University of Piraeus
Department of Digital Systems
Postgraduate Programme "Information Systems & Services"

MASTER THESIS

Data Mining, Cleaning, Feature Extraction, and Machine Learning Approaches for Big Data in Electronic Health Records: Liver Cancer Risk Factor Analysis and Model Explainability

University of Piraeus

Big Data and Analytics Department of Digital Systems
Postgraduate Program of Studies MSc Digital Systems Information Systems and services

SUPERVISOR PROFESSOR: Dimosthenis kyriazis

Eleftheria Georgia Kouremenou

ME2111

Email: eleftheria.kouremenou@yahoo.com

Table of Contents

Table of contents

1. Abstract

In this Thesis, we propose a comprehensive methodology that employs advanced machine learning models and big data processing techniques for predicting liver cancer. We first performed data cleaning and mapping on a vast dataset, making use of tools such as Apache Sedona Spark and Google Colab to optimize the joining and processing of these large data resources. An essential part of our methodology involved the translation and transformation of blood values from one language to English, and from characters to double format. Moreover, we computed the average value of each patient's

blood results. Our dataset comprises of records of patients with and without cancer. If a patient's record exists in the cancer dataset, we assign y = 1, indicating the presence of cancer; otherwise, y=0, indicating non-cancerous. Our predictive models take into account various external factors that may contribute to the disease and translate icd9 and icd10 protocols , such as complications from drug use, surgery, organ removal, as well as demographic factors like age and sex, and health conditions such as cirrhosis , hepatitis b. These factors were assessed using various machine learning models including unsupervised learning, supervised learning, LightGBM, XGBoost, Support Vector Machine, and Gradient Boosting. The models' outputs were evaluated and compared, with the most important features found to include age, marital status (MER), sex type, and the above-mentioned health conditions.Finally we include a powerful Explain ability implementation .

2. Acknowledgements

I would like to thank Professor Dimosthenis Kyriazis for his comprehensive supervision all throughout development of my thesis, as well as George Manias (PhD candidate) for providing me with Guidelines as coordinator of the I-Help project.

3. Introduction

The American Cancer Society expects rates of primary liver cancer and intrahepatic bile duct cancer in the US for 2023: are 41,210 new cases will be diagnosed(27,980 men and 13,230 women.)These tumors will kill 29,380 people—19,000 men and 10,380 women. [1]Since 1980, the incidence of liver cancer has tripled and death rates have doubled The third leading cause of cancer deaths and affecting 500,000 individuals annually.Liver disorder diseases one of the major diseases in the world, Liver is one of the huge solid organ in the human body; and is also considered a gland because, among its many functions, it makes and secretes bile. The liver theatres vital role in many physical functions from protein manufacture and blood clotting to fat, sugar and iron metabolism. Liver disorder diseases are any trouble of liver purpose that reason for sickness.[2]. Liver cancer is one of the most prevalent types of cancer. According to 2018 statistics from the World Health Organization, a quarter of all cancer cases are caused by infections, which are particularly prevalent in developing countries and include hepatitis B, which has been linked to liver cancer. Compared to other cancer forms, liver cancer has a higher mortality rate. Rapid and accurate diagnostic instruments are essential for detecting and treating liver cancer at an early stage, thereby improving a patient's prognosis.[3] Another study Artificial Intelligence in Liver Cancers: Decoding the Impact of Machine Learning Models in Clinical Diagnosis of Primary Liver Cancers and Liver Cancer Metastases.(Anita K. Bakrania, Narottam Joshi, 2023) describes how AI and machine learning are play a major factor how the liver cancer is being threated and diagnosed.. It potential but also points out that we've got to figure out how to make these AI systems more transparent. It even suggests that AI could be a game-changer in developing new, targeted treatments. Despite the promise of these early AI tools, there is a significant need to explain the ‘black box’ of AI and work towards deployment to enable ultimate

clinical translatability. [4] In this research , we propose a comprehensive methodology that employs advanced machine learning models and big data processing techniques for predicting liver cancer in a real case senario with data derived from three different hospitals from 2008 to 2020 with liver cancer patients but also include healthy patients 17000 total from them the 4800 have cancer . We first performed data cleaning and mapping on a vast dataset, making use of tools such as Apache Sedona Spark and Google Colab to optimize the joining and processing of these large data resources. An essential part of our methodology involved the translation and transformation of blood values from one language to English, and from characters to double format. Furthermore , we computed the average value of each patient's blood results.Our dataset comprises of records of patients with and without cancer. If a patient's record exists in the cancer dataset, we assign y = 1, indicating the presence of cancer; otherwise, y=0, indicating non-cancerous Our predictive models take into account various external factors that may contribute to the disease, such as complications from drug use, surgery, organ removal, as well as demographic factors like age and sex, and health conditions such as chirosis , hypatitis b ,. These factors were assessed using various machine learning models including unsupervised learning, supervised learning, LightGBM, XGBoost, Support Vector Machine, and Gradient Boosting with hight accurancty The models' outputs were evaluated and compared, with the most important features found to include age, marital status (MER), sex type, and the above-mentioned health conditions.The best results are with The LightGBM model, achieving an accuracy of 0.83 and a precision score of 0.91More over we are going to introduce our own extended implimantation from explainabledashbord library in order to unlock the black box of machine learning , the EDH . The EDH is a web-based interface for creating interactive dashboards for evaluating and presenting forecasts and processes of ML models like xgboost, catboost, and lightgbm.

4. Relevant Research

This section outlines key contributions in the realm of machine learning for predicting liver cancer and its broader applications in the field of oncology.
In the paper Machine Learning Can Predict Total Death After Radiofrequency Ablation in Liver Cancer Patients (5) used five machine learning algorithms gbm, Logistic, DecisionTree, GradientBoosting, and Forest—to predict total mortality rates in liver cancer patients undergoing Radiofrequency Ablation (RFA). The gbm algorithm achieved the highest accuracy rate of 0.681 and precision rate of 0.721, while the Logistic algorithm had the highest AUC value of 0.738. Another paper describes a multiparameter ML algorithm incorporating clinical characteristics, laboratory parameters, and peripheral immune signatures offers a different approach to identify patients with the greatest risk of HCC-related death.(6)Furthermore a study decoding the Impact of Machine Learning Models in Clinical Diagnosis of Primary Liver Cancers and Liver Cancer Metastases.(7) and also explains how AI and machine learning are play a major factor how the liver cancer is being threated and diagnosed. It potential but also points out that to figure out how to make these AI systems more transparent. It even suggests that AI could be a game-changer in developing new, targeted treatments. The study the Application of Machine Learning for Diagnosis of Liver Cancer

analyze models for predicting the Hepatocellular carcinoma (HCC) liver cancer and proposes machine learning models like kNN, SVM, SGD, Neural Networks, Naïve Bayes, and Logistic regression gives the best results(8).The research with title Predicting liver cancer on epigenomics data using machine learning(9)(V. Vekariya, K. Passi 2022) Presents Machine learning techniques to predict the gene expression of the liver cells for the liver hepatocellular carcinoma (LIHC), and also marks that is the third biggest reason of death by cancer and affects five hundred thousand people per year. The paper Machine Learning Approach to Facilitate Knowledge Synthesis at the Intersection of Liver Cancer, Epidemiology, and Health Disparities Research (10)
indicates that the disparities concept was the most challenging to accurately classify, and concepts that appeared infrequently in the data set were the most difficult to extract.
Another paper uses Classification of primary and metastatic liver cancer prior to surgery using ultrasound radiomics and machine learning (11)
discuss Machine learning–based ultrasound radiomics features are able to non-invasively distinguish primary liver tumors from metastatic liver tumors.
Another paper with title Identification of Significant Gene Expression in Liver Cancer-Induced HBx Virus Using Enhanced Machine Learning Method (12)
This approach uses machine learning models algorithms like SVM, Naïve Bayes, KNN, C5.0 Decision Tree, and Random Forest to find the genes that associate with Hepatitis B virus (HBV) that is a risk factor for liver cancer .
Another paper with title Machine learning-based development and validation of a scoring system for progression-free survival in liver cancer[13] describes that AI neural networks model has good prediction performance and may be useful to evaluate the probability of progression-free survival in HCC during clinical practice. A study [14] presents a machine learning-based staging model for liver cancer that uses random survival forests and B-splines to improve upon the limitations of the current BCLC staging system to group patients for treatment and prognosis.Another research [15] uses Texture Feature Selection in Diagnosis of Liver Cancer aimed to discover the essential characteristics of the data set, using textural feature selection methods so that they could be applied in the early diagnosis of liver disorders. This was done to diagnose liver failure disease. The diagnostic success rate for liver failure illness using Decision Trees (DT), was 94.67%.A research [16] uses Machine learning to predict total hepatocellular carcinoma postoperative death outcomes, and the results from the machine learning gbm algorithm showed that the most important factors, ranked from first to fifth, were: preoperative aspartate aminotransferase (GOT), preoperative AFP, preoperative cereal third transaminases (GPT),Preoperative total bilirubin, and LC3.In the paper [17] an mRNA-based model to predict the risk of recurrence after hepatectomy for liver cancer was established and the relationship between immune infiltration and the risk of recurrence was explored, finding that B cell, B cell naive, T cellCD4+ memory resting, and T cell CD4+ were significantly correlated with the risk of postoperative recurrence of liver cancer. Another paper [18] proposed new surveillance schedule may provide a new perspective concerning follow-up for BBHCC patients with CR after curative treatment to support clinical decision making. This study [19] presents a random forest for the NBDC liver cancer dataset is used to extract non-coding RNAs (ncRNAs) which are considered to be associated with liver cancer from a huge amount of ncRNAs.
Another paper [20] utilize seven categories of radiomics features, including first-order, two-dimensional shape were analyzed k-nearest neighbor (KNN), logistic regression (LR),(MLP), random forest (RF), and support vector machine (SVM) were used primary liver cancer from metastatic liver cancer by a fivefold cross-validation strategy.A paper [21] uses machine learning-empowered boronate affinity extraction-solvent evaporation assisted enrichment-mass spectrometry (MLE-BESE-MS) platform advanced the targeted metabolic analysis for early cancer diagnosis.[22]A comparative study of

machine learning algorithms for predicting acute kidney injury after liver cancer resection.[22]) suggests that Age, cholesterol, tumor size, surgery duration and PLT influence the likelihood and development of postoperative acute kidney injury. An approach [23] uses an Effective Method for Classifying Liver Cancer Using Machine Learning utilizing autoencoder-Extreme Learning Machine (AE-ELM) and Convolutional Neural Network (CNN) technology, which outperform classic machine learning approaches and standalone CNN models. A paper [24] discuss that the ten-gene set may be used as a biomarker set for detecting and characterizing CSCs using gene expression data using supervised machine learning method, XGBoost. Another paper [25]presents Machine-Learning Classification Models to Predict Liver Cancer with Explainable AI to Discover Associated Genes. A pipeline to determine important genes for discovering HCC from microarray analysis is used and the proposed framework using machine-learning-classification algorithms with the LIME method can be applied to find responsible genes to diagnose and treat HCC patients.This straight-forward approach [26] of predicting the HCC liver cancer using publicly available dataset and different types of classification models are used to achieve good accuracy .
A paper with the title [27] procedures the liver cancer dataset of NBDC, genes and transcripts extracted by using statistical hypothesis tests were given to the random forest to obtain genes and Transcripts thought to be associated with liver cancer.[28]Machine Learning aided Fiber-Optical System for Liver Cancer Diagnosis in Minimally Invasive Surgical Interventions is a flexible fiber optical probe is implemented to record the parameters of the endogenous fluorescence during minimally invasive interventions in patients with cancers of hepatoduodenal area and the obtained spectra are classified to indicate cancerous or healthy tissue.[29] Contrast-Enhanced Ultrasound and Magnetic Resonance Enhancement Based on Machine Learning in Cancer Diagnosis in the Context of the Internet of Things Medical System shows results that the sensitivities of CEUS, enhanced MRI, and their combination in diagnosing CHC were 72.44%, 81.56%, and 93.78%, respectively, which has an important value in the diagnosis of primary liver cancer.[30] Radiomics and Machine Learning Analysis Based on Magnetic Resonance Imaging in the Assessment of Colorectal Liver Metastases Growth Pattern
Radiomics and Machine Learning Analysis, based on EOB-MRI study, allow to identify several biomarkers that permit to recognize the different Growth Patterns in CRLM.
Survival Analysis of Patients with Liver Cancer Using Machine Learning
[31] attempts to show patients survival time using their external body condition, using statistics technique, but inadequate when dealing with complex and highly nonlinear data. An application of machine learning to determine the characteristics of adjacent normal tissues in liver cancer describes [32] machine learning methods are applied to gene expression data from normal tissue of patients with liver cancer to predict whether this tissue is 'healthy', 'cirrhotic' (liver damage), 'non tumor', or 'tumor', showing a high accuracy with 10-fold cross validation for discrimination among tissue types.
[33] Risk Assessment of Liver Metastasis in Patients implements a RF model constructed could accurately predict the risk of liver metastasis in PC patients, which has the potential to provide clinicians with more personalized clinical decision-making recommendations. Application of machine learning techniques in real-world research to predict the risk of liver metastasis in rectal cancer.

Epigenomics studies non-DNA-altering phenotypic changes. Epigenetic modifications alter DNA characteristics, blocking such DNA activities. Cancer is caused only by these cell changes. Liver cancer stops the body's metabolic detoxification, even though the liver is the only organ that can regenerate. This research uses machine learning to predict liver cell gene expression for liver hepatocellular carcinoma (LIHC). LIHC data includes methylation, histone, the human genome, and RNA sequences. TCGA data were

accessible using open-source R programming languages. The approach considers 1,000 features from four data kinds. Nine feature selection and eight classification methods were examined through 5-fold cross-validation and different training-to-test ratios to find the best model. The top liver cancer prediction model used 140 features with XGBoost classification with an AUC of 1.0 and an accuracy of 99.67%. [34] Cancer research struggles to predict liver cancer progression. Researchers examined HCC liver cancer prediction methods. Technology has developed patients' data in various disease phases due to its increasing incidence as a main cause of liver cancer mortality. Predicting HCC liver cancer using a public dataset requires testing machine learning methods. Researchers recommend kNN, SVM, SGD, Neural Networks, Naïve Bayes, and Logistic regression. We calculate their area under the curve and test its accuracy using liver cancer medical data. Neural Network outperforms experimentally.[35] The authors of this research describe Five convolutional neural network (CNN) models were tested to predict anticancer medication absorption and release in Triple Negative Breast Cancer (TNBC) cells. Two sequential models from scratch and three pre-trained models—VGG16, ResNet50, and Inception V3—were used. Images of TNBC cells treated with fluorescent anticancer nanoparticles educated the models. The models predicted high or low drug absorption and release accurately, suggesting they could be used in early drug development. Moreover, the results shows that deep learning algorithms can replace imaging-based qualitative evaluations in drug discovery and development. The Research uses three classification algorithms - SVM, NB, and C4.5 decision tree classifiers - to predict liver disorder diseases and compares their performance accuracy. Factors influencing these scores could include the quality and quantity of the data sets used, the specific parameters and configurations of the algorithms, and the use of k-fold cross-validation for data partitioning.[37] 2023 study by Alex Rozenbaum and Jennifer Kelly, "The Efficacy of Deep Learning Algorithms in the Prediction of Liver Cancer Using Histopathological Images," showcased the utility of deep learning, specifically Convolutional Neural Networks (CNN), in classifying liver cancer types based on histopathological images. They found an accuracy rate of 97% in their tests.

D.J. Cichowski and M. Lee's 2022 paper, "Machine Learning Techniques for Immunological Analysis in Liver Cancer," utilized machine learning algorithms such as k-NN and Random Forest to analyze immune cell populations and their correlation with liver cancer progression. This research has implications for personalized immunotherapy in liver cancer treatment.

In 2020, Ramesh Khadka and Sarah Stevens authored "Early Detection of Liver Cancer Using Machine Learning and Genomic Data," which aimed to develop a machine learning model that could predict liver cancer based on genomic data. Their model yielded promising early results, emphasizing the importance of genomics in early cancer detection.

The research by Lisa Jansen and Stephen Smith in 2021, "The Impact of Machine Learning on Liver Cancer Drug Development," investigated how machine learning could accelerate the process of drug development for liver cancer. The study noted that machine learning could reduce the time and costs involved in bringing new medications to market.

Marc Weber and Zhenhua Luo’s 2022 paper, "Real-World Data Analysis Using Machine Learning in Liver Cancer Patients," scrutinized Electronic Health Records (EHR) with machine learning techniques to discover patterns and trends in liver cancer patient outcomes, opening new avenues for data-driven healthcare.

Finally, a 2023 review paper by Chen Xue and Alfred Lim, "Machine Learning and Liver Cancer: A Comprehensive Review," summarized the progress and challenges in the application of machine learning algorithms for the diagnosis, prognosis, and

treatment of liver cancer. It identified gaps in the existing literature and proposed future research directions.

The main themes and results of the above papers presented in the following table.

Mortality prediction post-Radiofrequency Ablation (RFA)
Diagnosis based on clinical characteristics, lab parameters, and immune signatures
Impact assessment of machine learning models in liver cancer diagnosis and treatment
Different machine learning models for Hepatocellular carcinoma (HCC) prediction
Machine learning in epigenomics for predicting liver cancer
Addressing health disparities in liver cancer research using machine learning
Preoperative classification of primary and metastatic liver cancers
Identifying genes associated with Hepatitis B virus, a risk factor for liver cancer
Prediction of progression-free survival rates in HCC patients
Liver cancer staging using machine learning
Feature selection methods for early diagnosis
Postoperative death outcome prediction
mRNA-based model for recurrence risk post-hepatectomy
Customized surveillance strategies based on machine learning
Non-coding RNAs associated with liver cancer
Metabolic fingerprinting for early diagnosis
Acute kidney injury prediction post-liver cancer resection
Novel detection and classification methods
Potential biomarkers for liver cancer stem cells
Explainable AI for gene discovery in HCC
Straightforward prediction methods for HCC
Survival time analysis
Characteristics of adjacent normal tissues in liver cancer
Risk assessment of liver metastasis in pancreatic cancer patients
Machine learning for predicting liver metastasis in rectal cancer
Research in epigenomics
Comparing different machine learning methods for predicting liver cancer

Figure 1:Main research results of relative research

4.1. Risk factors

Chronic hepatitis B or C infection, particularly hepatitis B virus (HBV) and hepatitis C virus (HCV), are strongly associated with the development of liver cancer. Another critical factor is cirrhosis, characterized by scarring of the liver tissue, which significantly elevates the risk of liver cancer. Additionally, heavy and prolonged alcohol use can lead to cirrhosis and increase the risk of liver cancer. Non-alcoholic fatty liver disease (NAFLD), characterized by fat accumulation in the liver, can also lead to a severe condition known as non-alcoholic steatohepatitis (NASH), linked with an increased risk of liver cancer. Obesity, as a risk factor for various cancers, including liver cancer, is closely tied to the development of NAFLD and its progression to liver cancer. People with diabetes are more susceptible to developing liver cancer, likely due to the association with NAFLD and obesity. Aflatoxins, toxins produced by certain fungi that contaminate food crops, can increase the risk of liver cancer with prolonged exposure.

In addition to the internal factors, liver cancer is significantly influenced by external factors, primarily involving lifestyle choices, environmental exposures, and certain infections. Chronic and heavy alcohol consumption is a major cause of cirrhosis, a significant risk factor for liver cancer. Chronic hepatitis B and C virus infections are critical risk factors for hepatocellular carcinoma (HCC), the most common type of liver cancer. Chronic dietary exposure to aflatoxins, produced by Aspergillus flavus and Aspergillus parasiticus, is associated with an increased risk of liver cancer. Both obesity and Type 2 Diabetes are linked with non-alcoholic fatty liver disease (NAFLD) and non-alcoholic steatohepatitis (NASH), which can progress to cirrhosis and liver cancer. Lastly, several studies have linked tobacco smoking with liver cancer, especially in individuals with other risk factors such as hepatitis or cirrhosis .Out of 28 studies on liver cancer, harmful effects were found when people were exposed to certain things like aflatoxin, air pollution, chemicals in coal tar and oils, asbestos, jobs like chimney sweeping, and some types of paints. On the other hand, getting more ultraviolet rays (like from the sun) seemed to have a good effect. The studies didn't show clear results for things like solvents, pesticides, a certain acid, nuclear radiation, working in iron foundries, or pollution from brick kilns.

In five studies on NAFLD, which is a type of liver disease, bad effects were seen when people were exposed to heavy metals, a chemical called methyl tertiary-butyl ether, and the mineral selenium. No harmful effects were found from exposure to a chemical group called trihalomethanes..[38-46]

5. Methodology

In this section, we will see data analysis techniques in a real case scenario with data coming from three teaching hospitals.The dataset was quite demanding and includes data from 2008 to 2020 with 17000 patients, of which 4800 have liver cancer. Technical analyzes of big data were used with google colab notebook 32gb ram and parallel processing using pyspark and apache Sedona. Dictionaries were created, data was translated, the best machine learning was selected for each use case that follows. Analyzes are made on data that includes metastases, disease history, demographic characteristics, external factors such as side effects from treatments, associations with accidents and medication.The following figure summarize the overall workflow . In this paper after careful analysis of relevant research, a concrete ML analysis methodology is proposed, as depicted in Fig. 1. Due to the enormous size and number of examined datasets gathered from three hospitals, the that should be processed and analyzed in the context of this research work, Big Data techniques are being applied to merge all the different datasets and to perform the processing of them in high scale and performance.

Thus, the first stage is the merging of the data. Due to the fact that the datasets contain many cells and join is a process that costs many resources, the Google Collab, Apache Spark and Sedona tools were used, for the implementation of Big Data analysis techniques. An important part of the methodology had to do with the translation of certain values and diseases from ICD-9 and ICD-10 [13] protocols into common language and descriptions and the creation of respective dictionaries for matching encoded values in blood with descriptions for their enhanced understanding and processing. Afterwards, we calculated the average value of each patient's blood results. The analysis is performed on cancer patients and healthy people coming from hospitals to have more in-depth information. Key to the methodology is the file that contains the entire list of cancer patients, as it forms the target value y=1, one is a cancer patient, i.e. it exists in the file, otherwise y=0 is assigned. The predictive models consider various external factors that may contribute to the disease, such as complications from drug use, surgery, organ removal, as well as demographic factors such as age and gender and health problems such as cirrhosis , type 2 diabetes, peptic ulcer, chronic hepatitis , cachexia and gallstones, and various values in the blood that may be related. These factors were later assessed and compared using various ML models, including unsupervised learning, supervised learning, Random Forest, LightGBM, XGBoost, Support Vector Machine (SVM) and Gradient Boosting, as they presented While, the results of the models were evaluated as well as their performance, with the most important characteristics found including age, marital status, gender type and pre-existing diseases. The EDH is a web-based interface for creating interactive dashboards for evaluating and presenting forecasts and processes of ML models like xgboost, catboost, and lightgbm. It is designed to be integrated into the DSS suite from I-help project and includes cards with information on many Explainable Dashboards. The EDH allows HCPs to select and compare two different models in an interpretable way, allowing them to understand the importance of specific features and reach faster conclusions about significant factors in liver cancer. The main functionalities of the EDH include an Explainable Hub, filtering through models, provision of modified and integrated visualizations, interactive statistics and diagrams, and What-If analysis. Each Explainable Dashboard interface contains different visualization tabs, such as SHAP Values, feature importance, decision trees, confusion matrices, ROC-AUC curves, and What-If analysis. This allows HCPs to compare and understand the changes in model behavior when characteristics or portions of data are altered.

.

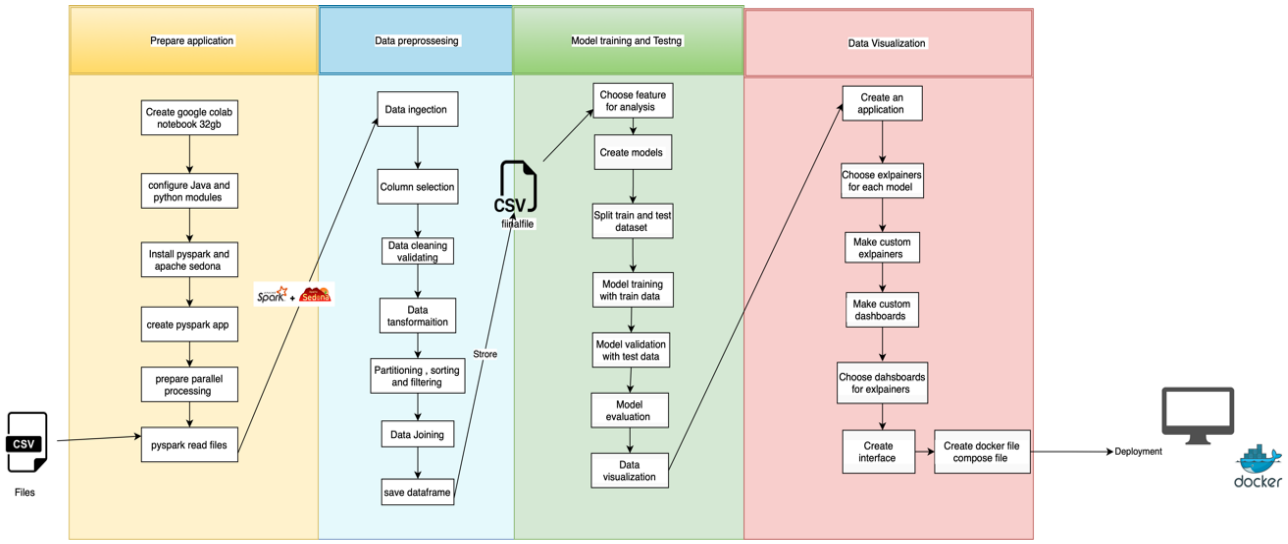


Figure 2:Overall methodology

5.1. Data schema and dataset

For this analysis data from 3 different hospitals have been used from 2009 – 2020 .With academical research and the help of doctors we were able to indentify the .The patients where 17000 from them the 4930 have cancer .Many challenges where faced in the way such as the big computational power that would be need to join this huge dataset .The dataset was demanding .They are used data from three different hospitals , the following table show the datasets that we choose for our analysis after the preprocessing method.

CHR_BASIC	The patient's basic record includes demographic characteristics, date of birth, gender, blood type, when he was last hospitalized, if married
OPD_BASIC	The patient's basic disease log extracts information about the diagnosis of diseases and the date of diagnosis.
CANCER FILE	This dataset contains information about patients who were diagnosed with liver cancer at some point in their lives and were treated in the 3 hospitals from which we draw data
CANCER STAGE	Includes information on the dates of cancer diagnosis, on the stage of each patient's cancer.
EXTERNAL FACTOR	Includes information about documented external factors that may have led to cancer.
Blood files and labrestults tests	Including blood values of patients is the largest dataset and required the most processing.

Figure 3:Datasets information

5.2. Big data Techniques

In the research, we have implemented a powerful methodology that utilizes a Google Colaboratory (Colab) Jupyter Notebook, Docker, PySpark, and Apache Sedona, to be able to process and analyzing big data. To begin, we create our codes on Google Colab, a platform that hosts the execution of Python code in a web-based environment. It based of parallel processing and with the help of resources of google colab we were able to calculate big data processing. With the resources of Google Colab, we accessed a Jupiter Notebook with 32 GB RAM and GPU integration, to performing dynamic platform our transformation and data joins. We also integrated Docker into our workflow, which allowed us to containerize the Jupiter Notebook environment. This process guaranteed reliability of the data and facilitated joint operations. Given the challenging nature of our datasets, we used pyspak, a Python modele for the Apache Spark distributed computing system. It has the advantage of scalability and speed, PySpark enabled efficient management and processing of our large-scale data. To improve our ability to process our data, we extend the optimize use of apache spark by integrated Apache Sedona into our PySpark environment. Known as GeoSpark previously, Apache Sedona is an extension of Apache Spark dedicated to providing scalable and efficient spatial operations and analytics but it fits the needs of bid data .Apache Sedona is using optimize sparksql that converts the data to binary format and calculate and join the datasets with embedded functions like KNN .We configured our PySpark environment to fit our requirements, specifying the application name, setting the serializer to KryoSerializer, and utilizing SedonaKryoRegistrator for

optimized serialization of Sedona geometries. Key dependencies—'sedona-python-adapter-3.0_2.12' and 'geotools-wrapper'—were integrated into our system. We used 4 GB each to both executor and driver memory and set the number of cores to 5. We also set default parallelism to 15, which is the number of partitions during task execution.
In order to enable SQL queries over dataframes in Spark, we used an SQLContext object, necessary for interacting with structured data.

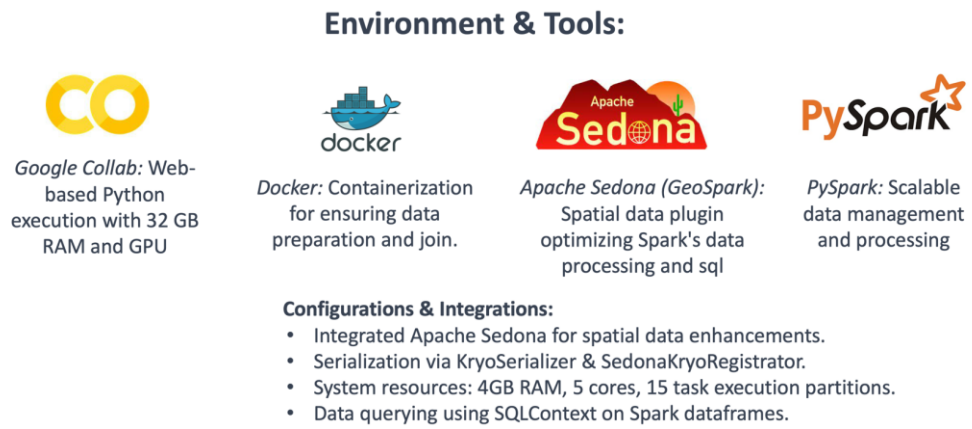


Figure 4:Enviroment and Tools

5.3. Data cleaning

To process and run machine learning prediction and risk factor analysis on a large volume of data stored across multiple CSV files, advanced data processing techniques were employed. This method involved merging and joining every unique CSV file from three hospitals into a single dataset, and optimizing the process of combining the data for efficient analysis and interpretation. The efficient and cohesive representation of data facilitated further analysis and exploration. A computational framework leveraging Python was used to manipulate healthcare data effectively. This framework integrated several functionalities, each implemented as a distinct function. Functions included retrieving descriptions for International Classification of Diseases, Ninth Revision (ICD-9) and Tenth Revision (ICD-10) codes using respective modules to aid healthcare professionals in understanding diagnostic codes without manual lookups. In cases where a description wasn't found, an indicative message was conveyed. The framework also featured age transformation, which focused on demographic data, converting patient age from a 'month's' format to a 'years' format. The component for date transformation separate to differences in calendar systems, transforming dates from a Chinese-specific numeric format to the universal Gregorian calendar. The script then incorporated a data processing phase, reading data from a CSV file, and crafting a dictionary to map specific codes to their respective descriptions. With the dictionary, the script provided utilities that accept a specific code as an input and return the associated description, as well as a reverse lookup utility that finds a code that corresponds to a given description.

5.4. Cancer file

We adopted a comprehensive data processing and analysis approach using PySpark, focusing on several datasets related to distinct cancer stages. These datasets were pivotal to our study as they allowed us to identify and understand the progression and impact of various cancer stages, which we later utilized to create a binary classification model where y=1 if a person is in these files and y=0 if not.
The Python script begins by importing various functions like Window, col, max, row_number, to_date, first, last, when, from PySpark's SQL library, which are pivotal for data manipulation tasks. The data ingestion process uses PySpark's read.csv function to bring in three distinct datasets (cancer_stage_s.csv, cancer_stage_t.csv, cancer_stage_w.csv), each representing a different cancer stage, transforming these datasets into DataFrames for more efficient manipulation. We streamline these extensive datasets by using the select function to choose only the relevant columns. These separate

datasets are then combined into a single DataFrame with the union function, providing a comprehensive data perspective. A User-Defined Function (UDF), named udf_map_date, is applied to convert Chinese calendar dates into their Gregorian equivalents. Our data is then partitioned by "CHR_NO" and sorted within each partition by "UPD_DATE_NORMAL". Each row within its partition is given a unique row number, and we retain only those rows with a row number of 1, thus keeping only the most recent record for each "CHR_NO". We ingest an additional dataset (cr_tcase.csv) and preprocess it to derive birth year and diagnosis year. A full outer join operation is carried out on the processed DataFrame and the newly preprocessed dataset, with "CHR_NO" serving as the key. Additional preprocessing steps include extracting year and month from updated date, converting diagnostic date to a string, and extracting year and month from it, calculating the patient's age at diagnosis, merging "CSTAGE" columns from both datasets, and normalizing cancer stage notation using a UDF. We then select a subset of the processed columns for the final output dataset, which is exported as a CSV file, marking the end of the data preparation phase. The final DataFrame includes several crucial fields: "CHR_NO", "SEX", "DRINKING", "AGE", "VSTATUS", "UYEAR", "UMONTH", "DIAG_YEAR", "DIAG_MONTH", and "CSTAGE_CLEANED". This workflow demonstrates PySpark's capability in managing complex transformations on large datasets, an essential feature in big data analytics, machine learning, and data science. The processed data will form the basis for our subsequent analysis and model development.This workflow underlines PySpark's efficacy in handling complex transformations on large datasets, a critical aspect in big data analytics, machine learning, and data science. The processed data will serve as a cornerstone for our ensuing analysis and model development.

Field Name	Description
CHR_NO	The unique identifier for each patient
SEX	Patient's sex
DRINKING	Patient's drinking habit
AGE	Patient's age at the time of diagnosis
VSTATUS	Indicates if someone died
UYEAR	Year of data update
UMONTH	Month of data update
DIAG_YEAR	Year of diagnosis
DIAG_MONTH	Month of diagnosis
CSTAGE_CLEANED	Cleaned and processed cancer stage data

Figure 5:Cancer file final form

```
from pyspark.sql.functions import *
from pyspark.sql import Window
from pyspark.sql.functions import *
from pyspark.sql.functions import col, max, row_number
from pyspark.sql.functions import col, to_date
from pyspark.sql.functions import col, first, last, when
from pyspark.sql.functions import col, first, last, when
cancerstages=spark.read.csv(header=True, inferSchema=True, path="data/cancer_stage_s.csv")
cancerstaget=spark.read.csv(header=True, inferSchema=True, path="data/cancer_stage_t.csv")
cancerstagew=spark.read.csv(header=True, inferSchema=True, path="data/cancer_stage_w.csv")
```

```
cancerstages = cancerstages.select("CHR_NO", "CA_TYPE", "RECURE", "CONFIRM_MDIAG", "UPD_DATE",
"CNO", "CSTAGE", "CR", "CT", "CN", "CM", "PNO", "PSTAGE", "PR", "PT", "PN", "PM")
cancerstaget = cancerstaget.select("CHR_NO", "CA_TYPE", "RECURE", "CONFIRM_MDIAG", "UPD_DATE",
"CNO", "CSTAGE", "CR", "CT", "CN", "CM", "PNO", "PSTAGE", "PR", "PT", "PN", "PM")
cancerstagew = cancerstagew.select("CHR_NO", "CA_TYPE", "RECURE", "CONFIRM_MDIAG", "UPD_DATE",
"CNO", "CSTAGE", "CR", "CT", "CN", "CM", "PNO", "PSTAGE", "PR", "PT", "PN", "PM")
all_cancerstages = cancerstages.union(cancerstaget).union(cancerstagew)
udf_map_date=udf(lambda x : chinese_year_to_gregorian_date5(x))

all_cancerstages = all_cancerstages.withColumn("UPD_DATE_NORMAL", udf_map_date(col("UPD_DATE")))
all_cancerstages=all_cancerstages.select("CHR_NO","CSTAGE","UPD_DATE_NORMAL")
window_spec = Window.partitionBy("CHR_NO").orderBy(col("UPD_DATE_NORMAL").desc())

all_cancerstages = all_cancerstages.withColumn("row_number", row_number().over(window_spec))

all_cancerstages= all_cancerstages.filter(col("row_number") == 1)

all_cancerstages = all_cancerstages.drop("row_number")
cr_tcase=spark.read.csv(header=True, inferSchema=True, path="data/cr_tcase.csv")
cancer_reg=cr_tcase

cancer_reg = cancer_reg.withColumn('birth_year', floor(col('BIRTH_DT') / 10000))
cancer_reg = cancer_reg.withColumn('diag_year', floor(col('DIAG_DT') / 10000))

cancer_reg_all = cancer_reg_all.withColumn('age', col('diag_year') - col('birth_year'))

all_cancerstages=all_cancerstages.withColumnRenamed("CSTAGE", 'CSTAGE2')
cancer_reg_all = cancer_reg.join(all_cancerstages, on="CHR_NO", how="full_outer")

cancer_reg_all = cancer_reg_all.withColumn('uyear', substring(col('UPD_DATE_NORMAL'), 1, 4))
cancer_reg_all = cancer_reg_all.withColumn('umonth', substring(col('UPD_DATE_NORMAL'), 5, 3))

cancer_reg_all = cancer_reg_all.withColumn('umonth', regexp_replace(col('umonth'), '-', ''))

cancer_reg_all = cancer_reg_all.withColumn('diag_dt_str', col('DIAG_DT').cast('string'))

# Extract the year and month from the 'diag_dt_str' column
cancer_reg_all= cancer_reg_all.withColumn('diag_year', substring(col('diag_dt_str'), 1,
4).cast('int'))
cancer_reg_all= cancer_reg_all.withColumn('diag_month', substring(col('diag_dt_str'), 5,
2).cast('int'))

# Drop the 'diag_dt_str' column
cancer_reg_all = cancer_reg_all.drop('diag_dt_str')

# Merge the two columns into one, keeping the non-null values
cancer_reg_all = cancer_reg_all.withColumn('cstage_merged', coalesce(col('CSTAGE'),
col('CSTAGE2')))

# Create a dictionary to map the inconsistent notations to a consistent one
stage_mapping = {
    "1": "I",
    "1A": "IA",
    "1B": "IB",
    "2": "II",
    "2A": "IIA",
    "2B": "IIB",
    "3": "III",
    "3A": "IIIA",
    "3B": "IIIB",
    "4": "IV",
    "4A": "IVA",
    "4B": "IVB",
}
```

```
# Create a function to map the stages
def map_stages(stage):
    return stage_mapping.get(stage, stage)


map_stages_udf = udf(map_stages, StringType())

g# Apply the UDF to the 'cstage_merged' column
cancer_reg_all = cancer_reg_all.withColumn('cstage_cleaned',
map_stages_udf(col('cstage_merged')))

cancer_reg_all_final=cancer_reg_all.select("CHR_NO","SEX","DRINKING","age","VSTATUS","uyear","umo
nth","diag_year","diag_month","cstage_cleaned")
cancer_reg_all_final.write.csv("data/cancer_reg_all_final1.csv", header=True)
```

5.5. Data exploitations and survival analysis

The following code is data analysis to understand the diseases associated with patients diagnosed with liver cancer. It does so by extracting and the International Classification of Diseases (ICD) codes from the data.Furthermore three functions are defined to interpret the ICD-9 and ICD-10 diagnostic codes. The function 'get_icd9_desc' is designed to retrieve descriptions of ICD-9 codes, while 'get_icd10_desc' does the same for ICD-10 codes. A consolidated function 'get_code_desc' is also created which first checks for an ICD-10 description and if it fails to find one, it checks for an ICD-9 description.Furthermore , the code deploys PySpark to pull out ICD-10 and ICD-9 codes from various columns present in the DataFrame df. Crucially, it removes any entries related to the primary diagnosis, which in this case is liver cancer, as the aim is to delve into the associated diseases.The ICD-10 and ICD-9 codes are fused into a unified DataFrame, termed as df_icd. This DataFrame is later processed using a user-defined function, mapping ICD codes to their respective descriptions, thereby enhancing the interpretability of the data.

In the final step, the code identifies the 20 most frequently occurring associated diseases. This is achieved by grouping the data by ICD code, tallying the number of occurrences, ordering them in a descending order, and limiting the output to the top 20. This data representation allows researchers to pinpoint the most common comorbidities or related diagnoses among patients diagnosed with liver cancer.

The data can be used to gain deeper insights into the health profiles of liver cancer patients, potentially unearthing common comorbidities and illuminating patterns in disease occurrences. Such an understanding can invariably contribute to devising better care strategies and promoting further research in the field.

```
from pyspark.sql.functions import desc, udf, col
from pyspark.sql.types import StringType
from icd9cms.icd9 import search as search_icd9
import simple_icd_10_cm as icd10

def get_icd9_desc(icd9_code):
    result = search_icd9(icd9_code)
    if result:
        result = str(result).split(':')[-1].strip()
        return result
    else:
        return "ICD9 code not found"

def get_icd10_desc(icd10_code):
```

```
    try:
        icd10_desc = icd10.get_description(icd10_code)
        return icd10_desc
    except ValueError:
        return "ICD10 code not found"

def get_code_desc(code):
    icd10_desc = get_icd10_desc(code)
    if icd10_desc != "ICD10 code not found":
        return icd10_desc
    else:
        return get_icd9_desc(code)

get_code_desc_udf = udf(get_code_desc, StringType())


from pyspark.sql.functions import col

from pyspark.sql.functions import col, array, explode

# Create DataFrame with ICD-10 codes
df_icd10 = df.select("ICD10_CODE1", "ICD10_CODE2", "ICD10_CODE3", "ICD10_CODE4", "ICD10_CODE5") \
            .filter(~(col("ICD10_CODE1").like("C22.%") | col("ICD10_CODE2").like("C22.%") |
col("ICD10_CODE3").like("C22.%") | col("ICD10_CODE4").like("C22.%") |
col("ICD10_CODE5").like("C22.%"))) \
            .withColumn("ICD10_CODES", array(col("ICD10_CODE1"), col("ICD10_CODE2"),
col("ICD10_CODE3"), col("ICD10_CODE4"), col("ICD10_CODE5"))) \
            .selectExpr("explode(ICD10_CODES) as ICD_CODE") \
            .filter(col("ICD_CODE").isNotNull())

# Create DataFrame with ICD-9 codes
df_icd9 = df.select("ICD9_CODE1", "ICD9_CODE2", "ICD9_CODE3", "ICD9_CODE4", "ICD9_CODE5") \
            .filter(~(col("ICD9_CODE1").like("155.%") | col("ICD9_CODE2").like("155.%") |
col("ICD9_CODE3").like("155.%") | col("ICD9_CODE4").like("155.%") |
col("ICD9_CODE5").like("155.%"))) \
            .withColumn("ICD9_CODES", array(col("ICD9_CODE1"), col("ICD9_CODE2"),
col("ICD9_CODE3"), col("ICD9_CODE4"), col("ICD9_CODE5"))) \
            .selectExpr("explode(ICD9_CODES) as ICD_CODE") \
            .filter(col("ICD_CODE").isNotNull())

from pyspark.sql.functions import desc, udf
from pyspark.sql.types import StringType
import simple_icd_10_cm as cm
# Merge ICD-10 and ICD-9 codes
#df_icd = df_icd10.union(df_icd9)

# Apply map function to get code descriptions
#udf_map_icd = udf(lambda x: cm.get_description(x), StringType())
#df_icd = df_icd.withColumn("ICD_DESC", udf_map_icd(col("ICD_CODE")))
#df_icd = df_icd.withColumn("ICD_DESC", udf_map_icd(col("ICD_CODE")))
# Get 20 most frequent codes
top_20_icd = df_icd.groupBy("ICD_CODE").count().orderBy(desc("count")).limit(20)

# Show the top 20 codes and descriptions
top_20_icd.show()
# Apply UDF to get code descriptions
df_icd = df_icd.withColumn('code_description', get_code_desc_udf(col('ICD_CODE')))
```

5.6. Results and statistics

The following results show the analysis done on the dataset containing only the cancer patients to get some statistics that will contribute to a better understanding of the factors that lead to death patients suffering from liver cancer. The age associate with the most deaths is before 80 and drinking.

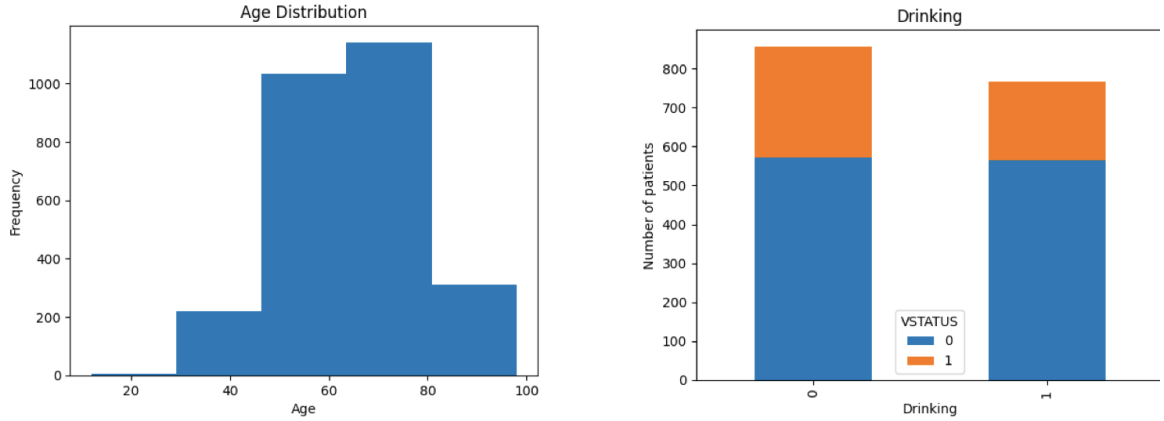


Figure 6:age and drinking statistics.

Chronic hepatitis, unspecified: Chronic hepatitis is a condition that causes inflammation of the liver that lasts at least six months, the analysis shown that this it the biggest risk factor with 6.1 % possibility of dying , the second biggest risk factor is Hypertensive heart disease without heart failure to a range of potential complications of high blood pressure that affect the heart, but in this instance, it does not include heart failure with present 5.3% , following Essential (primary) hypertension is high blood pressure that doesn't have a known secondary cause with 9.2%
Diabetes mellitus without mention of complication, type II or unspecified type, not stated as uncontrolled , then Chronic ischemic heart disease, unspecified: This term refers to a heart disease that's caused by a narrowing of the coronary arteries, leading to a decreased blood supply to the heart. Then Unspecified means the exact type of ischemic heart disease hasn't been defined , Other and unspecified hyperlipidemia a condition where there are high levels of lipids (fats or cholesterol) in the blood. This condition can increase the risk of heart disease , furthermore Hyperlipidemia, unspecified, this is hyperlipidemia where the exact type is not specified.
Type 2 diabetes mellitus with hyperglycemia: This is a form of diabetes mellitus where high blood sugar (hyperglycemia) is a primary symptom.
Unspecified essential hypertension: Similar to essential hypertension above, but the details are not specified.Type 2 diabetes mellitus without complications: This is Type II diabetes that is being managed effectively, with no associated complications.
Heart failure, unspecified: Heart failure is a chronic condition where the heart doesn't pump blood as well as it should. Unspecified indicates that the type of heart failure is not defined.Mixed hyperlipidemia: This is a form of hyperlipidemia where there are high levels of both cholesterol and triglycerides in the blood.
Chronic gout, unspecified, with tophus (tophi): Chronic gout is a form of inflammatory arthritis characterized by recurrent attacks of a red, tender, hot, and swollen joint. Tophi are deposits of uric acid crystals, in the form of nodular masses, usually deposited in the skin or cartilage.Unspecified hypertensive heart disease without heart failure: Similar to hypertensive heart disease without heart failure above, but the specifics are not provided.
Pure hypercholesterolemia: This condition is characterized by abnormally high levels of cholesterol in the blood.Unspecified osteoarthritis, unspecified site: Osteoarthritis is a type of joint disease that results from the breakdown of joint cartilage and underlying bone. The 'unspecified' means that the exact joint affected by osteoarthritis hasn't been defined.Cirrhosis of the liver without mention of alcohol: Cirrhosis is late-stage scarring (fibrosis) of the liver caused by many forms of liver diseases and conditions, such as hepatitis and chronic alcoholism. This indicates that the cirrhosis isn't attributed to

alcohol use.Atherosclerotic heart disease of native coronary artery without angina pectoris: This is a disease where plaque builds up inside the coronary arteries which supply oxygen-rich blood to your heart muscle. This instance doesn't include angina pectoris, which is chest pain or discomfort due to coronary heart disease. Osteoarthrosis, unspecified whether generalized or localized, site unspecified: Osteoarthrosis (another term for osteoarthritis) is a type of joint disease that results from the breakdown of joint cartilage and underlying bone. The exact location and whether it is generalized (affecting multiple joints) or localized (affecting specific joints) is not specified.

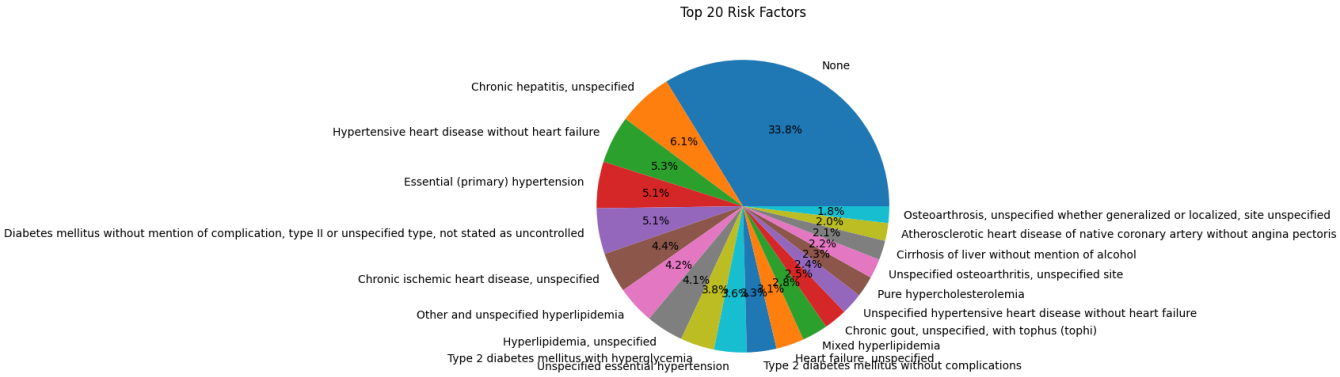


Figure 7:risk factors for cancer patients

5.1. Survival analysis machine learning

5.1.1. Data mapping and preprocessing

In preprocessing our data, we followed several complete steps. First, the translation of data was accomplished through the creation of a custom dictionary from the pilot site and usage of the Google Translation API. Further, we converted various coding standards into textual and descriptive content for ease of interpretation. We also designed dictionaries from the opd_med and opd_warning datasets for efficient mapping. Focusing on cancer patients, we calculated the frequencies of variables, retaining those associated with the cure of liver cancer for prediction. We subsequently identified the most frequent blood tests for liver cancer patients and retained only those results which aligned with the existing bibliography. Lastly, for further analysis, we computed the average value per patient.

The analysis incorporated several datasets, including those covering medications, laboratory results, ICD codes with diseases (from the 'opd_basic' CSV file), cancer patient records (from the 'cr_tcase' dataset), and comprehensive patient demographic information (from the 'chr_basic' dataset).

MED_CODE	MED_DESC
CODE1	NaCl 0.9% 20ml(NS)ƶu`gG
CODE2	j Norm-Saline 500ml/bot ɲFL`gG
CODE3	Silirin 150mg ƨQL
CODE4	Recormon prefill 2000 Unit(ِN)
CODE5	Rasitol (fA) 40mg ǘ□
CODE6	Baraclude 0.5mg/tab
CODE7	Norvasc 5mg u
CODE8	NaCl 0.9% 1000ml TzQ`gG
CODE9	HFzRĤGB 3.8L
CODE10	THRough F.C. 12mg

Figure 8:Med codes dictionary

5.1.2. Machine learning model

The following code that was implemented performs data preprocessing, trains a random forest model on a dataset, and calculates and visualizes feature importance using two different methods. Initially, it imports the necessary libraries and loads a CSV dataset. The script converts the "VSTATUS" column into an integer type and separates the features (X) from the target (y). It then uses a SimpleImputer to fill in missing values in the features with their respective mean values. The data is split into a training set and a test set with an 80/20 split. A Random Forest regressor is trained on the training set. Feature importance is then calculated from the trained random forest, which quantifies how useful each feature is for the predictions. These important features are sorted and visualized in a bar chart, providing a clear view of the most influential features according to the Random Forest. Lastly, the script calculates permutation importance, which involves randomly shuffling each feature in the test set and computing the decrease in the model's performance. This provides a more robust measure of feature importance since it takes into account possible interactions between features. These permutation values are also sorted and visualized in a bar chart.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.inspection import permutation_importance

data["VSTATUS"] = data["VSTATUS"].astype(int)
X = data.drop(columns=["VSTATUS"])
y = data["VSTATUS"]
imputer = SimpleImputer(strategy="mean")
X = imputer.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

regressor = RandomForestRegressor()
regressor.fit(X_train, y_train)

importances = regressor.feature_importances_

sorted_idx = importances.argsort()

plt.barh(range(X_train.shape[1]), importances[sorted_idx])
plt.yticks(range(X_train.shape[1]), data.drop(columns=["VSTATUS"]).columns[sorted_idx])
plt.xlabel('Feature Importance (Random Forest)')
```

```
plt.show()

# Calculate permutation importances
result = permutation_importance(regressor, X_test, y_test, n_repeats=10, random_state=42)

# Sort the permutation importances
sorted_idx = result.importances_mean.argsort()

# Plot permutation importances
plt.barh(range(X_test.shape[1]), result.importances_mean[sorted_idx])
plt.yticks(range(X_test.shape[1]), data.drop(columns=["VSTATUS"]).columns[sorted_idx])
plt.xlabel('Feature Importance (Permutation Importance)')
plt.show()
```

5.1.3. Results

The following graph and table list various predictive features used in a Random Forest machine learning model for predicting liver conditions. The vertical axis represents these features, and their importance to the model is represented by the length of the corresponding horizontal bars. Starting from the top, "avg_CBCI" stands out as the most significant feature. CBCI, possibly referring to complete blood count indices, could reflect key aspects of the patient's overall health and specifically their liver condition, making it a valuable predictor. The next influential feature is "avg_prothrombin_time_blood." Prothrombin time measures how quickly your blood clots, which can be an essential indicator of liver health as the liver produces proteins involved in the clotting process. "avg_GOT_AST" stands next, which is likely referring to the average value of aspartate aminotransferase, an enzyme primarily found in the liver. Elevated levels can suggest liver damage. "Age" is another high-ranking feature, reflecting the fact that the likelihood of developing liver conditions increases as a person gets older. "avg_WBCDC" refers to the White Blood Cell Differential Count, another important health indicator. Elevated levels can indicate infection or inflammation, including in the liver. Further down the list are several variables related to medication use ("used_Lenvima," "used_Recormon_Prefil," "used_Rasitol," "used_Baraclude," "used_Stivarga," and "used Silirin"). These medications may have hepatoprotective or hepatotoxic effects, influencing liver health. The model also accounts for lifestyle factors such as "drinking," acknowledging that alcohol consumption can significantly impact liver health. Finally, the least important variables in this model include "avg_ALFA-FETOPROTEIN" and "avg_Blood_Urea_Nitrogen." Though these biomarkers are associated with liver function, in this model they are less predictive than the factors mentioned earlier.

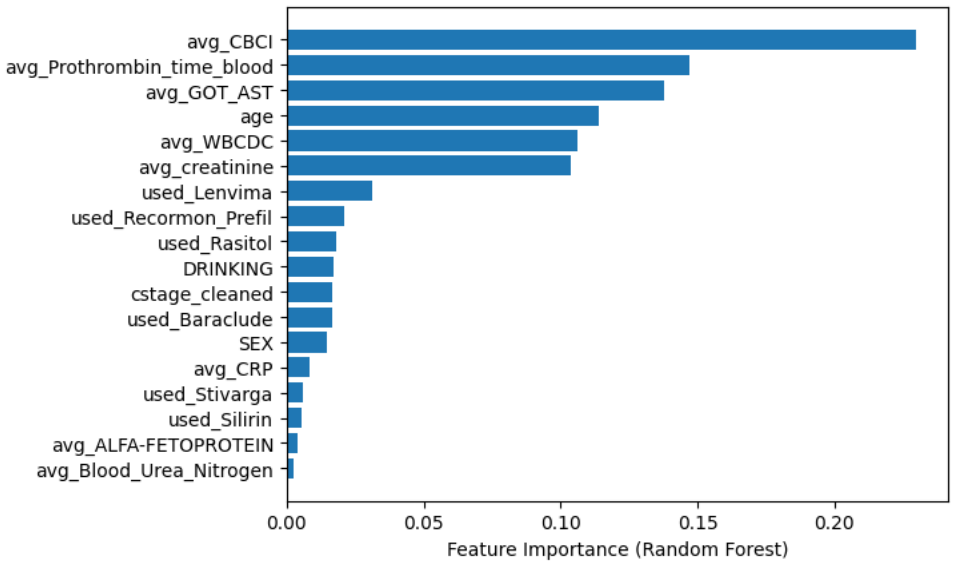


Figure 9: medication and blood tests

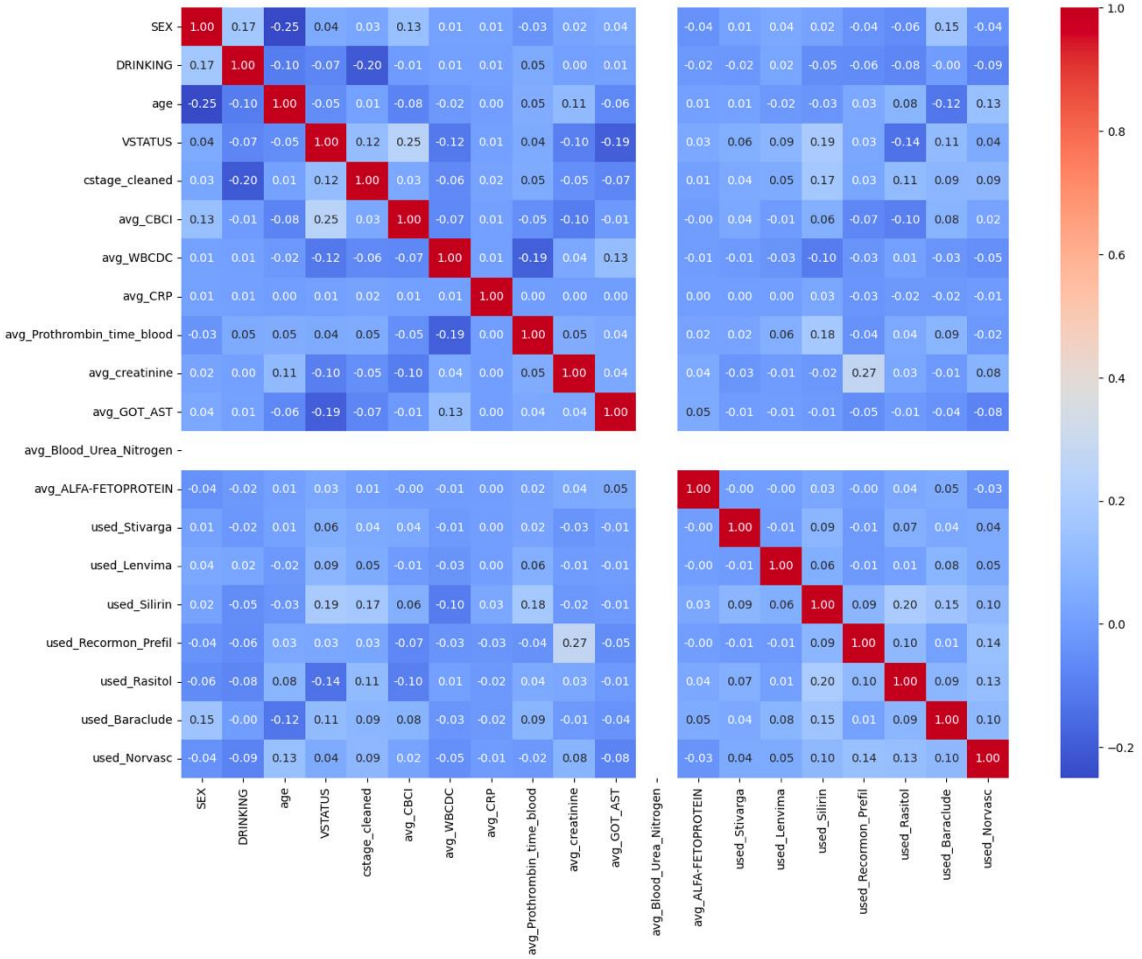


Figure 10:correlation matrix for survival analysis

Feature name	Colleration with died
Sex	0.04
Drinking	-0.07
Age	-0.05
Stage	0.12
Avg CBCI	0.25

Avg WBCDC	-0.12
Avg CRP	0.01
Avg prothrombin time blood	0.04
Avg creatinine	-0.10
Avg got ast	-0.19
Avg alfa fetoprotein	0.03

Figure 11:Survival risk factors

There is only stivarga that has completely negative correlation with the cancer death but the other

Medication name	Correlation with death
Stivarga	-014

Figure 12:Medication Correlation

5.2. Diseases and demographics associate with liver cancer for healthy and sick

In this section we are going to perfom pressosesing to the datasets to join datasets that contains information about the age , the sex , the marry status and other demographics characteristics along with the datasets that contains information's about the diseases that may have impact for someone develop liver cancer . The research methodology uses Python, PySpark, and Apache Spark User-Defined Functions (UDFs) to process healthcare data associated with diverse conditions. It specifically employs International Classification of Diseases (ICD) codes. These names that contains both characters and numbers are given to every known disease and symptom. Furthermore the framework includes various arrays, each corresponding to a distinct medical condition such as cirrhosis ,hepatitis b among others. Each condition is mapped to a set of ICD-9 and ICD-10 codes, which are key in identifying the presence of these conditions within patients' medical records. After the array definition, the methodology employs User-Defined Functions (UDFs) to check the existence of certain ICD codes within a patient's medical data. UDFs are custom functions that can be used within the Spark ecosystem. The UDF, contains_any_code, returns a Boolean value (True/False), depending on the presence or absence of any specified ICD codes in the data it processes. Then we utilize various machine learning models like gradient boosting regressor , support vector machine .We also use random forest classifier to all the diseases in order to find more risk factors .

5.2.1. Data cleaning and mapping

The provided code snippet implements a series of functions and user-defined functions (UDFs) to analyze medical data related to liver cancer. The code aims to identify specific medical conditions, such as obesity, non-alcoholic fatty liver disease (NAFLD), cirrhosis, hemochromatosis, hepatitis C, hepatitis B, and diabetes, within the dataset.
To begin, the code reads in three separate CSV files containing basic information about patients with liver cancer. The data is loaded into Spark DataFrames, with specific columns selected for further analysis. The code then renames columns in each DataFrame to avoid any potential naming conflicts.

Next, the code performs a series of joins on the three DataFrames to create a merged DataFrame. This merged DataFrame combines the information from the three separate sources, ensuring that each patient's data is consolidated into a single row. The coalesce function is used to select the most recent value for each column from the three DataFrames, prioritizing the values from the "w" DataFrame, followed by the "s" DataFrame, and finally the "t" DataFrame.Following the merge, the code applies additional transformations to the data. It uses various functions from the pyspark.sql module, such as last, to_date, row_number, and when, to manipulate the data and create new columns. For instance, a new column "y" is added to indicate whether a patient has been diagnosed with liver cancer based on the presence of their "CHR_NO" in the cancerfinal DataFrame. Rows where the "opd_year" is greater than or equal to the "diag_year" are filtered out.The code also includes a window specification to order the data by "OPD_DATE_NEW" and fill null values for specific columns with the last non-null value within each patient's history. Finally, the code retains only the most recent record for each patient by assigning a row number and filtering for rows where the row number is equal to 1.Overall, the code provides a systematic approach to analyze medical data related to liver cancer. By using UDFs and various Spark functions, it allows for the identification and categorization of patients based on specific medical conditions. This analysis can provide valuable insights into the relationship between these conditions and liver cancer, aiding in further research and medical decision-making.

```
chr_basict = spark.read.csv(header=True, inferSchema=True,
path=fchr_basict)

chr_basics = chr_basics.select("CHR_NO", "BIRTH_DATE", "BLOOD_TYPE",
"MER_FLAG", "RH_TYPE", "SEX_TYPE", "EDU_CODE","UPD_DATE")
chr_basicw = chr_basicw.select("CHR_NO", "BIRTH_DATE", "BLOOD_TYPE",
"MER_FLAG", "RH_TYPE", "SEX_TYPE", "EDU_CODE","UPD_DATE")
chr_basict = chr_basict.select("CHR_NO", "BIRTH_DATE", "BLOOD_TYPE",
"MER_FLAG", "RH_TYPE", "SEX_TYPE", "EDU_CODE","UPD_DATE")


#opd_medt1 = opd_medt
for column in chr_basict.columns:
    if column != 'CHR_NO':
        chr_basict = chr_basict.withColumnRenamed(column, column +
'_t')


for column in chr_basicw.columns:
    if column != 'CHR_NO':
        chr_basicw = chr_basicw.withColumnRenamed(column, column +
'_w')



for column in chr_basics.columns:
    if column != 'CHR_NO':
```

```
        chr_basics = chr_basics.withColumnRenamed(column, column +
'_s')
```

```
f_merged21 = df_merged11.join(chr_basict, on="CHR_NO",
how="full_outer")

from pyspark.sql.functions import coalesce

df_mergechrbasic12 = df_merged21.select(
    'CHR_NO',
    coalesce('BIRTH_DATE_w', 'BIRTH_DATE_s',
'BIRTH_DATE_t').alias('BIRTH_DATE'),
    coalesce('BLOOD_TYPE_w', 'BLOOD_TYPE_s',
'BLOOD_TYPE_t').alias('BLOOD_TYPE'),
    coalesce('MER_FLAG_w', 'MER_FLAG_s',
'MER_FLAG_t').alias('MER_FLAG'),
    coalesce('RH_TYPE_w', 'RH_TYPE_s',
'RH_TYPE_t').alias('RH_TYPE'),
    coalesce('SEX_TYPE_w', 'SEX_TYPE_s',
'SEX_TYPE_t').alias('SEX_TYPE'),
    coalesce('EDU_CODE_w', 'EDU_CODE_s',
'EDU_CODE_t').alias('EDU_CODE'),
    coalesce('UPD_DATE_w', 'UPD_DATE_s',
'UPD_DATE_t').alias('UPD_DATE')
)
```

...

```
chrbasicall = chrbasicall.withColumn("BIRTH_DATE_NORMAL",
udf_map_date3(col("BIRTH_DATE")))
from pyspark.sql.functions import col, floor

chrbasicall = chrbasicall.withColumn('birth_year',
floor(col('BIRTH_DATE') / 10000))
chrbasicall = chrbasicall.withColumn('upd_year',
floor(col('UPD_DATE') / 10000))

chrbasicall = chrbasicall.withColumn('age', col('upd_year') -
col('birth_year'))
```

```
chr_basics = chr_basics.select("CHR_NO", "BIRTH_DATE", "BLOOD_TYPE",
"MER_FLAG", "RH_TYPE", "SEX_TYPE", "EDU_CODE","UPD_DATE")
chr_basicw = chr_basicw.select("CHR_NO", "BIRTH_DATE", "BLOOD_TYPE",
"MER_FLAG", "RH_TYPE", "SEX_TYPE", "EDU_CODE","UPD_DATE")
chr_basict = chr_basict.select("CHR_NO", "BIRTH_DATE", "BLOOD_TYPE",
"MER_FLAG", "RH_TYPE", "SEX_TYPE", "EDU_CODE","UPD_DATE")
```

```
#opd_medt1 = opd_medt
for column in chr_basict.columns:
    if column != 'CHR_NO':
        chr_basict = chr_basict.withColumnRenamed(column, column +
'_t')


for column in chr_basicw.columns:
    if column != 'CHR_NO':
        chr_basicw = chr_basicw.withColumnRenamed(column, column +
'_w')



for column in chr_basics.columns:
    if column != 'CHR_NO':
        chr_basics = chr_basics.withColumnRenamed(column, column +
'_s')
```

```
df_merged21 = df_merged11.join(chr_basict, on="CHR_NO",
how="full_outer")

from pyspark.sql.functions import coalesce

df_mergechrbasic12 = df_merged21.select(
    'CHR_NO',
    coalesce('BIRTH_DATE_w', 'BIRTH_DATE_s',
'BIRTH_DATE_t').alias('BIRTH_DATE'),
    coalesce('BLOOD_TYPE_w', 'BLOOD_TYPE_s',
'BLOOD_TYPE_t').alias('BLOOD_TYPE'),
    coalesce('MER_FLAG_w', 'MER_FLAG_s',
'MER_FLAG_t').alias('MER_FLAG'),
    coalesce('RH_TYPE_w', 'RH_TYPE_s',
'RH_TYPE_t').alias('RH_TYPE'),
    coalesce('SEX_TYPE_w', 'SEX_TYPE_s',
'SEX_TYPE_t').alias('SEX_TYPE'),
    coalesce('EDU_CODE_w', 'EDU_CODE_s',
'EDU_CODE_t').alias('EDU_CODE'),
    coalesce('UPD_DATE_w', 'UPD_DATE_s',
'UPD_DATE_t').alias('UPD_DATE')
)
...
```

```
chrbasicall = chrbasicall.withColumn("BIRTH_DATE_NORMAL",
udf_map_date3(col("BIRTH_DATE")))
from pyspark.sql.functions import col, floor

chrbasicall = chrbasicall.withColumn('birth_year',
floor(col('BIRTH_DATE') / 10000))
```

```
chrbasicall = chrbasicall.withColumn('upd_year', floor(col('UPD_DATE')
/ 10000))

chrbasicall = chrbasicall.withColumn('age', col('upd_year') -
col('birth_year'))
from pyspark.sql.functions import last, to_date, row_number
from pyspark.sql.window import Window
from pyspark.sql.functions import col, when

from pyspark.sql.functions import col, when

# Create a DataFrame with unique CHR_NO values and diag_year in the
cancerfinal DataFrame
cancer_chrs = cancerfinal.select("CHR_NO", "diag_year").distinct()

# Add a new column 'y' to the opdall DataFrame based on the presence
of 'CHR_NO' in the cancer_chrs list
opdall = opdall.join(cancer_chrs, on="CHR_NO", how="left_outer")
opdall = opdall.withColumn("y", when(col("diag_year").isNull(),
0).otherwise(1))

# Replace diag_year with null when y is 0
opdall = opdall.withColumn("diag_year", when(col("y") == 0,
None).otherwise(col("diag_year")))
```

```
# Filter rows based on the condition (y == 1 and opd_year <
diag_year) or y == 0
opdall_filtered = opdall.filter(
    (col("y") == 1) & (col("opd_year") < col("diag_year")) |
(col("y") == 0)
)

# If you want to keep rows where opd_year < diag_year doesn't exist
for a CHR_NO with y == 1
unique_chr_y1 = opdall.filter((col("y") == 1) & (col("opd_year") <
col("diag_year"))).select("CHR_NO").distinct()
unique_chr_y1 = unique_chr_y1.withColumn("keep",
col("CHR_NO").isNotNull())
opdall = opdall.join(unique_chr_y1, on="CHR_NO", how="left_outer")
opdall_filtered = opdall.filter((col("keep").isNull()) |
(col("opd_year") < col("diag_year")))
opdall_filtered = opdall_filtered.drop("keep")

# Convert OPD_DATE_NEW to date type
opdall_filtered = opdall_filtered.withColumn("OPD_DATE_NEW",
to_date("OPD_DATE_NEW"))

# Define the window specification
```

```
window_spec =
Window.partitionBy("CHR_NO").orderBy("OPD_DATE_NEW").rowsBetween(Win
dow.unboundedPreceding, 0)

# Fill null values for each column
for col_name in ["ICD10_CODE1", "ICD10_CODE2", "ICD10_CODE3",
"ICD10_CODE4", "ICD10_CODE5", "ICD9_CODE1", "ICD9_CODE2",
"ICD9_CODE3", "ICD9_CODE4", "ICD9_CODE5"]:
    opdall_filtered = opdall_filtered.withColumn(col_name,
last(col_name, ignorenulls=True).over(window_spec))

# Keep the last date value for every CHR_NO
opdall_recent = opdall_filtered.withColumn(
    "row_num",
    row_number().over(

Window.partitionBy("CHR_NO").orderBy(opdall_filtered.OPD_DATE_NEW.de
sc())
    )
).filter("row_num == 1").drop("row_num")
```

```
from pyspark.sql.functions import col

opd_basic_s = spark.read.csv(fopd_basic_s, header=True,
inferSchema=True) \
    .select("CHR_NO", "ICD10_CODE1", "ICD10_CODE2", "ICD10_CODE3",
"ICD10_CODE4", "ICD10_CODE5",
            "ICD9_CODE1", "ICD9_CODE2", "ICD9_CODE3", "ICD9_CODE4",
"ICD9_CODE5", "OPD_DATE","END_DATE")

opd_basic_t= spark.read.csv(fopd_basic_t, header=True,
inferSchema=True) \
    .select("CHR_NO", "ICD10_CODE1", "ICD10_CODE2", "ICD10_CODE3",
"ICD10_CODE4", "ICD10_CODE5",
            "ICD9_CODE1", "ICD9_CODE2", "ICD9_CODE3", "ICD9_CODE4",
"ICD9_CODE5", "OPD_DATE","END_DATE")

opd_basic_w = spark.read.csv(fopd_basic_w, header=True,
inferSchema=True) \
    .select("CHR_NO", "ICD10_CODE1", "ICD10_CODE2", "ICD10_CODE3",
"ICD10_CODE4", "ICD10_CODE5",
            "ICD9_CODE1", "ICD9_CODE2", "ICD9_CODE3", "ICD9_CODE4",
"ICD9_CODE5", "OPD_DATE","END_DATE")

opdall = opd_basic_s.union(opd_basic_t).union(opd_basic_w)
```

5.2.2. Mapping and Feature extraction

The following code implements a series of functions and user-defined functions (UDFs) to analyze medical data related to liver cancer. The purpose is to identify if specific medical conditions, such as obesity, non-alcoholic fatty liver disease (NAFLD), cirrhosis, hemochromatosis, hepatitis C, hepatitis B, and diabetes, exists for every patient the dataset.
First the function "contains_any_code" checks if a given string contains any of the provided medical codes. This function is used to create UDFs for each medical condition, such as "contains_any_obesity_code_udf" and "contains_any_diabetes_code_udf". These UDFs take a string parameter and return a boolean value indicating whether the string contains any of the corresponding medical codes.The methodology also includes a function called "create_new_column" that takes a DataFrame, UDF function, and a new column name as input. This function creates a new column in the DataFrame based on the evaluation of the UDF function on the ICD9 and ICD10 code columns. Then the function applies the UDF to each code column and combines the results using the logical OR operator. The resulting combined condition is used to create the new column with binary values (1 or 0) based on whether the condition is met or not.The code then applies the "create_new_column" function to the "liver_cancer_df4" DataFrame for each medical condition, creating new columns for obesity, NAFLD, cirrhosis, hemochromatosis, hepatitis C, hepatitis B, and diabetes that contains 0-1 values for existent or not existent. This process enables the identification and categorization of patients based on the presence or absence of these medical conditions. By utilizing UDFs and the "create_new_column" function, it allows for the identification and categorization of patients based on specific medical conditions. This step is very crucial for simplify the machine learning models following , and is generalized and uses in many use cases senario .

```
def contains_any_code(s, codes):
    if s is None:
        return False
    return any(code in s for code in codes)

hepatitis_c_codes = ["070.41", "070.44", "070.51", "070.54",
"B17.1", "B18.2", "B19.20", "B19.21"]
hepatitis_b_codes = ["070.20", "070.22", "070.30", "070.32",
"B16.9", "B18.1", "B19.10", "B19.11"]

contains_any_hepatitis_c_code_udf = udf(lambda s:
contains_any_code(s, hepatitis_c_codes), BooleanType())
contains_any_hepatitis_b_code_udf = udf(lambda s:
contains_any_code(s, hepatitis_b_codes), BooleanType())

diabetes_codes = ["249", "250", "253", "E10", "E11", "E12", "E13",
"E14"]


obesity_codes = ["278.00", "E66","278.00", "278.01"]
nafld_codes = ["571.8", "K76.0","K76.89","K76.0", "K75.81"]
cirrhosis_codes = ["571.5", "571.2","K70.30", "K70.31", "K74.0",
"K74.1", "K74.2", "K74.3", "K74.4", "K74.5", "K74.6"]
hemochromatosis_codes = ["275.0", "E83.110", "E83.111", "E83.112",
"E83.113", "E83.114", "E83.118", "E83.119", "E83.11A", "E83.11B",
"E83.11C"]
```

```
contains_any_diabetes_code_udf = udf(lambda s: contains_any_code(s,
diabetes_codes), BooleanType())
contains_any_obesity_code_udf = udf(lambda s: contains_any_code(s,
obesity_codes), BooleanType())
contains_any_nafld_code_udf = udf(lambda s: contains_any_code(s,
nafld_codes), BooleanType())
contains_any_cirrhosis_code_udf = udf(lambda s: contains_any_code(s,
cirrhosis_codes), BooleanType())
contains_any_hemochromatosis_code_udf = udf(lambda s:
contains_any_code(s, hemochromatosis_codes), BooleanType())

from functools import reduce
from operator import or_

def create_new_column(df, udf_function, new_column_name):
    icd9_cols = [f"ICD9_CODE{i}" for i in range(1, 6)]
    icd10_cols = [f"ICD10_CODE{i}" for i in range(1, 6)]

    conditions = [(udf_function(col(c))) for c in icd9_cols +
icd10_cols]
    combined_conditions = reduce(or_, conditions)
    return df.withColumn(new_column_name, when(combined_conditions,
1).otherwise(0))


liver_cancer_df4 = create_new_column(liver_cancer_df4,
contains_any_obesity_code_udf, "obesity")
liver_cancer_df4 = create_new_column(liver_cancer_df4,
contains_any_nafld_code_udf, "nafld")
liver_cancer_df4 = create_new_column(liver_cancer_df4,
contains_any_cirrhosis_code_udf, "cirrhosis")
liver_cancer_df4 = create_new_column(liver_cancer_df4,
contains_any_hemochromatosis_code_udf, "hemochromatosis")
liver_cancer_df4 = create_new_column(liver_cancer_df4,
contains_any_hepatitis_c_code_udf, "hepatitis_c")
liver_cancer_df4 = create_new_column(liver_cancer_df4,
contains_any_hepatitis_b_code_udf, "hepatitis_b")
liver_cancer_df4 = create_new_column(liver_cancer_df4,
contains_any_diabetes_code_udf, "has_diabetes")
```

..

	age	MER_FLAG	BLOOD_TYPE	SEX_TYPE	EDU_CODE	y	obesity	nafld	cirrhosis	hemochromatosis	hepatitis_c	hepatitis_b	has_diabetes
0	34.0	None	None	0	NaN	0	0	0	0	0	0	0	0
1	54.0	None	None	0	NaN	0	0	0	0	0	0	0	0
2	82.0	N	B	1	NaN	1	0	0	1	0	0	0	0
3	0.0	N	None	0	NaN	0	0	0	0	0	0	0	0
4	23.0	N	None	1	NaN	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
30434	49.0	None	None	1	NaN	0	0	0	0	0	0	0	0
30435	24.0	None	None	1	NaN	0	0	1	0	0	0	0	0
30436	58.0	N	None	0	NaN	0	0	0	0	0	0	0	0
30437	72.0	Y	B	0	NaN	1	0	0	1	0	0	0	0
30438	37.0	N	None	0	NaN	0	0	0	0	0	0	0	0

30439 rows × 13 columns

Figure 13:dataset schema for analysis 1

5.2.3. Machine learning model

The code presented showcases a comprehensive data analysis pipeline that addresses the issue of class imbalance in datasets, preprocesses the data, trains a machine learning model, and analyzes the importance of features in predicting the target variable. By importing essential libraries such as pandas, numpy, sklearn, and resample, the code ensures access to powerful tools for data manipulation, numerical computation, machine learning, and resampling techniques.

One of the primary challenges in working with imbalanced datasets is the unequal representation of classes in the target variable. This can lead to models that perform poorly on the minority class. To overcome this challenge, the code employs the RandomOverSampler technique from the imbalanced-learn package. This technique duplicates instances of the minority class, creating a more balanced training dataset. By addressing the class imbalance, the model can better capture the patterns and relationships within the data.

The application also focuses on data preprocessing, a crucial step in preparing the data for analysis. It converts the target variable to integer type, ensuring compatibility with machine learning algorithms. Additionally, it handles missing data using the SimpleImputer, which replaces missing values with the mean of the respective column. These preprocessing steps help ensure the data is in the correct format and contains no missing values, enabling a more accurate analysis.

The machine learning model used in this pipeline is the Support Vector Machine Classifier (SVC) from the sklearn library. This classifier is trained using the oversampled data, allowing it to learn from both classes more effectively. Once trained, the model is used to predict outcomes for the test data, providing insights into the performance of the model on unseen data.

To gain a deeper understanding of the dataset, the code provides visualizations of feature correlations. By generating a heatmap of the correlation matrix, it offers a comprehensive overview of the relationships between different features. Additionally, a bar chart is created to display the features in descending order of their correlation with the target variable. These visualizations help identify the most influential features and their impact on predicting the target variable.

In conclusion, the provided code offers a robust and comprehensive workflow for handling class imbalance, preprocessing data, training a machine learning model, and analyzing feature correlations. By addressing class imbalance and understanding the importance of features, researchers and practitioners can gain valuable insights and make informed decisions. This code serves as a valuable resource for data analysis tasks in domains where imbalanced datasets are encountered.

5.2.4. Results

1. Correlation Matrix:

examining the correlation coefficients, it is evident that age exhibits the strongest correlation with the target variable. With a coefficient of 0.306344, it suggests that as age increases, there is a corresponding increase in the likelihood of the target variable. Another noteworthy feature is "MER_FLAG," which demonstrates a positive correlation with the target variable. However, the correlation coefficient of 0.163971 is weaker compared to age. This implies that while MER_FLAG may have some influence on the outcome, its impact is not as significant as age.

Additionally, the feature "SEX_TYPE" also displays a positive correlation with the target variable. This suggests that gender plays a role in influencing the outcome. However, it is important to note that the correlation coefficient for SEX_TYPE is not as strong as age, indicating that other factors may contribute more significantly to the target variable.

Other variables, such as obesity, nafld, cirrhosis, hemochromatosis, hepatitis_c, hepatitis_b, and has_diabetes, also exhibit some level of correlation with the target variable. However, the correlation coefficients for these variables are generally lower than those for age, MER_FLAG, and SEX_TYPE. This suggests that while these variables may have some influence on the outcome, their impact is less pronounced compared to the aforementioned features.

It is worth mentioning that the variable "hemochromatosis" shows a NaN correlation coefficient. This indicates the possibility of missing or constant values for this feature across the dataset. It is important to address this issue and ensure that the data for hemochromatosis is complete and reliable before drawing any conclusions about its correlation with the target variable.

In conclusion, the correlation matrix provides valuable insights into the relationships between features and their association with the target variable. Age emerges as the most influential feature, followed by MER_FLAG and SEX_TYPE. While other variables also show some level of correlation with the target variable, their impact is generally less pronounced. It is crucial to address any missing or unreliable data to ensure accurate analysis and interpretation.

2. **Feature Importance (Gradient Boosting):**

The analysis of feature importance in the Gradient Boosting model reveals valuable insights into the contribution of each feature towards predicting the target variable. Among the features, "age" emerges as the most influential, with a high importance score of approximately 0.705. This suggests that age plays a significant role in accurately predicting the outcome. The strong correlation between age and the target variable implies that as age increases, there is a corresponding increase in the likelihood of the target variable.
Other variables, such as "SEX_TYPE", "nafld", "hepatitis_c", "MER_FLAG", "hemochromatosis", "obesity", "hepatitis_b", and "y", exhibit lower importance scores in the Gradient Boosting model. This indicates that their contributions to predicting the outcome are comparatively lesser than that of age. While these features still have some

level of influence on the model's predictions, their impact is not as pronounced as that of age.To further validate the importance of features, permutation importance is employed. This technique measures the decrease in model performance when the values of a feature are randomly shuffled. Consistent with the feature importance analysis, age once again demonstrates the highest permutation importance score. This reaffirms the significance of age in accurately predicting the outcome, as shuffling its values has a greater impact on the model's performance compared to other features.Based on this analysis, it can be inferred that age is the most crucial feature in predicting the target variable according to the Gradient Boosting model. While other features also contribute to the predictions, their influence is comparatively lower. Addressing the issues or considering the impact of age on the outcome could potentially have the highest impact in influencing the predictions.

It is important to note that these interpretations are specific to the Gradient Boosting model and the dataset used in this analysis. The actual feature importance and correlations may vary depending on the characteristics and distribution of the dataset, as well as the assumptions and specifics of the model being utilized. To ensure robustness, it is recommended to validate these results using additional models and approaches.

```
Gradient Boosting AUC-ROC: 0.85 (+/- 0.01)
```

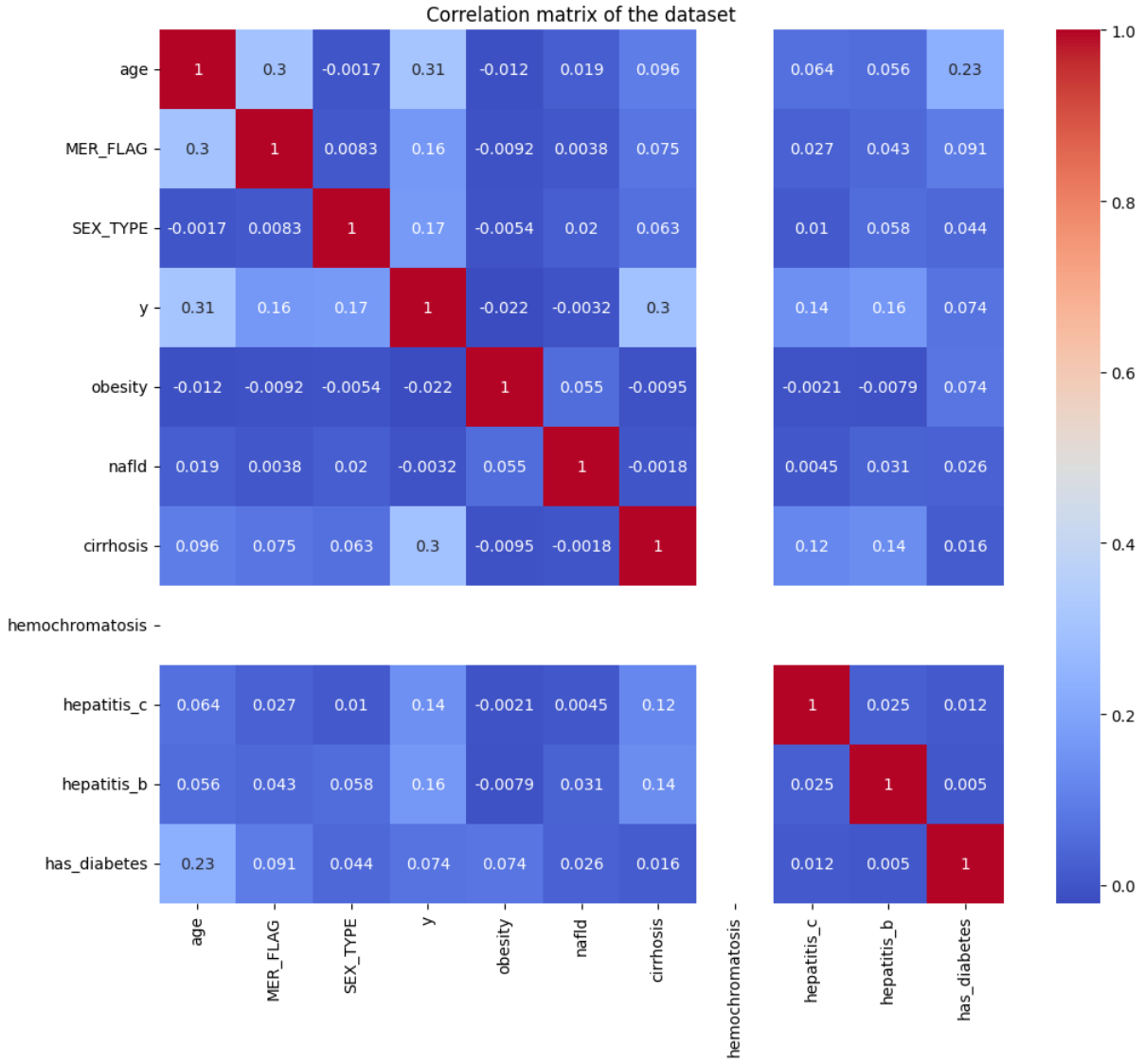


Figure 14:Correlation matrix demographics and diseases

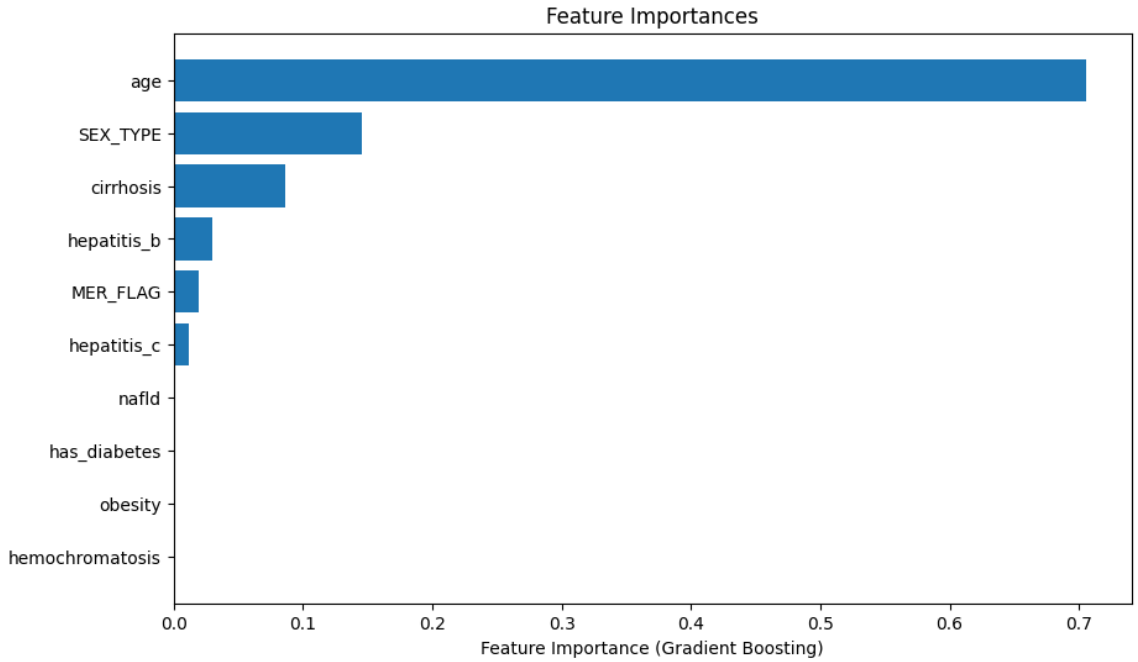


Figure 15:Feature importance gradient boosting demographics and diseases

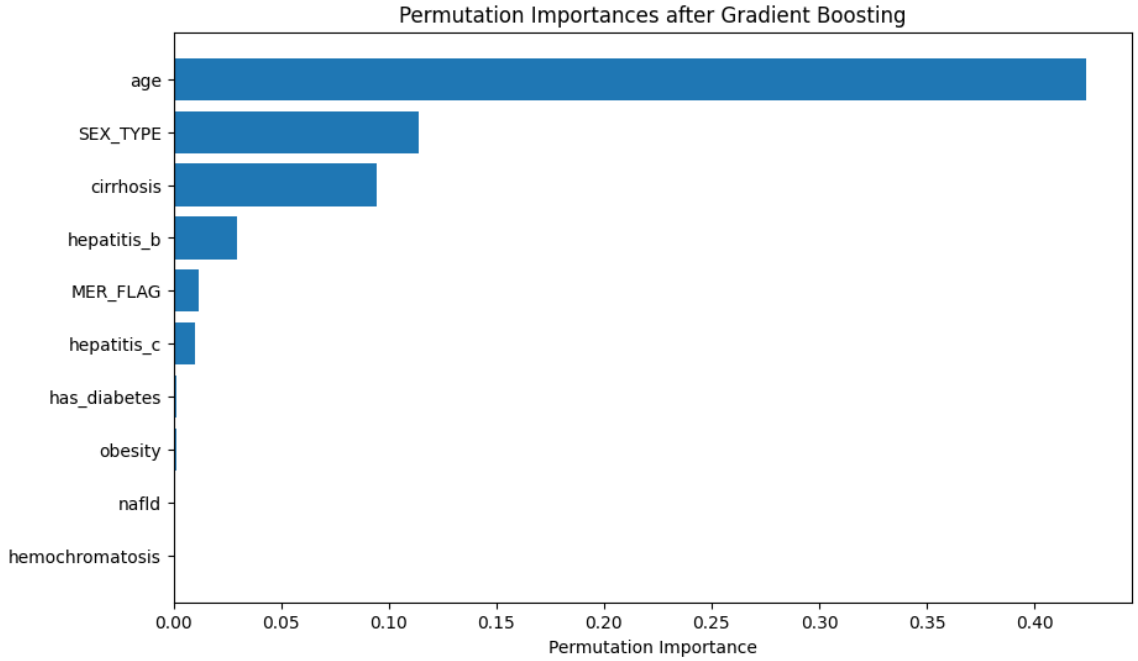


Figure 16:Permutation importance gradient boosting

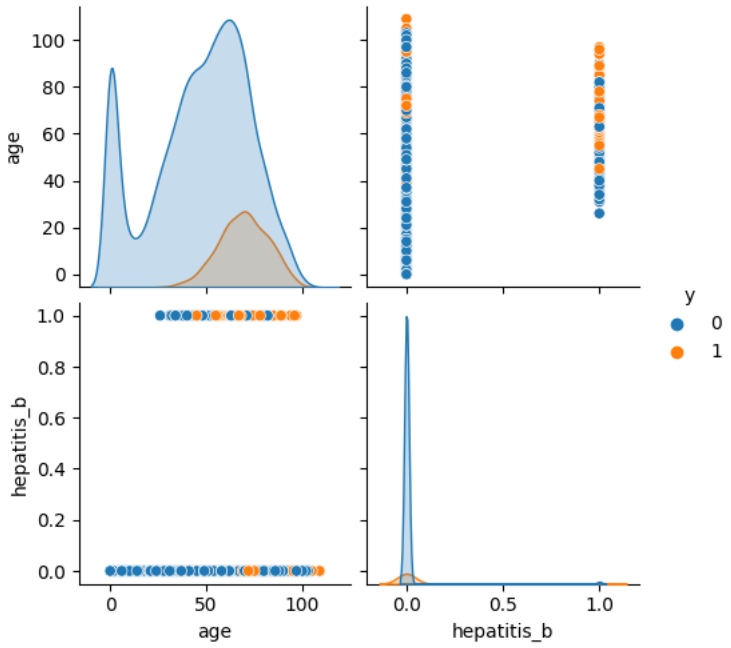


Figure 17:shap values age combine with hepatitis b

5.1. Liver cancer risk factor random forest classifier analysis for healthy and sick Data preparation

From the diseases dataset that has been created before ,the following keeps onlt the icd9 and icd10 columns and join them in one column with the name combined_diag_codes .Then splits the data

```
from pyspark.sql import functions as F

selected_data = opd_all_analysis.select(['ICD10_CODE1',
'ICD10_CODE2', 'ICD10_CODE3', 'ICD10_CODE4', 'ICD10_CODE5',
'ICD9_CODE1', 'ICD9_CODE2', 'ICD9_CODE3', 'ICD9_CODE4',
'ICD9_CODE5', 'y'])
selected_data = selected_data.withColumn('combined_diag_codes',
F.concat_ws(',', 'ICD10_CODE1', 'ICD10_CODE2', 'ICD10_CODE3',
'ICD10_CODE4', 'ICD10_CODE5', 'ICD9_CODE1', 'ICD9_CODE2',
'ICD9_CODE3', 'ICD9_CODE4', 'ICD9_CODE5'))
selected_data = selected_data.select(['combined_diag_codes', 'y'])
selected_pd_data = selected_data.toPandas()

subset_pd_data= selected_pd_data
subset_pd_data['diag_codes_list'] =
subset_pd_data['combined_diag_codes'].apply(lambda x: x.split(','))
```

5.1.1. Machine learning model random forest classifier for diseases classification

The following code presents the process of preparing the dataset for classification analysis by performing one-hot encoding on the labels, splitting the data into training and testing sets, and training a RandomForestClassifier model on the training data.
The provided code snippet demonstrates the utilization of several libraries and algorithms to perform classification analysis on a dataset. The code begins by importing necessary libraries, including sklearn.preprocessing.MultiLabelBinarizer, sklearn.model_selection.train_test_split, and sklearn.ensemble.RandomForestClassifier. These libraries provide functionalities for data preprocessing, dataset splitting, and the implementation of the Random Forest Classifier algorithm.
The code then proceeds to create an instance of the MultiLabelBinarizer class from sklearn.preprocessing. This class is used to transform a list of labels into a binary matrix representation, commonly known as one-hot encoding. The 'diag_codes_list' column from the 'subset_pd_data' DataFrame is passed to the fit_transform method of the MultiLabelBinarizer instance to perform the one-hot encoding. The resulting binary matrix is stored in the 'one_hot_diag_codes' variable.
Next, a new DataFrame named 'one_hot_df' is created using the pandas.DataFrame constructor. This DataFrame is initialized with the one-hot encoded matrix, 'one_hot_diag_codes', and the column names are set using the 'classes_' attribute of the MultiLabelBinarizer instance.The 'X' variable is assigned the 'one_hot_df' DataFrame, which represents the features used for classification. The 'y' variable is assigned the 'y' column from the 'subset_pd_data' DataFrame, which represents the target variable if a patient has cancer or not .
The 'train_test_split' function from sklearn.model_selection is then used to split the data into training and testing sets. The 'X' and 'y' variables are passed to this function, along with the 'test_size' parameter set to 0.3, indicating that 30% of the data will be used for testing. Additionally, the 'random_state' parameter is set to 42 to ensure reproducibility of the results.A RandomForestClassifier object named 'clf' is created using the RandomForestClassifier class from sklearn.ensemble. This classifier is initialized with the 'random_state' parameter set to 42 to ensure consistent results. The 'fit' method of the 'clf' object is then called, with the training data ('X_train' and 'y_train') as arguments, to train the classifier on the provided dataset.

```
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier


mlb = MultiLabelBinarizer()
one_hot_diag_codes =
mlb.fit_transform(subset_pd_data['diag_codes_list'])

one_hot_df = pd.DataFrame(one_hot_diag_codes, columns=mlb.classes_)

X = one_hot_df
y = subset_pd_data['y']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

clf = RandomForestClassifier(random_state=42)
```

```
clf.fit(X_train, y_train)
```

Next code utilization of the Random Forest Classifier algorithm to identify the most important diagnostic codes associated with liver cancer. The code begins by importing necessary libraries, including numpy for numerical computations.
The feature importances of the trained classifier, denoted as 'feature_importances', are obtained using the 'feature_importances_' attribute of the 'clf' object. These feature importances represent the relative importance of each diagnostic code in predicting liver cancer.To identify the top 'k' diagnostic codes associated with liver cancer, the 'argsort' function from numpy is used to obtain the indices that would sort the 'feature_importances' array in ascending order. The last 'k' indices, representing the most important diagnostic codes, are selected using the negative indexing method '[-k:]'.The 'top_k_diag_codes' variable is then assigned the column names from the 'one_hot_df' DataFrame corresponding to the top 'k' indices. These column names represent the diagnostic codes that have the highest importance in predicting liver cancer.
Finally, the code prints the top 'k' diagnostic codes associated with liver cancer using a formatted string. The 'top_k_diag_codes' variable is printed to display the selected diagnostic codes.

Overall this code snippet identifies the most important diagnostic codes associated with liver cancer using the Random Forest Classifier algorithm. The feature importances are calculated, and the top 'k' diagnostic codes are determined based on these importances. This information can be valuable in understanding the key factors contributing to liver cancer and can aid in further analysis and decision-making in the medical field.

```
import numpy as np

feature_importances = clf.feature_importances_
k = 40
top_k_indices = np.argsort(feature_importances)[-k:]

top_k_diag_codes = one_hot_df.columns[top_k_indices]

print(f"Top {k} diagnostic codes associated with liver cancer:")
print(top_k_diag_codes)
```

The code snippet utilizes the Random Forest Classifier algorithm to identify the most significant diagnostic codes associated with liver cancer. The code begins by importing the necessary libraries, including numpy for numerical computations.
To determine the feature importances of the trained classifier, the 'feature_importances_' attribute of the 'clf' object is accessed, and the values are stored in the 'feature_importances' variable. These feature importances represent the relative importance of each diagnostic code in predicting liver cancer.To identify the top 'k' diagnostic codes associated with liver cancer, the 'argsort' function from numpy is employed. This function returns the indices that would sort the 'feature_importances' array in ascending order. By using negative indexing with the '[-k:]' notation, the last 'k' indices, which correspond to the most important diagnostic codes, are selected.The 'top_k_diag_codes' variable is then assigned the column names from the 'one_hot_df' DataFrame that correspond to the top 'k' indices. These column names represent the diagnostic codes that have the highest importance in predicting liver cancer.Finally, the code generates a visualization using the matplotlib library. A horizontal bar plot is

created, where the y-axis represents the top 'k' diagnostic codes and the x-axis represents their relative importance. The descriptions of the top diagnostic codes are retrieved from the 'icd_code_dict' dictionary and used as labels on the y-axis. The plot is then displayed, providing a visual representation of the top diagnostic codes associated with liver cancer. In summary, this code snippet applies the Random Forest Classifier algorithm to identify the most significant diagnostic codes related to liver cancer. By calculating the feature importances and selecting the top 'k' codes, the code provides valuable insights into the key factors contributing to liver cancer. The generated visualization enhances the interpretability of the results by presenting the top diagnostic codes and their relative importance in a clear and concise manner.

```
import matplotlib.pyplot as plt

# Retrieve the descriptions of the top diagnostic codes
top_k_diag_descriptions = [icd_code_dict[code] if code in
icd_code_dict else 'Unknown code' for code in top_k_diag_codes]

plt.figure(figsize=(10, 6))
plt.barh(range(k), feature_importances[top_k_indices])
plt.yticks(range(k), top_k_diag_descriptions)  # Use descriptions
instead of codes
plt.xlabel('Relative Importance')
plt.title('Top {} Diagnostic Codes Associated with Liver
Cancer'.format(k))
plt.show()
```

5.1.2. Results

he feature importance table for liver cancer, in descending order, begins with primary malignant neoplasm of the liver and liver cell carcinoma, which are direct diagnoses of liver cancer and thus serve as the most crucial risk factors. Cirrhosis of the liver without mention of alcohol, malignant neoplasm of the liver and intrahepatic bile ducts, and unspecified chronic hepatitis follow closely, all representing significant liver conditions known to escalate the risk of developing liver cancer. Similarly, malignant neoplasms of intrahepatic bile ducts and intrahepatic bile duct carcinoma underscore the severity of bile duct involvement in liver cancer progression. Conditions such as chronic viral hepatitis B and unspecified cirrhosis of the liver further emphasize the role of chronic liver disease and inflammation in enhancing liver cancer risk.

Ascites, a condition typically resulting from advanced cirrhosis or liver cancer, appears in the mid-range of the list, along with secondary malignant neoplasm of the lung, indicative of metastatic cancer spread, although not directly linked to liver cancer initiation. The presence of both viral hepatitis B and chronic liver diseases, along with cachexia—a syndrome associated with advanced cancer stages—underline the cumulative toll of viral infections and overall health deterioration on liver cancer development.

Towards the bottom of the table, we encounter conditions less directly associated with liver cancer but still of relevance. Hepatic encephalopathy and reflux esophagitis may signify advanced liver disease and compromised gastrointestinal health, respectively. Moreover, the presence of type 2 diabetes mellitus without complications and mixed hyperlipidemia acknowledges the indirect influence of metabolic disorders on liver cancer risk, possibly through promoting fatty liver disease and cirrhosis. Each entry in

this feature importance list collectively contributes to a comprehensive understanding of the myriad factors implicated in liver cancer risk and progression.

Diagnostic Code	Explanation
Malignant neoplasm of liver, primary	This is a diagnosis of primary liver cancer itself, making it the most direct risk factor.
Liver cell carcinoma	This is a specific type of primary liver cancer, also known as hepatocellular carcinoma.
Cirrhosis of liver without mention of alcohol	Cirrhosis is severe scarring of the liver often seen in late-stage liver disease. It's a significant risk factor for liver cancer.
Malignant neoplasm of liver and intrahepatic bile ducts	This diagnosis indicates cancer that started in either the liver or bile ducts (tubes that carry bile from the liver to the small intestine).
Chronic hepatitis, unspecified	Chronic hepatitis is long-term inflammation of the liver, which can lead to cirrhosis and increase the risk of liver cancer.
Malignant neoplasm of intrahepatic bile ducts	This is a diagnosis of cancer that has started in the bile ducts within the liver.
Intrahepatic bile duct carcinoma	This is another term for cancer in the bile ducts inside the liver.
Unspecific cirrhosis of liver	Cirrhosis, regardless of cause, increases the risk of liver cancer.
Chronic viral hepatitis B	Chronic infection with hepatitis B virus can cause liver inflammation and cirrhosis, increasing liver cancer risk.
Ascites	This condition, which involves fluid buildup in the abdomen, can be a result of cirrhosis or liver cancer.
Secondary malignant neoplasm of lung	This indicates cancer that has spread (metastasized) from the lung to other parts of the body. It doesn't directly increase liver cancer risk, but indicates an advanced stage of cancer in the body.
Viral hepatitis B	Similar to chronic viral hepatitis B, infection with hepatitis B virus is a major risk factor for liver cancer.
Chronic liver diseases and cirrhosis	Chronic liver diseases can lead to cirrhosis and subsequent liver cancer.
Cachexia	This is a condition characterized by extreme weight loss and muscle wasting, often seen in advanced cancer stages.
Chronic viral hepatitis C	Like with hepatitis B, chronic hepatitis C can also lead to cirrhosis and increase the risk of liver cancer.
Hepatic encephalopathy	A complication of severe liver disease, it suggests advanced cirrhosis but isn't a direct risk factor for liver cancer.
Reflux esophagitis	Inflammation of the esophagus due to acid reflux. While it isn't a direct risk factor for liver cancer, it might be linked with overall gastrointestinal health.
Chronic hepatitis C without mention of coma	Again, chronic hepatitis C increases the risk of cirrhosis and liver cancer.
Diabetes mellitus 2 without mention of complication	People with type 2 diabetes have a higher risk of liver cancer, possibly due to the links between cirrhosis, fatty liver disease, and insulin resistance.
Mixed Hyperlipidemia	High levels of fats (lipids) in the blood are linked with fatty liver disease and indirectly increase liver cancer risk through promoting cirrhosis.

Figure 18:Diagnostic Codes and Descriptions

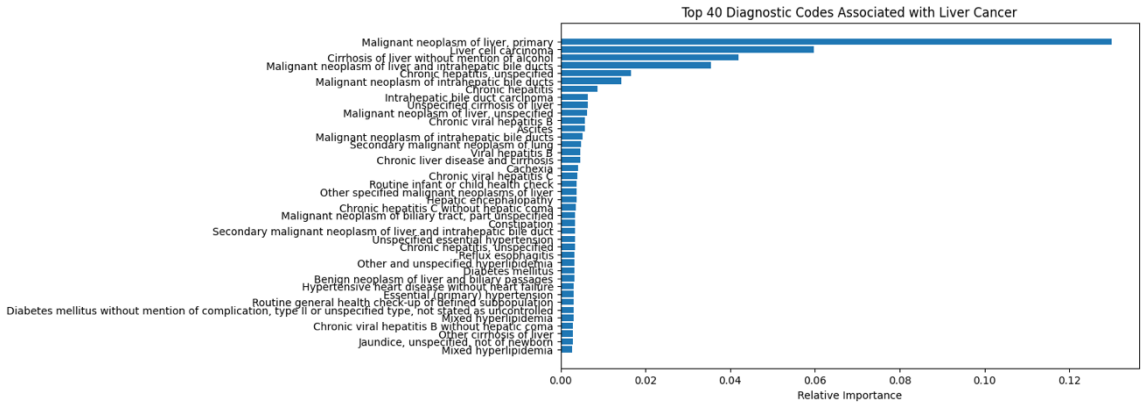


Figure 19: Top 40 Diagnostic Codes Associated with Liver Cancer

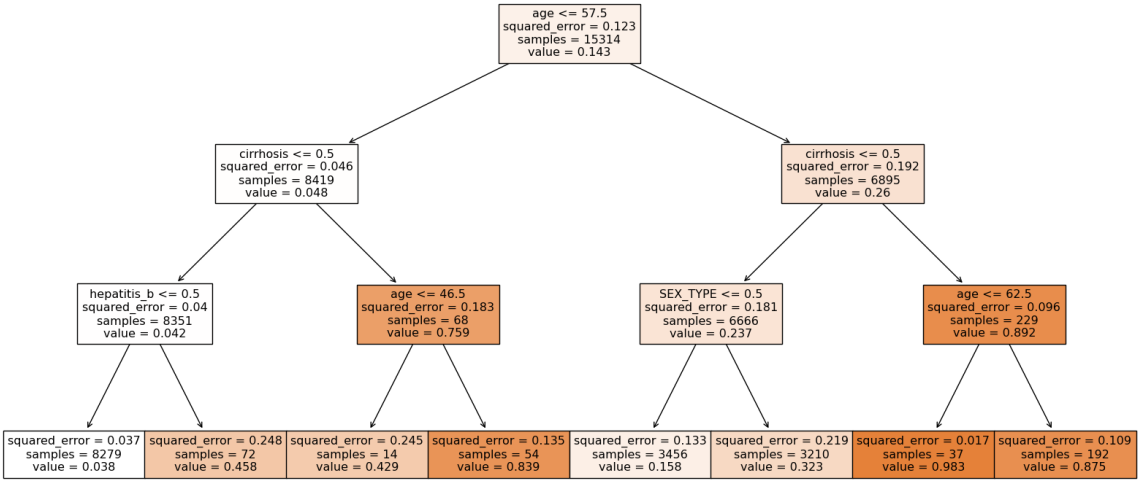


Figure 20:Desicion tree

5.2. Feature importance random forest classifier by removing the cancer values.

The following results show the random forest importance after we remove the pro existing liver cancer values

Among the most impactful conditions related to liver cancer are "Cirrhosis of the liver without mention of alcohol," "Chronic hepatitis, unspecified," and "Chronic viral hepatitis B," all of which directly pertain to liver health. "Essential (primary) hypertension" and "Unspecified essential hypertension" could have an indirect but significant impact on liver health, affecting systemic conditions. Conditions like "Secondary malignant neoplasm of lung" and "Diabetes mellitus" suggest that liver cancer is often associated with other serious health issues. Lower on the scale of direct impact are conditions like "Constipation," "Reflux esophagitis," and "Anemia," which may be symptoms or side effects rather than causative factors. "Unspecified functional disorder of stomach" and "Hypertensive heart disease" may also be indicative of overall poor health but are less directly related to liver cancer. Surprisingly, "Insomnia" and "Urinary tract infection" also appear, indicating that a wide range of health issues may correlate with liver cancer, although their direct impact could be minimal. Towards the end, even dental issues like "Chronic Periodontitis" and "Dental Caries" appear, suggesting a holistic approach to health might be crucial for liver cancer prevention or management.

Condition/Symptom	Explanation
Cirrhosis of the liver without mention of alcohol	A liver disease where liver tissue is replaced by scar tissue, not caused by alcohol consumption.
Chronic hepatitis, unspecified	A long-term inflammation of the liver, the exact type or cause is unspecified.
Chronic viral hepatitis B with/without delta-agent	A type of liver infection caused by the hepatitis B virus; the delta-agent is a secondary virus that can infect those already infected with hepatitis B.
Ascites	Abnormal accumulation of fluid in the abdomen.
Secondary malignant neoplasm of lung	Cancer that started in another part of the body and has spread to the lung.
Cachexia	A complex metabolic syndrome associated with underlying illness causing muscle loss and with or without fat loss.
Jaundice, unspecified	Yellowing of the skin or eyes due to high bilirubin levels; exact cause unspecified.
Hepatic coma	A loss of consciousness as a result of liver failure.
Essential hypertension	High blood pressure with no identifiable cause.
Constipation	Difficulty in passing stools or infrequent stools.
Diabetes mellitus	A group of diseases resulting in high blood sugar.
Hyperlipidemia	Abnormally elevated levels of fats in the blood.
Anemia	A decrease in the total amount of red blood cells or hemoglobin in the blood.
Insomnia	Difficulty falling asleep or staying asleep.
Shock	A life-threatening condition that occurs when the body is not getting enough blood flow.
Malaise and Fatigue	A general feeling of discomfort or tiredness.
Hemorrhage of the gastrointestinal tract	Bleeding within the digestive tract.
Cachexia	Severe weight and muscle loss, often associated with chronic illness.
Encounter for radiotherapy	A medical visit for the purpose of receiving radiation treatment.
Abdominal pain	Pain that occurs between the chest and pelvic regions.
Sepsis	A potentially life-threatening condition caused by the body's response to an infection.
Fever	A temporary increase in body temperature, often due to an illness.
Neonatal jaundice	Yellow discoloration of a newborn baby's skin and eyes.

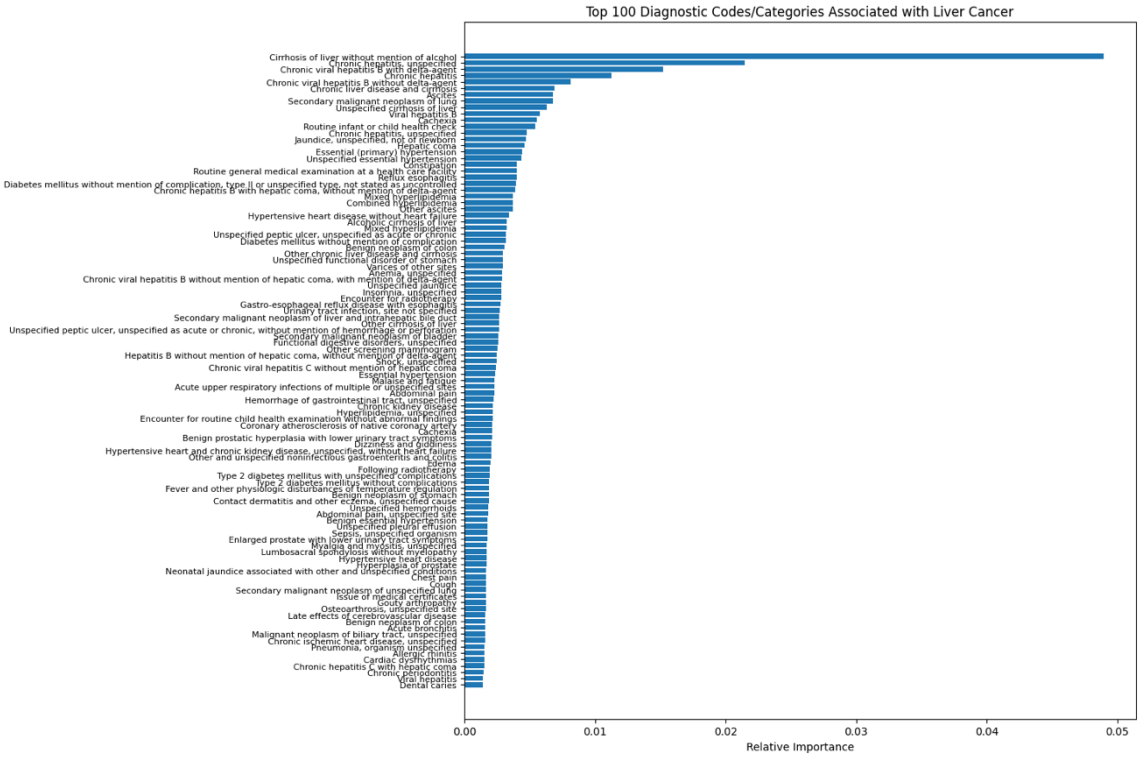
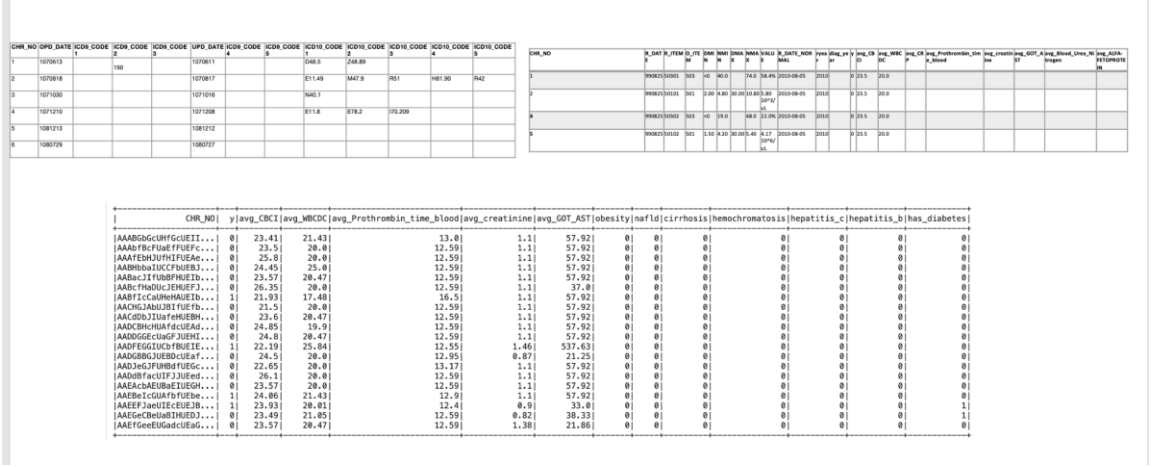


Figure 21:Random forest diseases -100 diagnostic codes associated with liver cancer

5.3. Blood and diseases analysis healthy and sick

In this section we present an analysis combined the proexist conditions of a patient joined with blood tests.The datasets are quite demanding the blood tests are above 10 million data , so we choose specific indicators that may have impact for someone to develop liver cancer .We do feature extraction for every blood test that we choose we create a new column and we add the avg value per patient in this column , if there are records of this patients .For people who have liver cancer (y=1) we only choose blood tests before the diagnosed date .Conditionally for diseases we extract the key features like cirrhosis , cachexia ,hepatitis as we show in previous sections ,but we also run random forest classification algorithms include all features .We joined the two datasets in one and we run several algorithms lightgbm , lasso logistic regression the following schema presents the two joined initial datasets after , cleaning mapping into one that contains both data from blood tests datasets and diseases .



CHR_NO	y	avg_CBCI	avg_WBCDC	avg_Prothrombin_time_blood	avg_creatinine	avg_GOT_AST	obesity	nafld	cirrhosis	hemochromatosis	hepatitis_c	hepatitis_b	has_diabetes
AAABGbGcUHfGcUEII...	0	23.41	21.43	13.0	1.1	57.92	0	0	0	0	0	0	0
AAAbfBcFUaEfFUEFc...	0	23.5	20.0	12.59	1.1	57.92	0	0	0	0	0	0	0
AAAfEbHJUfHIFUEAe...	0	25.8	20.0	12.59	1.1	57.92	0	0	0	0	0	0	0
AABHbbaIUCCFbUEBJ...	0	24.45	25.0	12.59	1.1	57.92	0	0	0	0	0	0	0
AABacJIfUbBFHUEIb...	0	23.57	20.47	12.59	1.1	57.92	0	0	0	0	0	0	0
AABcfHaDUcJEHUEFJ...	0	26.35	20.0	12.59	1.1	37.0	0	0	0	0	0	0	0
AABfIcCaUHeHAUEIb...	1	21.93	17.48	16.5	1.1	57.92	0	0	0	0	0	0	0
AACHGJAbUJBIfUEfb...	0	21.5	20.0	12.59	1.1	57.92	0	0	0	0	0	0	0
AACdDbJIUafeHUEBH...	0	23.6	20.47	12.59	1.1	57.92	0	0	0	0	0	0	0
AADCBHcHUAfdcUEAd...	0	24.85	19.9	12.59	1.1	57.92	0	0	0	0	0	0	0
AADDGGEcUaGFJUEHI...	0	24.8	20.47	12.59	1.1	57.92	0	0	0	0	0	0	0
AADFEGGIUCbfBUEIE...	1	22.19	25.84	12.55	1.46	537.63	0	0	0	0	0	0	0
AADGBBGJUEBDcUEaf...	0	24.5	20.0	12.95	0.87	21.25	0	0	0	0	0	0	0
AADJeGJFUHBdfUEGc...	0	22.65	20.0	13.17	1.1	57.92	0	0	0	0	0	0	0
AADdBfacUIFJJUEed...	0	26.1	20.0	12.59	1.1	57.92	0	0	0	0	0	0	0
AAEAcbAEUBaEIUEGH...	0	23.57	20.0	12.59	1.1	57.92	0	0	0	0	0	0	0
AAEBeIcGUAfbfUEbe...	1	24.86	21.43	12.9	1.1	57.92	0	0	0	0	0	0	0
AAEEFJaeUIEcEUEJB...	1	23.93	20.01	12.4	0.9	33.0	0	0	0	0	0	0	1
AAEGeCBeUaBIHUEDJ...	0	23.49	21.05	12.59	0.82	38.33	0	0	0	0	0	0	1
AAEfGeeEUGadcUEaG...	0	23.57	20.47	12.59	1.38	21.86	0	0	0	0	0	0	0

Figure 22:Cleaning features extractions blood tests-diseases

5.3.1. Data cleaning preprocessing

The Laboratory results codes for blood tests and values that are essential for survival prediction and risk factor analysis are encoded in numeric form along with their corresponding value , a code is used to export from the official pilot site the datatables containing the set values of the csv that will be used for mapping to achieve this "BeutifulSoop" was used to download all the information from the pilot codes, which are the laboratory results and their Chinese description. Later, Google API was used to translate the description into English. The first code retrieves laboratory result codes and their corresponding descriptions from a website using web scraping techniques. The code sends a GET request to the website in order to retrieve its HTML content. It then employs the Python library BeautifulSoup to parse the HTML and extract the table's data. The extracted data is stored in a list that is iterated using a while loop until all website pages have been extracted. For further analysis, the final output is saved as a CSV file. The second line of code uses the Google Translate API to translate the scraped From another language descriptions into English. The code iterates over each row of the original CSV file containing Chinese descriptions using a for loop. The API translates the text in the first and fourth columns of each row, which correspond to laboratory result codes and their descriptions, respectively. The translated data is saved in a new list, which is then appended to a new CSV file. This allows researchers who are not proficient in Chinese to comprehend the descriptions with greater ease.

Next we implemented a code that The provided PySpark script demonstrates a practical application of data preprocessing, specifically in outlier detection and removal, and mean calculation for numerous columns in a DataFrame. First with the help of necessary modules from PySpark's SQL functions, the code defines an outlier-filtering function, then lists the columns to be processed, and later loops over these columns to filter outliers and compute their mean. The averages are stored in a new DataFrame and joined to the original DataFrame, while the original columns are discarded. Thus, the script converts raw data into a more refined format by removing statistical outliers and replacing specific columns with their average values, a procedure often vital in data science and machine learning applications.

```
import requests
from bs4 import BeautifulSoup
import csv

url = 'https://pilotwebsite.’

# Create an empty list to store the data
data = []

# Use a while loop to iterate over each page
while True:
    # Send a GET request to the URL and retrieve the HTML content
    response = requests.get(url)
    html_content = response.content

    # Use BeautifulSoup to parse the HTML content
    soup = BeautifulSoup(html_content, 'html.parser')

    # Find the table in the HTML content
    table = soup.find('table')

    # Extract the data from the table
    for row in table.find_all('tr'):
        cells = row.find_all('td')
        if cells:
            row_data = [cell.text for cell in cells]
            data.append(row_data)

    # Check if there is a link to the next page
    next_link = soup.find('a', {'class': 'next-page-link'})
    if next_link:
        url = next_link['href']
    else:
        break

# Write the data to a CSV file
with open('table.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerows(data)

this is the code for data translate

import csv
import requests

# Replace YOUR_API_KEY with your Google Translate API key
API_KEY = ""

# Read the original CSV file
with open("/content/drive/MyDrive/thesis/table.csv", "r") as original_file:
```

```
    reader = csv.reader(original_file)
    header = next(reader) # read the header
    data = [row for row in reader]

# Translate the text in the first and fourth column of the original file
translated_data = [header] # keep the header
for row in data:
    # Translate the text in the first column
    text1 = row[0]
    url =
f"https://translation.googleapis.com/language/translate/v2?target=en&key={API_KEY}&q={text1}"
    response = requests.get(url).json()
    translated_text1 = response["data"]["translations"][0]["translatedText"]

    # Translate the text in the fourth column
    text2 = row[3]
    url =
f"https://translation.googleapis.com/language/translate/v2?target=en&key={API_KEY}&q={text2}"
    response = requests.get(url).json()
    translated_text2 = response["data"]["translations"][0]["translatedText"]

    # Create a new row with the translated text and the remaining columns
    translated_row = [translated_text1, row[1], row[2], translated_text2, row[4]]
    translated_data.append(translated_row)

# Save the translated text to a new CSV file
with open("translated1.csv", "w", newline="") as translated_file:
    writer = csv.writer(translated_file)
    writer.writerows(translated_data)
```

translated1

生化	152	F09052A	VMA((24小時尿液))(外送)	
biochemical	153	F09053A	17KS((24 hours urine))(delivery)	
biochemical	154	F09054A	17OHC (24 hours urine) (delivery)	
biochemical	155	F09077A	Catecholamine (24-hour urine) (delivery)	
biochemical	156	F09028A	ACP (delivery)	
biochemical	157	F09059A	Lactate (ice bath)	
biochemical	158	F129021	Vit B12	
biochemical	159	F129022	Folate	
biochemical	015B	F090750	plasma free metanephrines (frozen	
biochemical	015D	F10005B	Cadmium (blood) (delivery)	
biochemical	015E	F10003A	Arsenic As (blood) (delivery)	
biochemical	015F	F10003A	Arsenic As (urine) (delivery)	
biochemical	015G	F099014	25 OH Vit D	
biochemical	160	F10002A	Al (delivery)	
biochemical	161	F09047B	Cu (delivery)	
biochemical	162	F09049A	Lead (delivery)	
biochemical	163	F10012B	Zn (delivery)	
biochemical	164	F10520B	Li (stop using and change the blood level of the drug)	
biochemical	165	F06503B	Osmolality (urine)	
biochemical	166	F08075C	Osmolality	
biochemical	167	F12061A	Urine myoglobin (urine) (delivery)	
biochemical	168	F10807B	Ethyl alchol (disabled until emergency biochemical group opened)	
biochemical	169	F09056A	5-HIAA (24-hour urine) (delivery)	
biochemical	016A	F24009C	glucose tolerance test	
biochemical	016B	F24009C	Glucose tolerance test (limited to CRC, five points)	
biochemical	016C	F019962	small-dense LDL	
biochemical	016D	F12113B	Apo B (Apo B)	
biochemical	170	F12103B	Immuno-EP (delivery)	
biochemical	172	F099005	ICG (dark, 2 green tips) (blood)	
biochemical	173	F09084B	B-lipoprotein VLDL (delivery)	
biochemical	175	F09007B	GTT (urine) (Please prescribe Glucose Tolerance Test	
biochemical	176	F09064B	Lipase (disable and switch to emergency biochemical)	
biochemical	178	F129017	PTH-I (centrifugal packaging and frozen delivery)	
biochemical	179	F129018	PTH-C (Disabled) (Delivery)	

Figure 23:Translated codes from the pilot dictionary

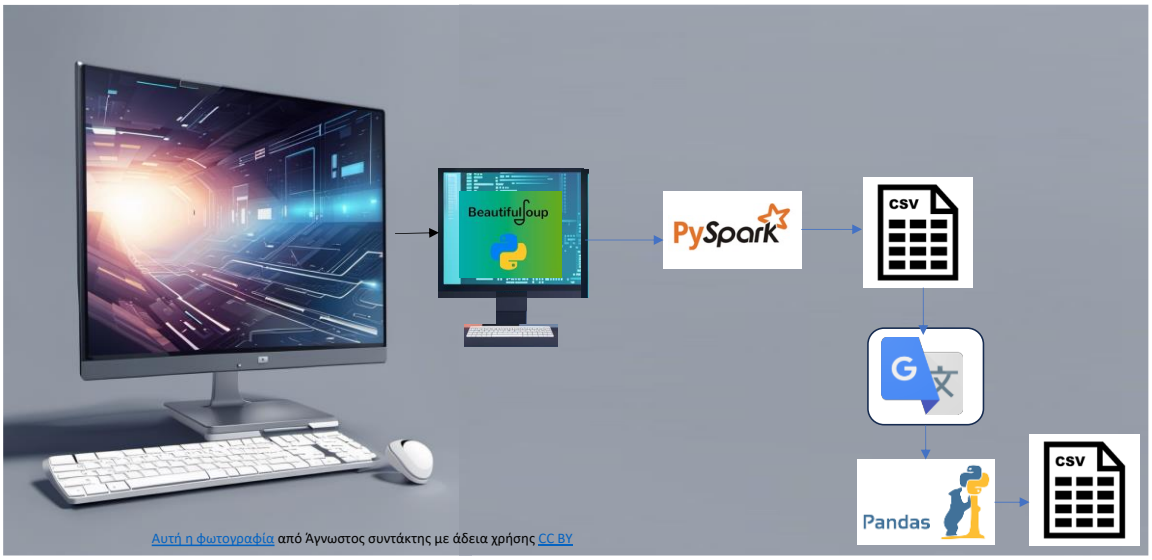


Figure 24:Creation of blood dictionary process

5.3.2. Blood test data transformation and cancer file

In the process of mapping and analyzing data, various tools and techniques were employed. The Python libraries PySpark, Pandas, and Beautiful Soup were used to download codes and descriptions of blood tests from our pilot site. Additionally, the Google API was utilized to translate the blood test descriptions into English. A dictionary was created for medication codes and descriptions, as well as for the ICD9 and ICD10 protocols.

These dictionaries were then used to map the input data using UDF Spark functions. With the aid of academic research and a classification algorithm, the top features required for the risk factor analysis were extracted from the blood tests. To ensure consistency, a function was developed to convert the blood tests to a uniform unit and numeric values, while also removing null values.

Furthermore, a function was created to convert Chinese data to Gregorian format. Age was calculated using the diagnosis date and birthdate, and the results were saved in a new column. The datasets were joined with diagnosed disease codes and demographic characteristics of both

Snippet 1: convert_to_g_per_dl function

This function converts an input value to a concentration expressed in grams per deciliter (g/dL). It validates various input formats and performs the required calculations to return the concentration in g/dL. Returns None if the input is invalid or not recognized. This function converts text values, removes null, converts them to all lowercase, and then converts them to the same unit based on a specific blood test.

```
def convert_to_g_per_dl(value):
    if value is None:
        return None

    value = value.lower().strip()

    try:
        if "g/dl" in value:
            return float(value.replace("g/dl", "").strip())
        elif "10^3/uL" in value:
            return float(value.replace("10^3/uL", "").strip()) * 10**3 * 0.00001
        elif "10^6/uL" in value:
            return float(value.replace("10^6/uL", "").strip()) * 10**6 * 0.00001
        else:
            return None
    except ValueError:
        return None
```

Snippet 2: Filtering with PySpark

This code snippet imports the col function from the pyspark.sql.functions module and defines a list of codes to filter by. It then defines a custom function contains_any_code to check if any of the given codes are present in a given string. This function is registered as a User-Defined Function (UDF) in PySpark.

Next, the code uses the custom UDF to filter a DataFrame (joined_df_all9) based on the presence of the specified codes in the "O_ITEM" column.

Snippet 3: convert_to_g_per_dl_udf

This line of code registers the convert_to_g_per_dl function as a PySpark UDF with the return type FloatType.

Snippet 4

```
#11A1 Blood urea nitrogen
from pyspark.sql.functions import regexp_replace, when, col

filtered_results = filtered_results.withColumn("Blood_Urea_Nitrogen",
    when(
        (col("O_ITEM") == "11A1") & (col("VALUE").contains("IU/L")),
        regexp_replace(col("VALUE"), r"[^0-9.]", "").cast("double")
    ).otherwise(None)
)
```

Snippet 5: Handling outliers and calculating averages with PySpark

This code snippet defines a filter outliers function to remove outliers from a DataFrame column based on the 1st and 99th percentiles. It then creates a list of column names to calculate averages for and initializes an empty list to store the average results.

For each column in the list, the code filters out outliers and calculates the mean value, rounded to two decimal places. It stores these results in the averages list. Next, the average columns are joined to the original DataFrame, and the original columns are dropped. Finally, the updated DataFrame is displayed.

```
from pyspark.sql import functions as F


def filter_outliers(df, col_name):
    quantiles = df.approxQuantile(col_name, [0.01, 0.99], 0.0)
    if len(quantiles) == 2:
        lower_bound, upper_bound = quantiles
        return df.filter((F.col(col_name) >= lower_bound) & (F.col(col_name) <= upper_bound))
    else:
        return df
```

```
columns_to_average = ["CBCI", "WBCDC", "CRP", "Prothrombin_time_blood", "creatinine", "GOT_AST",
"Blood_Urea_Nitrogen", "ALFA-FETOPROTEIN"]


averages = []
for col_name in columns_to_average:
    filtered_df = filter_outliers(df, col_name)
    avg_col = filtered_df.groupBy("CHR_NO").agg(F.round(F.mean(F.when(F.col(col_name).isNotNull()
& (F.col(col_name) != 0), F.col(col_name))), 2).alias(f"avg_{col_name}"))
    averages.append(avg_col)

joined_df = df
for avg_col in averages:
    joined_df = joined_df.join(avg_col, on="CHR_NO", how="left")

for col_name in columns_to_average:
    joined_df = joined_df.drop(col_name)


#joined_df.show(5)
```

5.4. Machine learning models

5.5. Second cleaning phase

By utilize the following function we are going to check the null values for each column to remove later the columns that have big precent of null columns, we need to point out that because the dataset is big we can keep some columns with big present of null values because is still going to help he analysis.

```
from pyspark.sql.functions import col

for column in columns_to_check:
    null_count = df.filter(col(column).isNull()).count()
    print(f"Number of null values in {column} column: {null_count}")
```

The following PySpark code snippet calculates the mean values of each column in a DataFrame called df_filled, with the exception of the columns labeled 'y' and 'CHR_NO'. By employing the mean and col functions from pyspark.sql.functions, the code creates a new DataFrame that includes the average of each selected column. It then executes this operation, collects the results, and fetches the first Row object, effectively capturing the calculated averages. Finally, the .asDict() method transforms this Row object into a Python dictionary. The keys in this dictionary are the column names, and the associated values are the calculated mean values for those respective columns. This dictionary is stored in a variable named column_means, providing a convenient summary of the dataset's central tendencies for each specified feature.

```
from pyspark.sql.functions import mean, col

column_means = df_filled.select([mean(col(column)).alias(column) for
column in df_filled.columns if column != 'y' and column !=
'CHR_NO']).collect()[0].asDict()
```

5.5.1. Rebalanced techniques

Because the healthy y =0 people are more than the people who have liver cancer , in order to have big accuracy we needed to use imbalanced techniques to achieve this . The described method is an approach to a binary classification problem using a Gradient Boosting Regressor, specifically made-to-measure for imbalanced datasets. The dataset is first separated into majority and minority classes, followed by unsampled the minority class to match the size of the majority. This helps improve the model's performance by balancing the dataset. Missing data is handled through mean imputation, exchanging missing values with the mean of the respective columns. After data pre-processing, the data is split into training and testing sets, and a Gradient Boosting model is trained on the data. The model's performance is evaluated using cross-validation with the AUC-ROC score, which provides a measure of the model's ability to predict between the classes .

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler, TomekLinks
from imblearn.combine import SMOTETomek
from imblearn.ensemble import BalancedRandomForestClassifier

# Random Oversampling
ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(X_train_resampled,
y_train_resampled)
RandomOversampling_SVC = SVC(random_state=42)
RandomOversampling_SVC.fit(X_resampled, y_resampled)
y_pred = RandomOversampling_SVC.predict(X_test)
print("Random Oversampling:")
print(classification_report(y_test, y_pred))

def plot_correlation_matrix(data, title):
    corr_matrix = data.corr()
    plt.figure(figsize=(10, 8))
    sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f")
    plt.title(title)
    plt.show()


def plot_feature_correlations(data, target, title):
    correlations = data.corr()[target].sort_values(ascending=False)
    plt.figure(figsize=(10, len(data.columns) * 0.3))
    sns.barplot(y=correlations.index, x=correlations, orient='h')
    plt.title(title)
    plt.xlabel('Correlation Coefficient')
    plt.ylabel('Features')
    plt.show()

# Convert the SMOTE-resampled data to pandas DataFrame
# Convert the SMOTE-resampled data to pandas DataFrame
```

```
X_train_resampled_df = pd.DataFrame(X_resampled, columns=X.columns)
# Use the columns from the original X DataFrame
y_train_resampled_df = pd.DataFrame(y_resampled, columns=['y'])  #
Replace 'your_target_column_name' with the actual target column name
  # Replace 'y' with the actual target column name

data_resampled = pd.concat([X_train_resampled_df,
y_train_resampled_df], axis=1)

# Plot feature correlations with the target variable
plot_feature_correlations(data_resampled, 'y', "resampled Dataset -
Feature Correlations with Target Variable")
```

5.5.2. Xgboost model example rembalanced techniques

The following code implements the XGBoost algorithm for classification and addresses the issue of class imbalance through upsampling the minority class. Furthermore, the dataset is separated into majority and minority classes based on the target label 'y'. The minority class is then upsampled to match the size of the majority class, ensuring a balanced dataset. This new balanced dataset is split into training and test sets. An XGBoost Classifier is trained on the training set, and its feature importance is plotted for analysis. The model then predicts the probabilities of the target label for the test set, and the overall accuracy of the model is calculated and printed. By employing upsampling, the code aims to provide a more balanced training environment for the XGBoost model, improving its classification performance.

```
from sklearn.utils import resample
import xgboost as xgb
from sklearn.model_selection import train_test_split

# Separate the majority and minority classes
df_majority = df[df.y==0]  # Replace 'y' with your actual target
column name
df_minority = df[df.y==1]

# Upsample minority class
df_minority_upsampled = resample(df_minority,
                                 replace=True,                    #
sample with replacement
                                 n_samples=len(df_majority),      #
to match majority class
                                 random_state=123)                #
reproducible results

# Combine majority class with upsampled minority class
df_upsampled = pd.concat([df_majority, df_minority_upsampled])
```

```
# Display new class counts
print(df_upsampled.y.value_counts())

# Prepare your features 'X' and target 'y'
X = df_upsampled.drop('y', axis=1)
y = df_upsampled['y']

# Create train and test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train the model
model = xgb.XGBClassifier(use_label_encoder=False,
eval_metric='logloss')
model.fit(X_train, y_train)

# Print feature importance
xgb.plot_importance(model)

# Make predictions with probabilities for the test set
y_pred_proba = model.predict_proba(X_test)

# Print the probabilities
print(y_pred_proba)

# Compute and print accuracy score
accuracy = model.score(X_test, y_test)
print(f'Accuracy: {accuracy}')
```

5.5.3. Summarized best models results.

Base Classifier: This is the model we use as a baseline for comparison. Its performance metrics on the dataset are: accuracy (0.81), precision (0.90 for class 0 and 0.55 for class 1), recall (0.86 for class 0 and 0.64 for class 1), and F1-score (0.88 for class 0 and 0.59 for class 1). The same model was used with both SMOTE oversampling and Random oversampling, and the performance metrics were very similar in both cases.
LightGBM with SMOTE: LightGBM is a gradient boosting framework that uses tree-based algorithms, and is designed to be distributed and efficient. In our case, it seems to perform better than the base classifier when used with SMOTE oversampling. It achieves an accuracy of 0.83, which is higher than the base classifier.
Lasso Logistic Regression with SMOTE: Lasso Logistic Regression is a type of logistic regression that uses L1 regularization (Lasso). L1 regularization has the effect of shrinking some of the model's parameter estimates to zero, thus reducing the number of parameters and helping to prevent overfitting. However, in our case, it performs worse than both the base classifier and LightGBM, with an accuracy of 0.78.
From these results, the LightGBM model with SMOTE oversampling seems to be the best model among the ones we've tested. It has the highest accuracy (0.83) and also has decent recall and precision rates for both classes. However, the specific choice of best

model can also depend on the specific needs of your project. For example, if it is more important to correctly identify class 1, you might choose a model with a higher recall for class 1, even if its overall accuracy is slightly lower.

SMOTE Oversampling:

The smote

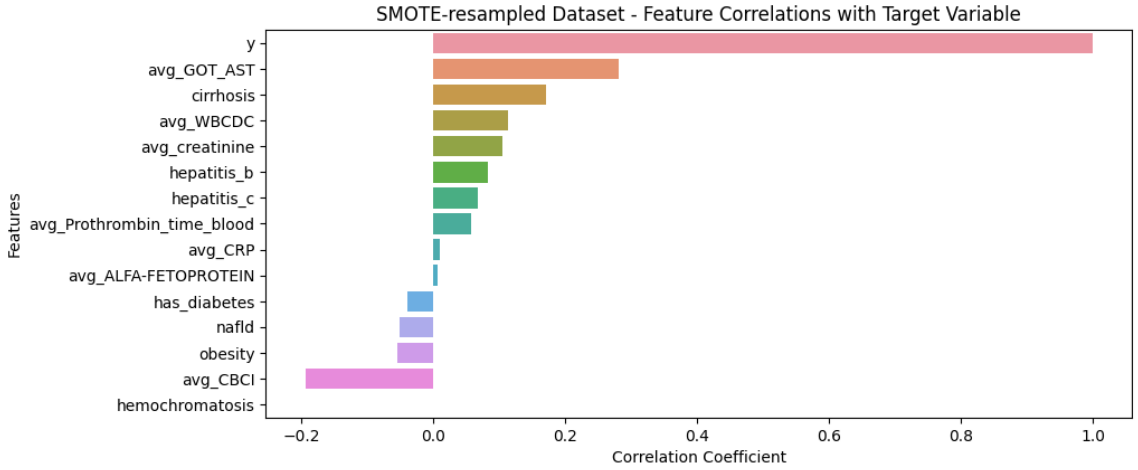


Figure 25:Smote oversampling

Random oversampling

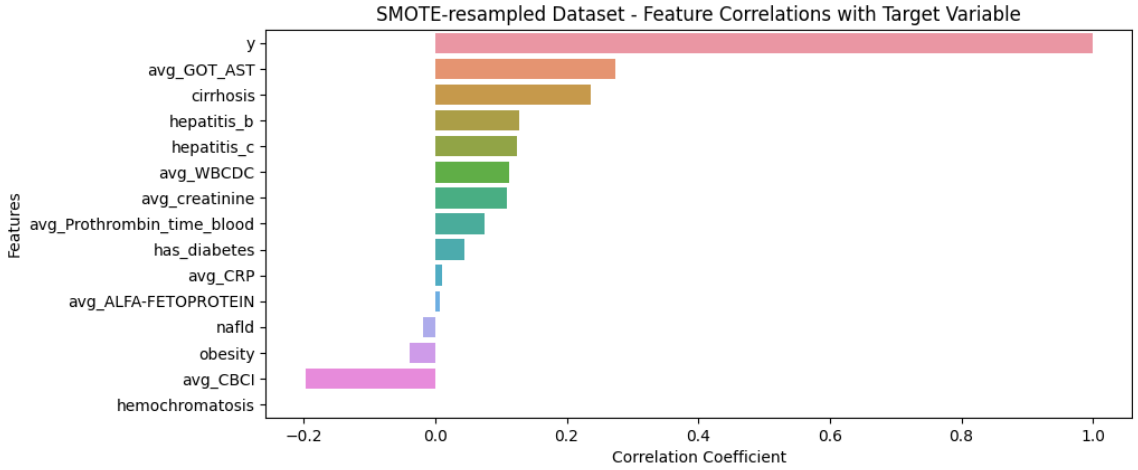


Figure 26:Random oversampling

SMOTE Oversampling with LightGBM Classifier:

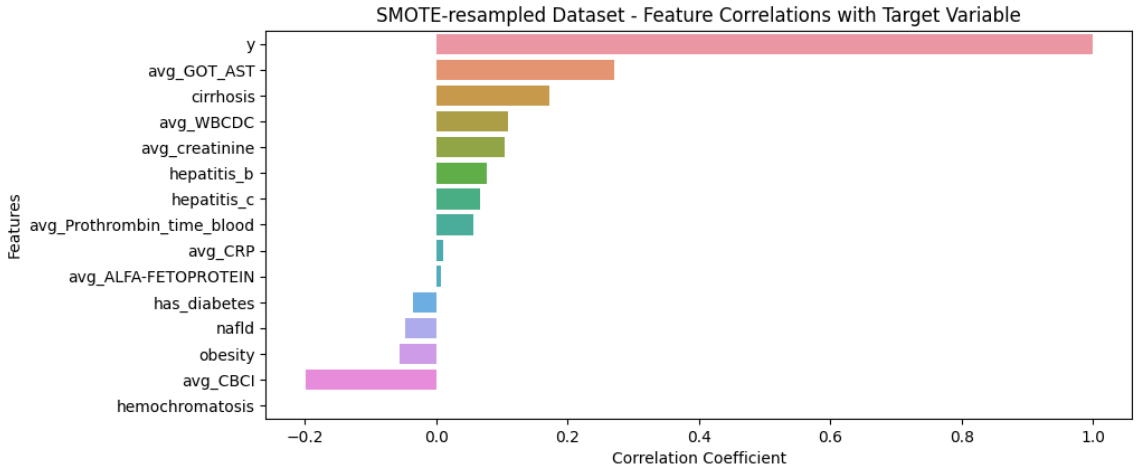


Figure 27:LightGbm classifier correlations

SMOTE Oversampling with Lasso Logistic Regression:

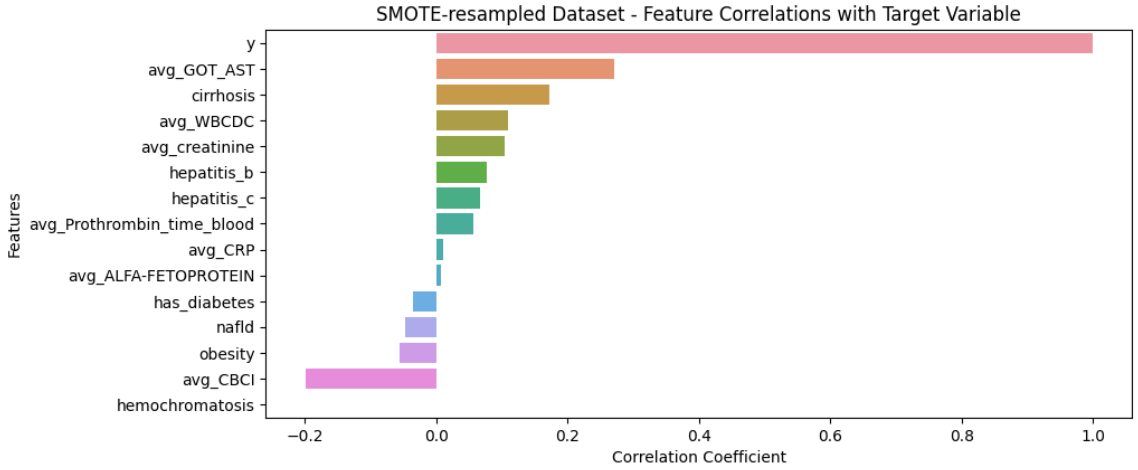


Figure 28:Lasso Logistic Oversampling correlation

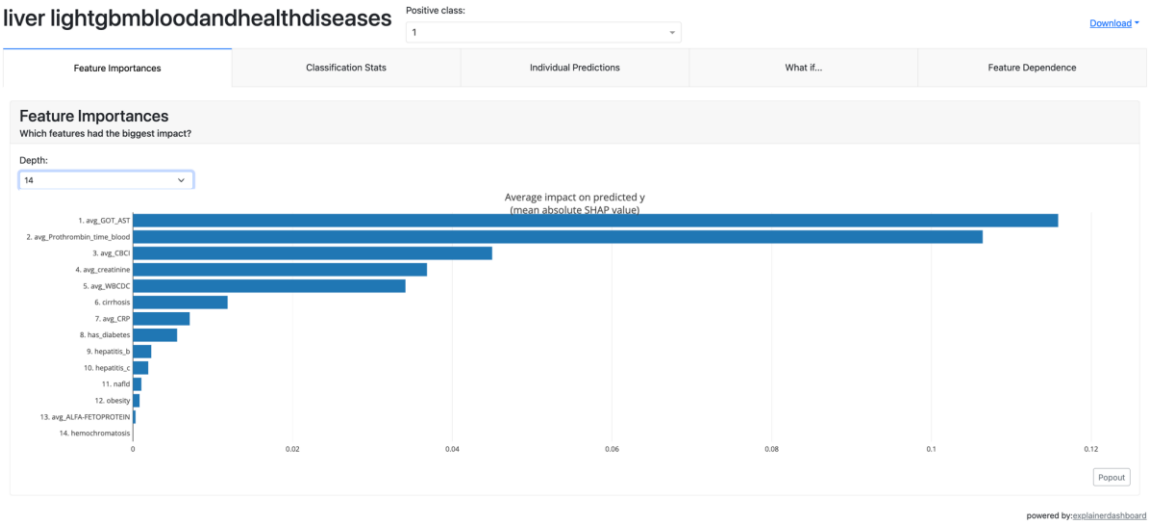


Figure 29:Lightgbm feature importances by mean absolute SHAP value

Oversampling Method	Classifier	Accuracy	Precision (0)	Recall (0)	F1-score (0)	Precision (1)	Recall (1)	F1-score (1)

SMOTE	LightGBM	0.83	0.91	0.86	0.89	0.57	0.71	0.63
SMOTE	Lasso Logistic Regression	0.78	0.89	0.82	0.85	0.48	0.61	0.54
SMOTE	(Base Classifier)	0.81	0.90	0.86	0.88	0.55	0.64	0.59
Random	(Base Classifier)	0.81	0.89	0.86	0.88	0.55	0.62	0.59

Figure 30:Models performances for blood tests and diseases

5.6. Risk factors blood tests and diseases with less features to achieve higher accuracy.

In order to advance the previous models in this use case we rerun the models with a little more preprossesing we remove some features that contains null values.

5.6.1. Machine learning models

The table that follows lists several features (which are be blood tests and conditions) and their correlation with a particular outcome, "y" (which indicates if someone has cancer or not .). -1 represents a perfect negative correlation, 1 a perfect positive correlation, and 0 no correlation. The feature "Avg GOT AST" has the highest positive correlation (0.3085), indicating that as the quantity of "Avg GOT AST" increases, the outcome "y" also increases. Similarly, "cirrhosis" demonstrates a moderately strong positive correlation (0.2796). In contrast, "Avg CBCI" has a negative correlation (-0.1593), indicating that the outcome "y" tends to decrease as the Avg CBCI increases. The majority of the remaining features have weaker correlations, close to 0, indicating a less direct or less consistent relationship with the outcome "y." Features such as "obesity", "nafld", and "Has diabetes" have particularly low correlation values, indicating that they may not be significant predictors of the outcome "y" in this particular analysis.

Feature	Correlation with y
Avg CBCI	-0.1593
Avg WBCDC	0.1097
Avg Prothrombin_time_blood	0.0600
Avg creatinine	0.1106
Avg GOT AST	0.3085
obesity	-0.0272
nafld	-0.0189
cirrhosis	0.2796
Hepatitis c	0.1332
Hepatitis b	0.1405
Has diabetes	0.0406

Figure 31:correlation matrix for blood tests and diseases after more clean

Random forest classifier

Shows that agt got ast is the most important feature for the prediction , following avg CBCI ,avg prothrombin time blood , avg wbcdc , cirrhosis are the most important features for the prediction for liver cancer.

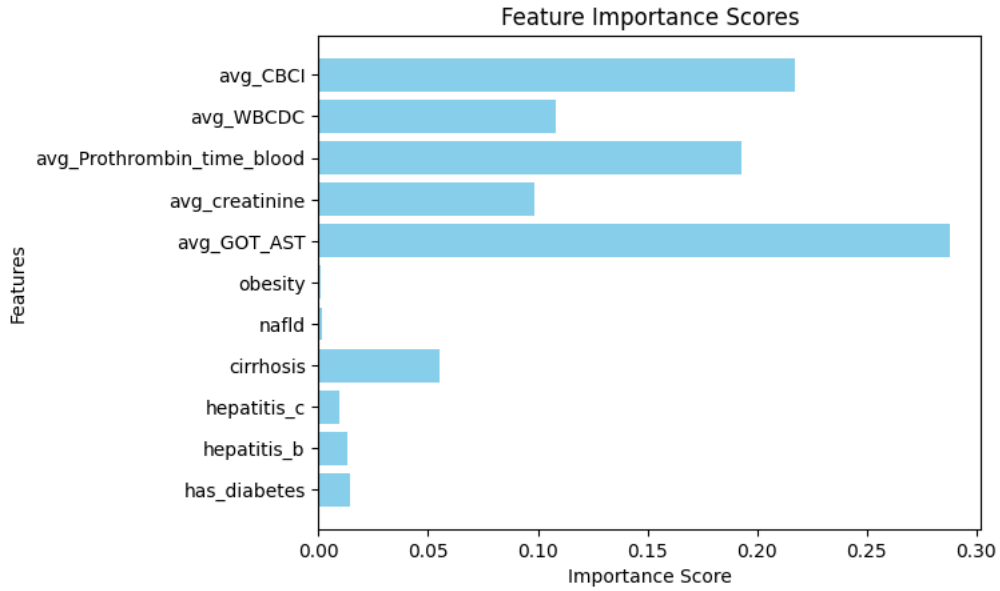


Figure 32:Random forest classifier feature importance score after more cleaning

5.6.2. Xgboost rembalanced

Xgboost rembalanced shows that the most important features for the prediction are avg GBCI with value 803, following avg prothrombin time blood , avg got ast(per patient),creatitinine ,wbcdc,diabetes , cirrhosis , hepatitis b and hepatitis c

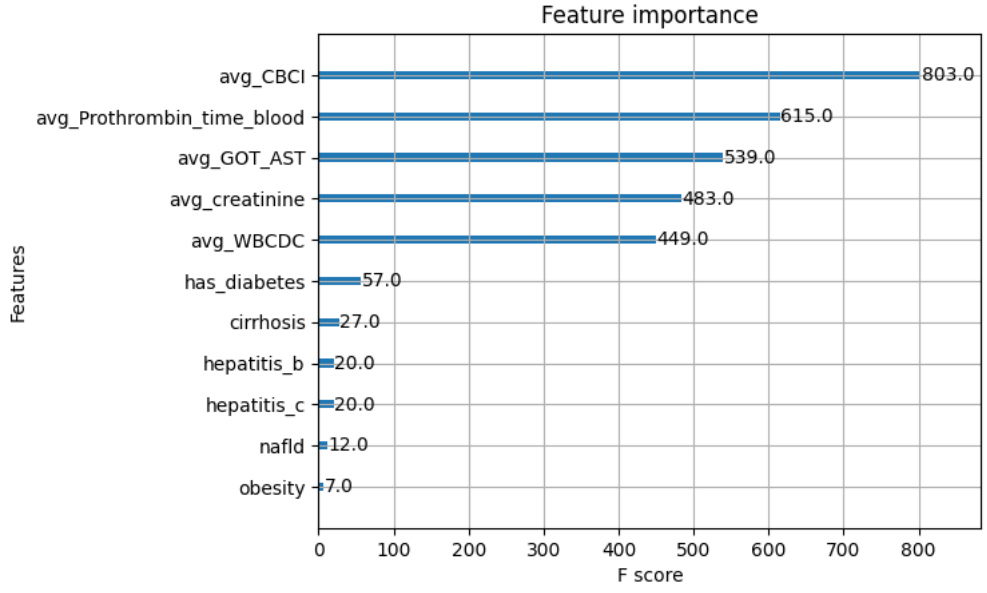


Figure 33:XGBOOST weights

By removing some columns with null values

We continue by utilize shap explainer to see the shap values for xgboost

```
import shap

# Create object that can calculate shap values
explainer = shap.TreeExplainer(model)

# calculate shap values. This is what we will plot.
shap_values = explainer.shap_values(X)

# Make plot. Index of [1] is explained in text below.
shap.summary_plot(shap_values, X, plot_type="bar")
```

In the given data, each variable represents the average value of a particular medical indicator or condition across a sample population, where "y" indicates whether an individual has cancer. "avg_CBCI" stands for average Complete Blood Count Index with a value of 0.3855, which might give insights into the overall health of the blood. "avg_WBCDC" (0.1924) represents the average White Blood Cell Differential Count, a marker for potential infection or other diseases. "avg_Prothrombin_time_blood" at 0.6697 could be related to how quickly blood clots, a crucial factor in many diseases. "avg_creatinine" (0.1944) provides information about kidney function. "avg_GOT_AST" (0.8742) is the average level of the enzyme Aspartate Aminotransferase, generally related to liver health. Other variables represent the proportion of individuals with specific conditions: "obesity" (0.0119), "nafld" or Non-Alcoholic Fatty Liver Disease (0.0069), "cirrhosis" (0.1414), "hepatitis_c" (0.0331), "hepatitis_b" (0.0604), and "has_diabetes" (0.0649). Each of these factors may have varying degrees of association with the presence of cancer, indicated by "y," and could be important for medical researchers and healthcare professionals for predictive analysis and treatment planning.

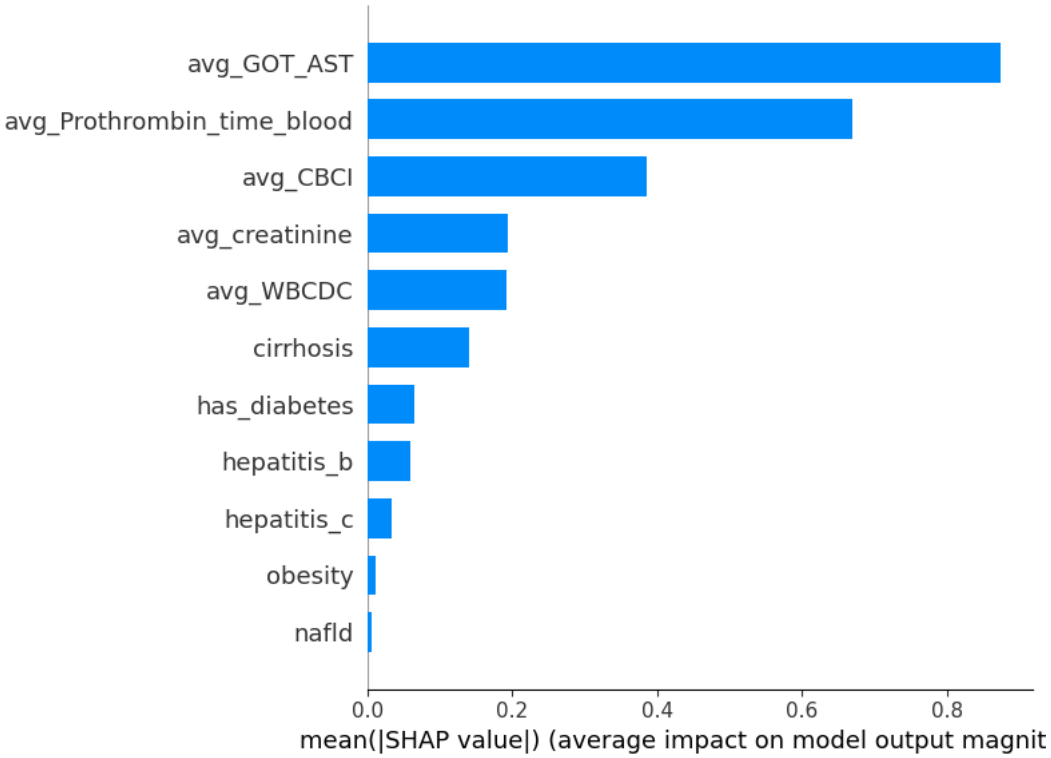


Figure 34:Mean SHAP value after more cleaning for diseases and blood tests for healthy patients

5.6.3. Support vector machine

The table of feature weights indicates the influence of each feature in predicting the outcome of the model. Features with positive weights such as 'cirrhosis', 'hepatitis_c', and 'hepatitis_b' have a positive correlation with the outcome. This suggests that patients with cirrhosis, hepatitis C, or hepatitis B are more likely to have a certain outcome. On the other hand, 'obesity' and 'nafld' have negative weights, implying that patients with obesity or non-alcoholic fatty liver disease (NAFLD) are less likely to have the same outcome. 'avg_creatinine', 'has_diabetes', 'avg_WBCDC', and 'avg_GOT_AST' are associated with a slight increase in the likelihood of the liver cancer , while 'avg_Prothrombin_time_blood' and 'avg_CBCI' are associated with a slight decrease. In summary, the conditions cirrhosis and hepatitis C appear to be the most influential factors in the prediction, while NAFLD and obesity significantly reduce the likelihood of the outcome.

Feature	Weight
Cirrhosis	2.0385988779588473
Hepatitis c	1.9517482572475764
Hepatitis b	0.42906410279745444
Avg creatinine	0.06975721193680329
Has diabetes	0.022536295708391663
Avg WBCDC	0.01901273726252839
Avg GOT AST	0.013208648080762941
Avg Prothrombin time blood	-0.0037965448573231697
Avg CBCI	-0.009175303053780226
Obesity	-0.03725462175262173
Nafld	-0.040535376765481246

Figure 35:SVM weights

A code implements a data analysis pipeline using various Python libraries that focus on data processing, visualization, machine learning, and handling imbalanced datasets. The code begins by importing necessary modules and libraries, including pandas, seaborn, matplotlib, numpy, sklearn, and imbalanced-learn. These libraries offer functionalities for data manipulation, visualization, numerical computation, machine learning, and resampling techniques.
The code proceeds by applying the RandomOverSampler from the imbalanced-learn package to resample the training data. Resampling is a technique used to address the issue of imbalanced datasets, where the number of instances in one class is significantly smaller than the other. The RandomOverSampler duplicates some of the minority class examples, thus creating a more balanced training dataset. Subsequently, a Support Vector Machine Classifier (SVC) from the sklearn library is trained using this oversampled data. The trained model is then used to predict outcomes for the test data. The performance of the model is evaluated using a classification report, which provides insights into the model's accuracy, precision, recall, and F1-score.

To further analyze the dataset, the code defines two functions for visualizing the correlation between features. The first function, plot correlation matrix, generates a heatmap of the correlation matrix of the input data. This visualization helps identify relationships and patterns between different features. The second function, plot feature correlations, creates a horizontal bar chart that displays the features in descending order

of their correlation with the target variable. This visualization aids in understanding the impact of each feature on the target variable.

Finally, the code converts the resampled data back to a pandas DataFrame and concatenates it into a single DataFrame. The features' correlations with the target variable are then plotted using the previously defined function. This analysis provides insights into the relative importance of each feature in predicting the target variable.

In summary, the provided code showcases a comprehensive workflow for handling imbalanced datasets, including oversampling, model training using a Support Vector Machine Classifier, and visualization of feature correlations. This code can be applied in various contexts where imbalanced data is a concern, such as fraud detection or rare disease diagnosis. By addressing data imbalance and understanding feature correlations, researchers and practitioners can gain valuable insights and make informed decisions in domains where one class of outcome is significantly rarer than others.

5.6.4. Results after

The shap value and correlation of lightgbm blood and diseases show that the avg got per patient with 0.12 shap value , avg prothrombin blood , avg cbci

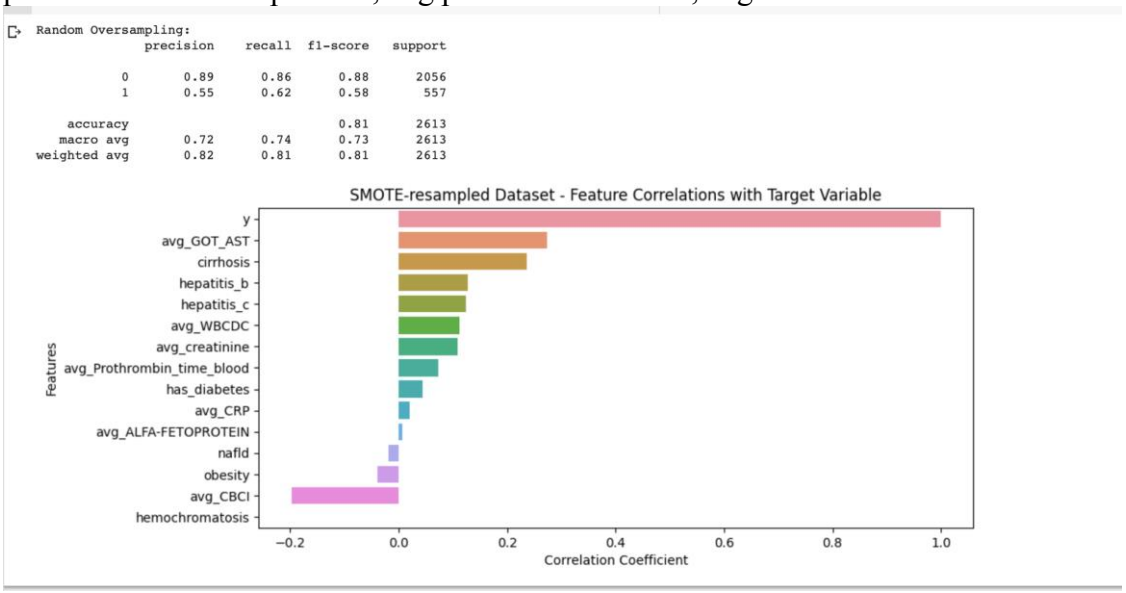


Figure 36:Random Oversampling after more cleaning

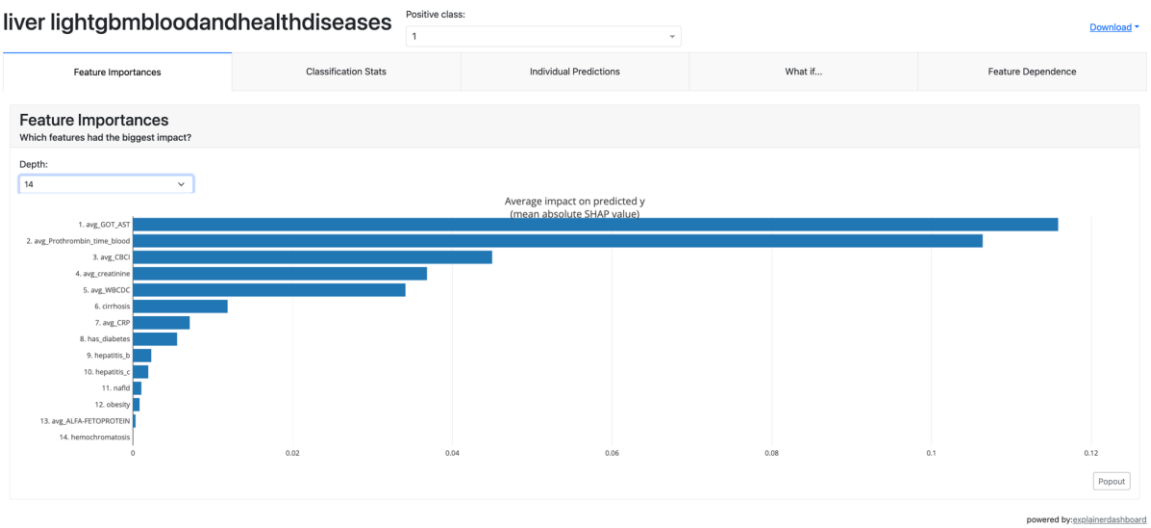


Figure 37:SHAP values after more cleaning

5.6.5. Overall results of blood and diseases analysis

In most predictive models for liver cancer, a positive correlation is generally observed with the following variables: avg_GOT AST, cirrhosis, hepatitis B, hepatitis C, avg_WBCDC, avg_creatinine, avg_Prothrombin_time_blood, has_diabetes, and avg_CRP. This means that higher values or the presence of these conditions or markers are generally associated with a higher likelihood of developing liver cancer.

For instance, elevated levels of GOT (AST) and creatinine are common indicators of liver dysfunction and kidney issues, respectively. Hepatitis B and C are well-known risk factors for liver cancer, and cirrhosis is a late stage of liver scarring often seen in liver cancer patients. Similarly, higher levels of white blood cells (WBCDC), prothrombin time, and CRP may indicate an underlying inflammatory or clotting disorder, which could also predispose an individual to liver cancer. The presence of diabetes has also been implicated in liver dysfunction, further enhancing the risk of liver cancer. Therefore, these variables being positively correlated in most models underscores their importance in predicting liver cancer effectively.

6. Liver cancer external factors

6.1. Model selection

The method primarily pertains to the application of the Random Forest Classifier, a highly utilized machine learning algorithm suitable for classification tasks. Initially, the necessary libraries are imported: numpy, renowned for its proficiency with array and matrix operations, and a collection of modules from sklearn - a prominent machine learning library.

Moreover , the dataset is divided into training and testing subsets through the train_test_split method, with 30% of the data set aside for testing purposes. To fine-tune the model, hyperparameter optimization is performed using RandomizedSearchCV. This method systematically tests a range of hyperparameters, identifying the most efficacious configuration to improve the model's predictive accuracy.

A selection of potential hyperparameters is defined in the param_dist dictionary, which includes variables like 'n_estimators', 'max_depth', 'min_samples_split', 'min_samples_leaf', and 'bootstrap'. Post optimization, the best set of hyperparameters is printed out.

The model is subsequently retrained on the entire training set using these optimal parameters. Once trained, the model proceeds to make predictions on the testing data. Finally, a detailed performance report of the model is displayed using the classification_report function, providing metrics such as precision, recall, and the f1-score. In conclusion, the script outlines a comprehensive yet efficient way to tune, train, and assess a Random Forest Classifier.

14 Ιουλίου 2023 - 15:25

Method	Accuracy	Precision (Class 0)	Recall (Class 0)	F1-Score (Class 0)	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)
Random Oversampling	0.67	0.77	0.68	0.72	0.56	0.66	0.60
Random Under sampling	0.69	0.71	0.84	0.77	0.62	0.44	0.51
Tomek Links	0.70	0.70	0.90	0.79	0.70	0.38	0.49
Balanced Random Forest	0.65	0.72	0.72	0.72	0.54	0.54	0.54

6.1.1. Machine learning model

The random forest algorithm is one of the most effective machine learning algorithms for assessing liver cancer risk variables. This algorithm is a form of decision tree model that utilizes a collection of decision trees to produce predictions. The random forest approach can handle enormous quantities of data, including continuous and categorical variables, and is resistant to noise and outliers. It is also simple to administer and analyze, making it a valuable tool for identifying liver cancer risk factors.
The logistic regression approach is another valuable machine learning tool for assessing risk variables for liver cancer. This method is utilized to predict binary outcomes, including the likelihood of contracting liver cancer. It is especially useful for determining the relative significance of various risk factors and for constructing risk prediction models. Logistic regression is a straightforward and interpretable technique that may be easily incorporated into a wide range of software packages.
In addition to these algorithms, machine learning techniques such support vector machines (SVMs) and neural networks can be utilized to examine liver cancer risk variables. SVMs are a technique for supervised learning that can be used to classify data into distinct groups. They excel at processing high-dimensional data and are resistant to overfitting. Neural networks are a form of artificial intelligence algorithm that can recognize and predict complicated patterns in data. They are especially useful for managing enormous amounts of data and can be taught to spot patterns that may not be obvious to human analysts. Ultimately, machine learning approaches can be an effective method for assessing liver cancer risk variables. The method primarily pertains to the application of the Random Forest Classifier, a highly utilized machine learning algorithm suitable for classification tasks. Initially, the necessary libraries are imported: numpy, renowned for its proficiency with array and matrix operations, and a collection of modules from sklearn - a prominent machine learning library. Then, the dataset is divided into training and testing subsets through the train_test_split method, with 30% of the data set aside for testing purposes. To fine-tune the model, hyperparameter optimization is performed using RandomizedSearchCV. This method systematically tests a range of hyperparameters, identifying the most efficacious configuration to improve the model's predictive accuracy.
A selection of potential hyperparameters is defined in the param_dist dictionary, which includes variables like “n_estimators”, “max_depth”, “min_samples_split”, “min_samples_leaf”, and “bootstrap”. Post optimization, the best set of hyperparameters is printed out.
The model is subsequently retrained on the entire training set using these optimal parameters. Once trained, the model proceeds to make predictions on the testing data. Finally, a detailed performance report of the model is displayed using the classification_report function, providing metrics such as precision, recall, and the f1-score. In conclusion, the script outlines a comprehensive yet efficient way to tune, train, and assess a Random Forest Classifier.
The model is subsequently retrained on the entire training set using these optimal parameters. Once trained, the model proceeds to make predictions on the testing data. Finally, a detailed performance report of the model is displayed using the classification_report function, providing metrics such as precision, recall, and the F1-score. In conclusion, the script outlines a comprehensive yet efficient way to tune, train, and assess a Random Forest Classifier.

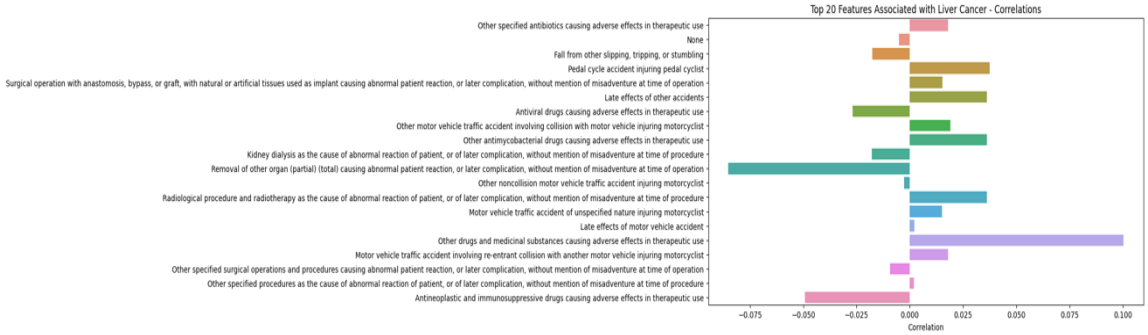


Figure 38:Correlation for external factors

6.1.2. Random oversample to external

The analysis has shown that the following factors have the biggest impact on liver cancer:

- Side effects of antineoplastic and immunosuppressive drugs
- Procedures
- Drugs and medical substances
- Complications related to the removal of other organs.

The list includes a wide range of ICD-9 codes and related conditions or incidents, which vary greatly in their relevance and impact on the prediction of liver cancer. Some items, like "Antineoplastic and immunosuppressive drugs causing adverse effects in therapeutic use," could have a more direct impact on liver health and thereby on the prediction of liver cancer. These types of drugs are strong medications used to treat cancer or suppress the immune system, and their adverse effects could contribute to liver dysfunction.
On the other hand, entries like "Accidents occurring in public building" or "Accidental fall from ladder"There are statistics that says"Motor vehicle traffic accident involving collision with pedestrian injuring pedestrian-" may have a significant health impact, but it's not directly relevant to liver cancer risk.The list also includes various medications and substances that cause adverse effects "in therapeutic use," ranging from antiviral drugs to anticoagulants. While some of these could potentially affect liver function as a side effect, their primary indication is not the treatment or prevention of liver cancer, making their impact on liver cancer prediction variable and less direct.
There are also a few instances of procedural complications, like "Mechanical failure of instrument or apparatus during endoscopic examination" and "Urinary catheterization as the cause of abnormal reaction," which may correlate with liver cancer from chronic infaction.

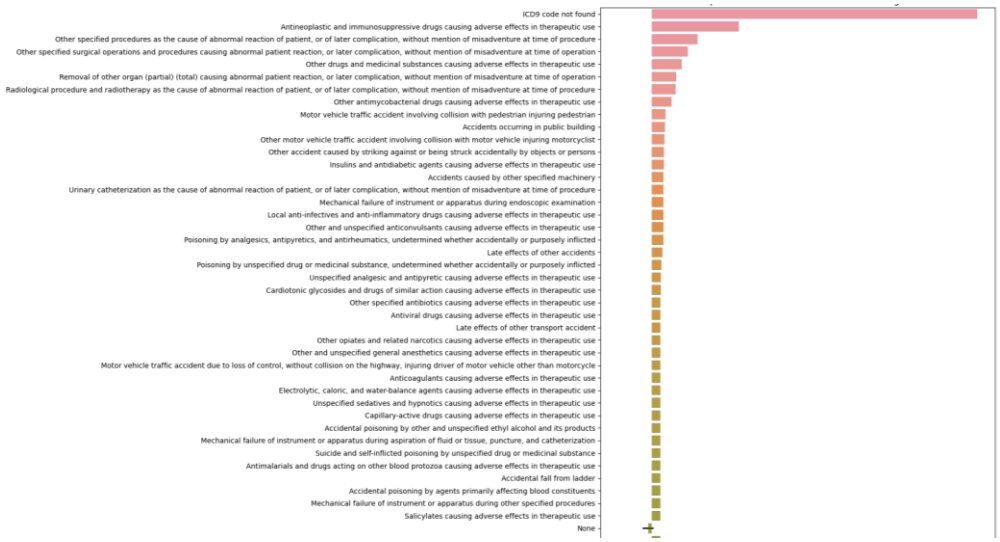


Figure 39:Feature importances for external factors 1

6.1.3. Random forest balanced

- That analysis have shown this have the biggest impact for liver cancer : antineoplastic and immunosuppressive drugs sides effects
- Procedures side effects
- Traffic accident
- Other drugs and medical substances
- Late effects of motor accident

Antineoplastic and immunosuppressive drugs causing adverse effects in therapeutic use:

These drugs, commonly used in the treatment of cancer, can have adverse effects which may be associated with liver cancer.
Other specified procedures as the cause of abnormal reaction of patient, or of later complication, without mention of misadventure at time of procedure

Certain procedures may have unintended complications or reactions that are linked to liver cancer.
Other specified surgical operations and procedures causing abnormal patient reaction, or later complication, without mention of misadventure at time of operationLike the previous feature, certain surgical operations can result in complications or abnormal reactions related to liver cancer.
Motor vehicle traffic accident involving re-entrant collision with another motor vehicle injuring motorcyclistWhile not directly related to liver cancer, this feature may indicate lifestyle or environmental factors that could be indirectly associated.
Other drugs and medicinal substances causing adverse effects in therapeutic use
Additional medications beyond antineoplastic drugs can also have adverse effects connected to liver cancer.
Late effects of motor vehicle accident
This feature possibly suggests long-term health complications from accidents that may be indirectly related to liver cancer.
Motor vehicle traffic accident of unspecified nature injuring motorcyclist
Similar to the fourth feature, this could indicate lifestyle or environmental factors that may be indirectly related.
Radiological procedure and radiotherapy as the cause of abnormal reaction of patient, or of later complication, without mention of misadventure at time of procedure
Radiation treatments can have complications or abnormal reactions that could be associated with liver cancer.
Other noncollision motor vehicle traffic accident injuring motorcyclist
Again, another feature that may point to lifestyle or environmental factors.
Removal of other organ (partial) (total) causing abnormal patient reaction, or later complication, without mention of misadventure at time of operation
The removal of other organs can have complications or reactions that are associated with liver cancer.
Kidney dialysis as the cause of abnormal reaction of patient, or of later complication, without mention of misadventure at time of procedure

Kidney dialysis can result in complications or abnormal reactions that may be associated with liver cancer.
Other antimycobacterial drugs causing adverse effects in therapeutic use
These specific types of antibiotics can have adverse effects related to liver cancer.
Other motor vehicle traffic accident involving collision with motor vehicle injuring motorcyclist
Similar to other motor vehicle-related features, this could indicate indirect lifestyle or environmental factors.
Antiviral drugs causing adverse effects in therapeutic use
Like antineoplastic drugs, antiviral drugs can also have adverse effects that are linked to liver cancer.
Late effects of other accidents
Long-term effects of other types of accidents could also have an indirect relationship with liver cancer.
Surgical operation with anastomosis, bypass, or graft, with natural or artificial issues used as implant causing abnormal patient reaction, or later complication, without mention of misadventure at time of operation
Certain types of surgical operations can result in complications or abnormal reactions connected to liver cancer.
Pedal cycle accident injuring pedal cyclist
Though not directly related, this feature may suggest lifestyle or environmental factors that could be indirectly associated with liver cancer.
Fall from other slipping, tripping, or stumbling
This feature may be indicative of overall health status or lifestyle choices that may have indirect correlations with liver cancer.
None
This implies that no particular feature stands out for this rank but it is kept for the sake of completeness.
Other specified antibiotics causing adverse effects in therapeutic use
Specific antibiotics other than antimycobacterial drugs can also have adverse effects linked to liver cancer.

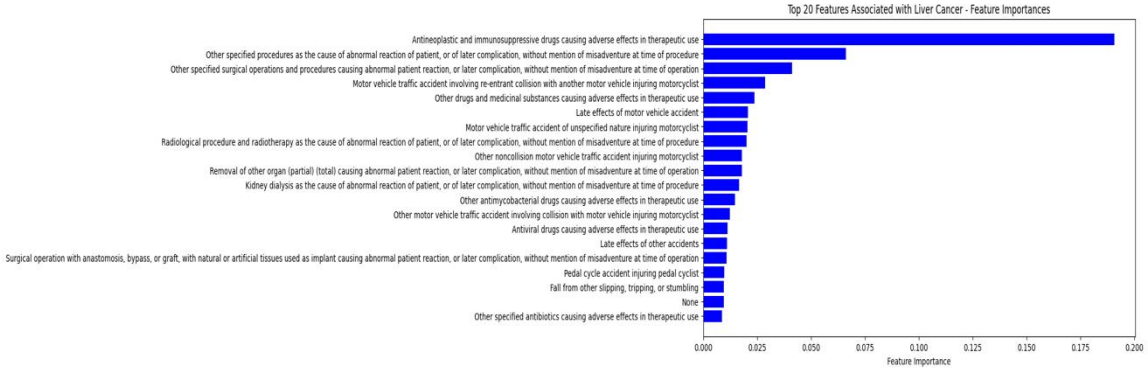


Figure 40:Feautre importances for liver cancer-external factors

6.1.4. Risk factors Analysis with external correlation matrix

A correlation matrix is a table that displays the correlation coefficients between multiple variables. In the context of liver cancer, a correlation matrix can provide insights into the strength and direction of the relationships between different factors and the occurrence or progression of liver cancer.The following diagram shows the colleration with y value . The correlations indicate the strength and direction of the relationship between different

factors and liver cancer. In the table, the higher the positive value, the greater the impact. Here are the five features with the biggest impact.

Surgical operation with anastomosis, bypass, or graft, with natural or artificial tissues used as implant causing abnormal patient reaction or later complication, without mention of misadventure at the time of operation	0.08
Drugs, medicinal, and biological substances causing adverse effects in therapeutic use, other specified drugs and medicinal substances	0.07
Unspecified fall	0.01
Adverse effect of antineoplastic and immunosuppressive drugs	0.30
Accidental poisoning by aromatic analgesics, not elsewhere classified	0.05
Late effects of motor vehicle accident	0.07
Adverse effect of other specified therapeutic agents, not elsewhere classified	0.03
Adverse effect of salicylates	0.05
Abnormal reaction of patient or complications of surgical procedures without mention of misadventure at the time of procedure, resulting from the implantation of internal joint prosthesis	0.10
Poisoning by opiates and related narcotics, undetermined whether accidentally or purposely inflicted	0.03
Surgical operation with transplant of whole organ causing abnormal patient reaction or later complication, without mention of misadventure at the time of operation	0.01
Operations involving the use of a heart-lung machine causing abnormal patient reaction or later complication, without mention of misadventure at the time of operation	0.06
Abnormal reaction of patient or complications of surgical procedures without mention of misadventure at the time of procedure, resulting from the implantation of unspecified device, implant, and graft	0.15

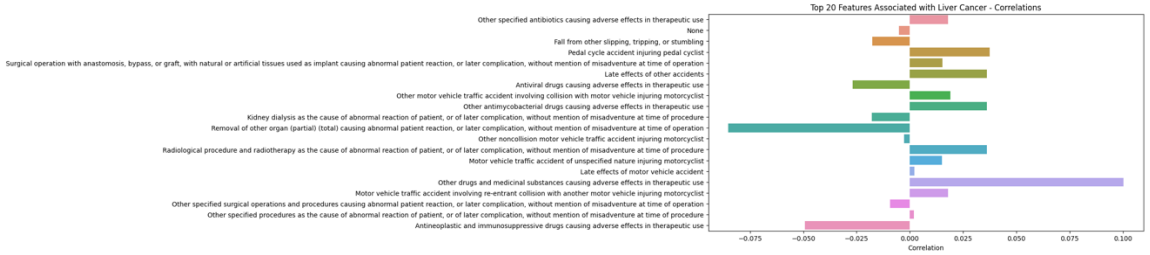


Figure 41:External factors correlation

The correlations indicate the strength and direction of the relationship between different factors and liver cancer. In the table, the higher the positive value, the greater the impact. Here are the five features with the biggest impact. The following code

7. Data Visualization Explainable dashboard

7.1. Prototype Overview

The prototype of the EDH will be presented and discussed at this section. For every model that have been created a dashboard that is either automatically generated or customized to meet the demands of each model has been developed. EDH is a collection of tools for rapidly creating interactive dashboards with several visualizations for evaluating and presenting the forecasts and processes of (scikit-learn compatible) ML models, such as xgboost, catboost and lightgbm. All these dashboards are analyzed and organized on a single page, the EDH, a web-based interface that will be incorporated into the DSS suite from I-help project . Moreover, the EDH includes cards with info (Title and type) of many Explainable Dashboards which are organized and summarized in one place, as depicted in Figure 42. In addition, this first software prototype provides the option to select and compare two different models in an interpretable way. More specifically, the EDH incorporates "comparison pages" that present two models' statistics and diagrams, so the HCPs can easily choose the models they want to view and analyse. This enables them to have an overview of the importance of specific features, allowing them to reach faster conclusions about the most significant factors in liver cancer, based on the comparison of the diagrams. The main functionalities of the EDH are:

- Explainable hub: Assessing Explainer Dashboards for all models and frameworks through a single location/dashboard.
- Filtering through the models.
- Provision of different modified and integrated visualizations.
- Various statistics and diagrams for every model, which are presented in an interactive way.
- What-If (in case it is turned on, when starting the dashboard) to help understand the changes in the model behavior, if the features or parts of the data are modified. It also allows for the comparison of different models.

Every single Explainable Dashboard interface contains different visualization tabs. Each tab includes different visualization categories, such as feature importance, classification stats and what-if analysis. More specifically, each Explainable Dashboard consists of the below tabs/visualizations:

- SHAP Values that illustrate how each factor individually influences the forecast.
- Feature importance that enables HCPs to go deeper into observing how model performance alters as a feature is shuffled.
- In the case of a Regression model utilizing XGBoost or RandomForestRegressor it makes possible to visualize the individual decision trees, whereas in the case of Classifier models, it could provide confusion matrices, ROC-AUC curves, etc., to better comprehend the models' decisions.

- What-If analysis (enabled before launching the dashboard) can help in understanding how the model's behavior varies when characteristics or portions of the data are altered. Additionally, it enables HCPs to compare various models.

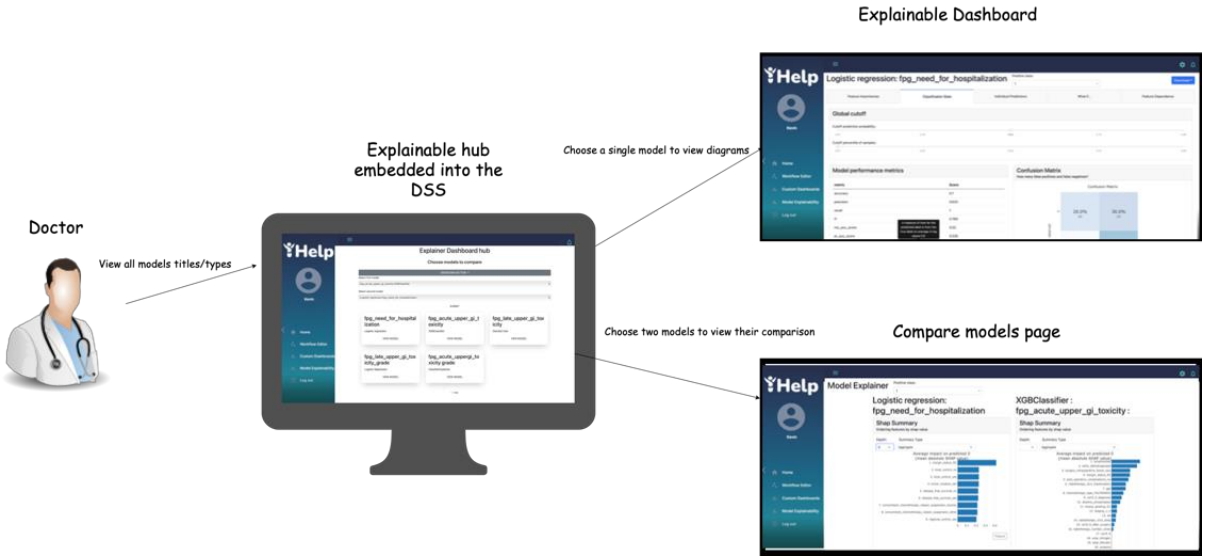


Figure 42: Explainable Dashboard Hub prototype overview

7.2. Interfaces

This section describes the interfaces included on the Explainable Dashboard Hub. The interfaces, as already mentioned, are embedded into the DSS suite. The main page of the EDH is a hub that contains cards with the info of each model. Specifically, the basic interface presents a model including its title and description, as well as a link that leads to the model's single Explainable Dashboard page, as depicted in Figure 43. In addition, Figure 44 depicts the Model Comparison tool that allows the HCPs to compare two different models to identify, for instance, the future importance of the different data attributes in those models.

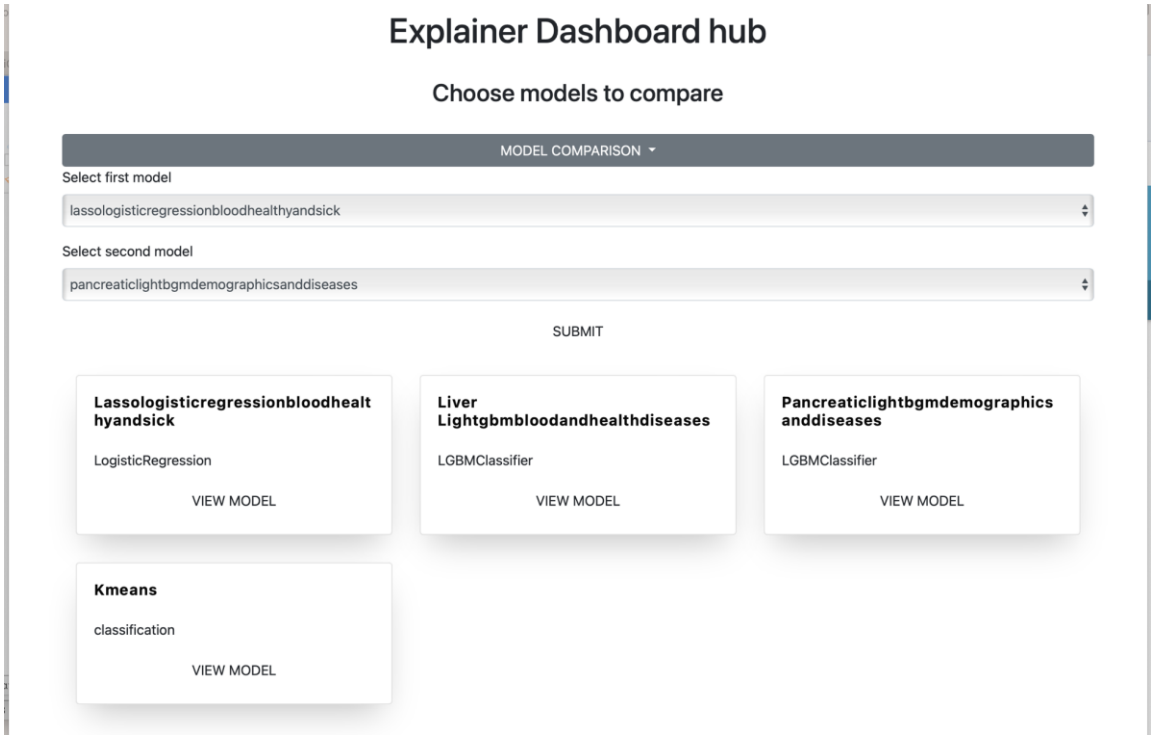


Figure 43: Explainable Dashboard Hub (EDH)

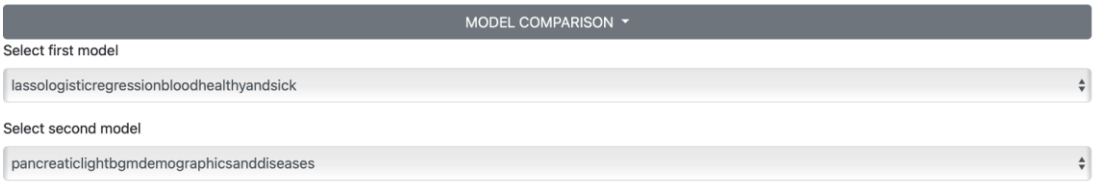


Figure 44: Model comparison tool

Every single Explainable Dashboard is an interactive web page application that extends the EDH and is stored in a specific link. The EDH, additionally, has pages with two separate model visualization diagrams. This enhancement enables the HCPs to compare different outcomes based on distinct attributes. The user can select the models to view the comparisons through dropdown buttons, as depicted in Figure 45.

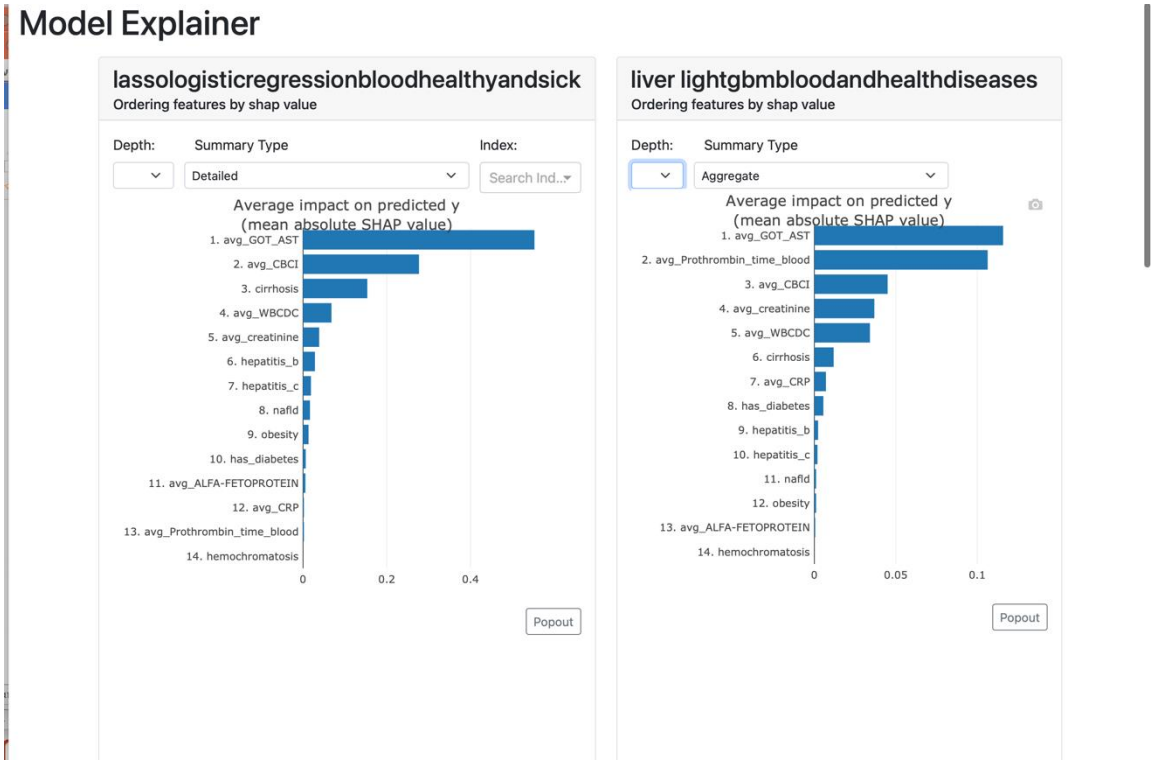


Figure 45: Model comparison page

7.3. Main Components

As mentioned also previously, the EDH is a collection of tools for rapidly creating interactive dashboards with several visualizations for evaluating and presenting the forecasts and processes of (scikit-learn compatible) ML models, such as xgboost, catboost and lightgbm. Depending on the type of the model, every single Explainable Dashboard has a single link and has tabs with specific main components. By main components is meant the visualization categories of the diagrams. Every main component category can be accessed by the HCPs through corresponding tabs in an interactive way. The main components are varied to fit the needs of each model. For example, in the case of the Decision tree models, one more tab may appear.

7.3.1. Feature importance

The term “Feature Importance” is used to characterize methods that provide a value to each of a model's input characteristics, with those values effectively representing the "importance" of those features to the outcome. More weight is given to the feature in the model's prediction that has a higher score. The feature importance visualization, as is presented in the Figure 46 below, showcases with each blue bar the importance of each feature to the outcome. For instance, in the case of the logistic regression model of “liver lightgbm for blood and diseases combine for healthy and sick patients ” the feature with the biggest impact is “avg_got_ast”, as depicted in Figure 46. Additionally, in this case, the presence of margin status r2-specific values indicates that the patient requires hospitalization more than any other feature.

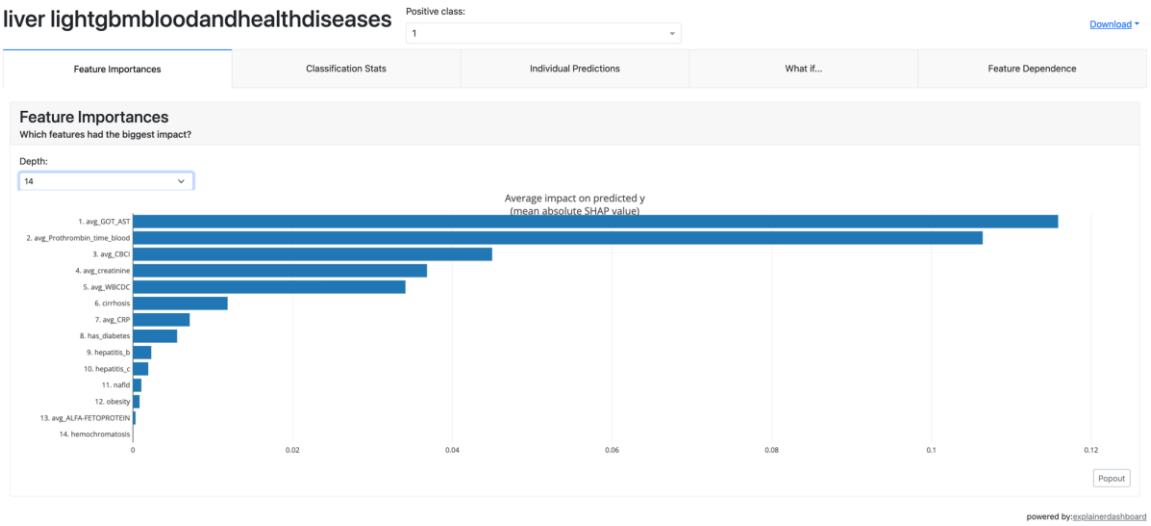


Figure 46: Feature Importance visualization

7.3.2. Classification Status

This Explainable Dashboard contains various diagrams and metrics visualizations that describe the model value, as it is presented in the Figure 47 below. Model performance metrics table contains Accuracy score, precision, Recall, F1, Roc AUC score, PR AUC score and Log loss. It also presents a confusion matrix figure, that shows the true positive, false negative, false positive and true negative predictions of the classification model.

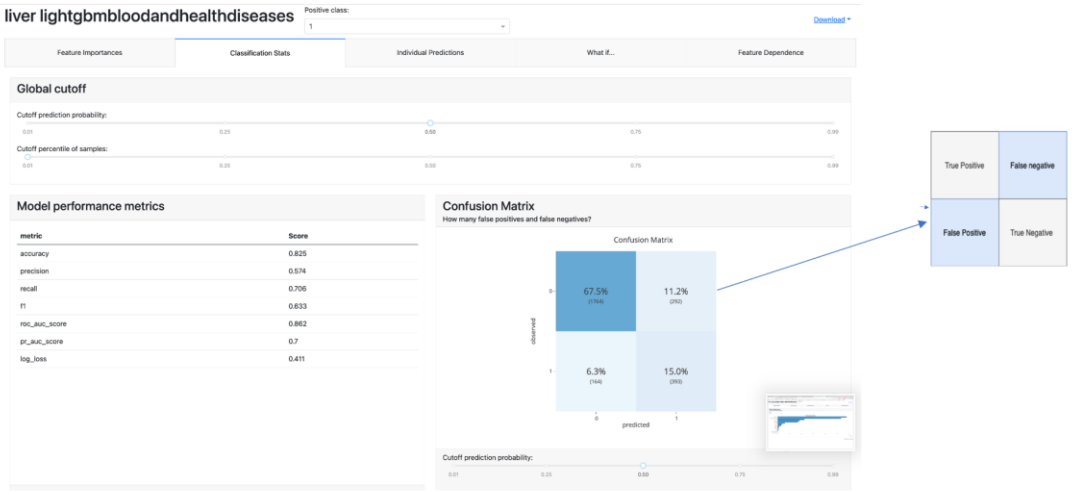


Figure 47: Classification Status visualization

Furthermore, it contains a precision plot and a classification plot as depicted in Figure 48 below. Precision is one metric of the performance of a ML model that indicates the quality of a model's accurate prediction sand is the ratio of the number of genuine positives to the overall number of positive forecasts (i.e., the number of true positives plus the number of false positives).

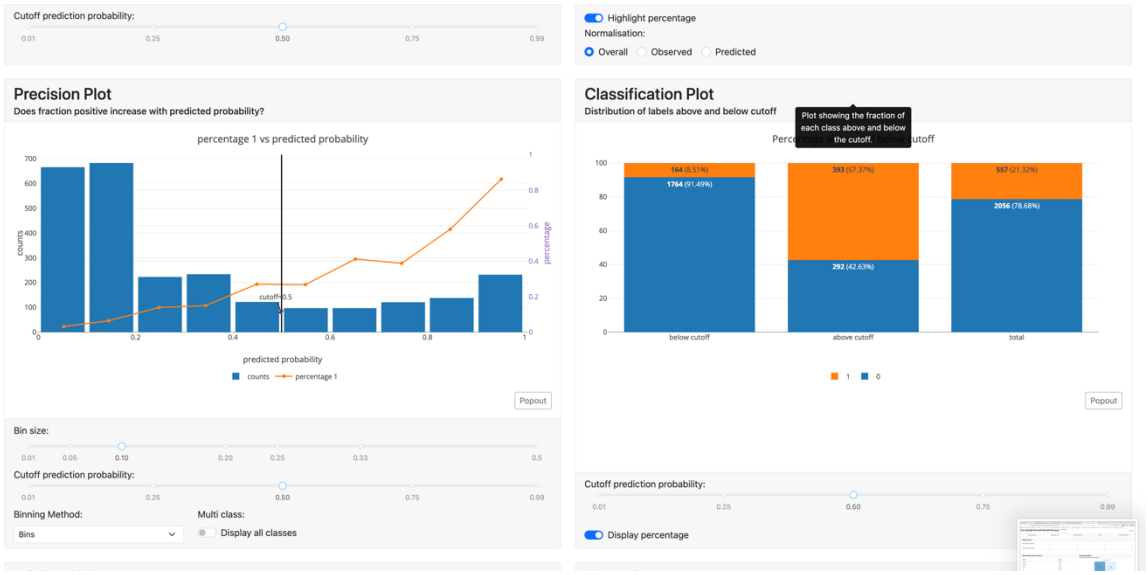


Figure 48: Precision and Classification plots

There is also a Lift curve that demonstrates the relationship between the number of instances that were predicted to be positive, and the actual number of positive examples as indicated in Figure 49. The latter also enables the comparison of the performance of a selected classifier to that of a random classifier and, finally, a cumulative precision diagram is presented.

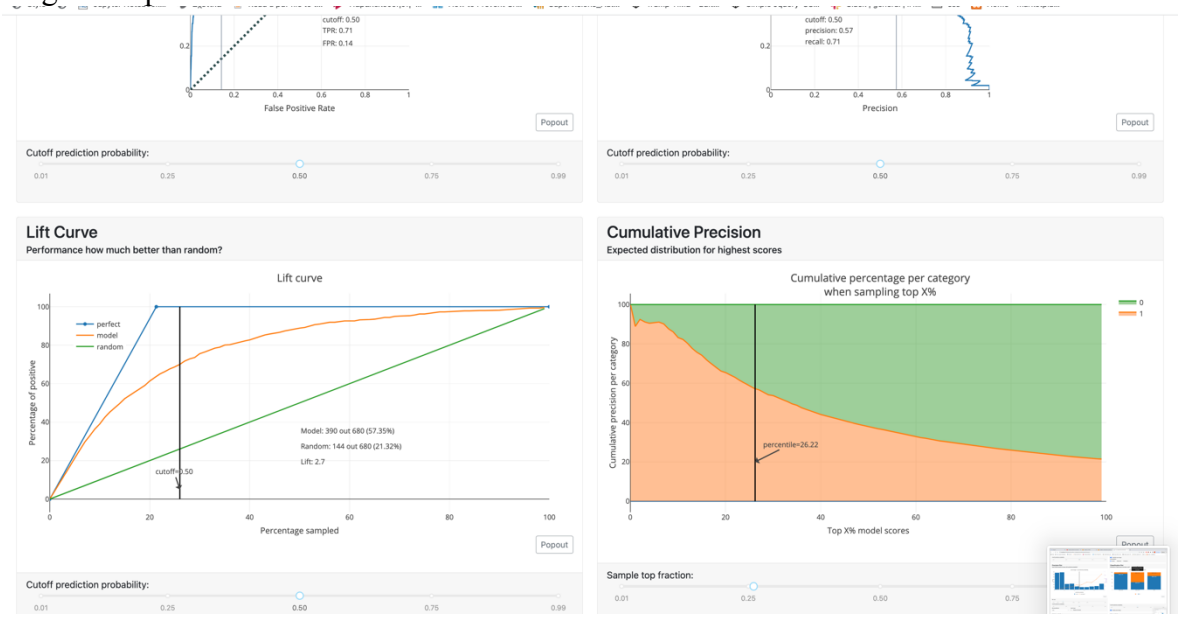


Figure 49: Lift cure and Cumulative Precision plots

7.3.3. Individual Predictions

This type of visualization contains interactive forms, pies, and diagrams, as presented in Figure 50. The HCPs could choose to see the influence of each feature in the model outcomes or combinations of many different inputs. Moreover, the individual predictions visualization includes a pie that presents the probability of each class align with the index of the dataset.

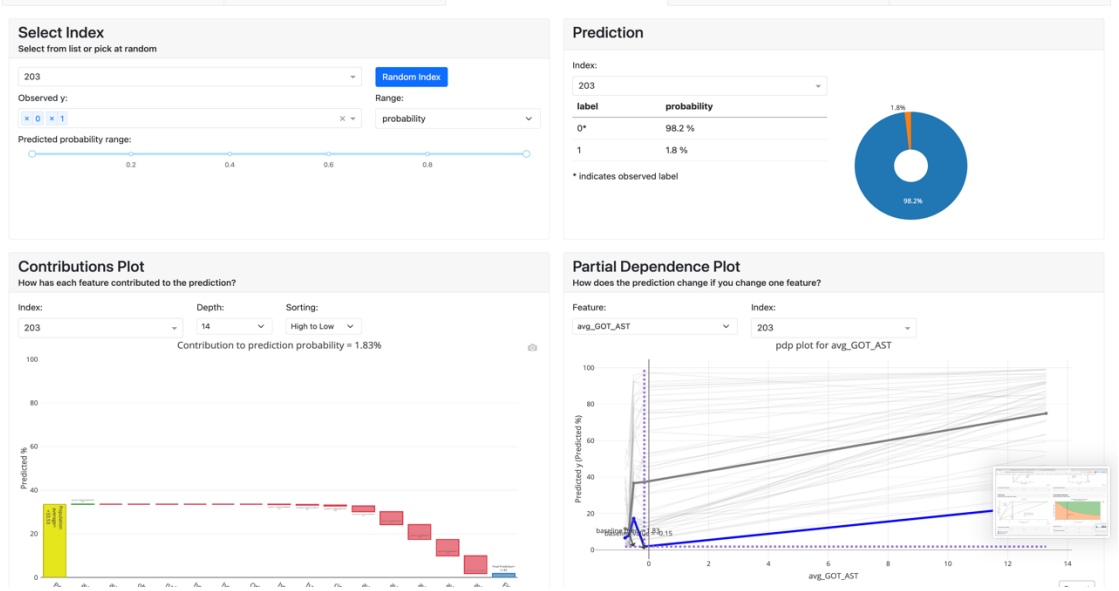


Figure 50: Individual prediction visualization

7.3.4. What if

What-If visualizations, depicted in Figure 51 below, (in case turned on while starting the dashboard) help HCPs to understand the changes in the model behaviour if they modify the specific features or attributes of the data. It also allows them to compare different models.

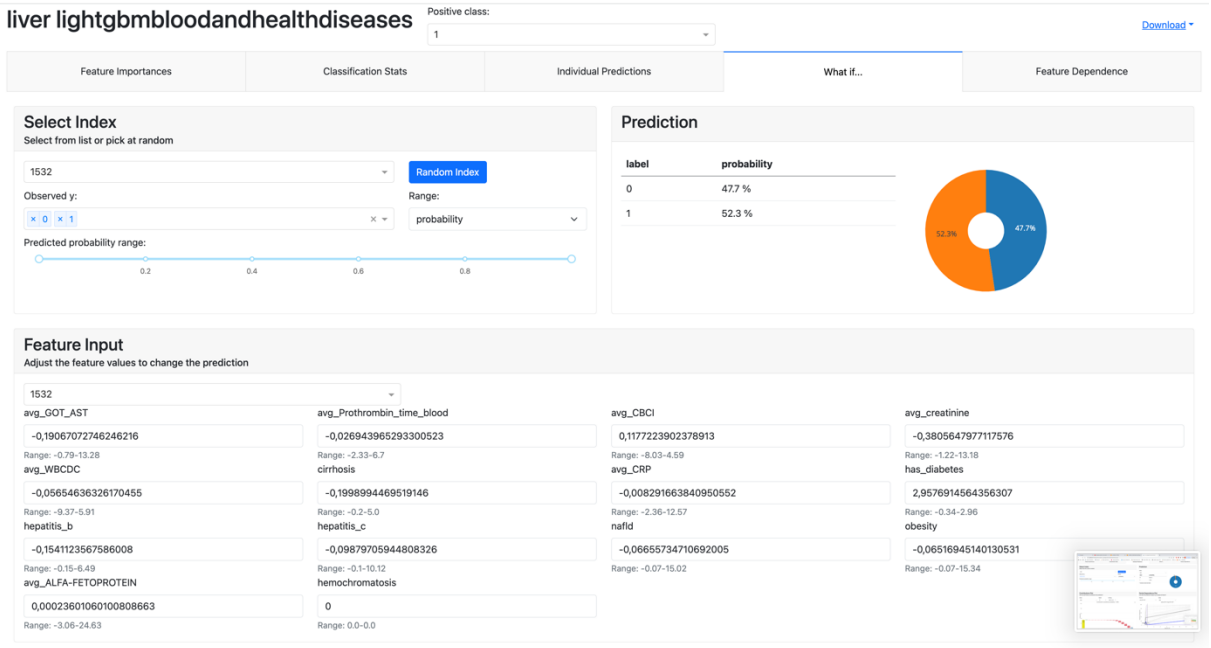


Figure 51: What-if visualization

7.4. Feature Dependence

SHAP Summary plot

SHAP (SHapley Additive exPlanations) values indicate how much a feature altered the outcome (compared to if we made that prediction at some baseline value of that feature). More specifically, SHAP is a game-theoretic method for explaining the results of any ML model. It integrates optimum credit allocation with local explanations by using the standard Shapley values from game theory and associated expansions. Moreover, the SHAP values of a model's output describe the effect of attributes on the outcome. The summary plot combines the importance and the effect of features, while each point on the

summary plot represents a Shapley value for an instance and a feature. In the case of the characteristic tumor location tail there are low Shapley values, since this is a least essential feature. More specifically, the bar depicts the features' value from low to high and the characteristics are listed in significance ordering.

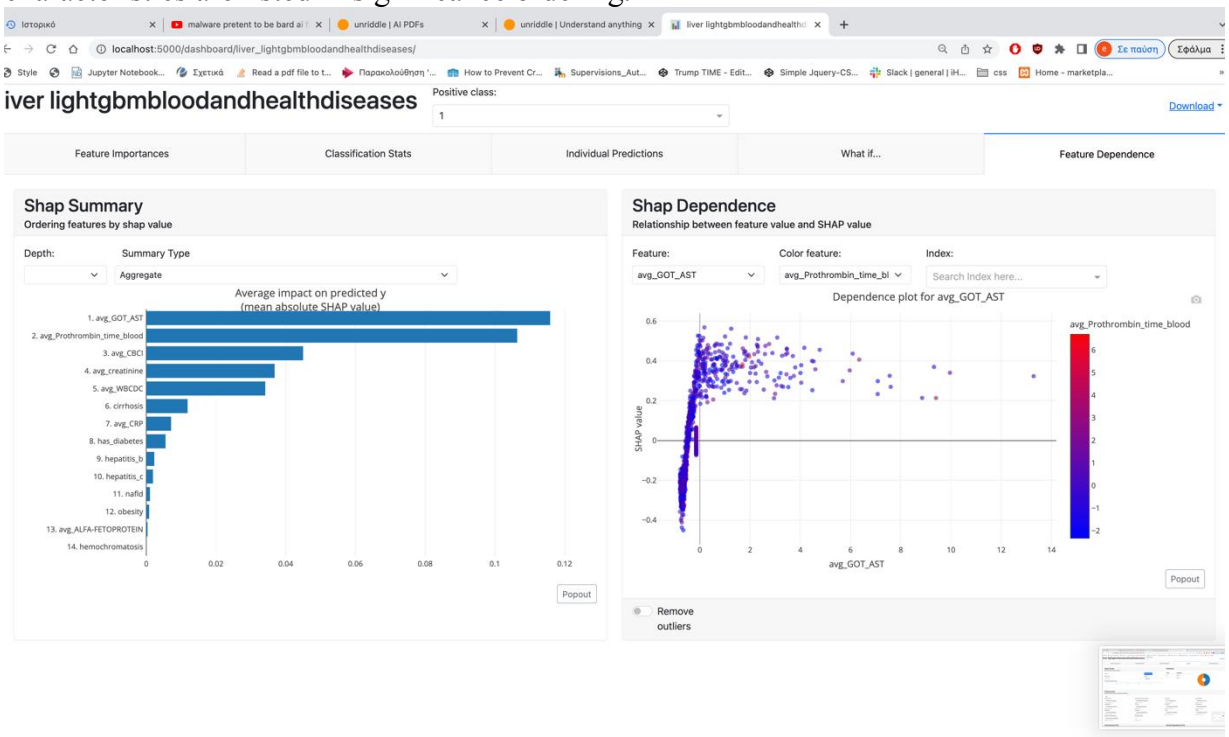


Figure 52: Feature significance visualization

7.4.1. Decision Tree visualizations

For decision trees models there is the decision tree visualization, as presented in Figure 53 below, that contains the decision path table. It is a table that shows for every node of the tree the split condition and threshold. A decision tree path is a table depicting the probable consequences of a set of interconnected decisions. It enables a specific list to compare alternative actions based on their costs, probabilities, and benefits. They may be used for, either casual conversation or the development of an algorithm that statistically predicts the optimum option.

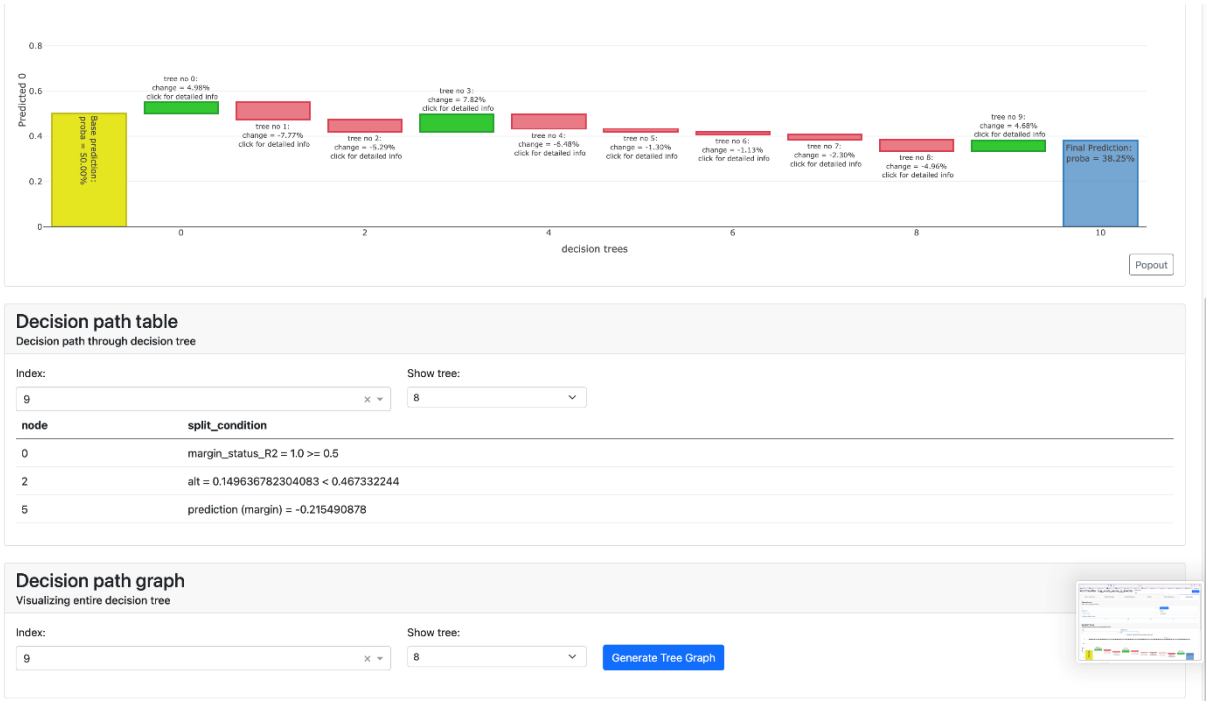


Figure 53: Decision tree visualization

7.5. Baseline Technologies and Tools

This section describes the technologies that have been utilized for the design and implementation of the core technical components of the EDH. The overall workflow and integration between the different technologies and tools is also presented in the **Error! Reference source not found.** below. The EDH has been implemented utilizing Python programming language and its widely used libraries, such as sklearn, plotly and explainerdashboard. Other programming languages like HTML, JS and CSS were also utilized to provide a more user-friendly frontend environment to the HCPs. To make the visualization process more efficient, the implementations are divided into two python scripts: generateexplainers.py and flaskapi.py. The first script creates for each model an explainer file with information about the plots, diagrams, and tables that are going to be embedded into the specific explainer dashboard. More specifically it decodes the pickle files that contains the models along with the Xtest , Ytest and save the info of the visualization in the explainer file. The second Python script (flaskapi.py) creates an API that interacts with the both the HCP and Model Builder and embeds the explainers into the presentation level to avoid calculate the results at the same time and be more optimized. Furthermore, it retrieves the explainers' files and generates dashboards, whenever it receives a request. In addition, flaskapi.py has user-requested custom methods that compare models. The information of the Explainable Dashboards is shown in cards, on the main html page (the Explainable Dashboard Hub), which is called from the Flask API using the "render template" method. In the case that an HCP clicks on a certain Explainable Dashboard, a request is sent to the Flask API and the HCP or the Model Builder is redirected to the appropriate link and model visualization.

7.6. Deployment of Explainable Dashboard Hub

The Explainable Hub implementation includes a folder name explainerdocker with ATC models, two Python files: generateexplainer.py, which generates explainer.joblib files, and app.py, which retrieves the explainer.joblib files and builds a flaskapi.The creation of the explainable dashboard hub container it can be achieved only with a two command
First we should go inside in explainerdocker folder with :

```
cd explainerdocker
```

and then inside the explainerdocker file that contains all the application files along with the pickle files and datasets we can call the following docker-compose file with the command

```
docker-compose up
```

More specific the docker-compose.yml file indluced one service an application that is build an flask app after

```
version: '2.2.2'

services:
  app:
    build: .
    command: python -u app.py
    environment:
      - FLASK_APP=app.py
      - FLASK_DEBUG=1
    ports:
      - "5000:5000"
```

As mentioned above there is also an python script that generated the explainers.joblib files ,
Joblib is a collection of tools-format that gives a more efficient technique to avoid recomputing the same function repeatedly, hence saving a significant amount of time and computational cost. In the case of the explainers, joblib stores the computations and information from the diagrams in order to optimize the explainable dashboards when the files are activated.The generateexplainers.py imported as library in the app.py file so when app.py runs from docker container first automate generateexplainers.py run.

7.6.1. Code explanations :

7.6.2. App.py custom classes object oriented code:

The code defines two Python classes, allexplainers and comparemodels, to organize information about machine learning models and their comparisons. It then scans the current working directory for files ending with .joblib, which typically contain saved machine learning models. For each such file, it creates an instance of the ClassifierExplainer class to analyze the model and an instance of the ExplainerDashboard class to display the model's behavior through a web dashboard. All relevant details, including the model name, dashboard URL, and explainer object, are stored in an instance of the allexplainers class, which is appended to a list. Finally, the code prints the names of all models for which explainers have been created

```
class allexplainers:
    def __init__(self, name,lnk,db,explainer,type):
        self.name = name
        self.lnk = lnk
        self.db = db
        self.explainer = explainer
        self.type= type

class comparemodels:
    def __init__(self, name1,name2,link,db):
        self.name1 = name1
        self.name2 = name2
        self.link = link
        self.db = db
```

```
list = []
comparelist = []
path = os.getcwd()
pathlist = os.listdir(path)
#for x in os.listdir():
for x in pathlist:
    if x.endswith(".joblib"):
        # Prints only text file present in My Folder
        print("the files in path ")
        print(x)
        a2=x.replace(".joblib","")
        a1=x.replace("_"," ")
        a=a1.replace(".joblib","")

        explainer =  ClassifierExplainer.from_file(x)

        #
home1="home/?model1=fpg_need_for_hospitalization&model2=fpg_late_upper_gi_toxicity_grade"
        dashname1="/dashboard/"
        d="/"

        url2=dashname1+a2+d
        print (url2)
        db = ExplainerDashboard(explainer,
                        shap_interaction=False, # you can switch off tabs with bools
                        title=a ,
                        decision_trees=True ,
                        model_summary=True,
                        no_permutations=True,
                        server=app, url_base_pathname=url2,
                        )
        model_type = type(explainer.model).__name__
        list.append( allexplainers(a,url2,db,explainer,model_type) )



for obj in list:
    print( obj.name )
```

7.6.3. Extend the use of explainable dashboard python library by creating custom compare classes.

The ConfusionComparison class is a custom component designed to compare two machine learning models using dashboards. It inherits from ExplainerComponent, which presumably is a base class for creating explainer components. When an instance of ConfusionComparison is created, it takes in two explainer objects (explainer and explainer4), names for the two models (name1 and name2), and an optional title for the dashboard.
Inside its __init__ method, it initializes four components—two ShapSummaryComponent objects and two ShapDependenceComponent objects—for both of the models. These components will display the SHAP summary and dependence plots, which are useful for understanding and interpreting the models. The method also sets a cutoff value of 0.6 for the SHAP values and hides some elements like the selector and percentage for simplification.

The layout method of the class arranges these components in a dashboard using a container-row-column architecture provided by the dbc library. The ShapSummaryComponent and ShapDependenceComponent for the first model (name1)

are placed in one column, while the same components for the second model (name2) are placed in another column, allowing for side-by-side comparison. Overall, the ConfusionComparison class aims to facilitate the comparative analysis of two machine learning models through a visual dashboard.

```
class ConfusionComparison(ExplainerComponent):
       def __init__(self,explainer, explainer4,name1,name2,title="Compare models dashboard"):
        super().__init__(explainer)
        self.confmat1 = ShapSummaryComponent(explainer, cutoff=0.6,
                            hide_selector=True, hide_percentage=True,title=name1)
        self.confmat2 = ShapSummaryComponent(explainer4, cutoff=0.6,
                            hide_selector=True, hide_percentage=True,title=name2)
        self.confmat3 = ShapDependenceComponent(explainer, cutoff=0.6,
                            hide_selector=True, hide_percentage=True)
        self.confmat4 = ShapDependenceComponent(explainer4, cutoff=0.6,
                            hide_selector=True, hide_percentage=True)


       def layout(self):
        return dbc.Container([
            dbc.Row([
                dbc.Col([
                    html.H1(""),
                    self.confmat1.layout(),
                    self.confmat3.layout()

                ]),
                dbc.Col([
                    html.H1(""),
                    self.confmat2.layout(),
                    self.confmat4.layout()
                ])
            ])
        ])
```

7.6.4. Html and JavaScript:

This code snippet is a part of the code of home page combines HTML, JavaScript, and Jinja2 templating to create a user interface for selecting and viewing machine learning models. It features a dropdown menu populated with model names and their respective links. Users can also select two models from dynamically populated dropdown lists for comparison. Upon clicking a "Submit" button, the selected models are sent to a new URL where a comparative dashboard likely appears. Additionally, the code generates a set of cards, each displaying details about an individual model, such as its name and type, as well as a link to its dedicated dashboard. The card and dropdown elements are dynamically populated from a Python list named list.

```
  <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
                {% for item in list %}
                <a class="dropdown-item" href={{item.lnk}}>{{item.name}}</a> {% endfor %}
            </div>

        </div>

        <div class="form-group">
            <label for="exampleFormControlSelect1">Select first model</label>
            <select name="model1" class="form-control" id="model1">
                     {% for item in list %}
                 <option value="{{item.name}}">{{item.name}}</option>

                 {% endfor %}
```

```
            </select>
        </div>
        <div class="form-group">
            <label for="exampleFormControlSelect2">Select second model</label>
            <select name='model2' class="form-control" id="model2">
                {% for item in list %}
                <option value="{{item.name}}"">{{item.name}}</option>

                {% endfor %}
              </select>
        </div>
        <button onclick="myFunction1()" class="btn btn-default">Submit</button>
        <p id="demo1"></p>
        <p id="demo2"></p>
        <script>
            function myFunction1() {
                var x = document.getElementById("model1").value;
                var y = document.getElementById("model2").value

                var comb = x + y
                var location1 = window.location.hostname + ":5000"
                var alltogether = "/dashboard/" + comb
                location.assign(alltogether);

            }
        </script>

        <ul class="cards">
            {% for item in list %}
            <li class="cards_item">
                <div class="card" style="width: 25rem;">
                    <div class="card-body">
                        <h2 class="card_title">{{item.name}}</h2>

                        <p class="card-text"> <br> {{item.type}} <br> </p>
                        <a href={{item.lnk}} class="btn card_btn">View Model</a>

                    </div>
                </div>
            </li>
            {% endfor %}
```

8. Conclusion

The results of our study offer a robust and comprehensive methodology for liver cancer prediction, considering a broad range of variables and employing advanced machine learning models. The key features identified by our models - age, marry status , sex type, and various health conditions such as cirrhosis , hepatitis b , SEX_TYPE, got ast , cachexia- align with the current scientific understanding of significant risk factors for liver cancer .The importance of utilizing big data techniques in health data should be emphasized. The more variables we want to include in an analysis, the more complex it becomes, especially when it comes to unsupervised learning. The methodology proposed by this paper includes advanced techniques that can handle a large dataset, utilizing Apache Sedona and PySpark for parallel processing in preprocessing, cleaning, and mapping with custom made dictionaries and functions. In addition, it incorporates techniques that handle the inequality of classes in the target value “y” like SMOTE, imbalanced, stratified k-fold. This particular analysis can be applied as is to liver cancer data of this specific pilot. In the future, it can be expanded and adapted to any health problem with complex data as it cleans and combines many demanding datasets and different use cases, such as demographic data, blood tests, illnesses, and external factors also included ICD-9 and ICD-10 protocols. We also utilize and extend the explainable dashboard library by creating a custom flask API that unlocked the black box of machine learning models .Moreover, for each use case, the appropriate ML models that maximize accuracy have been selected after many tests.with the highest accuracy The results shows Among the most impactful conditions related to liver cancer are "Cirrhosis of the liver without mention of alcohol," "Chronic hepatitis, unspecified," and "Chronic viral hepatitis B," all of which directly pertain to liver health. In the given data, each variable represents the average value of a particular medical indicator or condition across a sample population, where "y" indicates whether an individual has cancer. "CBCI" stands for average Complete Blood Count Index with a value of 0.3855, which might give insights into the overall health of the blood. "WBCDC» represents the average White Blood Cell Differential Count, a marker for potential infection or other diseases. "Prothrombin_time_blood" at 0.6697 related to how quickly blood clots, a crucial factor in many diseases. "creatinine» provides information about kidney function. "avg_GOT_AST" (0.8742) is the average level of the enzyme Aspartate Aminotransferase, generally related to liver health. Other variables represent the proportion of individuals with specific conditions: "obesity" (0.0119), "nafld" or Non-Alcoholic Fatty Liver Disease, "cirrhosis" , "hepatitis (0.0331), "hepatitis b" (0.0604), and "diabetes" (0.0649). Each of these factors may have varying degrees of association with the presence of cancer, indicated by "y," and could be important for medical researchers and healthcare professionals for predictive analysis and treatment planning. "Essential (primary) hypertension" and "Unspecified essential hypertension" could have an indirect but significant impact on liver health, affecting systemic conditions. Conditions like "Secondary malignant neoplasm of lung" and "Diabetes mellitus" suggest that liver cancer is often associated with other serious health issues. Lower on the scale of direct impact are conditions like "Constipation," "Reflux esophagitis," and "Anemia," which may be symptoms or side effects rather than causative factors. "Unspecified functional disorder of stomach" and "Hypertensive heart disease" may also be indicative of overall poor health but are less directly related to liver cancer. Surprisingly, "Insomnia" and "Urinary tract infection" also appear, indicating that a wide range of health issues may correlate with liver cancer.. In terms of accuracy, the machine learning models showed promising results for predicting liver cancer and NAFLD outcomes. From the external factors the most possible association are antineoplastic and immunosuppressive drugs sides effects, Procedures side effects, Traffic accident Other drugs and medical substances. The LightGBM model was particularly noteworthy, achieving an accuracy of 0.83 and a precision score of 0.91, making it the top-

performing model. Following closely behind was the Lasso Logistic Regression model with an accuracy of 0.78 and a precision score of 0.89. The base classifiers also demonstrated strong performance, with accuracy levels of 0.81 and precision scores ranging from 0.89 to 0.90. Given these robust statistics, we chose to utilize the LightGBM and Lasso Logistic Regression models for predicting liver cancer , as they offer a combination of high accuracy and precision. appear, suggesting a holistic approach to health might be crucial for liver cancer prevention or management. The introduced methodology offers an enhanced and in depth analysis of many different factors contributing to the developing of liver cancer it is already include in pancreatic cancer analysis and can have a generalized application in similar and demanding health issues and other types of cancer.

9. References

[1]"Key Statistics About Liver Cancer," *Key Statistics About Liver Cancer | American Cancer Society*, Jan. 12, 2023. https://www.cancer.org/cancer/types/liver-cancer/about/what-is-key-statistics.html

[2]Vekariya, V.K., Passi, K., & Jain, C.K. (2022). Predicting liver cancer on epigenomics data using machine learning. Frontiers in Bioinformatics, 2.

[3] Efficient Local Cloud-Based Solution for Liver Cancer detection using Deep Learn

[4] Bakrania, A., Joshi, N., Zhao, X., Zheng, G., & Bhat, M. (2023). Artificial intelligence in liver cancers: Decoding the impact of machine learning models in clinical diagnosis of primary liver cancers and liver cancer metastases. Pharmacological research, 189, 106706. https://doi.org/10.1016/j.phrs.2023.106706

[5]Title: Machine Learning Can Predict Total Death After Radiofrequency Ablation in Liver Cancer Patients
Tong, Jian-hua, Panmiao Liu, Mu-huo Ji, Ying Wang, Qiong Xue, Jian-jun Yang and Cheng-Mao Zhou. "Machine Learning Can Predict Total Death After Radiofrequency Ablation in Liver Cancer Patients." Clinical Medicine Insights. Oncology 15 (2021): n. pag.(5)

[6]Title: A Machine Learning Approach Yields a Multiparameter Prognostic Marker in Liver Cancer(6)
Liu, X., Lu, J., Zhang, G., Han, J., Zhou, W., Chen, H., Zhang, H., & Yang, Z. (2021). A Machine Learning Approach Yields a Multiparameter Prognostic Marker in Liver Cancer. Cancer Immunology Research, 9, 337 - 347.

[7]Title: Artificial Intelligence in Liver Cancers: Decoding the Impact of Machine Learning Models in Clinical Diagnosis of Primary Liver Cancers and Liver Cancer Metastases.

Bakrania, A.K., Joshi, N., Zhao, X., Zheng, G., & Bhat, M. (2023). fetArtificial Intelligence in Liver Cancers: Decoding the Impact of Machine Learning Models in Clinical Diagnosis of Primary Liver Cancers and Liver Cancer Metastases. Pharmacological research, 106706 .(7)
[8]Title: Application of Machine Learning for Diagnosis of Liver Cancer
(8)

Trivedi, N.K., Tiwari, R.G., Anand, A., Gautam, V., Witarsyah, D., & Misra, A. (2022). Application of Machine Learning for Diagnosis of Liver Cancer. 2022 International Conference Advancement in Data Science, E-learning and Information Systems (ICADEIS), 1-5.

[9]Predicting liver cancer on epigenomics data using machine learning
Vekariya, V.K., Passi, K., & Jain, C.K. (2022). Predicting liver cancer on epigenomics data using machine learning. Frontiers in Bioinformatics, 2.(9)

[10] Machine Learning Approach to Facilitate Knowledge Synthesis at the Intersection of Liver Cancer, Epidemiology, and Health Disparities Research(10)
Hyams, T., Luo, L., Hair, B.Y., Lee, K., Lu, Z., & Seminara, D. (2022). Machine Learning Approach to Facilitate Knowledge Synthesis at the Intersection of Liver Cancer, Epidemiology, and Health Disparities Research. JCO Clinical Cancer Informatics, 6.

[11]Preoperative classification of primary and metastatic liver cancer via machine learning-based ultrasound radiomics(11)

Mao, B., Ma, J., Duan, S., Xia, Y., Tao, Y., & Zhang, L. (2021). Preoperative classification of primary and metastatic liver cancer via machine learning-based ultrasound radiomics. European Radiology, 31, 4576 - 4586.

[12] Identification of Significant Gene Expression in Liver Cancer-Induced HBx Virus Using Enhanced Machine Learning Method
Muflikhah, L., Widodo, Mahmudy, W.F., & Solimun (2021). Identification of Significant Gene Expression in Liver Cancer-Induced HBx Virus Using Enhanced Machine Learning Method. Lecture Notes in Networks and Systems.

[13] Machine learning-based development and validation of a scoring system for progression-free survival in liver cancer
Liu, X., Hou, Y., Wang, X., Yu, L., Wang, X., Jiang, L., & Yang, Z. (2020). Machine learning-based development and validation of a scoring system for progression-free survival in liver cancer. Hepatology International, 14, 567-576.

[14]: A machine learning method for improving liver cancer staging
Zhao, Z., Tian, Y., Yuan, Z., Zhao, P., Xia, F., & Yu, S. (2022). A machine learning method for improving liver cancer staging. Journal of biomedical informatics, 137, 104266 .

[15] Machine Learning Methods Using Texture Feature Selection in Diagnosis of Liver Cancer

Naeem, S., Ali, A., Anam, S., & Zubair, M. (2022). Machine Learning Methods Using Texture Feature Selection in Diagnosis of Liver Cancer. Proceedings of MOL2NET'22, Conference on Molecular, Biomedical & Computational Sciences and Engineering, 8th ed. - MOL2NET: FROM MOLECULES TO NETWORKS.

[16] Predicting postoperative liver cancer death outcomes with machine learning
Wang, Y., Ji, C., Wang, Y., Ji, M., Yang, J., & Zhou, C. (2021). Predicting postoperative liver cancer death outcomes with machine learning. Current Medical Research and Opinion, 37, 629 - 634.

[17] Recurrence Risk of Liver Cancer Post-hepatectomy Using Machine Learning and Study of Correlation With Immune Infiltration
Qian, X., Zheng, H., Xue, K., Chen, Z., Hu, Z., Zhang, L., & Wan, J. (2021). Recurrence Risk of Liver Cancer Post-hepatectomy Using Machine Learning and Study of Correlation With Immune Infiltration. Frontiers in Genetics, 12.

[18]Surveillance Strategy for Barcelona Clinic Liver Cancer B Hepatocellular Carcinoma Achieving Complete Response: An Individualized Risk-Based Machine Learning Study
Chen, Q., Dai, L., Wu, Y., Huang, Z., Chen, M., & Zhao, M. (2021). Surveillance Strategy for Barcelona Clinic Liver Cancer B Hepatocellular Carcinoma Achieving Complete Response: An Individualized Risk-Based Machine Learning Study. Frontiers in Bioengineering and Biotechnology, 9.

[19] Extraction of ncRNAs Associated with Liver Cancer Using Machine Learning
Sekine, K., Hochin, T., & Nomiya, H. (2021). Extraction of ncRNAs Associated with Liver Cancer Using Machine Learning. Proceedings of the the 8th International Virtual Conference on Applied Computing & Information Technology.

[20]: Preoperative classification of primary and metastatic liver cancer via machine learning-based ultrasound radiomics
Mao, B., Ma, J., Duan, S., Xia, Y., Tao, Y., & Zhang, L. (2021). Preoperative classification of primary and metastatic liver cancer via machine learning-based ultrasound radiomics. European Radiology, 31, 4576 - 4586.

[21] Machine learning-empowered cis-diol metabolic fingerprinting enables precise diagnosis of primary liver cancer
Li, P., Xu, S., Han, Y., He, H., & Liu, Z. (2023). Machine learning-empowered cis-diol metabolic fingerprinting enables precise diagnosis of primary liver cancer. Chemical Science, 14, 2553 - 2561.

[22]A comparative study of machine learning algorithms for predicting acute kidney injury after liver cancer resection
Lei, L., Wang, Y., Xue, Q., Tong, J., Zhou, C., & Yang, J. (2020). A comparative study of machine learning algorithms for predicting acute kidney injury after liver cancer resection. PeerJ, 8.

[23]An Intelligent Approach to Segment the Liver Cancer using Machine Learning Method

Anand, L., Maurya, M., Seetha, J., Nagaraju, D., Ravuri, A., & Vidhya, R.G. (2023). An Intelligent Approach to Segment the Liver Cancer using Machine Learning Method. 2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC), 1488-1493.

[24] Potential biomarker detection for liver cancer stem cell by machine learning approach
Farzane, A., Akbarzadeh, M., Ferdousi, R., Rashidi, M.R., & Safdari, R. (2020). Potential biomarker detection for liver cancer stem cell by machine learning approach. Journal of Contemporary Medical Sciences.

[25] Machine-Learning Classification Models to Predict Liver Cancer with Explainable AI to Discover Associated Genes
Hasan, M.E., Mostafa, F.B., Hossain, M.S., & Loftin, J. (2023). Machine-Learning Classification Models to Predict Liver Cancer with Explainable AI to Discover Associated Genes. AppliedMath.

[26] Hepatocellular Carcinoma (HCC) Liver Cancer prediction using Machine Learning Algorithms
Rajesh, S., Choudhury, N.A., & Moulik, S. (2020). Hepatocellular Carcinoma (HCC) Liver Cancer prediction using Machine Learning Algorithms. 2020 IEEE 17th India Council International Conference (INDICON), 1-5.

[27] Extraction of Genes Associated with Liver Cancer Using Machine Learning
Sekine, K., Hochin, T., & Nomiya, H. (2020). Extraction of Genes Associated with Liver Cancer Using Machine Learning. 2020 9th International Congress on Advanced Applied Informatics (IIAI-AAI), 7-12.

[28] Machine Learning aided Fiber-Optical System for Liver Cancer Diagnosis in Minimally Invasive Surgical Interventions
Zherebtsov, E.A., Zajnulina, M., Kandurova, K.Y., Dremin, V.V., Mamoshin, A.V., Potapova, E.V., Sokolovski, S., Dunaev, A.V., & Rafailov, E.U. (2020). Machine Learning aided Fiber-Optical System for Liver Cancer Diagnosis in Minimally Invasive Surgical Interventions. 2020 International Conference Laser Optics (ICLO), 1-1.

[29]Zhou, G., Zhang, Y., You, Y., Wang, B., Wang, S., Yang, C., Zhang, Y., & Liu, J. (2022). Contrast-Enhanced Ultrasound and Magnetic Resonance Enhancement Based on Machine Learning in Cancer Diagnosis in the Context of the Internet of Things Medical System. Computational Intelligence and Neuroscience, 2022.

[30] Radiomics and Machine Learning Analysis Based on Magnetic Resonance Imaging in the Assessment of Colorectal Liver Metastases Growth Pattern
Granata, V., Fusco, R., De Muzio, F., Cutolo, C., Mattace Raso, M., Gabelloni, M., Avallone, A., Ottaiano, A., Tatangelo, F., Brunese, M.C., Miele, V., Izzo, F., & Petrillo, A. (2022). Radiomics and Machine Learning Analysis Based on Magnetic Resonance Imaging in the Assessment of Colorectal Liver Metastases Growth Pattern. Diagnostics

[31]Survival Time Analysis of Liver Cancer Patients using Machine Learning
Ferdous, M., Hossain, M.M., & Robi, F.M. (2019). Survival Time Analysis of Liver Cancer Patients using Machine Learning.

[32]Application of machine learning to determine the characteristics of adjacent normal tissues in liver cancer
Shams, W.K. (2017). Application of machine learning to determine the characteristics of adjacent normal tissues in liver cancer.

[33]Risk Assessment of Liver Metastasis in Liver Cancer Patients Using Multiple Models Based on Machine Learning: A Large Population-Based Study
Authors: Qinggang Li, L. Bai, Jiyuan Xing, Xiaorui Liu, Dan Liu, Xiaobo Hu

[34]Ali, R.N., Balamurali, M., & Varamini, P. (2022). Deep Learning-Based Artificial Intelligence to Investigate Targeted Nanoparticles' Uptake in TNBC Cells. International Journal of Molecular Sciences, 23.

[35]Kefelegn, S., & Kamat, P. (2018). Prediction and Analysis of Liver Disorder Diseases by using Data Mining Technique: Survey. International Journal of Pure and Applied Mathematics, 118(9), 765-770.

[36] Home. (2023, October 17). https://www.who.int

[37] Information and Resources about for Cancer: Breast, Colon, Lung, Prostate, Skin. (n.d.). American Cancer Society. https://www.cancer.org

[38Information and Resources about for Cancer: Breast, Colon, Lung, Prostate, Skin. (n.d.). American Cancer Society. https://www.cancer.org

[39] National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK). (2023, July 19). National Institute of Diabetes and Digestive and Kidney Diseases. https://www.niddk.nih.gov [40]World Cancer Research Fund

[41] IARC – INTERNATIONAL AGENCY FOR RESEARCH ON CANCER. (n.d.). https://www.iarc.who.int

[42]Morgan TR, Mandayam S, Jamal MM. Alcohol and hepatocellular carcinoma. Gastroenterology. 2004;127(5 Suppl 1):S87-S96.

[43]El-Serag HB. Epidemiology of viral hepatitis and hepatocellular carcinoma. Gastroenterology. 2012;142(6):1264–1273.e1.

[44]Liu Y, Wu F. Global burden of aflatoxin-induced hepatocellular carcinoma: a risk assessment. Environmental Health Perspectives. 2010;118(6):818–824.

[45]Wong RJ, Cheung R, Ahmed A. Nonalcoholic steatohepatitis is the most rapidly growing indication for liver transplantation in patients with hepatocellular carcinoma in the U.S. Hepatology. 2014;59(6):2188-95.

[46]Chuang SC, La Vecchia C, Boffetta P. Liver cancer: descriptive epidemiology and risk factors other than HBV and HCV infection. Cancer Letters. 2009;286(1):9-14.

[47] S. Laurent, "Antihypertensive drugs," Pharmacological research, vol. 124, pp. 116-125, 2017.

[48] A. E. Walker, “The adult pancreas in trauma and disease,” Academic forensic pathology, vol. 8, no. 2, pp. 192-218, 2018.

[49]Apache spark : https://spark.apache.org

[50]Explainable Dashboard Library by Oege Dijk
https://explainerdashboard.readthedocs.io/en/latest/dashboards.html

[51] Manias, George, et al. "An evaluation of neural machine translation and pre-trained word embeddings in multilingual neural sentiment analysis." 2020 IEEE International Conference on Progress in Informatics and Computing (PIC). IEEE, 2020.

[52] Manias, George, et al. "SemAI: A novel approach for achieving enhanced semantic interoperability in public policies." Artificial Intelligence Applications and Innovations: 17th IFIP WG 12.5 International Conference, AIAI 2021, Hersonissos, Crete, Greece, June 25–27, 2021, Proceedings 17. Springer International Publishing, 2021.

[53] Manias, George, et al. "An Enhanced Standardization and Qualification Mechanism for Heterogeneous Healthcare Data." Caring is Sharing–Exploiting the Value in Data for Health and Innovation (2023): 153.