APIHOST/api/v1/namespaces/{namespace}/actions](https://$APIHOST/api/v1/namespaces/{namespace}/actions)
- [https://\\$APIHOST/api/v1/namespaces/{namespace}/triggers](https://$APIHOST/api/v1/namespaces/{namespace}/triggers)
- [https://\\$APIHOST/api/v1/namespaces/{namespace}/rules](https://$APIHOST/api/v1/namespaces/{namespace}/rules)
- [https://\\$APIHOST/api/v1/namespaces/{namespace}/packages](https://$APIHOST/api/v1/namespaces/{namespace}/packages)
- [https://\\$APIHOST/api/v1/namespaces/{namespace}/activations](https://$APIHOST/api/v1/namespaces/{namespace}/activations)

The **\$APIHOST** is the Openwhisk API hostname (in our use case is 192.168.111.128). The namespace and activation endpoints support only **GET** requests. The actions, triggers, rules, and packages endpoints support **GET**, **PUT**, and **DELETE** requests. The endpoints of actions, triggers, and rules also support **POST** requests, which are used to invoke actions and triggers and enable or disable rules.

### 2.5.3.6 Serverless vulnerabilities

Whilst, the serverless computing has been evolved rapidly offering from developer's perspective: zero system administration, easier operational management, encouraging of adoption of Nanoservices, Microservices, SOA Principles, faster set up, scalability without worry about the number of concurrent requests, monitoring out-of-the-box, innovation and from business perspective: cost-based specific criteria, process agility, lower cost for hiring backend infrastructure engineers and reduced operational costs, still has some sensitive spots. Serverless reduces the complete control, so it is more difficult to monitor the entire procedure, acquires entirely trust to cloud providers (cloud vendors that control the infrastructure), is more susceptible to security risks and disaster recovery risks, and unpredictable as it concerns the cost due to the fact that the requests cannot be predefined. Nevertheless, all the mentioned situations can be mitigated with the benefits that the serverless computing provides.

# Chapter 3

---

## A Proposed ML-FaaS Business-Intelligence model

---

### 3.1 Overview

In a previous chapter, we presented the entire ordinary procedure to approach a problem (such as classification, regression) with the implementation of methods and techniques related to Data Analytics and Machine Learning; also, we introduced the Serverless Computing model along with Function as a Service concept (FaaS) as one of the most successful innovations that has evolved into the cloud promoting the automation. As we already referred in a preceding chapter, building an entire application demands a full-stack development knowledge, which in most of the cases acquires an entire group of hard-working, skilled employees in order to be implemented. What if we want to approximate a classification problem, create a model, and make it visible to third-users as an entire application avoiding massive effort? How can a developer elaborate his business logic and strategy in combination with cloud-offered-services without the support of backend infrastructure engineers to construct his application as a meaningful power service reducing the overall cost at the same time? Our proposed approach in this thesis answers those questions with the construction of a Machine Learning Model with the support of Data Analysis tools and visualization techniques in order to approach a specific use-case classification problem. The critical part of this thesis is that we split the entire logic into functional pieces that have been integrated into a serverless ecosystem, becoming an entire application visible from users and available for test purposes. In the next sections of this chapter, we will introduce our proposed approach to solve a classification problem, all the procedure steps -in detail - and the final deployment of the proposed logic as series of Openwhisk actions that synthesize the entire application, which is accessed by ordinary users.

## 3.2 The proposed approach

Our proposed approach is an experiment that consolidates data analytics skills and machine learning, along with the Apache Openwhisk Serverless platform. In particular, we selected to examine a dataset with labeled fraud and authorized e-commerce transactions due to the fact that online card fraud reported cases to constitute a crucial problem that has dramatically occupied the Financial Sector, Law Enforcement Agencies, and consumer society. The target was to build a machine learning classifier that can sharply recognize whether an e-commerce transaction is fraudulent or legitimate. For that reason we selected a dataset with historical e-commerce transactional data in order to train our classifier giving, as a result, the opportunity to an ordinary user to provide as input new unseen data concerning a unique transaction and finally predicting if his input data constitutes a fraud or an authorized transaction; The above-mentioned business logic has been transmitted into the selected Openwhisk serverless platform as an entire application which consists of a pipeline of three -3- microservices (functions) which each of these executes a specific task concerning the new unseen data that the user-provided as input. The related work was implemented in two phases the Offline phase and the Online Phase.

The Offline phase was held in our local machine and concluded the entire data analytics and machine learning procedure to manipulate the selected dataset started from the problem definition, the data collection and its explanation, an EDA Analysis in order to discover useful patterns and insights, the crucial part of preprocessing concerning imputation of missing values, the dropping of useless features, dimensionality reduction with PCA technique, implementation of feature engineering process, label encoding concerning categorical data, scaling and normalization process, the enrichment of dataset to improve the performance due to the fact that we dealt with imbalanced labeled data (the vast majority of transactions was not fraudulent), the selection of various machine learning algorithms that were fitted to the dataset, the training procedure with the support Stratify K-fold cross-validation model validation technique, and the final machine learning model selection according to best performance of Recall, Area Under Curve -AUC- and Confusion Matrix metrics. As it concerns the selection of the most suitable machine learning algorithm, we tested models like Logistic Regression, Decision Trees, Random Forest, and Artificial Neural Networks, the last one implemented with Keras framework in four -4- different cases. Especially we tested the chosen algorithms with four different techniques concerning the data manipulation: a) Raw dataset without any enrichment technique, b) by giving weight to minority class samples (fraudulent transactions), c) with the implementation of a simple oversampling technique and d) with the implementation of Synthetic Minority Oversampling technique -SMOTE- in combination with TOMER-LINK undersampling technique. At the end of the Offline phase, we saved the best pre-trained model in the local disk in order to use it Online phase.

The Online phase entails the creation and development of the source code (in python programming language), corresponding to the building of functions (connected actions) inside Openwhisk. More particular, we decided to build three connected Openwhisk actions (a sequence of

microservices) with python runtime. As we said in the previous chapter, each Openwhisk action accepts as input parameters in a python dictionary structure (a JSON format) and returns a result likewise the input's structure. When we deal with a sequence of actions, the first input is passing into the first action, an execution is implemented, and an output is produced, which will be the input for the second Openwhisk function and so forth until the last action give the final result. All of the three constructed functions concerning new unseen data; The first function receives the parameters (JSON format), undertakes to transform their structure in order to make data analysis fast and easy, checks if missing values are appeared and does the appropriate imputation according to the strategy that we followed in Offline phase, also, adds external features according to the feature engineering procedure that we followed similarly in Offline phase and forms capitalization to categorical data in order to maintain a consistent format for the given data. The second action receives as input the transformed data from the computations of the first function; its main task is to perform label encoding to these values that constitute categorical variables and do the normalization according to the normalization that we applied in Offline phase. The modified parameters from the second function pass as input in the third and final function. In the third action, we used the pre-trained model from the Offline phase, which becomes part of the mentioned action, and it is used to make the final prediction if the given data (parameters) constitutes a fraudulent or authorized transaction. The final action's output produces a message that refers to if the transaction is fraud or not.

As it concerns the construction of functions, taking mind the constraints and limitations that we referred in the previous chapter; OpenWhisk supports creating actions from archive files containing source files and project dependencies; but in our system, we used external docker containers mechanisms, which includes the appropriate machine learning libraries that we needed to complete each action's task, due to the fact that most of the third-party python packages which help to improve data performance are exceeding the limitation of the function's source code (48MB).

As we already know, Python is a popular language for machine learning and data science; public libraries like pandas, scikit-learn, and NumPy provide all the necessary tools to manipulate the given data in order to fulfill our target. Although, popular libraries like flask, requests, and Beautifulsoup are concluded in the Openwhisk supported runtimes as global packages. Any additional packages can also be imported using the construction of a virtual environment, which will be included along with the source code for the specific action during invocations. Nevertheless, Machine Learning libraries often use numerous shared libraries and compile native dependencies for performance; this can lead to hundreds of megabytes of dependencies. Setting up a new virtualenv folder and installing, for example, Pandas library leads to an environment with nearly 100MB of dependencies. Bundling these libraries within an archive file will not be possible due to the file size limit. We can overcome this limit by using a custom runtime image. The runtime will pre-install additional libraries during the build process and make them available during invocations. OpenWhisk uses Docker for runtime containers. Source files for the images are available through the official repository GitHub. Using custom runtimes with private source files is a fantastic feature of OpenWhisk because it enables developers to run larger applications on the platform but also enables lots of other use cases. Almost

any runtime, library, or tool can now be used from the platform. In our related work, we used and configured prefixed docker images, including libraries with specific editions such as Pandas, Scikit-learn, Keras, and NumPy, in order to create and run our custom Openwhisk actions.

When new data are inserted as input (because Openwhisk is an event-driven serverless platform), an HTTP request is created and is passing to all of the Openwhisk’s core components (NGIX Server, Controller, CouchDB, LoadBalancer, Action Container) and produce the response. The user practically, in Openwhisk language, makes an HTTP request through a REST API endpoint, which is passing into the pipeline of functions and follows the procedure mentioned above to fulfill the request and produce a result.

According to the information mentioned above, we introduce the graphical representation of the entire proposed architecture that constitutes the construction of an alert system for card fraud detection in a Serverless ecosystem:

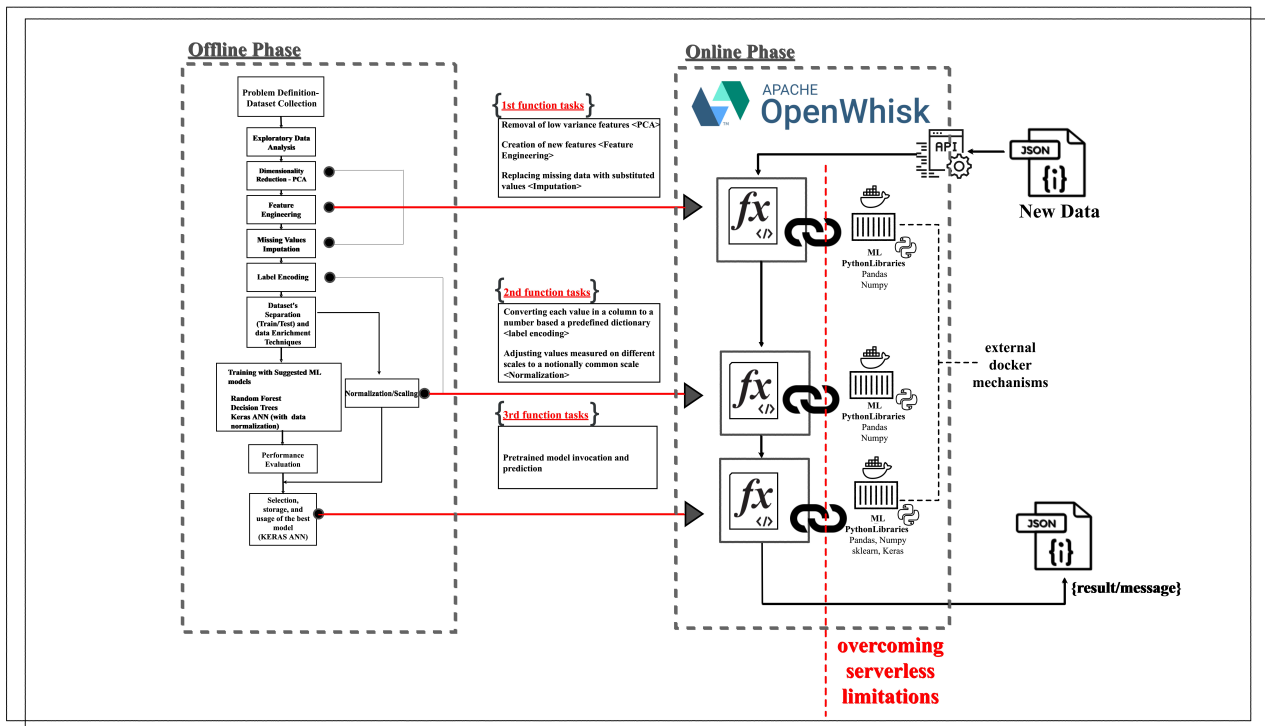


Figure 9 Proposed ML-FaaS Approach

In the next sections of this chapter, we will present each phase separately; first, the Offline Phase, which includes all the above steps starting from Problem Definition, continue with the Dataset Collection, Data Analysis Strategy, the training phase with the selected machine learning algorithms) and afterward in the Online phase (deployment), we will present the entire business logic for the construction of the functions inside Openwhisk.



## 3.3 Offline Phase

### 3.3.1 Definition of the problem

One of the most indispensable innovations that the Internet provides is automated online e-commerce systems, which enables to consumers to interact directly with vendors in order to make online transactions globally, ensuring speed, reliability, and a high level of security. On the other side, the Internet favors the criminals who invent day by day new methods and techniques in order to act illegally without performing physical appearance. More precisely, we are concentrated on card Thefts who are using various methods to capture data, including card skimming at ATMs or ticket machines and phishing. People are often unsuspecting that their credential card data has been stolen until it is too late. These data can be used to create fake-clone cards or used subsequently for Card-Not-Present fraud (CNP). Mainly, Fraudsters use this information to purchase goods or services in the name of victims or obtain unauthorized funds from the victim's accounts. Compromised card data may also put up for sale on Darknet markets. In numerous cases, the data stolen in one country is used elsewhere, making it harder to trace.

According to the Official Central's Bank Website (Eurosystem)<sup>25</sup>, in 2016 the total value of CNP fraud increased by 2.1% compared with the previous year, reaching € 1.32 billion furthermore, CNP was the most common type of fraud in 2016 by accounting for 76% of the total value of frauds.

Due to the Official Federal Trade Commission Website<sup>26</sup>, people reported losing \$1.48 billion to fraud last year – an increase of 38% over 2017. The most reports in 2018 were about imposter scams, debt collection, and identity theft. Additionally, in 2018 the 43% of people who reported fraud is in the age of 20s, while only 15% of people in their 70s did.

Data analysts, according to their skills, knowledge, and experience, contribute to combatting card fraud crime by producing models that can detect fraud transactions with a high level of success. Machine learning can be meaningful in order to extract valuable information and help face this grown-up crime activity due to the fact that it uses algorithms, statistical, and mathematical models that computer systems use to accomplish specific tasks without using explicit instructions, relying on patterns and inference instead. Machine learning models by using historical data, are capable of making predictions or decisions without being explicitly programmed to perform the task.

According to the sections, as mentioned above, we decided to introduce our solution in the above problem using extensive Data Analytics in order to gather all the necessary information we need, extract insights, implement visualization techniques, discover useful patterns in combination with machine learning algorithms in order to recognize fraudulent transactions with the highest sensitivity.

---

<sup>25</sup> <https://www.ecb.europa.eu/pub/cardfraud/html/ecb.cardfraudreport201809.en.html#toc5>

<sup>26</sup> <https://www.consumer.ftc.gov/blog/2019/02/top-frauds-2018>

### 3.3.2 Dataset Collection

Thus, the problem and business goal are defined; the typical data analysis logic starts with the determination of historical data for studying in order to build our business model. After that, the mentioned data feed the chosen machine learning algorithm in order to predict the result.

We collected our appropriate dataset concerning fraud e-commerce transactions from the official data repository Kaggle -purchased by Google in 2017- which is an online community for data scientists providing various projects for all kinds of analysis. The chosen dataset was published from Kaggle for competition due to the fact that the ability to identify fraud e-commerce transactions is a big challenge for the data science community, and it can semantically improve the security of consumers banking activity. Vesta Corporation, which is a world's leading payment service company, provided the dataset for this competition. Vesta Corporation is the forerunner in guaranteed e-commerce payment solutions. Founded in 1995, Vesta pioneered the process of fully guaranteed card-not-present (CNP) payment transactions for the telecommunications industry. Since then, Vesta has firmly expanded data science and machine learning capabilities across the globe and solidified its position as the leader in guaranteed e-commerce payments.

### 3.3.3 Dataset explanation

The chosen dataset was downloaded through Kaggle's official repository<sup>27</sup>, it is structured with a comma-separated values format and consists of two parts. The data are broken into two files identity and transaction, which are joined by the column TransactionID. A piece of additional information is that not all transactions have corresponding identity information. The size of the aforementioned files are 26,5 MB and 683,4 MB, respectively. Due to the fact that consumers transactions constitute sensitive personal data, the competition hosts, according to the Law, provided them fully anonymized. Dataset's features are masked and are corresponding to a pairwise dictionary, which was not provided for privacy protection and contract agreement. The identify dataset consists of forty (40) columns and one hundred forty-four thousand two hundred thirty-three (144233) rows. The transaction dataset consists of three hundred ninety-three (393) unique features and five hundred ninety thousand five hundred forty (590540) rows. Each row corresponds to one unique consumer's transaction. Based on competition instructions, both of datasets have numerical and categorical features. Specifically, as it concerns the identity dataset, the first thirty-eight (38) features are called ids and follow a numerical order from id\_01 up to id\_38 and the rest two features are named 'DeviceType' and 'DeviceInfo'. According to instructions, the competition hosts defined that the first eleven id features correspond to numerical variables, and the features from id\_12 up to id\_38 and 'DeviceType' and 'DeviceInfo' form categorical data by taking a limited, and a fixed number of possible values. Also, the variables in the mentioned dataset concern consumers identity information, network connection information (i.e., IP,

---

<sup>27</sup> <https://www.kaggle.com/c/ieee-fraud-detection/data>

ISP, Proxy), and digital signature (UA/browser/Operation System/version) associated with transactions.

From the other hand, the transaction dataset has a plenty of both numerical and categorical features which according to competition hosts are described below along with their definitions:

**Table 7 Main Features of transaction dataset**

• <b>TransactionDT:</b> timedelta from a given reference datetime (not an actual timestamp).
• <b>TransactionAMT:</b> transaction payment amount in USD.
• <b>ProductCD:</b> product code, the product for each transaction.
• <b>card1 - card6:</b> payment card information, such as card type, card category, issue bank, country, etc.
• <b>addr1-addr2:</b> address.
• <b>dist1-dist2:</b> distance.
• <b>P_ and (R_) emaildomain:</b> purchaser and recipient email domain.
• <b>C1-C14:</b> counting, such as how many addresses are found to be associated with the payment card, etc. The actual meaning is masked.
• <b>D1-D15:</b> timedelta, such as days between previous transaction, etc.
• <b>M1-M9:</b> match, such as names on card and address, etc.
• <b>Vxxx:</b> Vesta engineered rich features, including ranking, counting, and other entity relations.

According to the above table, the features ProductCD, card1 up to card6, addr1, addr2, P\_emaildomain, R\_emaildomain and M1 up to M9 corresponds to categorical variables.

Below, we collocate subsamples from the aforementioned files to provide a short view of them.

Table 8 Subsample of the file transaction.csv

TransactionID	is Fraud	TransactionDT	TransactionAmt	ProductCD	card1	card2	P_emaildomain	card4	M4	M5	addr1	C3	C4	dist2	V199	V200	M5	M6
2987000	0	86400	68.5	W	13926		NA	discover	M2	F	315.0	0.0	0.0	NA	NA	NA	F	T
2987001	0	86401	29.0	W	2755	404.0	gmail.com	mastercard	M0	T	325.0	0.0	0.0	NA	NA	NA	T	T
2987002	0	86469	59.0	W	4663	490.0	outlook.com	visa	M0	F	330.0	0.0	0.0	NA	NA	NA	F	F
2987003	0	86499	50.0	W	18132	567.0	yahoo.com	mastercard	M0	T	476.0	0.0	0.0	NA	NA	NA	T	F
2987004	0	86506	50.0	H	4497	514.0	gmail.com	mastercard		NA	420.0	0.0	0.0	NA	1.0	1.0	NA	NA
2987005	0	86510	49.0	W	5937	555.0	gmail.com	visa	M1	F	272.0	0.0	0.0	NA	NA	NA	F	T
2987006	0	86522	159.0	W	12308	360.0	yahoo.com	visa	M0	F	126.0	0.0	0.0	NA	NA	NA	F	F
2987007	0	86529	422.5	W	12695	490.0	mail.com	visa	M0	F	325.0	0.0	0.0	NA	NA	NA	F	F
2987008	0	86535	15.0	H	2803	100.0	anonymous.com	visa	NA	NA	337.0	0.0	0.0	NA	1.0	1.0	NA	NA
2987009	0	86536	117.0	W	17399	111.0	yahoo.com	mastercard	M0	T	204.0	0.0	0.0	NA	NA	NA	T	T
2987010	0	86549	75.887	C	16496	352.0	gmail.com	mastercard	M0	NA	NA	0.0	1.0	NA	1.0	1.0	NA	NA
2987011	0	86555	16.495	C	4461	375.0	hotmail.com	mastercard	M0	NA	NA	0.0	1.0	30.0	1.0	1.0	NA	NA
2987012	0	86564	50.0	W	3786	418.0	verizon.net	visa	M1	F	204.0	0.0	0.0	NA	NA	NA	F	F
2987013	0	86585	40.0	W	12866	303.0	aol.com	visa	NA	NA	330.0	0.0	0.0	NA	NA	NA	NA	F
2987014	0	86596	10.5	W	11839	490.0	yahoo.com	visa	M0	F	226.0	0.0	0.0	NA	NA	NA	F	F
2987015	0	86618	57.95	W	7055	555.0	NA	visa	NA	NA	315.0	0.0	0.0	NA	NA	NA	NA	NA
2987016	0	86620	30.0	H	1790	555.0	aol.com	visa	NA	NA	170.0	0.0	0.0	NA	1.0	1.0	NA	NA
2987017	0	86668	100.0	H	11492	111.0	yahoo.com	mastercard	NA	NA	204.0	0.0	0.0	NA	NA	1.0	NA	NA
2987018	0	86725	47.95	W	4663	490.0	gmail.com	visa	NA	NA	184.0	0.0	0.0	NA	NA	NA	NA	F
2987019	0	86730	186.0	W	7005	111.0	gmail.com	visa	M1	F	264.0	0.0	0.0	NA	NA	NA	F	F
2987020	0	86761	39.0	W	7875	314.0	gmail.com	mastercard	M0	F	299.0	0.0	0.0	NA	NA	NA	F	F
2987021	0	86769	159.95	W	11401	543.0	gmail.com	mastercard	NA	NA	204.0	0.0	0.0	NA	NA	NA	NA	F
2987022	0	86786	50.0	H	1724	583.0	gmail.com	visa	NA	NA	299.0	0.0	0.0	NA	NA	NA	NA	NA
2987023	0	86808	107.95	W	2392	360.0	gmail.com	mastercard	NA	NA	126.0	0.0	0.0	NA	NA	NA	NA	F

Table 9 Subsample of the file identity.csv

TransactionID	id_01	id_02	id_03	id_04	id_05	id_06	id_07	id_08	DeviceType	DeviceInfo	id_11	id_12	id_13	id_14	id_15	id_31	id_32	id_33
2987004	0.0	70787.0	NA	NA	NA	NA	NA	NA	mobile	SAMSUNG SM-G892A Build/NRD90M	100.0	NotFound	NA	-480.0	New	Samsung browser 6.2	32.0	2220x1080
2987008	-5.0	98945.0	NA	NA	0.0	-5.0	NA	NA	mobile	iOS Device	100.0	NotFound	49.0	-300.0	New	mobile safari 11.0	32.0	1334x750
2987010	-5.0	191631.0	0.0	0.0	0.0	0.0	NA	NA	desktop	Windows	100.0	NotFound	52.0	NA	Found	chrome 62.0	NA	NA
2987011	-5.0	221832.0			0.0	-6.0	NA	NA	desktop	NA	100.0	NotFound	52.0	NA	New	chrome 62.0	NA	NA
2987016	0.0	7460.0	0.0	0.0	1.0	0.0	NA	NA	desktop	MacOS	100.0	NotFound	NA	-300.0	Found	chrome 62.0	24.0	1280x800
2987017	-5.0	61141.0	3.0	0.0	3.0	0.0	NA	NA	desktop	Windows	100.0	NotFound	52.0	-300.0	Found	chrome 62.0	24.0	1366x768
2987022	-15.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NotFound	14.0	NA	NA	NA	NA	NA
2987038	0.0	31964.0	0.0	0.0	0.0	-10.0	NA	NA	mobile	NA	100.0	Found		-300.0	Found	chrome 62.0	32.0	1920x1080
2987040	-10.0	116098.0	0.0	0.0	0.0	0.0	NA	NA	desktop	Windows	100.0	NotFound	52.0	NA	Found	chrome 62.0	NA	NA
2987048	-5.0	257037.0	NA	NA	0.0	0.0	NA	NA	desktop	Windows	100.0	NotFound	52.0	NA	New	chrome 62.0	NA	NA
2987049	-5.0	287959.0	NA	NA	1.0	-11.0	NA	NA	desktop	Windows	100.0	NotFound	52.0	NA	New	chrome 62.0	NA	NA
2987057	0.0	88525.0	NA	NA	NA	NA	NA	NA	mobile	SM-G930V Build/NRD90M	100.0	NotFound	NA	-300.0	New	chrome 62.0 for android	32.0	1920x1080
2987066	-5.0	54927.0	0.0	0.0	0.0	-1.0	NA	NA	desktop	Windows	100.0	NotFound	52.0	-360.0	Found	chrome 62.0	24.0	1680x1050
2987069	0.0	69542.0	0.0	0.0	2.0	-4.0	NA	NA	desktop	NA	100.0	Found	NA	-300.0	Found	chrome 62.0	32.0	1920x1080
2987070	0.0	132356.0	NA	NA	1.0	-6.0	NA	NA	mobile	iOS Device	100.0	NotFound	NA	-300.0	New	mobile safari 11.0	32.0	1136x640
2987072	0.0	275611.0	NA	NA	0.0	0.0	NA	NA	mobile	BLADE A602 Build/MRA58K	100.0	NotFound	20.0	NA	New	chrome 62.0 for android	NA	NA
2987074	-5.0	419136.0	NA	NA	0.0	0.0	NA	NA	desktop	Windows	100.0	NotFound	52.0	NA	New	chrome 62.0	NA	NA
2987084	-5.0	436352.0	NA	NA	0.0	0.0	NA	NA	desktop	Windows	100.0	NotFound	52.0	NA	New	chrome 62.0	NA	NA

As a first view of the datasets, and their components, it is clear that we have to deal with raw decoded data that were extracted without any transformation or modification from Vesta Corporation Systems; also we observe that we have to deal with a plenty of missing values; Indeed in a well-designed and controlled study, missing data happens in almost all research. Missing data can reduce the statistical power of a study and can produce biased estimates, heading to invalid outcomes.

### **3.3.4 Fundamental instructions for data manipulation**

Before starting to apply our strategy to approach the datasets mentioned above, we must provide some vital information concerning the dataset's handling, such as the programming language to build the source code, and its third packages. In order to fulfill our target, we used the programming language Python (version 3.7); Python enables developers to roll out programs and get prototypes running, making the development process much faster. It has become the most preferred machine learning tool in the way it allows aspirants to do mathematical computations easily.

Because we had to deal with structured data, we used python specific packages which helped us to handle the datasets mentioned above; More particular, we mainly used the pandas library which is the supreme package for data structures, data analysis, and time-series, giving us the capability to allocate our data in a Dataframe structure, also we used NumPy package for extensive data transformation since it provides fundamental utilities for arrays computation and arguably we used the packages Matplotlib and Seaborn for statistical data visualization.

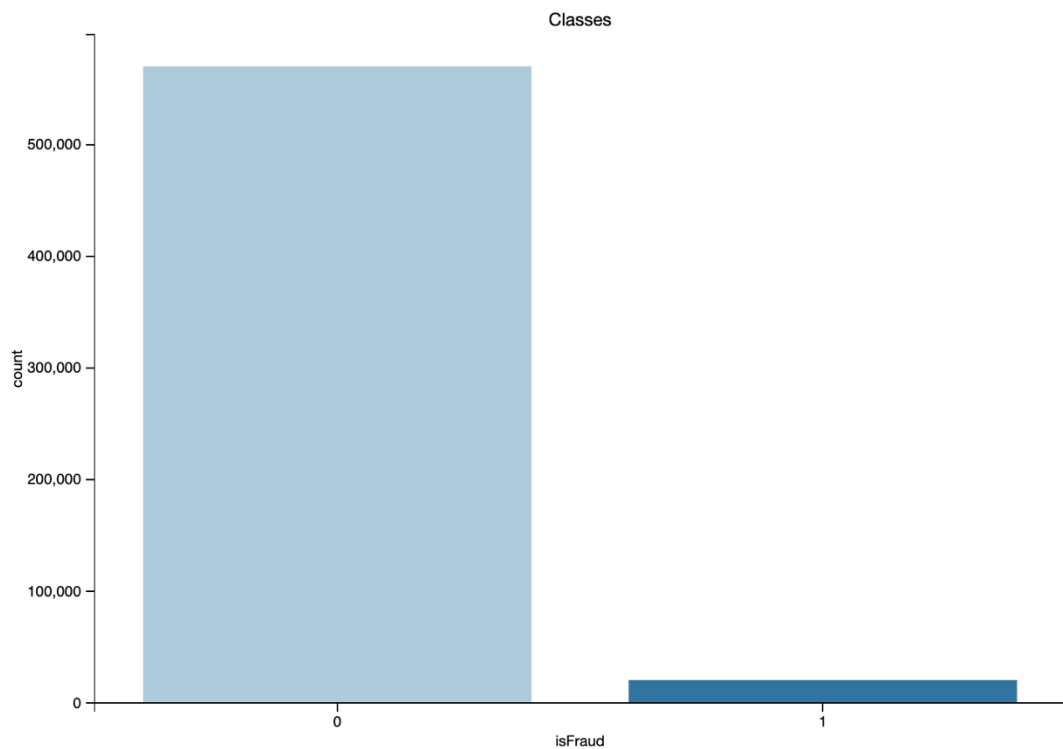
### **3.3.5 Exploratory Data Analysis - EDA**

As we have already introduced to the first chapter of this thesis, before starting to apply various methods and techniques concerning the data preprocessing procedure, the data analysts must follow a typical approach to analyze the dataset, extract useful information from it, discover insights and summarize its characteristics with the support of using visual and quantitative methods without making any assumptions about its contents. The aforementioned method is introducing as Exploratory Data Analysis (EDA). As a benefit, EDA often leads to insights that the business stakeholder or data scientist would not even consider investigating, but that can be hugely enlightening about the business.

Considering our datasets, the first step that we followed was to join them according to their common key, which is the TransactionID column, in order to have a sufficient overview.

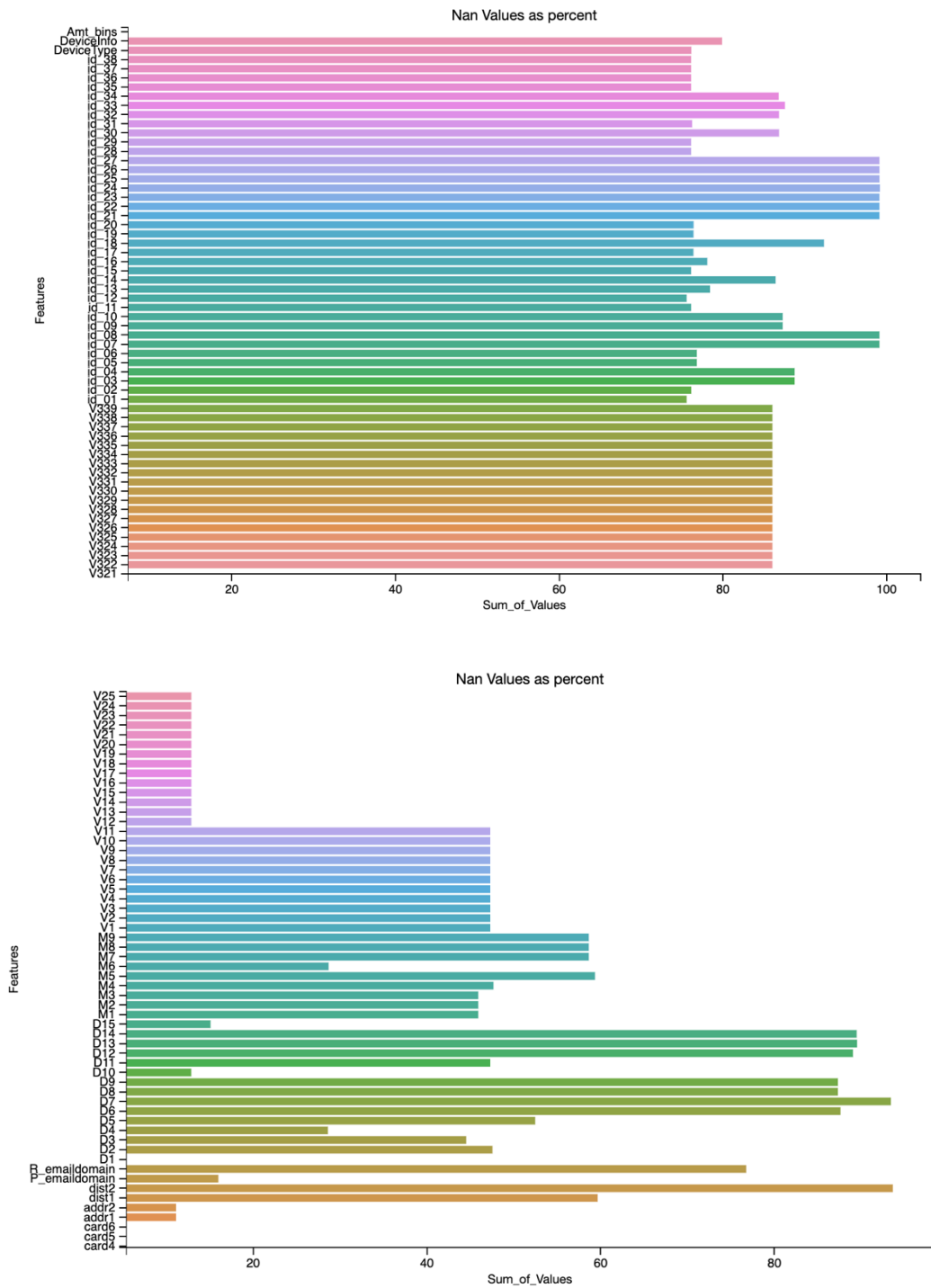
The first step towards the visualization was to investigate the number of observations corresponding to both classes. According to the figure below, we noticed how abnormal is our original dataset; Most of the transactions are non-fraud; in particular, only 3,5% of the entire observations constitute fraud transactions. The valuable information from this visualization was to understand that if we have used the Dataframe as it was along with our predictive models and analysis, we might get

many errors, and our chosen algorithms will possibly overfit since it will "assume" that most transactions are authorized. Thus, our dataset is imbalanced; Imbalanced means that the number of data points available for the different classes is much different.



**Figure 10 Imbalance Dataset – Fraud/Authorized Transactions**

Thereinafter, we observed that our datasets have a high percentage of missing values. Nearly all of the real-world datasets have missing values, and it is not just a minor nuisance; it is a severe problem that we need to taking mind; Missing data — is a hard problem, and, unfortunately, there is no best way to deal with it. In the following figure, we visualized the percentage of missing values of the merged dataset.

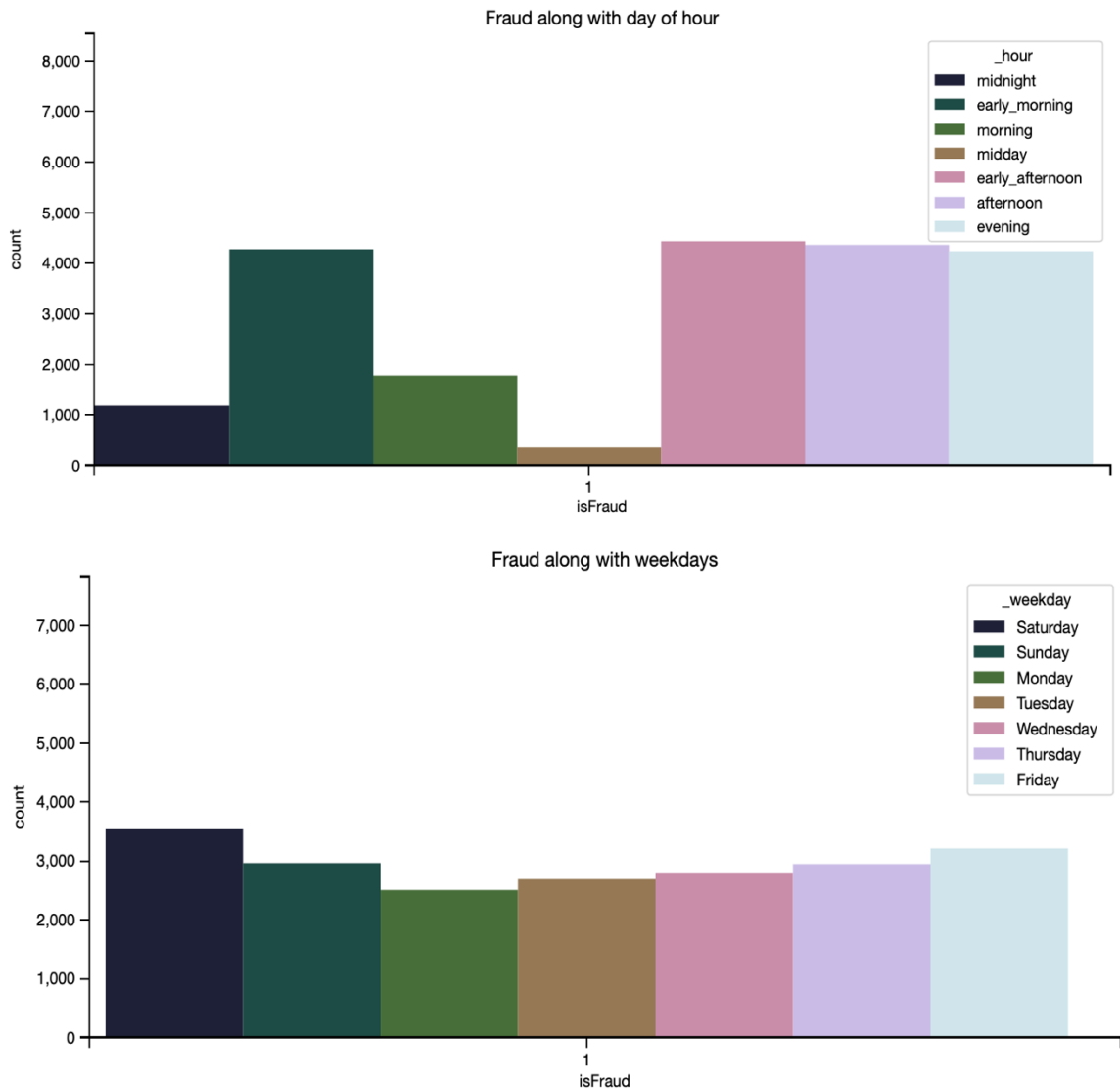


**Figure 11 Percentage of Missing Values in datasets train and identity**

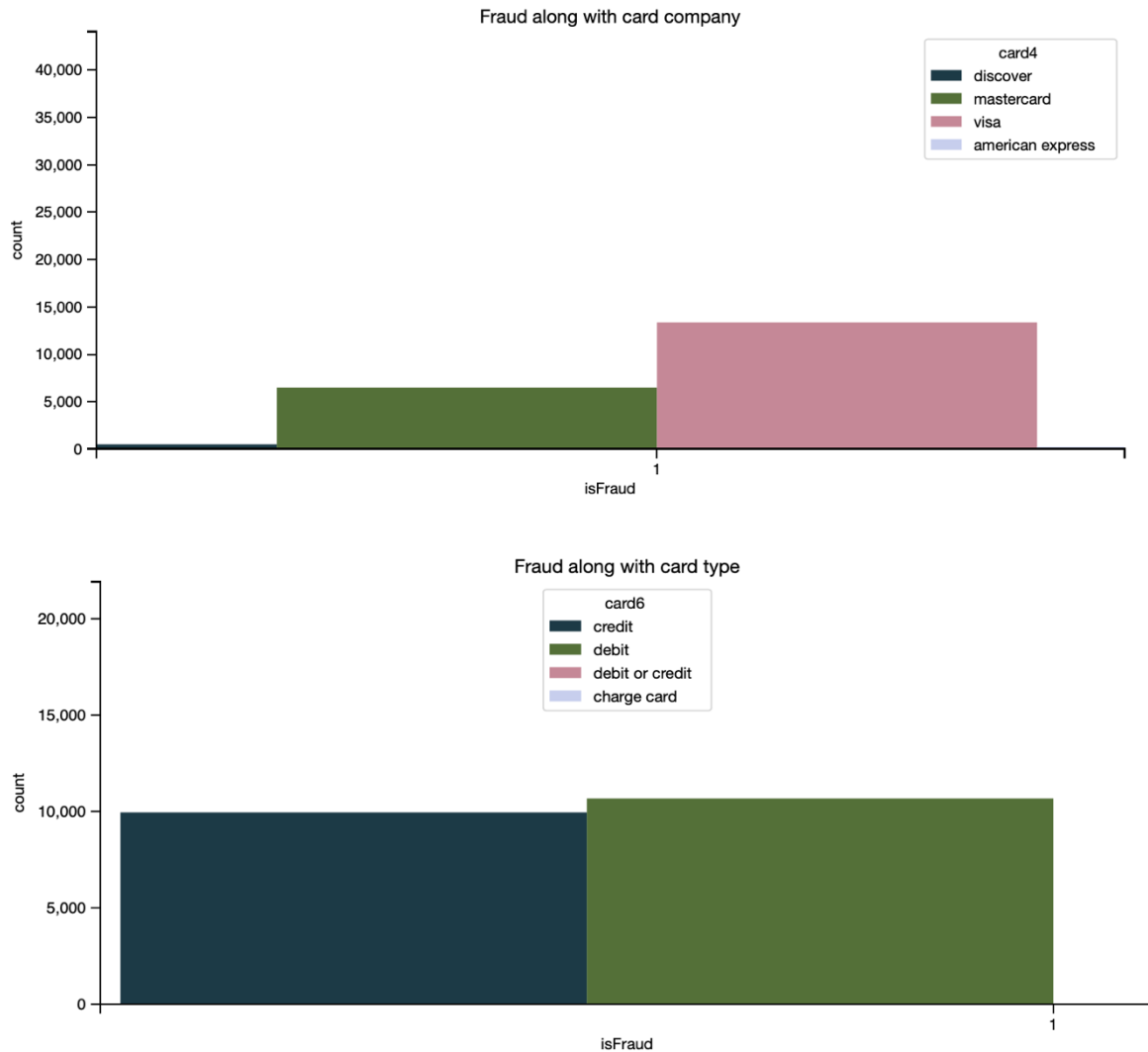
Features such as id21 up to id27, V321 up to V339, dist2 have missing values at their most significant percentage. This information is beneficial due to the fact that in the preparation procedure, we must define the greatest strategy to deal with them in order to have satisfactory results.



Furthermore, we observed that in our dataset, we have the Transaction Time in Timedelta<sup>28</sup> format. We thought of converting it in DateTime format because it will be more understandable, and may we could identify useful insights concerning the timer of the fraudulent transactions. Then, from DateTime format we broke the value into two new features: *Day* and *Time*, and we defined in which day and hour of the week the transaction corresponding. Thereby, we visualized the below components:



<sup>28</sup> A **timedelta** object represents a duration, the difference between two dates or times. class datetime. **timedelta** ([days[, seconds[, microseconds[, milliseconds[, minutes[, hours[, weeks]]]]]]]) All arguments are optional and default to 0

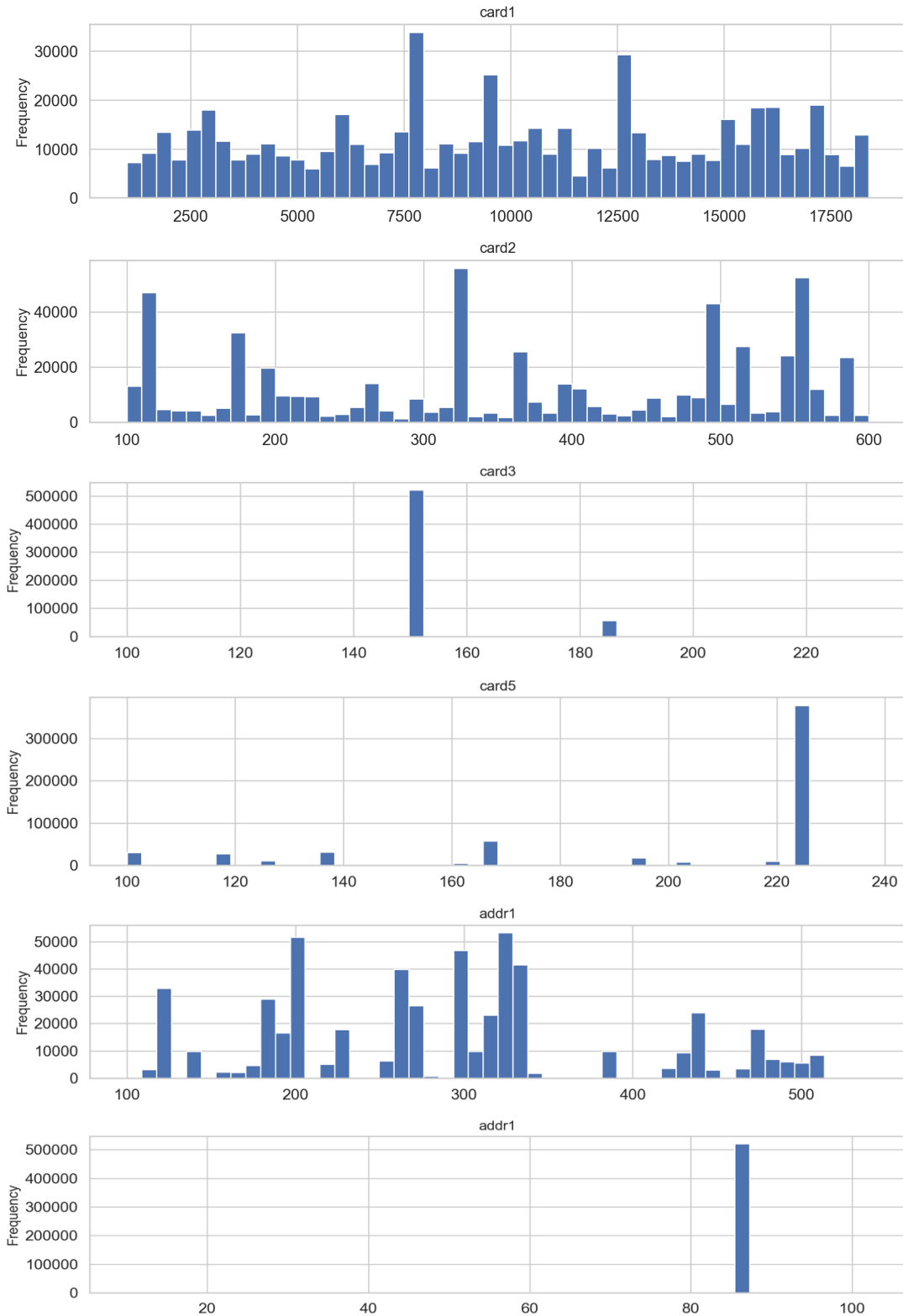


**Figure 12 Countplots from dataset features concerning the fraud transactions**

We visualized the frequency of fraudulent transactions (countplots) along with our newly constructed features Day and Time, and we observed that in the given dataset, the most fraudulent transactions took place during Saturday and Sunday, and as it concerns the time in the early afternoon. This pattern seems logical due to the fact that the perpetrators/thieves usually commit fraud at weekends because it is more difficult for Bank Sector and Law Enforcement Agencies to react fast and effectively.

Furthermore, we noticed that thieves and hackers prefer to break debit cards rather than credit or charge cards, and as it concerns the company, they prefer Visa cards rather than MasterCard and Discover. These insights can lead to useful information such as possible vulnerabilities concerning debit cards or the appearance of a weak security and fraud prevention department of that company, which gathers the most fraud transactions.

As the features card1, card2, card3, card5, addr1, and addr2 are numerical; we visualized the distribution of them:

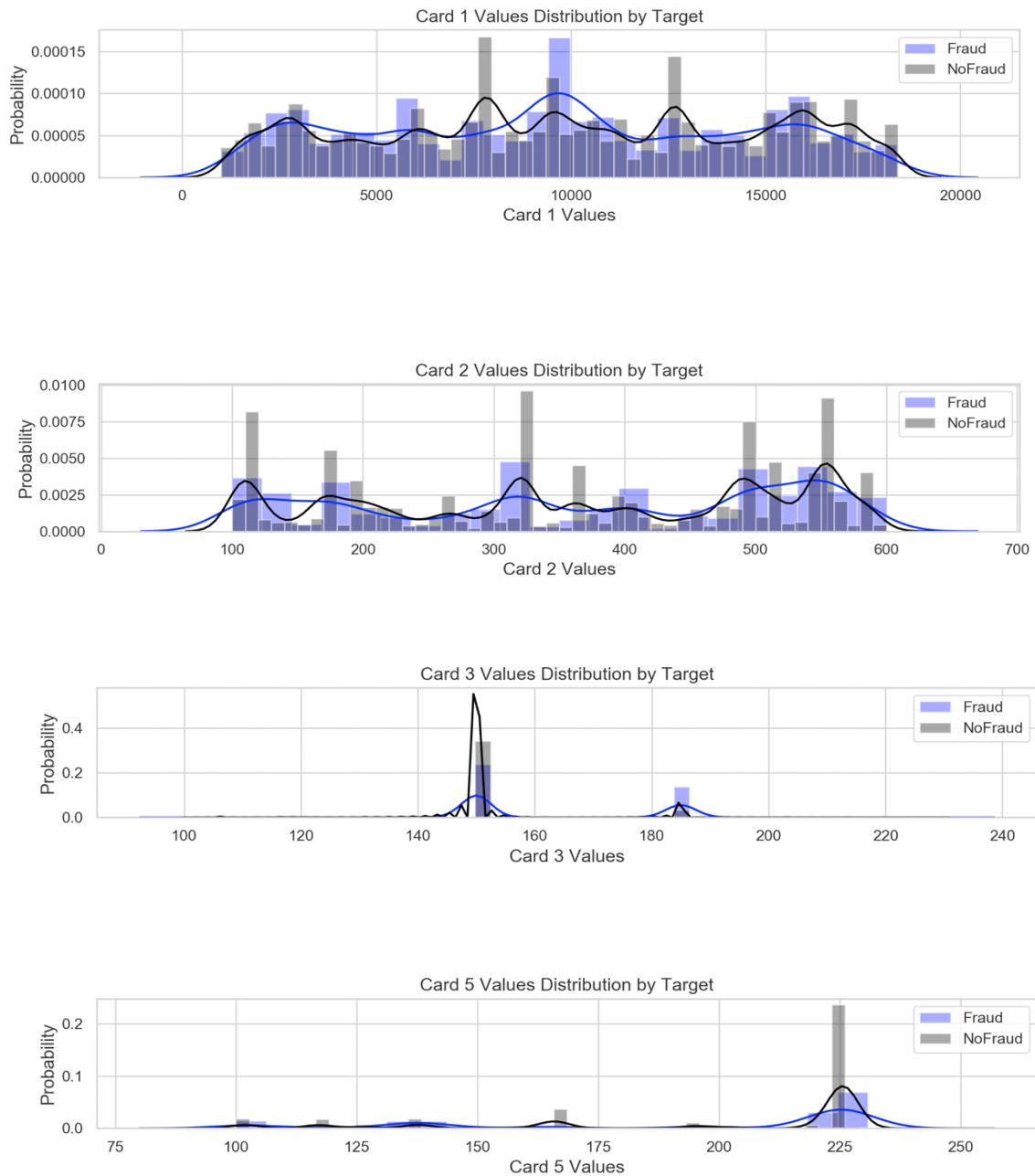


**Figure 13** Distribution plots for features card1, card2, card3, card5, addr1, addr2

The distribution plot is proper for comparing range and distribution for groups of numerical data. Data are plotted as value points along an axis. According to the above figures, we have an

overview for the values of features card1, card2, card3, card5, addr1, addr2. We observed that card3, card5 and addr2 have few values in contrast with card1 and card2 features. In addition, we observe that almost all entries in Addr2 feature are in the same value.

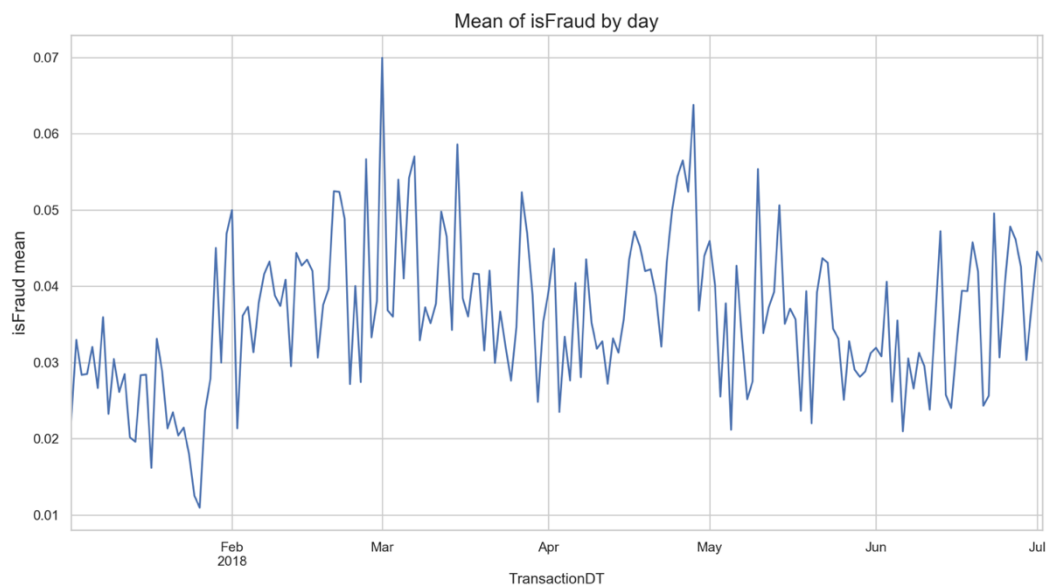
Interested approach would be to plot the features card1, card2, card3, card5 according to each transaction status (fraudulent and legitimate):



**Figure 14** Distribution of both transactional status (features card1, card2, card3, card5)

The most meaningful information that we can extract from the above figures is that we noticed that the values of card5 feature, which indicates a fraud transaction is around two hundred and twenty-five -225- value. For the card3 feature, the values that indicate a fraud transaction is around on hundred and fifty -150- value.

Another useful information would be the performance of the mean of fraud transactions per day. By applying a time plot in our given data, we possess the below graphical overview:

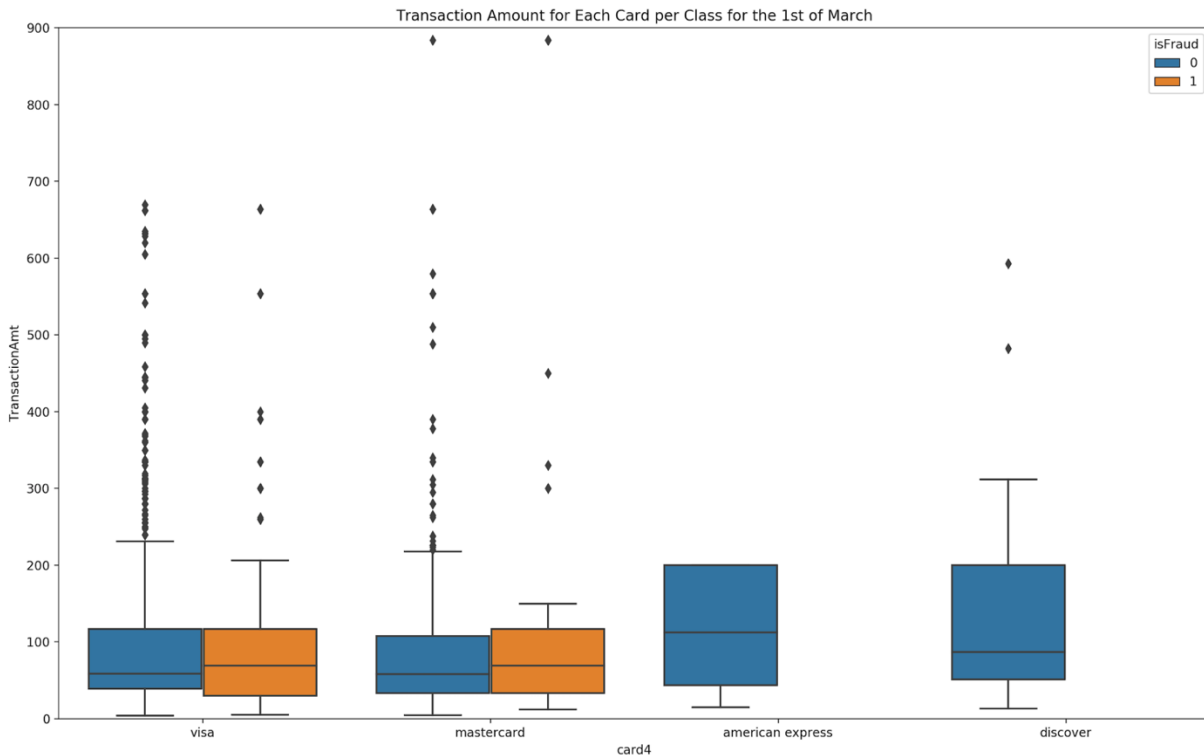


**Figure 15 Time-series fraud visualization**

The above graphical representation shows the mean value of the fraud transaction; this kind of plot constitutes a time-series (time plot) visualization. A time series, as the name suggests, is a series of data points with respect to time. The data points are indicators of some activity that takes place in a given period of time. Plots of the raw sample data can provide valuable diagnostics to identify temporary structures like trends, cycles, and seasonality that can affect the selection of models.

As a first overview, we noticed that the first day of March had been recorded a lot of fraudulent transactions due to the value of the mean; Also, we can recognize a seasonality pattern; because the first days of each month, we observe a low mean of fraud transactions. If we had a more representative sample with fraud and legitimate transactions, we could have traced more phenomena, and maybe we will be able to make valid, future forecasts of fraudulent transactions over time.

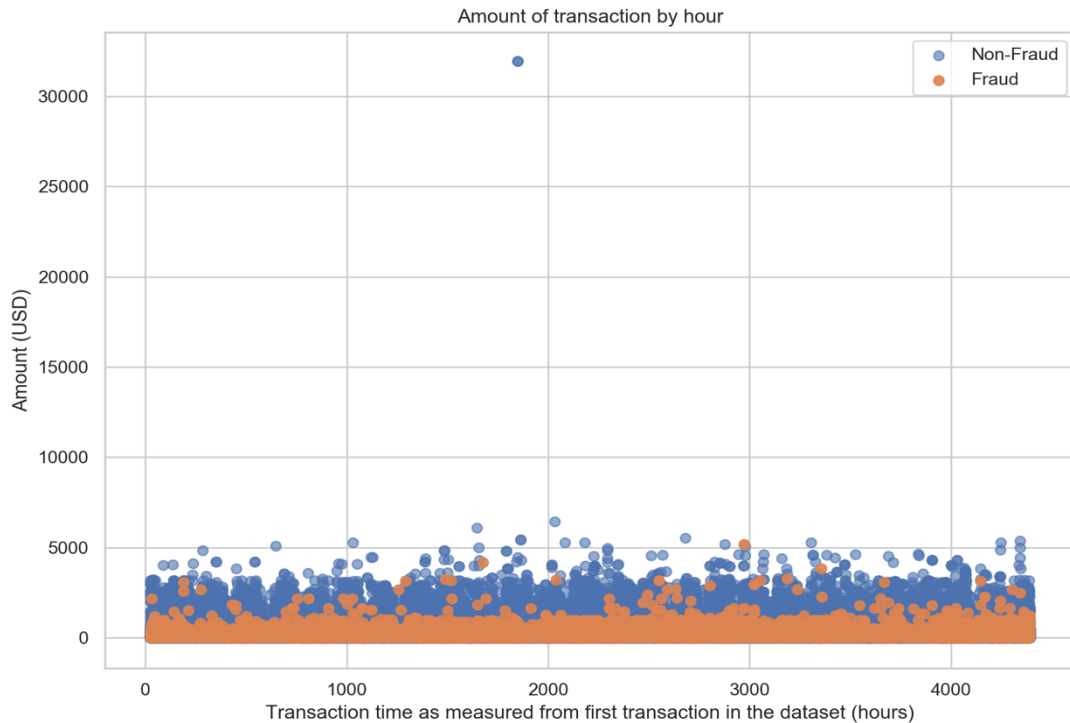
According to the above insight, we extracted a boxplot graphical representation to visualize the feature “card company” concerning the first of March, which had been recorded most of the fraud transactions.



**Figure 16 Transactional components for the first day of March**

According to the above figure, we noticed that the day that recorded the most of fraud e-commerce transactions, thieves/hackers, preferred to broke Visa and MasterCard cards, and hardly American express and Discover company cards. Additionally, we observed that in fraud status, Visa, and Mastercard had many outlier values. The most significant fraudulent transaction that took place on the first of March performed with a Mastercard card, and the financial loss was approximately nine hundred 900\$ USD. Furthermore, we observed that the range of values in fraud transactions was almost equal for Visa and Mastercard; Mastercard’s median was higher than Visa’s, Visa’s boxplot had greater max value and higher min value concerning the financial loss than the Mastercard’s, and also, Visa had more outlier values than Mastercard.

Concerning the financial loss, we created a scatterplot to have an overview for overall transactions of the given dataset concerning both fraud and authorized e-commerce transactions:



**Figure 17 Scatterplot - Amount of transaction by hour**

We clearly recognize that the financial loss of fraudulent transactions is lower than legitimate transactions. Also, we can distinguish some outlier values concerning fraud transactions, with the greatest of them being close to five thousand 5000\$ USD as financial loss.

Another interesting insight would be to identify the relationships between the values of various features of the given datasets; a useful technique to quickly examine correlations among dataset's features is by visualizing the correlation matrix as a heatmap. Heatmaps are graphical depictions of data that utilize systems coded by specific colors. The principal purpose of Heatmaps is to adequately visualize the volume of events within a dataset and support in directing the observers towards areas on data visualizations that matter the most. Heat Maps can be used for various data visualizations. Because of their dependence on color to communicate values, Heat Maps are possibly the most useful visualization to display a more generalized view of numeric values. In our problem, we used heatmaps to represent the relationships between Cs, Ds, and between mixed dataset's features.

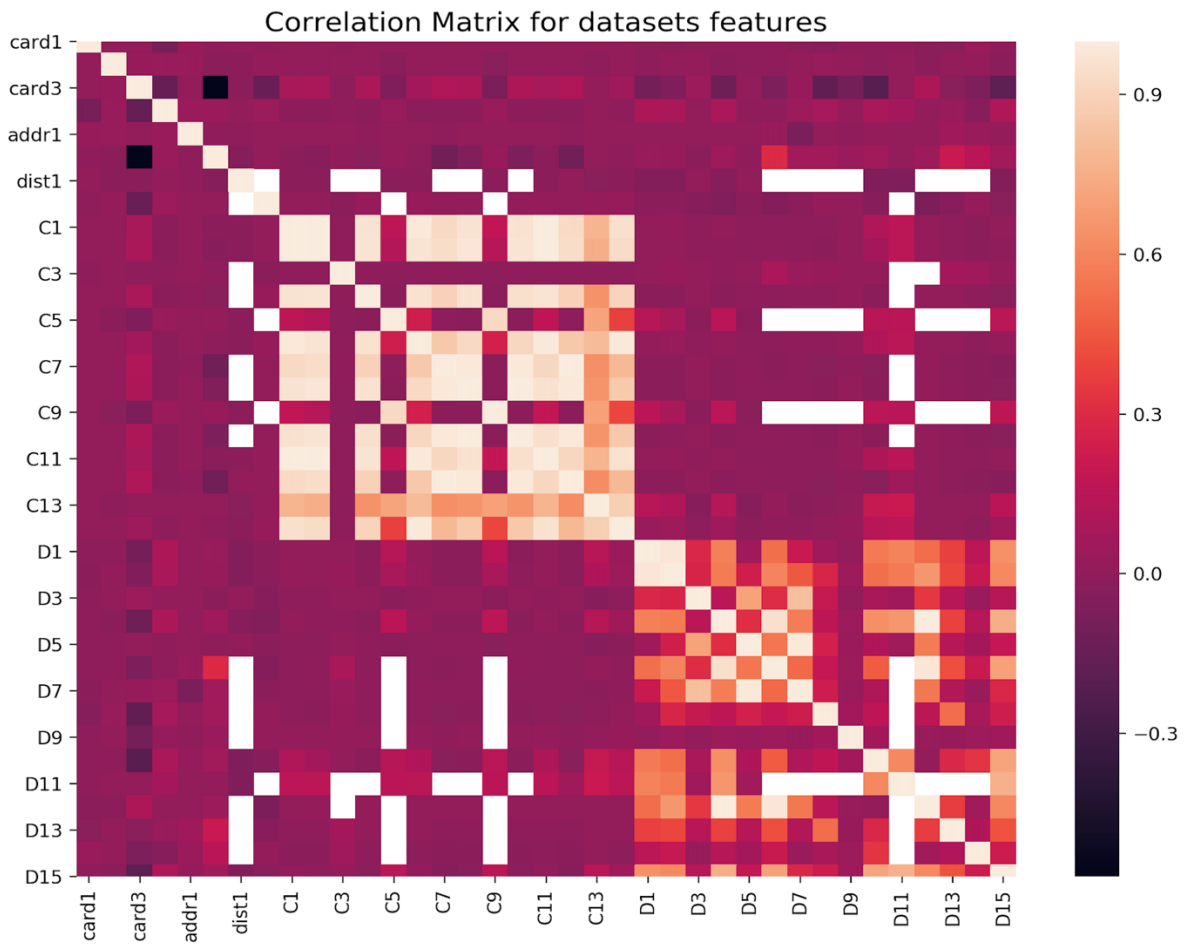
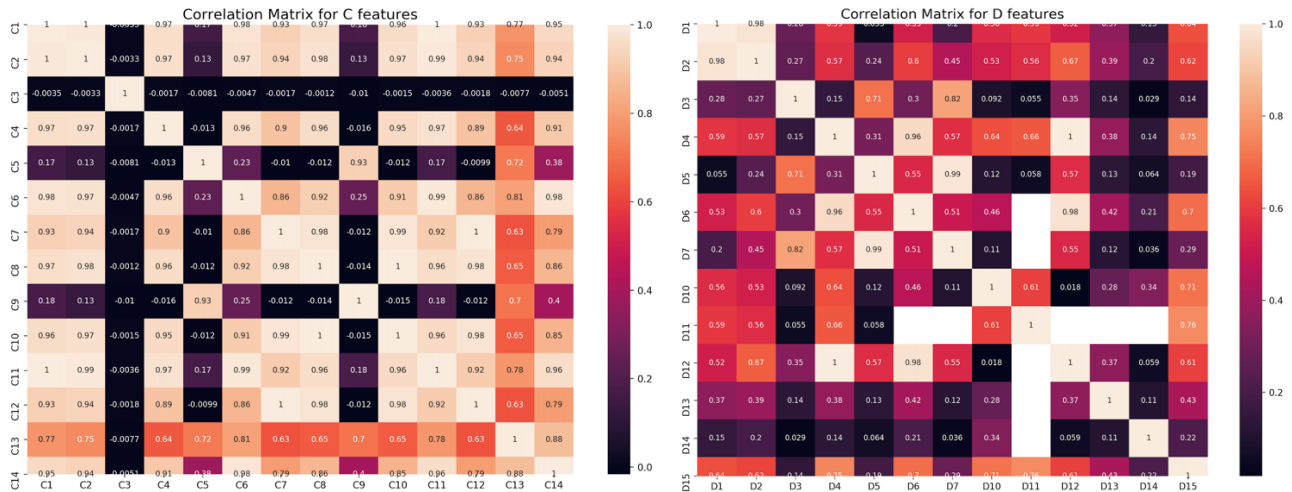


Figure 18 Correlation Heatmaps for various features



Heatmaps are based in hot and cold color zones on a map; The brighter the palette is, the higher is the correlation between variables and respectively, the darkest the shade, the weaker is the correlation.

According to the figures below, except for Cs and Ds, most of the remainder features are not linearly correlated since the biggest part of the heatmap has the same color, which is corresponding to correlation values around zero -0-. The C features correlation matrix presents many areas that depict a strong correlation between them; especially, we noticed that C1 is at the most strongly correlated with the other Cs features except C3, C5, and C9. Also, we observed that the C3 has a negative association with the rest of C features due to the low correlation values. The D feature correlation heatmap has few associations; particularly, their associations are in a medium-scale because the correlation values fluctuate between 0.4 and 0.6.

According to the above data visualizations, we have come to the conclusion that EDA is an endless and custom-made procedure that requires a lot of experience and knowledge to extract the most vital insights that can help the data analyst to approach his problem and determine the appropriate strategy to bring the most valuable result. Every analyst shapes his own strategy to apply EDA analysis according to his point of view. EDA procedure encourages the analyst to think critical questions about the given problem such as what he is trying to solve, what kind of data he has, and how to treat the different types of them, what is missing from the data, and how does it deal with this, where are the outliers, and why should they care about them, how can he append, change, or remove features to get more out of his data.

### **3.3.6 Dimensionality Reduction – PCA**

Our given data sets, which have been merged consist of four hundred and thirty-five -435- columns each of them represents a feature of a unique e-commerce transaction, except the “label” column which concerns the status of transaction (fraud/legitimate); The abundant quantity of features ascribes a high dimensional data set which is hard to plot it effectively. Also, the more dimensions the data set has, the harder it is to process it. Thus, we thought to reduce the dimensions of the given dataset without losing the information of it. The method that we implemented for dimensionality reduction was Principal Component Analysis – PCA. PCA is a method of extracting essential variables from a large set of variables available in a data set. It extracts a low dimensional set (a subset) of features from a high dimensional data set an into a new coordinate system with a purpose to capture as much information as possible before running a machine learning algorithm on the data. Thus, with fewer variables, visualization also enhances much more significant. The main target of the PCA method is the discovery of the directions of the maximal variance of data.

Generally, as we referred in a preceding chapter, Principal Components Analysis is a statistical technique based on linear algebra and matrices computations, and it used to explain data in high

dimension using a smaller number of variables called the principal components. Principle Components obtain the linear combinations of the original variables in the data set. The first principal component implies a linear combination of the original variables. In the new coordinate system, the first axis corresponds to the first principal component; it is the component that describes the highest amount of the variance in the data; the second principal component is selected such that it sprawls vertical to the first principal component. According to the procedure of the PCA method, we need to compute the covariance matrix, eigenvalues, and eigenvectors, as we already referred to in the first chapter of this thesis.

Specifically, as we mentioned, our dataset has about four hundred and thirty-three -433- features or variables (we do not include the TransactionID which is unique for each transaction, the label which corresponds to the status of the transaction and the exact DateTime that the transaction was executed); or -explaining it better- 433 dimensions in matrix algebra and one -1- target vector determines the transaction status (fraud/legitimate) dependent to the features mentioned above. Hence, the problem is in 433 -Dimensional. 433D is much; therefore, we reduce it to 2D for the illustration of PCA. Below we represent the steps concerning the PCA method that we applied in our given data:

- **Load the dataset**

We have loaded the data to a matrix having 590540 samples (x-axes) and 433 features (y-axes) for each sample.

- **Standardize the dataset**

The principal components are supplied with a normalized version of original features. It is an unavoidable procedure to implement the normalization process in the data before applying the PCA method. Different variables in the data set may be having different units of measurement (different scale). The PCA calculates a new projection of the data set, and the new axis is based on the standard deviation of the given variables. Performing PCA on un-normalized variables will lead to insanely high loadings for variables with high variance; thus, a variable with a high standard deviation will have a higher weight for the calculation of axis than a variable with a low standard deviation. If the analyst normalizes his data, all variables will have the same standard deviation. So, we standardized our data (e.g., subtracting the mean of each variable from the values), making the mean of each variable equal to zero.

- **Calculation of Covariance Matrix**

The next step was to calculate the covariance matrix of our dataset; the Covariance matrix is merely a square and symmetrical matrix of covariance of features (dimensions). Covariance is the variance of 2 features; more illustratively, how two -2 features vary from each other.

A large covariance value (positive or negative) indicates that the variables have a strong linear relationship with one another. Covariance values close to 0 indicate a weak or non-existent linear relationship. The below table is depicting a subsample of the calculation of the covariance square matrix in python:

**Table 10 Subsample of the calculation of covariance matrix**

```
Covariance matrix
[[ 1.00000169e+00  1.47910106e-01 -5.76480669e-03 ...  1.07813773e-01
  1.06280180e-01  5.76065917e-02]
 [ 1.47910106e-01  1.00000169e+00  2.82995930e-04 ...  8.43293982e-01
  8.08135824e-01  4.47703738e-01]
 [-5.76480669e-03  2.82995930e-04  1.00000169e+00 ...  7.47245030e-04
  2.36442205e-03  3.09804596e-04]
 ...
 [ 1.07813773e-01  8.43293982e-01  7.47245030e-04 ...  1.00000169e+00
  8.71211804e-01  5.04005854e-01]
 [ 1.06280180e-01  8.08135824e-01  2.36442205e-03 ...  8.71211804e-01
  1.00000169e+00  3.58788489e-01]
 [ 5.76065917e-02  4.47703738e-01  3.09804596e-04 ...  5.04005854e-01
  3.58788489e-01  1.00000169e+00]]
```

- **Calculation of Eigenvalues and Eigenvectors**

Due to the fact that the covariance matrix is square and symmetrical, it is also diagonalizable, which signifies that an eigendecomposition can be calculated on the matrix; this is where PCA pronounces Eigenvectors and Eigenvalues for the data set. Eigenvalues and eigenvectors are the essential components of PCA; Eigenvalues and Eigenvectors are strongly connected components, which constitute the root characteristics of a matrix equation. An *Eigenvector* is a vector whose direction prevails unchanged when a linear transformation is applied to it. The principle components are the eigenvectors of the covariance matrix of the original dataset. They correspond to the direction (as it concerns the original n-dimensional space) with the highest variance in the data. Each eigenvector has an analogous eigenvalue. From the other side, the *Eigenvalue* is corresponding in a number that indicates how much variance there is in the data along that eigenvector (or principal component). There are three -3- simple features we need to know about them: a) we can only calculate eigenvalues/eigenvectors of a square matrix (n x n, the covariance of the matrix), b) eigenvectors are perpendicular/orthogonal to each other. If we have an n-dimensional matrix, so we have n eigenvectors in n-space, and all of them are perpendicular; this makes sense because all of them constitutes the data they represent c) the length of eigenvectors is exactly one -1- and each has a corresponding eigenvalue which represents the power of the vector. Hence, we calculated with the suitable Python's NumPy modules, the eigenvectors, and eigenvalues, and a subsample of them is presented below:

**Table 11 Subsample of the calculation of Eigenvectors from covariance matrix**

Eigenvectors					
[	-1.26902518e-02	-7.42380172e-03	1.02379417e-02	...	2.17587044e-07
	1.07278943e-06	-6.71045874e-08]			
[	-9.43025359e-02	-4.08208742e-02	3.94083910e-02	...	-1.08162976e-05
	-3.40592692e-06	2.57668248e-05]			
[	1.29372445e-03	-6.75051869e-03	-2.46557385e-03	...	-1.05934670e-06
	-5.07661566e-07	-1.41012925e-06]			
...					
[	-9.82708243e-02	-2.99403157e-02	8.65830445e-03	...	3.15493783e-06
	-4.73531456e-06	-3.17792559e-06]			
[	-9.90404630e-02	-2.66086934e-02	-1.55544894e-02	...	1.57180466e-05
	1.10242411e-05	-5.99375950e-06]			
[	-5.27608921e-02	-2.41181444e-02	-3.78015587e-02	...	2.98306828e-06
	-2.15904627e-06	1.57734907e-06]			

**Table 12 Subsample of the calculation of Eigenvalues from covariance matrix**

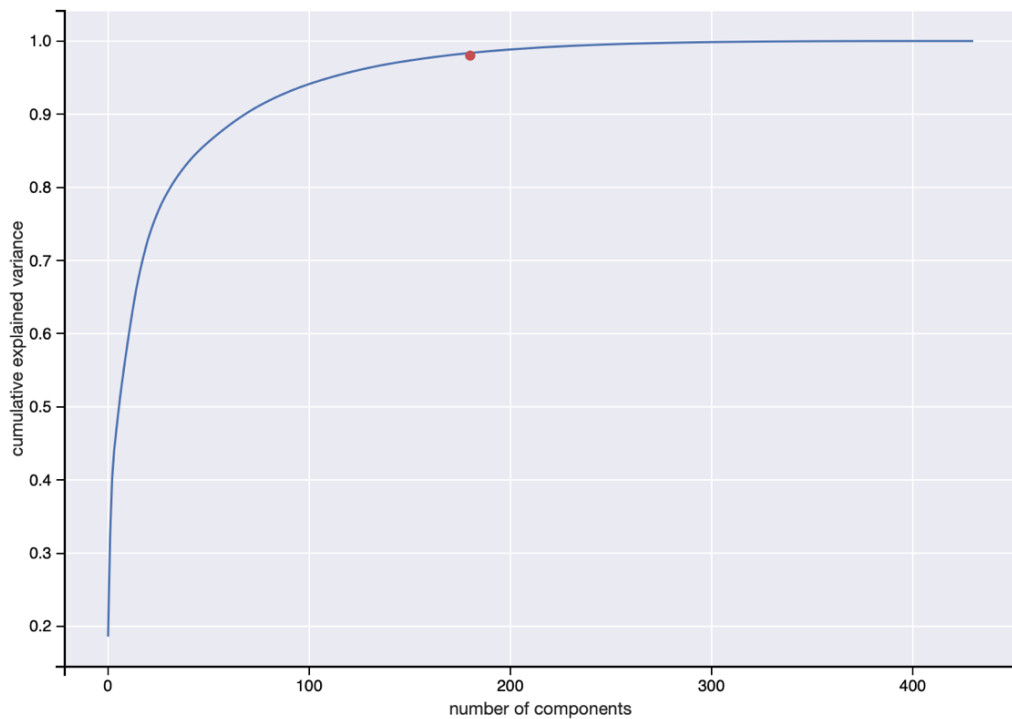
Eigenvalues				
[	8.00995308e+01	5.84139858e+01	3.41868776e+01	1.66944065e+01
	1.12746484e+01	1.04126294e+01	1.02321753e+01	9.04548667e+00
	8.72757540e+00	8.34451918e+00	8.05894126e+00	7.82796514e+00
	3.02036502e+00	2.91582854e+00	2.64813771e+00	2.58393442e+00
	2.23942022e+00	2.17485699e+00	2.11786031e+00	2.02436181e+00
	1.98839264e+00	1.83278451e+00	1.77526310e+00	1.74082695e+00
	1.60651721e+00	1.58382110e+00	1.52247432e+00	1.49456512e+00
	1.41717021e+00	1.39176773e+00	1.33566644e+00	1.27757045e+00.....]

- **Sorting Eigenvectors in a descending order according to their Eigenvalues**

Now that we have calculated the eigenvectors and eigenvalues, we hold an eigenvalue for each dimension in data and a corresponding eigenvector in results as listed above. What we need to do is to perform a descending order in each value of eigenvalues. Afterward, we choose k eigenvectors with the largest eigenvalues to form a **d×k** dimensional matrix.

- **Choosing the number of k components**

A vital part of using PCA in practice is the ability to estimate how many components are needed to describe the data. This can be determined by looking at the cumulative explained variance ratio as a function of the number of components. It is more efficient for us to visualize the variance of the data along with their original features and conclude which are the most important of them and how many retain the most significant part of the information of the data.



**Figure 19 Dataset's Feature importance**

This curve quantifies how much of the total, 433-dimensional variance is contained within the first  $N$  components. We conclude that around 98% of the variance of the data is included in the first 180 features. In other words, the first 180 features of the given dataset synthesize the 98% of the entire information from the data. Thus, the rest features involve only the last 2% of the aggregate information of the data. Our strategy oriented to keep these features that perform 85% of the entire information of the data because they have a low percentage of missing values, and they are more understandable from the other features. The idea behind the selection is that by choosing top  $k$  we have already decided that the variance which corresponds to those  $k$  feature space is enough to describe the data set. Furthermore, from the other side by losing the remaining variance of those not selected features will not cost the accuracy much, or we are prepared to drop that much accuracy that costs because of neglected variance. Hence, we worked the entire preparation with the features: 'ProductCD', 'card' features, 'addr1', 'addr2', 'dist1', 'dist2', 'P\_emaildomain', 'R\_emaildomain', 'C' features, 'D' features, and 'M' features. Regarding back our case study, we dropped columns like TransactionAmt and TransactionID and TransactionDT due to the fact that they were not valuable for the construction of the fraud alert model.

### 3.3.7 Feature Engineering

Proceeding with the features mentioned above, we possess the ability to extract some valuable knowledge and use them to interpret easier and more effectively our data. Here, the need for feature engineering arises. Feature Engineering efforts mainly have two purposes a) Preparing the proper input dataset, compatible along with the machine learning algorithm requirements, and b) Improving the performance of machine learning models. Feature Engineering demands a lot of experience and practice to discover insights and useful information which can contribute to the effectiveness of the machine learning model process.

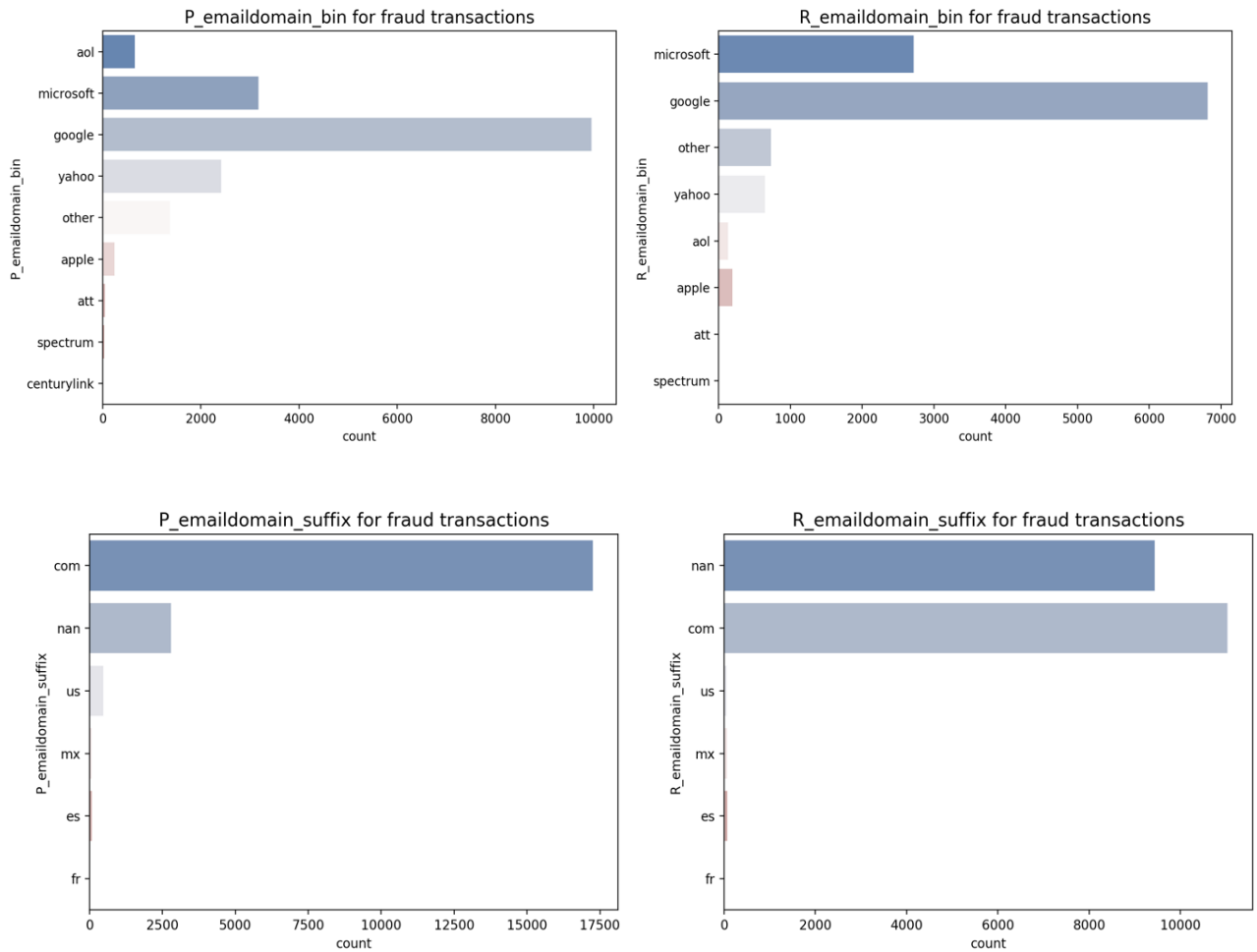
In our case study, we focused on features P\_emaildomain and R\_emaildomain. According to dataset's instructions, these two features constitute the purchaser and recipient email domain. We thought to create four -4- new columns concerning emails suffix, and email's host company. Thus, our new features are 'P\_emaildomain\_bin', 'R\_emaildomain\_bin', 'P\_emaildomain\_suffix' and 'R\_emaildomainsuffix'. The first two features constitute the company that hosts the Purchaser and Recipient email addresses (i.e. Google Inc, Microsoft Corporation), and the last two features indicate the suffix of the Purchaser and Recipient email addresses according to the country (com, us, es, fr).

Thereby, according to real information and our logic, we created one python dictionary, which helped as to create the features mentioned above:

**Table 13 Custom-made dictionary for implementation of feature engineering technique**

```
{'gmail': 'google', 'att.net': 'att', 'twc.com': 'spectrum', 'scranton.edu': 'other', 'optonline.net': 'other',
'hotmail.co.uk': 'microsoft', 'comcast.net': 'other', 'yahoo.com.mx': 'yahoo', 'yahoo.fr': 'yahoo',
'yahoo.es': 'yahoo', 'charter.net': 'spectrum', 'live.com': 'microsoft', 'aim.com': 'aol', 'hotmail.de':
'microsoft', 'centurylink.net': 'centurylink', 'gmail.com': 'google', 'me.com': 'apple', 'earthlink.net':
'other', 'gmx.de': 'other', 'web.de': 'other', 'cfl.rr.com': 'other', 'hotmail.com': 'microsoft',
'protonmail.com': 'other', 'hotmail.fr': 'microsoft', 'windstream.net': 'other', 'outlook.es': 'microsoft',
'yahoo.co.jp': 'yahoo', 'yahoo.de': 'yahoo', 'servicios-ta.com': 'other', 'netzero.net': 'other',
'suddenlink.net': 'other', 'roadrunner.com': 'other', 'sc.rr.com': 'other', 'live.fr': 'microsoft',
'verizon.net': 'yahoo', 'msn.com': 'microsoft', 'q.com': 'centurylink', 'prodigy.net.mx': 'att',
'frontier.com': 'yahoo', 'anonymous.com': 'other', 'rocketmail.com': 'yahoo', 'sbcglobal.net': 'att',
'frontiernet.net': 'yahoo', 'ymail.com': 'yahoo', 'outlook.com': 'microsoft', 'mail.com': 'other',
'bellsouth.net': 'other', 'embarqmail.com': 'centurylink', 'cableone.net': 'other', 'hotmail.es': 'microsoft',
'mac.com': 'apple', 'yahoo.co.uk': 'yahoo', 'netzero.com': 'other', 'yahoo.com': 'yahoo', 'live.com.mx':
'microsoft', 'ptd.net': 'other', 'cox.net': 'other', 'aol.com': 'aol', 'juno.com': 'other', 'icloud.com': 'apple'}
```

By creating the features as mentioned earlier, we noticed that useful patterns are shaping concerning fraud transaction; hence, we visualized their values and are depicted below:



**Figure 20** New features for both transaction status

According to preceding figures, we noticed that fraud transactions follow a specific pattern as it concerns the Purchaser and Recipient email address. Particularly, at most of the fraudulent transactions, the thief works with an email address, which belongs to the Google Company; the same is applicable for the Recipient email address. These new features, which are produced by feature engineering method, are necessary to adequately use machine learning algorithms and thus build predictive models, giving robust information about fraud and legitimate transactions. Generally, understanding the meaning of the features leads to the most significant gains in performance. Knowing the role of these features is vital to understanding machine learning. Mostly, performing feature engineering techniques constitutes a crucial procedure, but it can drive to excellent results. This is what data scientists focus on the majority of the time along with the data cleaning, which is introduced in-depth in the next subchapter that follows.

### 3.3.8 Data Cleaning

The most frequent phenomenon in data analysis is the appearance of missing values. Almost in any data case study, the analyst faces problems concerning the scarcity of data values, which can affect his entire analytical procedure. The concept of missing values is essential to understand in order to manage data successfully. If the researcher does not appropriately handle the missing values, then he probably ends up drawing an inaccurate inference about the data. Due to incorrect manipulation, the result obtained by the researcher will vary from one where the missing values are present. Missing data may come in a variety of ways; usually, missing values are indicating with symbols like N/A, nan, inf, or with a blank cell without any value. The best way to prepare for dealing with missing values is to understand the data which analyst has: to understand how missing values are represented, whence the data was collected, where missing values are not assumed to be, and where are explicitly used to represent the absence of data. Domain knowledge and data understanding are the most critical factors to deal with missing data successfully; moreover, these factors are the most important in any part of the data science project.

According to the research performed officially by Donald Rubin, the missing data mechanism was introduced. Missing data mechanism describes the underlying mechanism that generates missing data and can be categorized into three -3- types a) missing completely at random (MCAR), b) missing at random (MAR), and c) missing not at random (MNAR). MCAR indicates that the occurrence of missing values is entirely at random, not related to any variable. MAR implies that the missingness only relates to the observed data, and NMAR refers to the case that the missing values are related to both observed and unobserved variables, and the missing mechanism cannot be ignored. Researchers suggest various ways to manipulate efficiently and effectively the missing data problem, but the crucial process is the selection of the most appropriate strategy for data cleaning. In order to understand how to deal with missing data, the analyst needs to understand what types of missing data he has to deal with, and it might not be straightforward to grasp their differences.

According to our case study, and as we saw previously on EDA Analysis, we had to deal with a high percentage of missing values. We considered that our case's missing values indicate to MCAR type since the Vesta's Systems have provided us the given data completely unprocessed and hence some of their features are stated by the consumers plus possibly were not real or perhaps were reported mistakenly, and some of them did not match to the corresponded contextual case. According to our given data, and the competition's host guidance, we separated the features based on the type of them (numerical and categorical). In proceeding, we explored various methods in order to handle our missing values such as the removal of the columns which the 80% of them include missing values, or to test suitable machine learning algorithms to estimate the values; we believed that no one of the techniques mentioned above were effective for our case study, so we ended up by applying the below data imputation strategies by data type:



- As it concerns *numerical variables*, we tried various imputation and estimation techniques such as filling missing values with mean, median, mode or -999, but we end up filling the missing values with the most frequent value based on each column.
- As it concerns *categorical variables*, the first step was to convert all values with lower-case letters and then we filled missing values with the string “unknown”.

### 3.3.9 Label encoding for categorical variables

Arguably, most of the Machine learning algorithms cannot manipulate categorical variables unless they are converted to numerical values, and many algorithms performance varies based on whereby Categorical variables are encoded. There are plenty of methods that we can encode those categorical variables as numbers and use them in the algorithm, but the most well-known techniques are a) one-hot encoding and b) label encoding. In our case, we used the label encoding method. In label encoding, each category is assigned a value from 1 through N (where N is the number of category for the feature. One major issue with this approach is that there is no relation or order between these classes, but the algorithm might consider them as some order, or there is some relationship. With the appropriate source code and with the support of the scikit-learn <sup>29</sup>python package, we implemented Label Encoding technique to our categorical data; especially, with the usage of specific modules from the package as mentioned above, we read the entire set of categorical data, and nested dictionaries have been created in which each key indicated to a specific dataset's feature including all the numerical values that were corresponding to the real values.

**Table 14 Dictionary for implementation of label encoding technique**

```
{'ProductCD': {'c': 0, 'h': 1, 'r': 2, 's': 3, 'w': 4}, 'card4': {'american express': 0, 'discover': 1, 'mastercard': 2, 'unknown': 3, 'visa': 4}, 'card6': {'charge card': 0, 'credit': 1, 'debit': 2, 'debit or credit': 3, 'unknown': 4}, 'P_emaildomain': {'aim.com': 0, 'anonymous.com': 1, 'aol.com': 2, 'att.net': 3, 'bellsouth.net': 4, 'cableone.net': 5, 'centurylink.net': 6, 'cfl.rr.com': 7, 'charter.net': 8, 'comcast.net': 9, 'cox.net': 10, 'earthlink.net': 11, 'embarqmail.com': 12, 'frontier.com': 13, 'frontiernet.net': 14, .....}, 'M1': {'f': 0, 't': 1, 'unknown': 2}, 'M2': {'f': 0, 't': 1, 'unknown': 2}, 'M3': {'f': 0, 't': 1, 'unknown': 2}, 'M4': {'m0': 0, 'm1': 1, 'm2': 2, 'unknown': 3}, 'M7': {'f': 0, 't': 1, 'unknown': 2}, 'M8': {'f': 0, 't': 1, 'unknown': 2}, 'M9': {'f': 0, 't': 1, 'unknown': 2}, 'R_emaildomain_suffix': {'com': 0, 'de': 1, 'es': 2, 'fr': 3, 'jp': 4, 'mx': 5, 'uk': 6, 'unknown': 7, 'us': 8}.....}
```

<sup>29</sup> **Scikit-learn** is a free third-party package -ML library for the Python language. It includes various classification, regression and clustering algorithms and is designed to interoperate with well-known Python numerical and scientific libraries like NumPy and SciPy.

### 3.3.10 Dealing with imbalanced dataset

Another huge problem that analysts face during classification procedures concerns the imbalanced annotation of the datasets. Notably, the main field of this type of classification problem is that the examples of one class overcome the examples of the other one significantly. In current years, the imbalanced learning problem has received much attention from the machine learning community, and various techniques have been developed to face the problem as mentioned above. Concerning real-world domains, the importance of the imbalanced learning problem is increasing, since it is a recurring issue in various applications. Because most of the standard learning algorithms consider a balanced training set, this may generate suboptimal classification models, for example, good coverage of the majority examples, whereas the minority ones are misclassified frequently. Generally, the most painful spot for machine learning models that have been trained with unbalanced datasets is the appearance of low sensitivity, i.e., the lack of separability.

Therefore, those algorithms, which obtain a normal behavior in the framework of standard classification, do not significantly achieve the best performance for imbalanced datasets. The logic constitutes that many problems are tough to be solved with the support of machine learning because of the lack of samples. For example, problems concerning human disease diagnosis, terrorist attacks, online fraud cases, transportation accident occurrence are hard to be resolved due to the fact that we do not have adequate capacity of samples, which are annotated as favorable to the problem situation. The weakness to predict rare events constituting the minority class, plus the misleading accuracy detracts from the predictive models we build. The algorithm learns from the majority class at the most, making it “natural” for there to be a greater tendency towards it. The algorithm is then prone to overfitting the majority class. Just by predicting the majority class, models would score high on their loss-functions. In these instances, the phenomenon of high accuracy rate appears. Notably, in many paradigms, we have been seen that model which have been trained with imbalanced datasets achieves high scores of accuracy, which is normal since the model classifies new unseen test samples as being in majority class which is more reasonable to have appeared. From the other side, by using other evaluation metrics such as recall and Under the Area Curve - AUC- we have been observed that the mentioned model returned poor performance. Hence, in these cases, we have to choose evaluation metrics that provide guarantee and effective results and describes the real generalization performance of the model.

In our case study and according to the preliminary implementation EDA analysis, we captured as the first insight the unbalanced class that our given dataset constitutes. Fundamentally, we established that from 590540 e-commerce transactions, only 20663 had been annotated as fraud, i.e. the 3,50% of the entire datasets constitutes illegal transactions. Thus, according to the above information, we considered that possibly our approach for the card fraud detection would perform a poor performance concerning the evaluation of the machine learning models when they have to generalize since our algorithms will have been trained with a considerable amount of authorized transactions and a small collection of fraud transactions.

### 3.3.10.1 Proposed strategies for data transformation to test machine learning models

Because of the appearance of the above-mentioned problem, we decided, before trying to implement machine learning algorithms, to apply various techniques to enrich our dataset with minority class samples. Principally, in this thesis, we developed three -3- different approaches to face this situation and also to test the effectiveness of the chosen machine learning models that we used afterward. We chose to implement some very interested methods for datasets enrichment, which are widely used from the machine learning community.

- **Oversampling technique:**

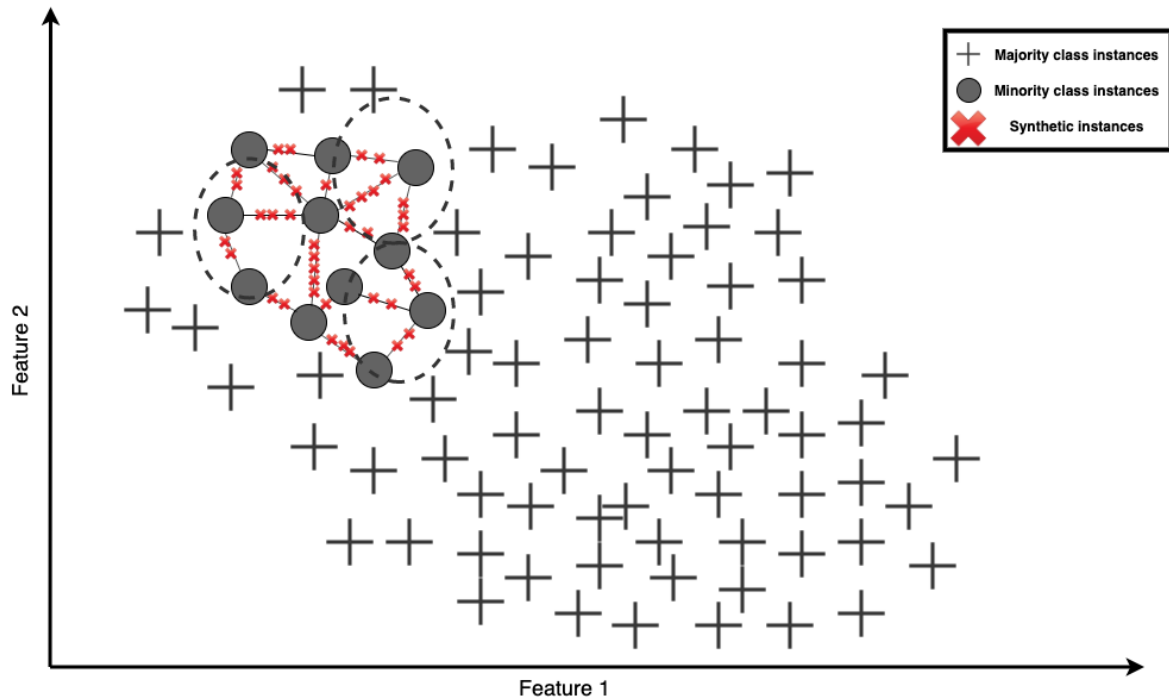
The first one method constitutes the dataset's enrichment with the support of the random resampling technique. Practically, the oversampling technique increases the weight of minority class samples by producing replicas of these samples until we have a balanced dataset. It is essential to mention that this technique does not increase the data information; it has been proved that it possibly causes overfitting issues, which make the chosen model being extremely specific. It may be the case that the accuracy for the training set is high, yet the performance for new datasets is genuinely worse. Also, possibly this technique bypasses useful data with essential information due to the fact that it chooses random samples. However, we decided to test this method along with our selected machine learning algorithms in order to comprehend its attitude to our given dataset.

- **Synthetic Minority Oversampling Technique – SMOTE and TOMEK LINK**

Another suggested proposal to increase the minority class - fraudulent samples, which is widely used from the machine learning community- is the combination of an undersampling technique named TOMEK-LINK along with the above-mentioned oversampling technique SMOTE because it can perform better classifier performance rather than using solely one technique. The second technique (SMOTE) was motivated by data augmentation technique for handwritten character recognition. The main scope of the SMOTE technique was to increase the decision boundaries as to concerns the minority class samples by producing synthetic examples rather than by over-sampling with replacement achieving, as a result, the increase of classifier sensitivity. The minority class is oversampled by using each minority class sample and injecting synthetic examples along the line portions joining any or all of the  $k$  minority class nearest neighbors. Depending upon the amount of over-sampling ordered, neighbors from the  $k$  nearest neighbors are randomly chosen.

According to the proposers of SMOTE, the technique follows typical algorithm steps to be implemented. The first step constitutes the calculation of the difference between the feature vector (sample) under consideration and its nearest neighbor; it follows the multiplication of the above difference by a random number between 0 and 1 and the sum with the feature vector under consideration; this procedure indicates the determination of a random point along the line segment between two specific features; this proposed approach efficiently pushes the decision region of the

minority class to become more general. The generated synthetic examples cause the classifier to create more extensive and less specific decision regions, which is the main scope of this technique, as we referred before.



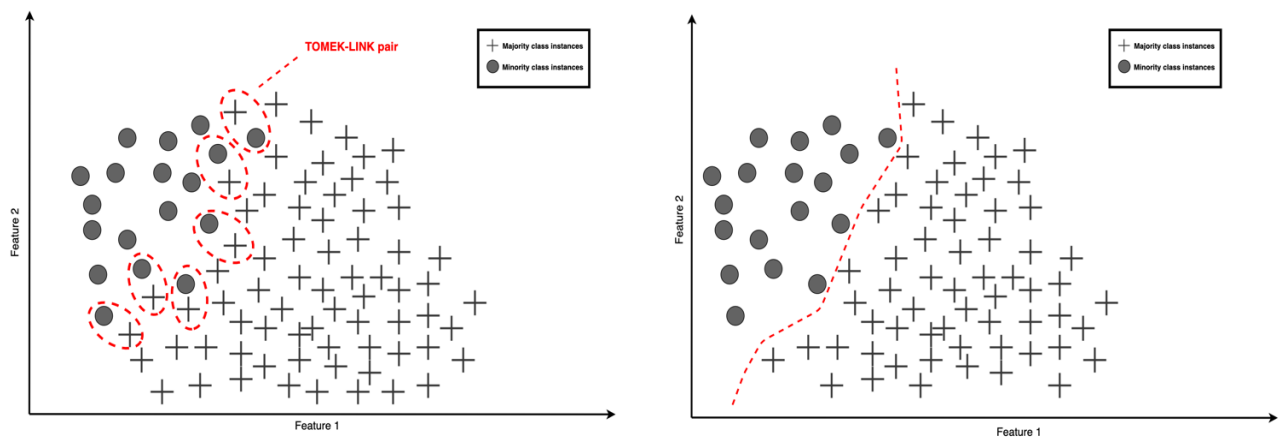
**Figure 21 SMOTE representation in 2D feature space with 3 nearest neighbor parameter**

The preceding figure represents an example of enrichment of minority instances with the support of the SMOTE Technique in 2-Dimension feature space. Particularly, given  $k=3$  as the nearest (closest) neighbors, SMOTE acts like drawing lines between existing minority instances, taking mind the three closest neighbors, which in reality are the distances between spots. Then the method proceeds by creating synthetic minority instances somewhere on these lines.

In python, the SMOTE technique is implemented with the `SMOTE()` function which is involved in `imblearn.over_sampling` package and operates principally with two parameters: `K` and `dup_size`; `K` parameter represents the  $k$ -nearest neighbor; i.e., how many neighbor minority instances the algorithm will take in mind to implement the above method (the algorithm provides by default  $k=5$ ). From the other side, the `dup_size` parameter responds to the question of how many times `SMOTE()` should loop through the existing, real minority instances. The above parameter symbolizes the percentage for sampling strategy; this parameter matches to the desired proportion of the number of observations in the minority class over the number of observations in the majority class after resampling, for instance, the value 0.70 means that the SMOTE algorithm will create 70% samples beside majority samples percentage. However, this method can be combined with an undersampling technique for majority class samples because, according to previous use-case experiences of analysts dealing with imbalanced datasets, it operates more effectively and achieves better evaluation performance without overfitting.

Our approach oriented to use TOMEK-LINK undersampling technique for majority class TOMEK-LINK, which is an effective method that can be used for the reduction of the majority by considering samples near the borderline. Notably, the method states that given two instances  $a$  and  $b$  belonging to different classes and are separated by a distance  $d(a,b)$ , the pair  $(a, b)$  is called a Tomek link only if there is no other instance  $c$  such that:  $d(a,c) < d(a,b)$  or  $d(b,c) < d(a,b)$ .

Instances participating in Tomek links are either borderline or noise, so both are removed. Thus, the implemented method calculates the distances between all instances belonging to different classes and considers as TOMEK-LINK pairs all these that ensures the above conjecture. Then, the user can choose to remove either entire pairs or only, for instance, the instances belonging to the majority class (according to his strategy).



**Figure 22 Implementation of TOMEK-LINK technique for majority class sample reduction**

For the thesis case study, the SMOTE technique was used to enrich the given imbalanced dataset semantically with minority class samples along with the TOMEK-LINK undersampling technique for majority class. At first, a reduction of the given dataset was implemented by removing the border majority class samples with TOMEK-LINK, and afterward, the SMOTE technique was used to enrich the dataset with synthetic minority class samples, which includes the 40% of the entire dataset observations. With the adoption of this undersampling strategy, any observations from the majority class are removed for which a Tomek's link is identified. Depending on the dataset, the technique, as mentioned above, will not actually achieve a balance among the classes - it will merely "clean" the dataset by removing some noisy and border majority class observations, which may result in a more natural classification problem.

- **Weight Balancing method**

Another suggestion for approaching the problem with the imbalanced dataset is the assignation of weight to the minority class. Weight balancing method balances data by altering the weight that each training example carries when computing the loss. According to the above suggestion, the user

can assign weight to the classes by merely multiplying the loss of each example by a particular factor depending on their class.

For the thesis given dataset, a dictionary -with two -2- key-pair values- was created that includes the weights that should have assigned to each class (legitimate and fraud status) according to the number of their samples; in particular, each weight was calculated as the fraction of 1 divided by the amount of each class samples. With the adoption of this method, a weight balancer was integrated into each observation in order to make both classes contribute equally to our loss. The assignation of class weight transfuses bias to the minority class samples causing higher sensitivity by giving more emphasis in these observations that constitute fraud transaction. This approach will cause the model to "pay more attention" to examples from an under-represented class.

As a synopsis, imbalanced data classification is an inherently challenging task considering there are so few samples to learn from. The analyst should always begin with defining the data first, then make a tremendous effort in order to gather as many samples as possible and provide substantial thought to what features may be relevant; thus, the model can get the most out of datasets minority class. At some point, the constructed model may strive to improve and yield the results that the analyst desire; thereby, it is essential to keep in mind the context of each custom problem and the trade-offs between different types of errors.

### **3.3.11 Machine Learning chosen models for card fraud detection**

According to the above-suggested methods for the transformation of the given dataset with a scope to achieve the most efficient results concerning the fraud detection, we ended by using four -4- different approaches to handle our dataset which already have mentioned: a) Raw imbalanced dataset as it was provided from the Kaggle repository, b) by assigning weight to minority class samples, c) by using the random oversampling method to enrich the dataset with replication minority class samples and finally d) by applying the TOMMEK-LINK and SMOTE techniques to similarly to enrich the minority class observations.

In this phase of the procedure, we are moving across to the machine learning part; this section mainly involves the most crucial task for building the fraud detection model, which is the training procedure of the chosen algorithmic models with the historical e-commerce transaction data. This chapter ends by testing four -4- different kinds of machine learning, and the model with the highest rank is selected to be deployed in the Openwhisk serverless platform.

### 3.3.11.1 Dataset's separation & the chosen Validation method

However, in the machine learning part, the "learning" phase, which is a crucial task, should be preceded by another typical procedure that constitutes the dataset's separation. In particular, the typical procedure includes the segregation of training and test sets and the adoption of a validation method named Stratify K-fold cross-validation, which constitutes a resampling procedure that is used to evaluate the performance of the ML models on a limited data sample. Since our data have been transformed and are ready to be fed into ML algorithms, we should consider splitting them into two - 2- groups for training and testing the chosen models. The first and most important group of data is the training set which establishes the sample of data that is used to fit the model, from the other side, the test data constitutes the sample of data used to provide an unbiased evaluation of a final model fit on the training dataset; The test dataset provides the gold standard used to evaluate the model. It is solely used once a model is thoroughly trained. Concerning the dataset's split ratio, the proportion to be divided is utterly dependent on each user's strategy and the task he has to face.

For this thesis case study, we followed a typical separation ratio and randomly chose the 80% of the given dataset to be the actual train set and the remaining (100-80) % to be the test set. Concerning the training set, which is the essential group of data for the chosen ML algorithm in order to achieve the generalization, we chose to use in our - case-study- a particular validation method named stratified k-fold cross-validation. In Machine learning, the datasets split approach may cause problems; in particular, due to sample variability between training and test set, the model gives a better prediction on training data but fail to generalize on test data; this situation leads to a low training error rate but a high test error rate. Also, when we separate the dataset into training and test sets, we use a subset of data solely, and we comprehend when we train on fewer observations, the model will not perform adequately and overestimate the test error rate for the model to fit on the entire dataset. Thereby, to solve the two-issue, we adopted the approach called cross-validation. Cross-validation, as we already mentioned, is a statistical procedure that involves partitioning the data into k subclasses, training the data on a subset, and work with the other subset in order to evaluate the model's performance. Our strategy oriented to select a particular type of cross-validation specified as Stratified cross-validation with ten -10- number to be the k parameter; Stratification is a technique in which we reconstruct the data in a way that each fold has a reliable description of the entire dataset. It forces each fold to have at least n instances of each class. This approach guarantees that one class of data is not overrepresented, mainly when the target variable is unbalanced, i.e., in our case-study, the technique, as mentioned before, ensures that each fold has a percentage of legitimate e-commerce transactions and a percentage of fraudulent transactions. This method constitutes the avoidance of both bias and variance.

### 3.3.11.2 Chosen ML models for the classifier's construction

Hence, since we defined the manipulation of the training set, we are proceeding with the selection of the ML algorithms. The idea was to choose well-known algorithms which are widely used for binary classification problems; we ended by choosing: a) Logistic Regression, b) Random Forests, c) Extra Trees Classifier, and d) Artificial Neural Networks. Consequently, each of these selected models has followed the training procedure along with each one of the techniques mentioned above for data transformation, and in the end, evaluation metrics were calculated to select the most appropriate model-classifier. Below we introduce each algorithm and the hyperparameters that we chose to accompany each model:

- **Logistic Regression:**

The logistic regression arises from the statistics field; it is delineated by a logistic function to model the conditional probability of the label  $Y^{30}$  according to variables  $X^{31}$ .

In our case,  $Y$  indicates the transactional status, and  $X$  represents the rest of the features that accompany the  $Y$  variable (e.g., card features,  $M$  features). The difference between the linear regression and the logistic is that the first one estimates continuous numbers to be the value of the dependent  $Y$  variable, in contrast with logistic, in which the output value which being modeled is a binary value (0 or 1) rather than a numeric value. The logistic regression classifier can be derived by analogy to the linear regression hypothesis which is:  $\mathbf{h}_\theta = \theta^T \mathbf{x}$ , however, the logistic regression hypothesis generalizes from the linear regression hypothesis in that it uses the logistic function:  $\mathbf{h}_\theta = g(\theta^T \mathbf{x})$ , where  $g(\mathbf{x}) = \frac{1}{1+e^{-z}}$ , according to the above mentioned formulas  $\mathbf{h}_{\theta(x)} = \frac{1}{1+e^{\theta^T \mathbf{x}}}$ . The function  $g(z)$  is the logistic function, also known as the sigmoid function which shapes an S-curve, has asymptotes at 0 and 1, and it crosses the y-axis at 0.5 point called “threshold”; sigmoid function can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits; in machine learning terms the sigmoid function calculates the probability for the  $Y$  variable indicating a binary classification problem. An example of a logistic regression equation is:

$$\mathbf{y} = \frac{e^{(-b_0 + b_1 * x)}}{1 + e^{(-b_0 + b_1 * x)}}$$

where  $\mathbf{y}$  is the predicted output (dependent value),  $b_0$  is the bias or intercept term and  $b_1$  is the coefficient for the single input value ( $x$ ). Each column (feature) in the input data has an associated  $b$  coefficient (a constant real value) that must be learned from the training data. Logistic regression models the probability of the default class (e.g. the first class). In our case-study, we are modeling the execution status for e-commerce transactions according their transformed features which derived from PCA, then the first class (1) could be the fraud status and the logistic regression model could be written

<sup>30</sup>  $\mathbf{Y}$  is indicating the dependent variable, the one that is predicted.

<sup>31</sup>  $\mathbf{X}$  is indicating the independent variables i.e the features.



as the probability of fraud transaction given its transformed features, or more formally:  $P(X) = P(Y = 1 | X_1, X_2, X_3, \dots, X_n)$ , where  $Y=1$  indicates the fraud status and  $X$ s indicating the features, so the above equation is transformed:

$$y = \frac{e^{(-b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_3 + \dots + b_n * x_n)}}{1 + e^{(-b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_3 + \dots + b_n * x_n)}}$$

Concerning the  $b$  coefficients (Beta values  $b$ ) of the logistic regression algorithm as we already referred, must be estimated from the training data, a task that it must be done by using maximum-likelihood estimation; the maximum-likelihood estimation is a standard learning algorithm used by a variety of machine learning algorithms. The conception for the maximum likelihood for logistic regression is a search procedure that seeks values for the coefficients (Beta values) aiming to minimize the error in the probabilities predicted by the model to those in the data. Once, the Beta coefficients have been calculated; we can use them to calculate the probability for the  $Y$  label; thus, when  $Y$  takes value under the threshold 0.5, then the transaction is legitimate due to the fact that we defined the first class to be the fraudulent transactions, from the other side, if the probability has been estimated with a value more than 0.5 then the  $Y$  belongs to the first label which represents the fraud status. This logic, in python terms, can be represented as one module derived from the third-party package “scikit-learn” named `LogisticRegression()`; this module initializes the logistic regression algorithm automatically, and the only thing we can do is to insert the data in order to train (fit) the model. We tested the logistic regression model four -4- times, each of them with one of the cases mentioned above a) raw data, b) weighted as it concerns the minority class, c) with applying the random oversampling technique and d) by applying TOMMEK-LINK and SMOTE techniques. The evaluation metrics are represented in the next chapter.

- **Random Forest:**

The main logic of decision tree algorithms is that they build blocks of "forests"; Breiman, 2001 introduced Random forest model; it is a special kind of decision tree algorithm, and, like its name, intimates it represents a group of individual decision trees that operate as a group. Actually, a random forest is a collection of decision trees which are splitting according to the optimal feature (decision workflow), where each tree is slightly altered from the others. Random forest logic inherits benefit from decision trees, and the central concept behind random forests is that each tree might do a relatively good job of predicting but will likely overfit on the part of the data. We chose the random forest algorithm because we wanted to avoid the overfitting phenomenon. Notably, if we build many trees, all of which work well and overfit in different spots; in this manner, the amount of overfitting is decreasing by using the average of their results. In order to implement this procedure, many decision trees must be built with each of them to do an acceptable job of predicting the target and should also be different from the other trees. Random forests received their names from introducing randomness

into the tree building for ensuring each tree's difference. Random forests for regression and classification problems are currently among the most widely used machine learning methods; they are compelling, and they are working efficiently without heavy parameter tuning, plus they do not demand to scale the data. In python, we can execute a module named `RandomForestClassifier()`, which is involved in scikit-learn library, and it implements the building of the random forest algorithm. In order to implement a random forest model, the number of trees to build indubitably must be defined; the `n_estimators` parameter implements this procedure. These trees will be built entirely independently from each other, and the algorithm will make different random choices for each tree to make sure the trees are distinct. In our case, we defined the `n_estimators` parameter to be one hundred -100-. Thus, the above algorithm was implemented by building ten -10- forests (according to 10 stratified k-fold cross-validation method) with each of them involving one hundred -100- individual trees. As we already mentioned in logistic regression implementation, we executed random forests for four -4- times for the different approaches concerning the amount of data.

- **Extra Trees Classifier:**

Extra Trees Classifier is an ensemble learning method which is based on decision trees; the algorithm name originates from Extremely Randomized Trees; Extra Trees Classifier, just like the Random Forest logic, randomizes individual decisions and subsets of data to minimize over-learning from the data and overfitting. The Extra Trees classifier model performs similarly to the Random Forest with a kind of difference that chooses the cut-off spots ultimately in random independently of the target variable; At each specific tree node, a random selection of a certain number of attributes among which the best of them is determined. In the extreme case, the method picks a single attribute as cut-point randomly at each node and hence builds completely randomized trees whose structures are autonomous of the target variable values of the learning sample. Another main difference with random forest is the measurement of variance; in machine learning terms, variance is a measure of how much the prediction would change if the classifier's training was implemented with different data. Thus, Random Forests ascribe medium variance and Extra Trees low variance.

For the given data, we used via python's scikit-learn library -similarly-, the module which initializes the Extra Trees Classifier model and it represented as `ExtraTreesClassifier()`; we chose to define one hundred -100- `n_estimators` parameter just like Random Forest model and letting the rest of the algorithm's parameter with its default values. The algorithm tested -similarly- for the four cases of data manipulation with the stratified 10-fold cross-validation method.

- **Artificial Neural Networks:**

Machine learning is controvertibly a semantic component of the Artificial Intelligence family; also, Deep Learning is. Deep Learning constitutes the fast "learning" similar to the human brain; it

encourages scalability in the training phase and excellent performance even in more complicated problems, including image and speech recognition. The Deep Learning is mainly based on Neural Networks, which adopted their philosophy from the human neurons. Due to the fact that the Deep Learning Chapter is too large to be discussed, we will be concentrated on ANN for structured data and especially how their architectures can be combined with our given data. In particular, we performed the well-known architecture for handling structured data called Multilayer Perceptron – MLP- but implemented with the high-level API named Keras.

### **MLP implemented with the Back-propagation Algorithm : A classical approach**

Concerning the architecture, an MLP is composed of one input layer, one or more internal layers named hidden layers, and the final layer named the output layer, which defines the outcome. Every layer, except the output layer, introduces a bias neuron and is fully connected to the next layer.

An efficient way for an MLP to be trained is the backpropagation algorithm; it is just a Gradient Decent procedure trying to achieve optimization as it concerns the prediction's error reduction. The backpropagation algorithm is capable of computing the gradient of the network's error with regards to every single model parameter. In other words, the backpropagation can find out how each connection weight and each bias term should be adjusted in order to reduce the error. Once it has these gradients, it just performs a regular Gradient Descent step, and the whole process is iterated until the network converges to the solution.

In more detail, the algorithm operates in subsamples of the whole dataset named mini-batches, it manipulates one mini-batch at a time, and it goes through the full training set multiple times; each pass of the dataset is called an epoch (one -1- epoch).

The skeleton of the steps are introduced: a) Each mini-batch is transmitted to the network's input layer and then sends it to the first hidden layer, b) The algorithm then computes -through the activation function- the output of all the neurons in this layer (for every instance in the mini-batch). Due to the fact that we desire the MLP to learn non-linear decision boundaries, we need to introduce non-linearity into the network. We achieve the task as mentioned above by introducing non-linearity, adding an activation function. Several kinds of activation function which can be used, but usually we use Rectified Linear Units -ReLU- which is one of the most widespread activation functions; ReLU function is a simple function which is zero for any input value under zero and the same value for values greater than zero, c) the procedure is continuing by passing the result to the next layer, its output is calculated and passed to the next layer, and so forth till we receive the outcome of the last layer, the output layer. The procedure mentioned above constitutes the forward pass; it is precisely

like making predictions, all the results are preserved – except the intermediates - since they are essential for the backward pass, d) As a next step, the algorithm estimates the network's output error; in particular, it operates with a loss function (MSE, categorical cross-entropy, binary cross-entropy) that compares the aspired output and the calculated output of the network and returns some measure

of the error; e) in the following step, it computes how much each output connection contributed to the error. The task mentioned above is done analytically by merely applying the chain rule, which makes this step fast and precise, e) Then the algorithm estimates how much of those error contributions grew from each connection in the previous layer, using the chain rule again—and so forth until the algorithm reaches the input layer. As we explained first, this reverse pass efficiently measures the error gradient across all the connection weights in the network by propagating the error gradient backward through the network, f) eventually, the algorithm performs the Gradient Descent procedure to tweak all the connection weights in the network, using the error gradients which are just computed. As a summary, for each training instance the backpropagation algorithm first makes a prediction (the forward pass procedure), measures the error, then goes through each layer in reverse to measure the error contribution from each connection (reverse pass), and finally slightly adjusts the connection weights to reduce the error.

MLP is appropriate for binary classification problems, like our case study; the MLP needs a single output neuron using the activation function: the output is the estimated probability, which is a number between 0 and 1.

### **Keras API contribution for building our MLP**

For our classification problem, we constructed an MPL with the support of Keras; Keras is a high-level Deep Learning API that allows users to efficiently build, train, evaluate and perform all sorts of neural networks. The model type that we used was `Sequential()`<sup>32</sup>. `Sequential` is the most comfortable way to build a model in Keras. It allows the user to build a model layer by layer. Mainly, each layer has weights that correspond to the layer the follows it; then, we used the `add()`<sup>33</sup> function in order to add layers to the constructed model. The function mentioned above comes with the parameter ‘`Dense`’<sup>34</sup>. `Dense` is a typical layer type that operates in most cases. In a dense layer, all nodes in the previous layer connect to the nodes in the current layer. The `add()` function accepts various parameters, which will be explicitly discussed in our case study. Below we introduce the constructed ANN architecture to approach our card fraud classification problem:

---

<sup>32</sup> <https://keras.io/models/sequential/>

<sup>33</sup> [https://keras.io/layers/merge/#add\\_1](https://keras.io/layers/merge/#add_1)

<sup>34</sup> <https://keras.io/layers/core/#dense>

**Table 15 Implemented ANN architecture****Code Snippet for the basic structure of the ANN (Python 3.7)**

```

#build the model
model = Sequential ()
model.add (Dense(500, input dim=n, activation='relu'))
model.add (Dense(256, activation='relu'))
model.add (Dropout(0.3))
model.add(Dense(256, activation='relu'))
model.add(Dropout (0.3))
model.add(Dense (1, activation='sigmoid'))
# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam')
# Fit the model
model.fit(train_features[train], train_labels[train], epochs=90, batch_size=900, verbose=1)

```

According to the above snippet, we built an ANN model with three -3- core layers and one -1- output label, which defines the final result. Concerning the first three -3- layers, we choose with trial and error method to define five hundred -500- units in the first layer and two hundred and fifty-six -256- units in the rest two layers; By Increasing the number of nodes in each layer, the model capacity is increasing. Furthermore, we used as an activation function for the first three -3- layers the ReLu function to take into account nonlinear relationships for our model, and also we used the Dropout() function that supports the overfitting's reduction. Dropout<sup>35</sup> consists of randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting. By setting the hyperparameter of the Dropout () function to 0.3, we are dropping randomly out 30%of nodes during training. Concerning the output layer, we used the Sigmoid<sup>36</sup> activation function because it is more suitable for our binary classification problem; the sigmoid function takes any range real number and returns the output value (probability), which falls in the range of 0 to 1.

After a model is created, we must call its compile()<sup>37</sup> method to specify the loss function and the optimizer to use. Compiling the model receives two -2- parameters: “optimizer” and “loss”. The optimizer controls the learning rate. We used ‘Adam’<sup>38</sup> to be our optimizer because it is generally a good optimizer to use for many cases. The Adam optimizer adjusts the learning rate throughout the training. As the loss function, we decided to use the binary cross-entropy to calculate the loss due to the fact that we had to deal with a binary classification problem and the labels corresponding to binary values (legitimate=0, fraud=1).

<sup>35</sup> <https://keras.io/layers/core/>

<sup>36</sup> <https://keras.io/activations/#sigmoid>

<sup>37</sup> <https://keras.io/models/model/#compile>

<sup>38</sup> <https://keras.io/optimizers/#adam>

After the definition of the above functions and their parameters, the model is ready to be trained; also it is important to mention that the data must be normalized in order to achieve better performance; For this task, we merely need to call its `fit()`<sup>39</sup> method. We insert it the input features and the target classes, as well as the number of epochs to train which we chose to be ninety -90- i.e., the algorithm sees the entire train data for ninety -90- times, also we defined the size of each mini-batch to be nine hundred -900- i.e., the model takes sub-samples of nine hundred -900- observations for its training.

The training was conducted for -4- times, each of them for a different data manipulation case, as it was implemented for the already mentioned ML algorithms. Its score and evaluation metrics are represented in the next chapter, along with the performance of the rest of the chosen ML algorithms.

---

<sup>39</sup> <https://keras.io/models/model/#fit>

### 3.4 Online Phase

The preceding procedure constitutes the entire business logic that we developed in order to build a system for detecting e-commerce fraud transactions according to the data provided by Vesta Systems Company. The second scope of this thesis was to make the entire procedure as a serverless application by integrating the strategy mentioned above into the Apache Openwhisk serverless platform building a pipeline/chain of functions as a service that receives the data, makes a transformation of them according to our implemented source code and finally, call the best model with the most significant results to make the prediction if the new unseen data (provided by the Vesta System) constitutes a fraud transaction by printing the result as a message in JSON format. For this procedure, we created three -3- connected functions with each of them to execute a specific pre-defined task that we chose and analytically will be discussed one by one in the mentioned section; also, we composed all of the functions in python programming language just like our implemented source code in the Offline phase, and of course, we point out the decisive role of docker containers that semantically contributed in the implementation of each custom-made function.

#### 3.4.1 Fundamental instructions-guidance for Openwhisk functions

Below we present you the basic instruction that we followed in order to integrate the above-mentioned logic into Openwhisk:

- As we already mentioned in the previous chapter, Openwhisk receives the input data in a JSON format. Mainly, the function accepts a dictionary as input and produces a dictionary as output. The input and output dictionaries are key-value pairs, where the key is a string, and the value is any valid JSON value. The dictionaries are canonically represented as JSON objects when interfacing to action via the REST API or the wsk CLI. Furthermore, the function must be called main or otherwise must be explicitly exported to identify it as the entry point. The mechanics may vary depending on the choice of language, but in general, the entry point can be specified using the --main flag when using the wsk CLI. Thus, all of the constructed functions, in our case study, were named `__main__.py`.
- Concerning the parameters passing, actions may receive parameters as input, and the wsk CLI makes it convenient to pass parameters to the actions from the command line. Briefly, this is done with the flag --param key-value where the key is the property name, and the value is any valid JSON value. Thus, the new unseen data was provided once in the first action but not via CLI but via the constructed REST API since the serverless application was constructed to be used from users, not only developers and Openwhisk's handlers.

- Also, all the functions were accompanied by a docker container, which includes fundamental machine learning libraries that are not including in the pre-fixed Openwhisk's python runtime (libraries like scikit-learn, NumPy, Pandas, and Keras). The mentioned docker containers contributed to the function construction because of the code and dependencies limitation (48MB).
- Furthermore, the second and the third constructed functions were included in zip files, which contain fundamental files in order to be operational.
- After the construction of all functions, we created the sequence (chain) of them in order to be connected as an entire serverless microservice.
- In addition, first invocations of functions have a cold start latency. That is the effect of the very first invocation of an action taking a bit long.
- The last operation was to create, through Openwhisk native commands- a REST API for providing the new data (parameters).

### 3.4.2 Defining the usage of supported Openwhisk Docker containers

Due to the fact that our entire service needed Machine Learning libraries to work, we tested if several of these are exceed the maximum limit that Openwhisk indicates. Notably, we ascertained that almost all the Machine Learning libraries use numerous shared libraries and compile native dependencies for performance, which can lead to hundreds of megabytes of dependencies. Indeed, we find that for example, that the pandas library needs at least 84M of dependencies. In order to overcome this problem, we made use of a fundamental benefit that Openwhisk provides; constructed custom-made runtime containers. Simply that means that the user can build his own custom-made docker container and integrate it along with the constructed function without affecting the limitation of the size. However, the construction of a suitable docker container for Openwhisk usage must follow a specific process; Custom runtime images must implement the Action interface; this is the protocol used by the platform to pass invocation requests to the runtime containers. Containers are expected to expose an HTTP server (running on port 8080) with /init and /run endpoints. In our case, we used pre-defined Docker containers which are made by other Openwhisk users; practically the entire procedure for a docker container creation is the below:

The first operation is the creation of a directory in our OS, which will include all the fundamental files needed for the construction of the appropriate docker container. Thus, inside the new directory, we create a new Dockerfile<sup>40</sup> which installs additional packages into the OpenWhisk Python runtime. Below we present a typical Dockerfile structure for the construction of an Openwhisk container:

---

<sup>40</sup> A **Dockerfile** is a text document that contains all the commands a user could call on the command line to assemble an image.



Table 16 Dockerfile contents

Dockerfile construction <sup>41</sup>	
1	<b>FROM</b> openwhisk/python3action
2	<b>RUN</b> echo "@community http://dl-4.alpinelinux.org/alpine/edge/community" >> /etc/apk/repositories
3	<b>RUN</b> apk add --no-cache \ # add package build dependencies
4	g++ \
5	lapack-dev \
6	gfortran
7	<b>RUN</b> pip install \ # add python packages
8	numpy pandas scipy sklearn

A valid Dockerfile must start with a **FROM** instruction. The **FROM**<sup>42</sup> native command initializes a new build stage and sets the Base image for subsequent instructions. The new docker image inherits features from the Base image (in our case openwhisk/python3action). The image can be any valid image – a base image can be pulled from the Docker Hub public repository.

The **RUN**<sup>43</sup> instruction will execute any commands in a new layer on top of the current image and commit the results. The resulting committed image will be used for the next step in the Dockerfile. Thus, with the RUN command we can install the chosen libraries along with their dependencies.

After the Dockerfile has been defined, we are running (in our OS terminal) the Docker **BUILD**<sup>44</sup> instruction to create a new image with these extra dependencies.

It is essential to mention that custom runtime images -for Openwhisk usage- must be available on Docker Hub. Docker Hub is the only container registry currently supported; this means all custom runtime images will need to be publicly available. Hence, we must execute specific commands to integrate the fixed docker container to the Docker Hub platform to be ready for Openwhisk usage. After the Docker's building, with the **TAG**<sup>45</sup> instruction - command in terminal- we can target the specific source image, and then with the **PUSH**<sup>46</sup> docker command we make it available in the Docker Hub repository ready for usage. When the docker image has been published, is available to be used in Openwhisk.

The above procedure constitutes the typical process to create and use a docker custom-made image for Openwhisk operations. In our case, we used prefixed docker images available in Docker

<sup>41</sup> <http://jamesthom.as/blog/2017/08/04/large-applications-on-openwhisk/>

<sup>42</sup> <https://docs.docker.com/engine/reference/builder/#from>

<sup>43</sup> <https://docs.docker.com/engine/reference/builder/#run>

<sup>44</sup> <https://docs.docker.com/engine/reference/commandline/build/>

<sup>45</sup> <https://docs.docker.com/engine/reference/commandline/tag/>

<sup>46</sup> <https://docs.docker.com/engine/reference/commandline/push/>

Hub/ Openwhisk branch<sup>47</sup> with some custom-made configurations concerned the versions of machine learning libraries. To integrate the docker images into Openwhisk, we use -for every action construction- (in Openwhisk CLI) the flag parameter **-docker** which is accompanied with the name of the specific docker image that we want to use.

### 3.4.3 Openwhisk functions (actions)

After the detailed explanation concerning the significant usage of Docker container, we are introducing the three constructed functions to build our fraud alert system inside Openwhisk implemented with the support of data analysis tools and ML algorithms:

#### **First Openwhisk action:**

The first Openwhisk action implements the feature engineering and the missing data imputation procedures (from the Offline phase). As we already said, the function reads the new parameters like one value from a key-value pair; thus, we split the value, i.e., the new fifty-three -53- chosen by the PCA method features that constitute the transaction and we fed them into a data frame. In this phase, it is crucial to mention that we used the python's third-party package Pandas built as a docker container, which was pulled by the Openwhisk Docker repository because it is the library that supports the data frame creation. By putting the features in a data frame, the next step was to replace the blank features with the nan value in order to have an overview of our data; then according to the feature engineering procedure, we inserted the new features concerning the P\_emaildomain and R\_emaildomain, i.e., the P\_emaildomain\_bin, R\_emaildomain\_bin, P\_emaildomain\_suffix, and R\_emaildomain\_suffix as well as we did in the Offline phase. Next, we defined the categorical and the numerical values as well as we did in the Offline phase similarly, and we imputed the nan values according to our data imputation strategy, i.e., we filled the numerical features with the most frequent value and the categorical features with the string "unknown". Then we had to define, again, the first's action output by converting it in a key-pair value because Openwhisk receives data in a dictionary and produces a result similarly in the same format. After the development of the above-mentioned source code (in python), we save it in a file named. `__main__.py`. Then, in the directory which the source code was saved, we executed -in our OS terminal- the above native command to build the first function:

**Table 17 First action construction**

#### **Command for the first function construction**

```
wsk -i action create first __main__.py --docker jamesthomas/openwhisk_python_ml --web true
```

<sup>47</sup> <https://hub.docker.com/u/openwhisk/>

As we already mentioned in the previous chapter, wsk is the acronym of Openwhisk which establishes the command to interact with the Openwhisk serverless platform; action and create are native commands for the action creation accompanied by the given action's name, the flag `--docker` determines the specific docker container that will contribute to the function implementation and lastly the `--web true` flag makes the action accessible to third users without requesting authentication.

### **Second Openwhisk action:**

The second Openwhisk action implements the label encoding and data normalization procedures. The construction of the mentioned action is a little different from the first because it was implemented as a zip file. Concerning the source code, we inserted the features -coming from the first's action result- in a data frame, then we used the dictionary that we developed in the Offline phase, which includes the label encoding features, and we mapped it in the new features. Next, we implement the normalization process; for that reason, we applied the exact same scaling as for the training data in the Offline phase; hence we created -in the Offline phase- a binary file including the average and the standard deviation from entire features; when we bundle an action file with some extra files that we need, we deploy them together as a zip file. Thus, we created a zip file which contained the `__main__.py` source code as the second function and the binary file, which implements the normalization process. Same as the first action, the output must correspond to a dictionary format. As it concerns the action creation, we produced the above-mentioned zip file which included the binary file named `scalar.sav` to do the normalization procedure and the `__main__.py` source code. Then, we open a terminal running bash and we executed the specific command for the zip creation which was "`zip second.zip __main__.py scalar.sav`". Then, and we executed the below command in order to build the second Openwhisk action:

**Table 18 Second action construction**

#### **Command for the second function construction**

```
wsk -i action create second second.zip --docker jamesthomas/openwhisk_python_ml --web true
```

As we observe, we used again the specific image which includes ML libraries and especially the Pandas library due to the fact that we used again the data frame structure.

### **Third Openwhisk action:**

The third and final action includes the chosen saved machine learning model, which achieved the best performance, and it was designed to predict if the new unseen transaction features concern a fraud or a legitimate e-commerce transaction. The following chapter introduces the evaluation metrics for each one of the above ML models -for each data case- that we experimented in our given data, but

in this chapter, we can briefly mention that the best model was an ANN by applying weight in the minority class. The mentioned model named “model\_ANN\_weight.h5” and it was stored in an h5<sup>48</sup> format, and it was included in a zip file since it was necessary in order to implement the third function, which gives the final prediction. The third action produces the message that informs the user for the transaction status.

Concerning the docker image, we needed a docker image, which includes the Keras ML library, because it is essential to import it for the third action implementation. The docker image that we used in the preceding two functions did not include the Keras Library; for that reason, we used another already-made image constructed by the Openwhisk developers, which included, among other ML libraries, the Keras framework (version 2.2). The mentioned docker image can be found in the official Docker Hub repository as well as in the Openwhisk’s official account in GitHub. The mentioned image included, except other fundamental files, a text file named requirements.txt with the versions of the third-party packages as well as Keras version 2.2. However, in the Offline phase, we developed our model with a later version of Keras framework (v. 2.4) and in order to avoid any compatibility issues we re-configured the version of Keras from 2.2 to 2.4 inside the requirements.txt and we built, tag and pushed again the docker container in the Docker Hub in order to be available for usage. Then, we created the appropriate source code for the third function named similarly `__main__.py` for predicting the transaction and next, in a terminal running bash we executed the specific command for the zip creation which was “**zip third.zip \_\_main\_\_.py model\_ANN\_weight.h5**”. Then, in the directory which the source code was saved, we executed `-in` our OS terminal- the above native command to build the third and final function:

**Table 19 Third action construction**

<b>Command for the third function construction</b>
<b>wsk -i action create third third.zip --docker pepi1989/openwhisk_python3aiactions --web true</b>

### Action chaining

Actions can be combined in many ways. The simplest way is chaining them into sequences. Chained actions use as input the output of the previous actions. Of course, the first action of a sequence will receive the parameters (in JSON format), and the last action of the sequence will produce the final result as a JSON string. According to the above information, we chained the above-mentioned constructed functions to develop an entire serverless application, which in the first two actions completes the preprocessing of the new data and, in the third one, performs the final prediction.

---

<sup>48</sup> An **H5** file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

Openwhisk provides the ability to build a pipeline of actions using -again- native commands. Therefore, in order to connect the above three -3- actions, we used the below command:

**Table 20 Sequence of functions construction**

**Command for the pipeline construction**

```
wsk -i action create sequence--sequence first,second,third --web true
```

Notably, the commands for the sequence construction are not much different from the preceding commands except the global flag `--sequence`, which defines the development of the pipeline of Openwhisk actions.

### 3.4.4 Using REST-API call with Openwhisk

Since the OpenWhisk environment is enabled, we can use OpenWhisk with our implemented application with a REST API call. Openwhisk implements the construction of REST API by using -again- native commands, which allows to associate it with the chosen actions. To create a REST API for our service we used the above command:

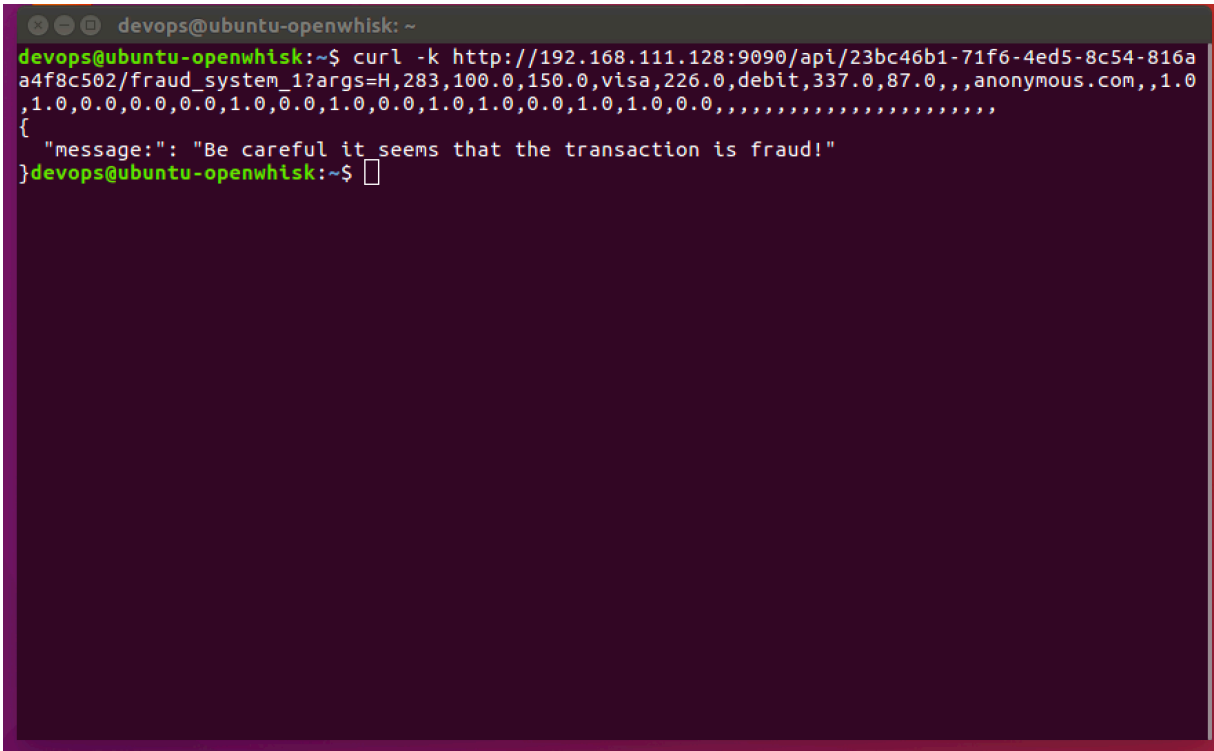
**Table 21 REST API construction**

**Command for the REST-API construction**

```
wsk -i api create /fraud_system get sequence --response-type json
```

As we can observe, we use -again- the `wsk` acronym; `create` and `get` are native commands for the REST-API creation accompanied by the given sequence name, the flag `--response-type` determines the output format. When we execute the command as mentioned above, Openwhisk produces a unique REST-API for our service.

After executing the above relevant commands, we can test our service by feeding the implemented REST-API with the new data:



```
devops@ubuntu-openwhisk: ~  
devops@ubuntu-openwhisk:~$ curl -k http://192.168.111.128:9090/api/23bc46b1-71f6-4ed5-8c54-816a  
a4f8c502/fraud_system_1?args=H,283,100.0,150.0,visa,226.0,debit,337.0,87.0,,,anonymous.com,,1.0  
,1.0,0.0,0.0,0.0,1.0,0.0,1.0,0.0,1.0,1.0,0.0,1.0,1.0,0.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,  
{  
  "message": "Be careful it seems that the transaction is fraud!"  
}  
devops@ubuntu-openwhisk:~$
```

**Figure 23 Testing Openwhisk constructed pipeline via the REST API call**

By using the curl<sup>49</sup> command tool, we call our REST-API along with the new transactional data; Then the REST-API calls the constructed pipeline of actions along with their connected dockers, feeds them with the associated data and produces the result which comes progressively from the execution of the action.

The next chapter drives to the explanation of the results and the evaluation performance concerning the chosen ML models as well as the performance of the Openwhisk’s sequence execution. Concerning the Offline phase, we annotate performance metrics i.e., recall, confusion matrix, and AUC, which constitute crucial factors for choosing the best model to be used as the leading prediction model for Openwhisk service. Concerning the Online phase, we evaluate the time-based performance of our constructed service.

---

<sup>49</sup> **cURL**, often just “**curl**,” is a free command line tool. It uses URL syntax to transfer data to and from servers.

# Chapter 4

---

## Evaluation Performance

---

### 4.1 Overview

Since in the preceding chapter, we described the implementation of the Offline and Online phases that combine the fraud alert system; in this section, we evaluate the tested ML models performance used in the Offline Phase by applying well-known measures which will be discussed in detail. From the other side, concerning the Offline phase, we evaluate the execution time of the constructed pipeline when a request is initialized.

In the Machine Learning area, the most common metric to evaluate the classification performance of our constructed model is Accuracy, which is the most intuitive performance measure, and it is calculated as the ratio of correctly predicted observation to the total observations. However, this is not the only way to summarize how well a supervised model performs on a given dataset. In practice, this Accuracy might not be the appropriate measure for the constructed application, and it is essential to choose the right metric when we are selecting between models and adjusting parameters. When we are selecting a metric, we should always consider the end goal of the machine learning application in mind; we are usually interested not just in making accurate predictions, but in using these predictions as a portion of a larger decision-making process. Before picking a machine learning metric, we should think about the high-level goal of the application. In our case study and due to the imbalanced dataset problem, we used measures like recall, AUC, and Confusion Matrix to evaluate the chosen Models. From the other side, as we already referred in a previous chapter, an Openwhisk function's execution time is the lesser of sixty -70- seconds (with configuration); thus, we tested the constructed pipeline by making ten -10- requests and we kept execution times to evaluate the time reaction.

## 4.2 Offline Phase evaluation performance

### 4.2.1 Evaluation measures description

In the initial stages of the development, it is often almost impossible to place models into production just for testing purposes without evaluating them, because of the great business or personal risks that can be involved. A non-evaluated model in production can cause disastrous consequences. Thereby, we always use metrics that can help us define the final model for production. Concerning the classification problems, we introduce the most well-known measures (metrics), which influence the efficiency of the chosen ML model:

- **Accuracy:** reflects the number of correct predictions made by the model over all kinds predictions made. Accuracy is the ratio of number of correct predictions to the total number of input samples, and it works efficiently only if there are equal number of samples belonging to each class.
- **Precision:** Usually named specificity, it defines the number of correct positive results divided by the number of positive results predicted by the classifier. Precision measures the percentage of the samples which are classified to one class and were correctly classified.
- **Recall:** Usually named sensitivity, it defines the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). Mainly, recall measures the percentage of how many samples found correctly classified.
- **F1-Score:** F1 Score defines the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. F1-score attributes how precise the chosen classifier is (i.e how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances). Mainly, F1-Score tries to find the balance between precision and recall.
- **Confusion Matrix:** Confusion Matrix as the name suggests gives us a matrix as output and describes the complete performance of the model containing information about the actual classifications and predicted classifications done by the classifier. In binary classification problems the CM consist of four -4- terms: a) True Positives -TP-, which defines the case we predicted as YES and the actual output was also YES, b) True Negatives -TN-, which defines the case we predicted we predicted NO and the actual output was NO, c) False Positive -FP-, which defines the case we predicted YES and the actual output was NO and d) False Negative -FN-, which defines the case we predicted NO and the actual output was YES. Thus, with the support of the above-mentioned tasks, a CM is a 2D table giving a completely overview of the model's performance.
- **AUC-ROC-Curve:** it is a curve that indicates a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represent degree or measure of separability. It tells how much model is capable of distinguishing between classes.



Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. By analogy, Higher the AUC, better the model is at distinguishing between two classes. An excellent model has AUC near to the 1 which means it has good measure of separability. A poor model has AUC near to the 0 which means it has worst measure of separability. In fact, it means it is reciprocating the result. It is predicting 0s as 1s and 1s as 0s. And when AUC is 0.5, it means model has no class separation capacity whatsoever.

#### **4.2.2 Chosen evaluation measures for the fraud detection classifier**

As we already referred, usually, we evaluate the performance of each chosen model with the accuracy measure; however, it is not always the appropriate measure to summarize how well a supervised model performs on a given dataset. Classification concerning data with imbalanced class distribution has confronted an essential disadvantage of the performance attainable by most standard classifier learning algorithms, which assume a relatively balanced class distribution and equal misclassification costs. In our case in which we deal with an imbalanced dataset

In this thesis, the purpose is to choose the most suitable model, which exposes the fraud sharply from legitimate e-commerce transactions. Thus, we must select that model which provides the greatest separability in transactions. Types of errors play an important role when one of the two classes are much more frequent than the other one as our case. Thus, we have an imbalanced dataset containing 3% of a minority class (fraud transactions) and 97% of the majority class (legitimate transactions), the algorithm can predict almost all cases as belonging to the majority class. The accuracy score of this algorithm will yield an accuracy of 99%, which seems impressive, but in reality, it is not. The minority class is ignored in that case, and the model is a total failure.

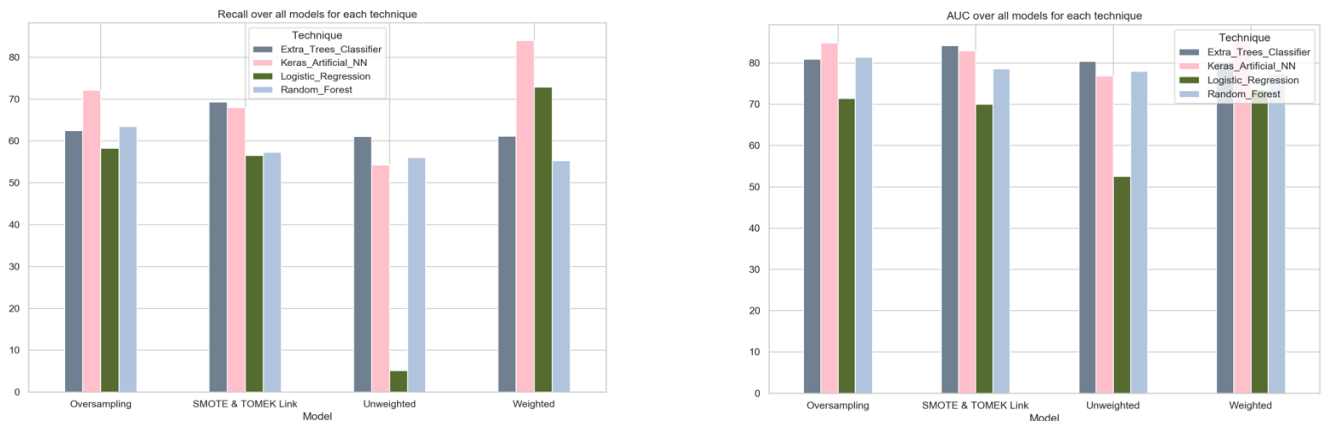
Thereby, we decided to use metrics like recall, Confusion Matrix, and AUC providing a more reliable overview of the performance.

Below, we present the calculated evaluation metrics for each chosen model with each one of the cases that concern the data manipulation that we discussed in the previous chapter and we chose the best model according to the highest percentage of the measures mentioned above, also we quote the calculated confusion matrices for each of the models.

Table 22 Evaluation Measures results

Chosen Model performance with testing data (Training with Stratify 10-Fold C.V)	Recall	AUC
Logistic Regression ( <i>raw dataset</i> )	5,13%	52.54 %
Logistic Regression ( <i>with applying weight to minority class samples</i> )	72.85 %	73.42 %
Logistic Regression ( <i>with oversampling 40% minority class</i> )	58.24 %	71.43 %
Logistic Regression ( <i>by applying SMOTE and Tomek Link technique 40% minority class</i> )	56.52 %	69.99 %
Random Forest ( <i>raw dataset</i> )	56%	78%
Random Forest ( <i>with applying weight to minority class samples</i> )	55.26 %	77.39 %
Random Forest ( <i>with oversampling 40% minority class</i> ) *Overfitting	63.46 %	81.37 %
Random Forest ( <i>by applying SMOTE and Tomek Link technique 40% minority class</i> )	57.32 %	78.56 %
Extra Trees Classifier ( <i>raw dataset</i> )	61.07 %	80.42 %
Extra Trees Classifier ( <i>with applying weight to minority class samples</i> )	61.17 %	80.23 %
Extra Trees Classifier ( <i>with oversampling 40% minority class</i> ) *Overfitting	62.47 %	80.89 %
Extra Trees Classifier ( <i>by applying SMOTE and Tomek Link technique 40% minority class</i> ) *Overfitting	69.32 %	84.18 %
Keras ANN ( <i>raw dataset</i> )	54.25 %	76.9 %
<b>Keras ANN (<i>with applying weight to minority class samples</i>)</b>	<b>84%</b>	<b>85%</b>
Keras ANN ( <i>with oversampling 40% minority class</i> )	72.1 %	84.84 %
Keras ANN ( <i>by applying SMOTE and Tomek Link technique 40% minority class</i> )	67.99 %	82.97 %

According to the table mentioned above, we identified some cases which, in the training phase, revealed the overfitting phenomenon, such as the Random Forest by applying the technique oversampling, Extra Trees Classifier similarly by using the method oversampling and Extra Trees Classifier by applying SMOTE and Tomek Link method, which practically means that the models had over-trained with the training dataset but failed to predict the new unseen data giving low rates on the collected measures.



**Figure 24 Barplots depicting Recall and AUC measures for the tested ML models**

The previous table, accompanying with the above barplot (which present the recall and AUC measures for each of the models - for each data case-) heads to the result that the ANN by applying weight in minority class achieved the most high-grade performance by finding the 85% of the test observations correct. Furthermore, the same model achieved the highest percentage concerning the AUC Measure. Nevertheless, in our experiments, we recognized that the model mentioned above suffers from low precision by having a rate near 20%; this low rate implies that the constructed classifier sometimes classifies the legitimate transactions wrong, but we do not consider much about this, because it does not cost too much to examine again a valid e-commerce transaction which is classified as fraud in contrast with an actual fraud which classified as legitimate. Thereby, we chose the specific model because we do not mind sacrificing some legitimate test observations in order to have efficient separability in fraud and legitimate transactions. Below we perform the confusion matrices for each one of the models depicting how many of the test observations correctly classified. Due to the fact that we defined the fraud transactions as positive examples (isFraud=1) and legitimate transactions as negative (isFraud=0), the confusion matrix has in its upper left-side position the True Positive -TP- samples, in its upper right-side position the False Positive -FP- samples, in its bottom left hand side the false negative - FN - examples and in its bottom right hand side the true negative - TN - samples. The diagonal entries represent correctly classified samples and the rest represent misclassifications. This enables us to visualize prediction results and related derived statistics in a class-wise manner from the data.

**Table 23 Confusion Matrix for Logistic Regression – all cases**

Logistic Regression (raw data)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>212</b>	<b>3921</b>
	Legitimate	<b>64</b>	<b>113911</b>
Logistic Regression (weight to minority class)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>3011</b>	<b>1112</b>
	Legitimate	<b>29658</b>	<b>84317</b>
Logistic Regression (random oversampling)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>2407</b>	<b>1726</b>
	Legitimate	<b>17531</b>	<b>96444</b>
Logistic Regression (SMOTE and TOMER-LINK)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>2336</b>	<b>1797</b>
	Legitimate	<b>18845</b>	<b>95130</b>

**Table 24 Confusion Matrix for Random Forest – all cases**

Random Forest (raw data)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>2335</b>	<b>1798</b>
	Legitimate	<b>194</b>	<b>113781</b>
Random Forest (weight to minority class)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>2284</b>	<b>1849</b>
	Legitimate	<b>551</b>	<b>113424</b>
Random Forest (random oversampling)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>2623</b>	<b>1510</b>
	Legitimate	<b>815</b>	<b>113160</b>
Random Forest (SMOTE and TOMER-LINK)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>2369</b>	<b>1764</b>
	Legitimate	<b>219</b>	<b>113756</b>

**Table 25 Confusion Matrix ExtraTrees – all cases**

ExtraTrees (raw data)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>2524</b>	<b>1609</b>
	Legitimate	<b>264</b>	<b>113711</b>
ExtraTrees (weight to minority class)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>2528</b>	<b>1605</b>
	Legitimate	<b>813</b>	<b>113162</b>
ExtraTrees (random oversampling)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>2582</b>	<b>1551</b>
	Legitimate	<b>786</b>	<b>113189</b>
ExtraTrees (SMOTE and TOMMEK-LINK)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>2865</b>	<b>1268</b>
	Legitimate	<b>1087</b>	<b>112888</b>

**Table 26 Confusion Matrix for ANN – all cases**

ANN (raw data)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>2242</b>	<b>1891</b>
	Legitimate	<b>505</b>	<b>113470</b>
ANN (weight to minority class)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>3337</b>	<b>796</b>
	Legitimate	<b>13142</b>	<b>100833</b>
ANN (random oversampling)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>2980</b>	<b>1153</b>
	Legitimate	<b>2758</b>	<b>111217</b>
ANN (SMOTE and TOMMEK-LINK)		<i>Actual</i>	
		<i>Fraud</i>	<i>Legitimate</i>
<i>Predicted</i>	<i>Fraud</i>	<b>2810</b>	<b>1323</b>
	Legitimate	<b>2280</b>	<b>111695</b>

The selected the ANN model by applying weight to minority class samples, identified correct three thousand thirty-seven test observation -**3337**- as fraud transactions and lost seven hundred ninety-six -**796**- fraud transaction which classified as authorized. This model had the best performance from all the tests models in this thesis.

### 4.3 Online Phase

#### 4.3.1 System’s time execution performance

As the next level of the potential of cloud computing, Serverless provides significant benefits such as scalability, the pay-for-request method, smooth migration of individual features, or partial workloads to functions, neither debugging nor controlling allocation resources and beyond any doubt fast response to function requests. Regarding our thesis, we performed several tests on our constructed service to monitor the response time for each request, which is passing in the pipeline. Especially, we performed ten -10- individual requests by transferring data -concerning e-commerce transactions- to the constructed pipeline. Below are pictured plots concerning the execution time of each of the custom-made functions and the entire pipeline’s response time in order to review the overall performance of the system.

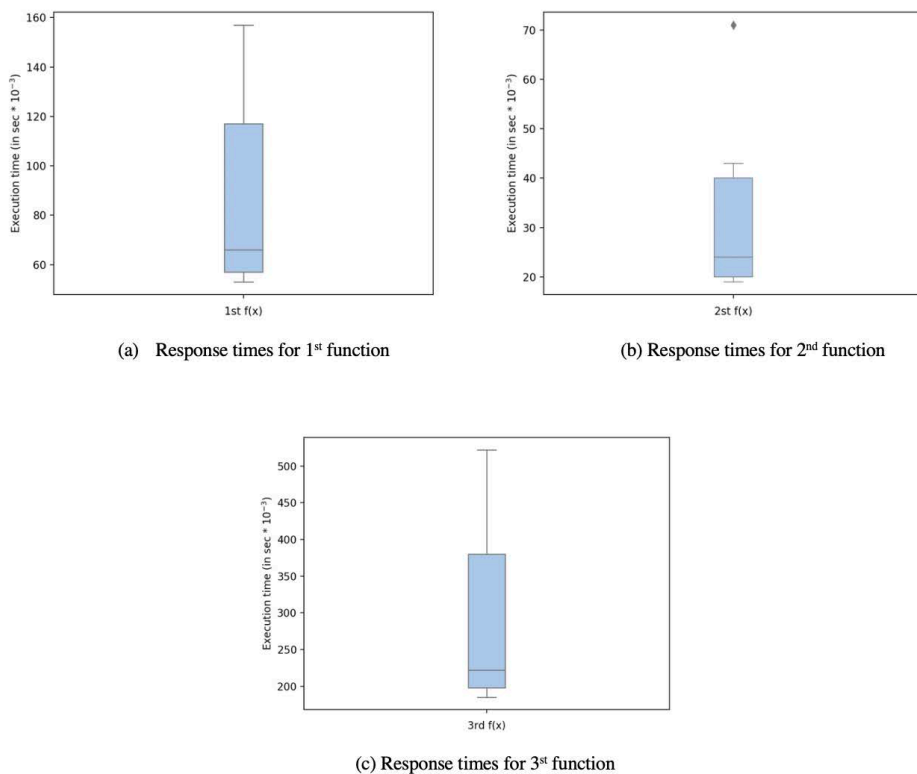
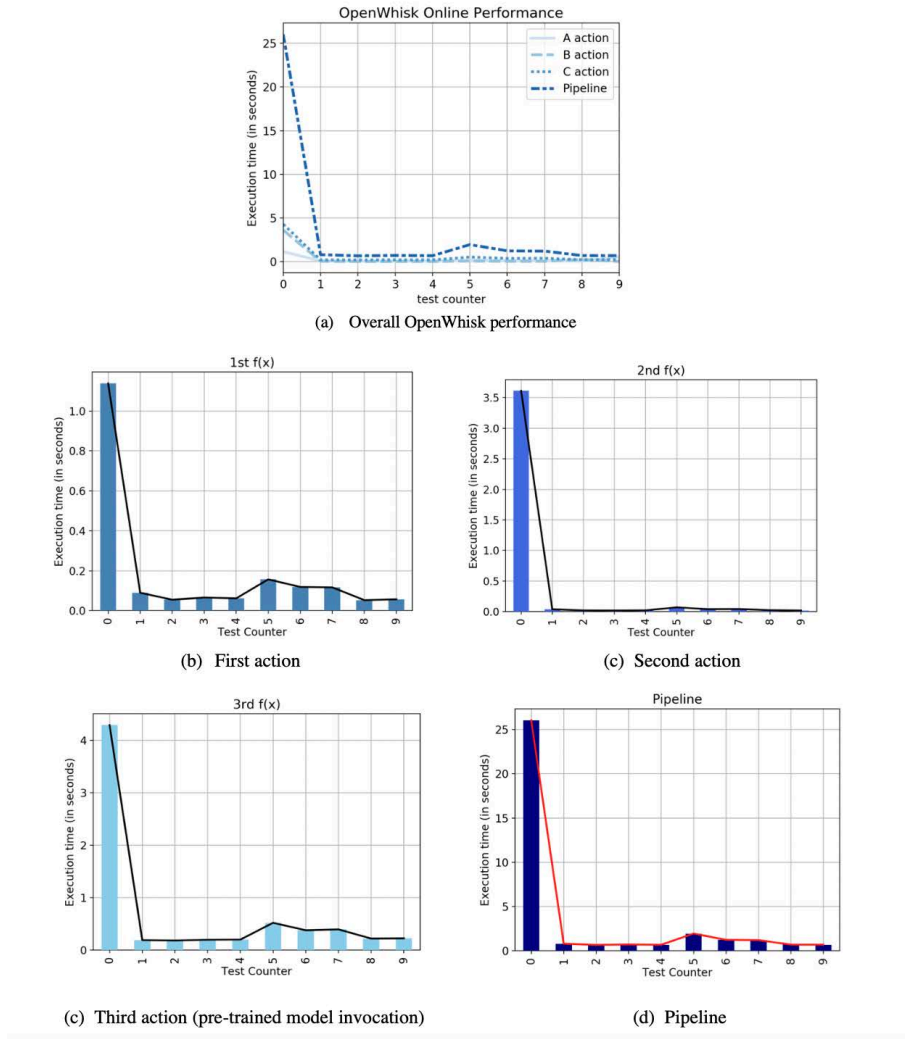


Figure 25 Boxplots for each action’s execution time (Duration in seconds-concerning only warm invocations)

We observed that each action – individually- is executing in a time range of seconds and milliseconds. For the first action the fastest execution time was 53 ms and the weakest 1.138 s. For the second function the fastest execution time was 19 ms and the weakest was 3.612 s. The third function- the one that calls the saved ML model, achieves the best execution time in 185 ms and the weakest performance in 4.291 s.



**Figure 26 Time-plots for Openwhisk actions (Duration in seconds)**

Notably, the execution time of the pipeline does not exceed twenty-seven (27) seconds, which is actually a very fast response with regards to our constructed system and concerning the complexity that provides. In addition, Openwhisk has a cold start spot; that is why the first request takes longer time to produce a response.





# Chapter 5

---

## Experience acquired from implementation

---

### 5.1 Framework compatibility problem

By closing with this performed approach, we expose the principal problem that we faced with the Openwhisk Serverless platform and the proposed solutions to overcome it. The main problem is that Openwhisk is not compatible with TensorFlow API since it cannot understand implemented ANN models with TensorFlow .h5 format.

Initially, in the Offline phase, we designed the proposed approach by using ANN models implemented with Tensorflow's latest update (version 2.0); because it is the most famous library used in production for deep learning models; also, it has a vast and impressive community. Furthermore, Tensorflow is used as a backend framework (in most cases) to the Keras API, which is more user-friendly and easier to use. Notably, Keras can perform as a standalone API, and the user can use it uniquely to construct deep learning models.

Recently<sup>50</sup>, Tensorflow's developers community produced its latest stable version ready to be used from third users (developers – data scientists). Specifically, there are multiple changes in TensorFlow 2.0 to make users more productive; they made more consistent APIs and removed the redundant ones; performed better integration with the Python runtime and refactored the modules in order users to create smaller functions. Also, the newest version supports distributed computing, enabling portions of the graph to be computed on different processes, which may be on entirely different servers. Also, this can be used to distribute computation to servers with powerful GPUs, and have other computations done on servers with more memory, and so forth.

According to the information mentioned above, we chose -initially- to construct our ANN model with Keras API and TF (v 2.0) framework as a backend. Also, in order to save time from the training phase, we selected to rent allocation resources from AWS. AWS Deep Learning AMI constitutes pre-configured environments to build deep learning applications quickly.

---

<sup>50</sup> [https://www.tensorflow.org/guide/effective\\_tf2](https://www.tensorflow.org/guide/effective_tf2)

Specifically, we chose spot AWS instance because it is cheaper than a default EC2 instance; in particular, a spot instance is an unused EC2 instance that is available for less than the On-Demand price; because Spot Instances are enabled to request unused EC2 instances at steep discounts, by lowering Amazon EC2 costs significantly. Thus, -for our goal - we selected an EC2 (p2.large ) spot instance along with a Deep Learning AMI (Ubuntu 16.04), which comes pre-built and optimized for deep learning on EC2 with NVIDIA CUDA, cuDNN, and Intel MKL-DNN, and also includes popular frameworks such as TensorFlow.

## 5.2 The proposed solutions to defeat the problem

After the determination of the most suitable ANN model, we saved it in a TF .h5 file format to deploy it into Openwhisk's third function. Therefore, we developed the source code of the third function, we integrated the saved model, as mentioned in the third chapter of this thesis, we created the sequence of functions and the REST API, and we attempted to test the constructed fraud alert system. Unfortunately, we discovered that when we tried to call the REST-API, along with the new unseen data, the entire OS was not responding. After a detailed examination of the problem, we thought the solutions mentioned below:

- To save the model in a protobuf format (.pb) since we assumed that Openwhisk does not understand tf .h5 format. The saved\_model.pb file stores the actual TensorFlow model, and a set of named signatures and variables that uses, each identifying a function that accepts tensor inputs and produces tensor outputs.
- To construct a new docker container with the latest TensorFlow image (v2.0), including Keras specific version (2.4) and use it as a support along with the implementation of the third function.
- To use AWS S3 object storage in order to save our constructed TF model and afterward, to call it inside the third Openwhisk function. *(This option constitutes a massive complexity because it demands advanced level knowledge concerning AWS S3 object storage and possibly it would acquire a hard system configuration).*
- To construct the same model using Keras model .h5 format.

We tested all the above -mentioned solutions except the third one. Unfortunately, neither the first nor the second worked with success, causing the same initial problem again. The last one was the key to implement our system effectively. Therefore, as we already referred to the third chapter of this thesis, we constructed the ANN with the support with the Keras API, and we deployed it -successfully- inside Openwhisk.

In the light of all the preceding chapters, we conclude our approach, and we are available for any contribution, optimization, or a higher-level proposal concerning the implemented fraud alert system in a serverless environment.



# Appendix\_A

---

## Installation and Configuration Apache Openwhisk Serverless Platform in a Local Machine (with Docker Compose)

---

### - Prerequisites

#### 1) Install the VMWare Fusion (our version 11.5 pro - paid)

- Download and import OS Linux Ubuntu 16.04 LTS Xenial
- Install VMWare tools
- Set VMware workstation's networking options to NAT

#### 2) Install JAVA (version 8)

- Install the "Main" repository with apt: **sudo apt-get update**
- Install OpenJDK 8: **sudo apt-get install openjdk-8-jdk**
- Verify that Java and the Java compiler have been properly installed: **java -version / javac -version**
- Set Java Home Environment: To set the variable for your system:  
**echo "JAVA\_HOME=\$(which java)" | sudo tee -a /etc/environment**
- Reload your system's environment variables: **source /etc/environment**
- Verify the variable was set correctly: **echo \$JAVA\_HOME**

#### 3) Install Docker CE Engine (Version: 19.03.1)

- Navigate to <https://docs.docker.com/install/linux/docker-ce/ubuntu/> and perform the installation steps according the chosen OS

#### 4) Install Docker Compose (version 1.24.1)

- Navigate to <https://docs.docker.com/compose/install/> and perform the installation steps

## - Main Installation steps for Openwhisk

### 1) Install the Openwhisk Serverless Platform

- Navigate to GitHub repo: (<https://github.com/apache/openwhisk/blob/master/README.md#quick-start>)
- Create a directory in the OS and perform the next command: **git clone <https://github.com/apache/incubator-openwhisk-devtools.git>**
- Execute the command: **cd incubator-openwhisk-devtools/docker-compose**
- Execute: **docker pull openwhisk/action-nodejs-v10:nightly**
- Because we used python runtime we also executed the command: **docker pull openwhisk/action-python-v10:nightly or latest**
- Then, in order to begin the full installation of Openwhisk, execute: **make quick-start**

### 2) Set and configure the Openwhisk's CLI and wsk abbreviation as a global variable

- Copy: copy in the .bashrc : **export WSK\_CONFIG\_FILE='/home/devops/incubator-openwhisk-devtools/docker-compose/.wskprops'**
- To set wsk acronym global in entire OS, navigate to .bashrc and execute the command: **export PATH="\$PATH:/home/<your host machine name>/incubator-openwhisk-devtools/docker-compose/openwhisk-src/bin/"**

---

# Bibliography

---

- [1] R. Camden, *Developing Serverless Applications: A Practical Introduction with Apache OpenWhisk*, B. Foster and V. Wilson, Eds., 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc., 2018, p. 71.
- [2] T. M. Mitchell, "Machine Learning", McGraw-Hill, 1997, p. 432.
- [3] R. Camden, *Developing Serverless Applications: A Practical Introduction with Apache OpenWhisk*, B. Foster and V. Wilson, Eds., 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc., 2018, p. 71.
- [4] A. Williams, "Guide to Cloud Native Microservices", The New Stack, 2018, p. 125.
- [5] A. Williams, "Guide to Serverless Technologies", The New Stack, 2018, p. 91.
- [6] A. Zheng and A. Casari, "Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists", 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc., 2018, p. 201.
- [7] A. C. Müller and S. Guido, "Introduction to Machine Learning with Python A Guide for Data Scientists", 1005 Gravenstein Highway North, Sebastopol, CA 95472.: Reilly Media, Inc., 2017, p. 378.
- [8] M. Harrison, "Machine Learning Pocket Reference", 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc., 2019, p. 708.
- [9] Chollet, F. (2018). *Deep learning with Python*. Manning Publications.
- [9] S. Bangera, *DevOps for serverless applications : design, deploy, and monitor your serverless applications using DevOps practices*, Packt Publishing, 2018.
- [10] P. C. Bruce and A. . G. Bruce, "Statistics for Data Scientists: 50 Essential Concepts", 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc., 2016, p. 302.
- [11] J. Cook, "Docker for Data Science", Santa Monica, California, USA: Apress Media, LLC, 2017, p. 257.

- [12] M. Sciabarrà, "Learning Apache OpenWhisk", Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc., 2019, p. 772.
- [13] S. Chaudhary, R. Buyya and G. Somani, Research Advances in Cloud Computing, Singapore: Springer Nature Singapore Pte Ltd., 2017.
- [14] M. Kleppmann, Designing Data-Intensive Applications, 1005 Gravenstein Highway North, Sebastopol, CA 95472 : O'Reilly Media, Inc., 2017.
- [15] K. Matthias and S. P. Kane, Docker: Up & Running - second edition, O'Reilly Media, Inc.: 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2018.
- [16] H. Sayers and I. M. Aidan , Docker in Practice- second edition, 20 Baldwin Road PO Box 761 Shelter Island, NY 11964: Manning Publications Co., 2019.
- [17] G. McGrath and P. R. Brenner, ""Serverless Computing: Design, Implementation, and Performance", " in IEEE 37th International Conference on Distributed Computing Systems Workshops, Atlanta, GA, USA, 2017.
- [18] G. C. Fox, V. Ishakian, V. Muthusamy, and A. Slominski, "Status of serverless computing and function-as-a-service(faaS) in industry and research," CoRR, vol. abs/1708.08028, 2017. [Online]. Available: <http://arxiv.org/abs/1708.08028>
- [19] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, and P. Suter, Serverless Computing: Current Trends and Open Problems. Singapore: Springer Singapore, 2017, pp. 1–20.
- [20] M. Kalske, M. Kalske, N. Mäkitalo and T. Mikkonen, "Challenges When Moving from Monolith to Microservice Architecture," in International Conference on Web Engineering, Spain, 2018.
- [21] P. J. Garcia-Laencina, J. ,L. Sancho-Gomez and A. R. Figueiras-Vidal, "Pattern classification with missing data: a review," *Neural Comput & Applic*, pp. 263-282, 2009.
- [22] A. Gelman and J. Hill, "Missing-data imputation," in Data Analysis Using Regression and Multilevel/Hierarchical Models, Cambridge: Cambridge University Press, 2006, pp. 529–544.
- [23] V. Ganganwar, "An overview of classification algorithms for imbalanced datasets," International Journal of Emerging Technology and Advanced Engineering, 2012.
- [24] N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of artificial intelligence research, 16, 321-357, 2002.
- [25] I. Tomek, "Two modifications of CNN," In Systems, Man, and Cybernetics, IEEE Transactions on, vol. 6, pp 769-772, 2010.



- 
- [26] P. Geurts, D. Ernst., and L. Wehenkel, "Extremely randomized trees", *Machine Learning*, 63(1), 3-42, 2006.
- [27] Breiman, "Random Forests", *Machine Learning*, 45(1), 5-32, 2001.
- [28] M. Bekkar, H. K. Djemaa and T. A. Alitouche, "Evaluation Measures for Models Assessment over Imbalanced Data Sets," *Journal of Information Engineering and Applications* , vol. 3, pp. 27-38, 2013.
- [29] H. Zhang, "RWO-Sampling: A Random Walk Over-Sampling Approach to Imbalanced Data Classification," *Information Fusion*, 2013.
- [30] V. Menon and S. Kalyani, "Structured and Unstructured Outlier Identification for Robust PCA: A Fast Parameter Free Algorithm," in *IEEE Transactions on Signal Processing*, vol. 67, no. 9, pp. 2439-2452, 1 May1, 2019.
- [31] Chiroma, Haruna & Ali, Usman & Abdulhamid, Shafi'i & AlArood, Ala & Gabralla, Lubna & Rana, Nadim & Shuib, Liyana & Hashem, Ibrahim & Dada, Emmanuel & Abubakar, Adamu & Zeki, Akram & Herawan, Tutut. (2018). Progress on Artificial Neural Networks for Big Data Analytics: A Survey. IEEE Access. PP. 1-1. 10.1109/ACCESS.2018.2880694.
- [32] G. Lahera, "Unbalanced Datasets & What To Do About Them," 22 January 2019. [Online]. Available: Unbalanced Datasets & What To Do About Them.
- [33] M. Konkiewicz , "Splitting your data to fit any machine learning model," 23 October 2019. [Online]. Available: <https://towardsdatascience.com/splitting-your-data-to-fit-any-machine-learning-model-5774473cbed2>.
- [34] W. Koehrsen , "Beyond Accuracy: Precision and Recall," 3 March 2018. [Online]. Available: <https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c> .
- [35] S. Biswas, "Importance of K-Fold Cross Validation in Machine Learning," Medium, 21 May 2019. [Online]. Available: <https://medium.com/towards-artificial-intelligence/importance-of-k-fold-cross-validation-in-machine-learning-a0d76f49493e>.
- [36] D. Berdikulov , "Dealing with Missing Data," Medium, 3 April 2019. [Online]. Available: <https://medium.com/@danberdov/dealing-with-missing-data-8b71cd819501>.
- [37] Dataman, "Using Over-Sampling Techniques for Extremely Imbalanced Data," *Towards Data Science*, 10 August 2018. [Online]. Available: <https://towardsdatascience.com/sampling-techniques-for-extremely-imbalanced-data-part-ii-over-sampling-d61b43bc4879>.
- [38] R. Agarwal, "The 5 most useful Techniques to Handle Imbalanced datasets," Medium, 20 November 2019. [Online]. Available: <https://towardsdatascience.com/the-5-most-useful-techniques-to-handle-imbalanced-datasets-6cdba096d55a>.

## Bibliography

---

- [39] G. Lahera, "Unbalanced Datasets & What To Do About Them," Medium, 22 January 2019. [Online]. Available: <https://medium.com/strands-tech-corner/unbalanced-datasets-what-to-do-144e0552d9cd>.
- [40] W. Badr, "Having an Imbalanced Dataset? Here Is How You Can Fix It.," Towards Data Science, 22 February 2019. [Online]. Available: <https://towardsdatascience.com/having-an-imbalanced-dataset-here-is-how-you-can-solve-it-1640568947eb>.
- [41] J. Brownlee, "Discover Feature Engineering, How to Engineer Features and How to Get Good at It," MachineLearningMastery, 26 September 2014. [Online]. Available: <https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/>.
- [42] D. Bourke, "A Gentle Introduction to Exploratory Data Analysis," Medium, 13 January 2019. [Online]. Available: <https://towardsdatascience.com/a-gentle-introduction-to-exploratory-data-analysis-fl1d843b8184>.
- [43] G. Seif, "Handling Imbalanced Datasets in Deep Learning," Medium, 19 November 2018. [Online]. Available: <https://towardsdatascience.com/handling-imbalanced-datasets-in-deep-learning-f48407a0e758>.
- [44] D. Hornung, "Binary Classification with Logistic Regression," Towards Data Science, 24 November 2019. [Online]. Available: <https://towardsdatascience.com/binary-classification-with-logistic-regression-31b5a25693c4>.
- [45] J. Brownlee, "Logistic Regression for Machine Learning," Machinelearningmastery, 1 April 2016. [Online]. Available: <https://machinelearningmastery.com/logistic-regression-for-machine-learning/>.
- [46] F. Ceballos, "An Intuitive Explanation of Random Forest and Extra Trees Classifiers," Medium, 14 July 2019. [Online]. Available: <https://towardsdatascience.com/an-intuitive-explanation-of-random-forest-and-extra-trees-classifiers-8507ac21d54b>.
- [47] P. Ramchandani, "Random Forests and the Bias-Variance Tradeoff," Towards Data Science, 10 October 2018. [Online]. Available: <https://towardsdatascience.com/random-forests-and-the-bias-variance-tradeoff-3b77fee339b4>.
- [48] A. Tripathi, "A Complete Guide to Principal Component Analysis — PCA in Machine Learning," Medium - Towards Data Science, 11 July 2019. [Online]. Available: <https://towardsdatascience.com/sampling-techniques-for-extremely-imbalanced-data-part-ii-over-sampling-d61b43bc4879>.
- [49] S. A. Rahim, "SMOTE AND NEAR MISS IN PYTHON: MACHINE LEARNING IN IMBALANCED DATASETS," Medium, 4 June 2018. [Online]. Available: <https://medium.com/@saeedAR/smote-and-near-miss-in-python-machine-learning-in-imbalanced-datasets-b7976d9a7a79>.
- [50] A. Kharenko, "Monolithic vs. Microservices Architecture," 9 October 2015. [Online]. Available: <https://articles.microservices.com/monolithic-vs-microservices-architecture-5c4848858f59>.