ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΜΣ «ΠΛΗΡΟΦΟΡΙΚΗ»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής:	***Ελαχιστοποίηση πιστωτικού κινδύνου με χρήση αλγορίθμων μηχανικής μάθησης*** *Minimization of credit risk with the use of machine learning algorithms*
Ονοματεπώνυμο φοιτητή:	*ΓΕΩΡΓΙΟΣ ΦΡΑΓΚΑΚΗΣ*
Πατρώνυμο:	*ΔΗΜΗΤΡΙΟΣ*
Αριθμός Μητρώου:	*ΜΠΠΛ19065*
Επιβλέπων:	*ΔΙΟΝΥΣΙΟΣ ΣΩΤΗΡΟΠΟΥΛΟΣ, ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ*

Ημερομηνία Παράδοσης **Οκτώβριος 2022**

Τριμελής Εξεταστική Επιτροπή

Γ. Τσιχριντζής Καθηγητής	Δ. Σωτηρόπουλος Επίκουρος Καθηγητής	Ε. Σακκόπουλος Αναπληρωτής Καθηγητής

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Διονύσιο Σωτηρόπουλο για την επιμονή, την υπομονή αλλά και την καθοδήγηση που μου έδωσε σε κάθε βήμα των μεταπτυχιακών μου σπουδών και ιδιαίτερα για την ολοκλήρωση της μεταπτυχιακής μου διατριβής.

Τίποτα όμως δεν θα ήταν εφικτό χωρίς την στήριξη αλλά και την βοήθεια των ανθρώπων που στέκονται δίπλα μου σε κάθε μου βήμα. Ένα μεγάλο ευχαριστώ στους γονείς μου, την αδερφή μου αλλά και την κοπέλα μου που είναι πάντα εκεί για μένα και με βοηθούν να επιτύχω κάθε μου στόχο και κάθε μου όνειρο.

Περίληψη

Τα επονομαζόμενα πιστωτικά χαρακτηριστικά, δηλαδή τα χαρακτηριστικά βάσει των οποίων κάποιος αξιολογείται για τον αν είναι φερέγγυος ως προς την πίστωση του δανείου ή όχι, αποτελούν τον σημαντικό μέρος της τραπεζικής καθημερινότητας, αφού όλο και περισσότεροι άνθρωποι επιθυμούν την απόκτηση καταναλωτικών δανείων. Φυσικά στην εποχή μας τα χαρακτηριστικά, τα οποία αξιολογούν κάποιον για την πιστωτική του ικανότητα, είναι τόσα πολλά όπου καθιστούν την ανάλυση από τον άνθρωπο έως και ακατόρθωτη. Στην παρακάτω διπλωματική εργασία θα δούμε αν οι αλγόριθμοι μηχανικής μάθησης μπορούν να λάβουν υπ' όψη τους, τους αριθμούς αλλά και τις λέξεις, που διατίθενται σαν χαρακτηριστικά πίστωσης και να αποφανθούν αν κάποιος θα αποπληρώσει το δάνειό του ή εάν ο χρηματοπιστωτικός οργανισμός διατρέχει κίνδυνο ζημίας.

Abstract

The so-called credit features, that is the features based on which someone can be evaluated as to whether one is creditworthy for the loan repayment or not, conduct a very important part of the banking everyday routine since more and more people wish to obtain customer loans. Of course, now a days, the features based on which one can be evaluated for their creditworthiness are so many that consists human analysis even infeasible. In the following thesis we will see if the machine learning algorithms can take into consideration, the numbers as well as the words, that are available as credit features and decide whether the loan receiver will repay the loan or if the financial institute is under the risk of loss.

Περιεχόμενα

Ευχαριστίες 3

Περίληψη 4

Abstract..... 5

1.Εισαγωγή..... 7

2.Τα Δεδομένα..... 9

2.1 Περιγραφική Ανάλυση 11

2.2 Επεξεργασία Δεδομένων και επιλογή χαρακτηριστικών 24

3. Επεξεργασία Φυσικής Γλώσσας 38

3.1 Η Βιβλιοθήκη spaCy 38

3.2 Ο Αλγόριθμος PCA..... 41

4. Μέθοδοι μηχανικής μάθησης και η συμβολή τους στην μείωση πιστωτικού κινδύνου.....45

4.1 Ο Αλγόριθμος SMOTE (Synthetic Minority Oversampling Technique) 49

4.2 Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks)..... 50

4.3 Ο αλγόριθμος XGBoost..... 53

4.4 Λογιστική Παλινδρόμηση (Logistic Regression) 55

5. Αποτελέσματα 56

Συμπεράσματα & Μελλοντική Έρευνα 59

Βιβλιογραφία 60

1.Εισαγωγή

Σε αυτή την φουρτούνα εξελίξεων, την οποία βιώνουμε τις τελευταίες δεκαετίες, η χρηματοοικονομική διοίκηση αναγκάστηκε να ακολουθήσει αυτές τις επιχειρηματικές, οικονομικές αλλά και τεχνολογικές μεταβολές. Η αυξημένη πολυπλοκότητα που διέπει τους χώρους αυτούς καθιστά αντίστοιχα περίπλοκη και την διαδικασία λήψης χρηματοοικονομικών αποφάσεων. Στην διαδικασία αυτή εμπλέκονται χαρακτηριστικά διαφορετικής φύσης, τα οποία πολλές φορές οδηγούν σε συμπεράσματα τα οποία είναι αρκετά δύσκολο, αλλά και χρονοβόρο να ερμηνευθούν [12].

Ο πιστωτικός κίνδυνος είναι άρρηκτα συνδεδεμένος με τις καθημερινές δραστηριότητες μιας επιχείρησης / χρηματοπιστωτικού οργανισμού και ορίζεται ως ο κίνδυνος αθέτησης των υποχρεώσεων των πιστούχων της. Αυτό μπορεί να δημιουργήσει σημαντικές ζημίες και επιδείνωση των οικονομικών της και, κάτω από αρκετά ακραίες συνθήκες να υπάρξει κίνδυνος ακόμα για την συνέχιση της ύπαρξής της.

Ο πιστωτικός κίνδυνος θα μπορούσε να αναλυθεί σε τέσσερεις επιμέρους συνιστώσες.

- Κίνδυνος Πτώχευσης (Default Risk)
- Κίνδυνος Ανοίγματος (Exposure Risk)
- Κίνδυνος ανάκτησης σε περίπτωση πτώχευσης (Recovery Risk)
- Κίνδυνος Περιθωρίων (Credit Spread Risk)

Ο κίνδυνος πτώχευσης αναφέρεται στην πιθανότητα πτώχευσης των πιστούχων ενός χρηματοπιστωτικού ιδρύματος. Ο κίνδυνος ανοίγματος αναφέρεται στο συνολικό ποσό που εκτίθεται στον πιστωτικό κίνδυνο. Ο κίνδυνος ανάκτησης σε περίπτωση πτώχευσης αναφέρεται στο ποσοστό ικανοποίησης της τράπεζας από το συνολικό ποσό που είναι εκτεθειμένο σε κίνδυνο σε περίπτωση πτώχευσης του πιστούχου. Τέλος, ο κίνδυνος περιθωρίων αναφέρεται στην πιθανότητα μείωσης της αξίας μιας πιστοδότησης ως αποτέλεσμα της αύξησης πιστωτικών περιθωρίων [13].

Αφού λοιπόν είδαμε, πόσο σημαντικός είναι ο πιστωτικός κίνδυνος για τις επιχειρήσεις αλλά και τα χρηματοπιστωτικά ιδρύματα εν γένει, πρέπει να βρεθεί ένας τρόπος να μπορεί να προβλεφθεί αυτός ο κίνδυνος ώστε οι εταιρείες να μην διατρέχουν τόσο μεγάλο οικονομικό κίνδυνο.

Όπως θα δούμε παρακάτω το σύνολο των δεδομένων μας αφορά μια Peer-to-Peer εταιρία δανεισμού. Η αξιολόγηση της φερεγγυότητας των ενδιαφερόμενων αποτελεί πρόκληση στην μικροοικονομία όπου τα δάνεια, των peer-to-peer οργανισμών, δεν είναι τόσο ασφαλή [16]. Επιπρόσθετα, τα δάνεια στους P2P οργανισμούς δίνονται κάτω από συνθήκες όπου υπάρχουν αρκετά κενά στα δεδομένα, συνθήκες όπου ο οργανισμός δεν έχει αρκετές πληροφορίες για το πιστωτικό παρελθών του ενδιαφερόμενου ή ακόμα και να υπάρχουν αυτές οι πληροφορίες, ο οργανισμός μπορεί να μην ξέρει πως να κερδίσει χρήσιμες πληροφορίες από αυτές [17].

Οι παραδοσιακές τεχνικές αξιολόγησης ενός δανείου προϋποθέτουν ένα ισορροπημένο σύνολο δεδομένων. Όμως, ασύμμετρα σύνολα δεδομένων είναι πολύ πιο συνηθισμένα στις P2P εταιρείες δανεισμού. Τα προβλήματα άνισων κλάσεων δημιουργούνται όταν υπάρχουν πάρα πολλές παρατηρήσεις στη μια και πάρα πολύ λίγες παρατηρήσεις στην άλλη. Η πρόβλεψη πιστωτικού κινδύνου σε τέτοιες συνθήκες είναι δύσκολη διότι αυτή η ασυμμετρία επηρεάζει την ικανότητα του μοντέλου, να διαχωρίσει τους αξιόπιστους ενδιαφερόμενους από τους αναξιόπιστους, αφού εστιάζει περισσότερο στην κλάση πλειοψηφίας και αμελεί αρκετά την μειοψηφία [15].

Από το 1960 έως και σήμερα πολλές χρηματοπιστωτικές εταιρείες χρησιμοποιούσαν το credit scoring ώστε να αξιολογήσουν αν οι ενδιαφερόμενοι δανειολήπτες θα αποπληρώσουν στην ώρα τους ή όχι. Το credit scoring αναφέρεται σε σχεδιασμό μαθηματικών μοντέλων τα οποία μετέτρεπαν τα σχετικά χαρακτηριστικά του ενδιαφερόμενου σε μια και μοναδική τιμή. Λόγω της ραγδαίας ανάπτυξης τεχνικών μηχανικής μάθησης, πολλές μέθοδοι ταξινόμησης, αλγόριθμοι, έχουν προταθεί για να διευκολύνουν την αξιολόγηση της πιστωτικής συμπεριφοράς ενός δανειολήπτη [18].

Οι αλγόριθμοι μηχανικής μάθησης χρησιμοποιούνται τα τελευταία χρόνια με αρκετά μεγάλη επιτυχία από χρηματοπιστωτικούς οργανισμούς και επιχειρήσεις κάθε είδους με σκοπό την γρήγορη επεξεργασία δεδομένων και φυσικά την εξαγωγή συμπερασμάτων τα οποία χρήζουν ερμηνείας. Πιο συγκεκριμένα ένας αλγόριθμος μηχανικής μάθησης για να προβλέψει την πιστωτική φερεγγυότητα ενός δανειολήπτη θα πρέπει να μπορεί να τον αξιολογήσει σωστά μέσα από κάποια πιστωτικά χαρακτηριστικά.

Η παρούσα διπλωματική εργασία επικεντρώνεται σε αλγορίθμους μηχανικής μάθησης και πώς αυτοί μπορούν να βοηθήσουν χρηματοπιστωτικούς οργανισμούς, έχοντας στην διάθεσή τους δεδομένα με άνισες κλάσεις, να μειώσουν την ύπαρξη ρίσκου και κατ' επέκταση την αποφυγή κακών επενδυτικών αποφάσεων, κατηγοριοποιώντας τους δανειολήπτες σε φερέγγυους και μη.

2.Τα Δεδομένα

Τα δεδομένα που χρησιμοποιήθηκαν σε αυτή την διπλωματική εργασία λήφθηκαν από τον ιστότοπο του Kaggle [3] και αφορούν την εταιρεία χρηματοοικονομικών υπηρεσιών LendingClub. Η LendingClub, η οποία διαθέτει δάνεια online χωρίς ενδιάμεσους οικονομικούς οργανισμούς όπως είναι οι τράπεζες [15], ήταν η μεγαλύτερη peer-to-peer πλατφόρμα δανεισμού στον κόσμο. Το LendingClub επέτρεψε στους δανειολήπτες να δημιουργήσουν μη εξασφαλισμένα προσωπικά δάνεια μεταξύ $1.000 και $40.000. Η τυπική περίοδος δανείου ήταν τρία χρόνια. Οι επενδυτές μπόρεσαν να αναζητήσουν και να περιηγηθούν στις λίστες δανείων στον ιστότοπο του LendingClub και να επιλέξουν δάνεια στα οποία ήθελαν να επενδύσουν με βάση τις πληροφορίες που παρείχαν σχετικά με τον δανειολήπτη, το ποσό του δανείου, τον βαθμό δανείου και τον σκοπό του δανείου [2].

Το σύνολο αρχικά, αποτελείται από τα 28 παρακάτω χαρακτηριστικά:

- loan_amnt: Το επιθυμητό χρηματικό ποσό που έχει δηλώσει ο δανειολήπτης.
- term: Ο αριθμός των δόσεων. Οι τιμές είναι σε μήνες (36 ή 60).
- int_rate: Το επιτόκιο του δανείου.
- installment: Η μηνιαία δόση που οφείλει ο δανειολήπτης.
- grade: Η κατηγορία στην οποία έχει ενταχθεί ο δανειολήπτης από την LendingClub.
- sub_grade: Η υποκατηγορία στην οποία έχει ενταχθεί ο δανειολήπτης από την LendingClub.
- emp_title: Το επάγγελμα του δανειολήπτη.
- emp_length: Το χρονικό διάστημα το οποίο ο δανειολήπτης κάνει το συγκεκριμένο επάγγελμα.
- home_ownership: Ποια είναι η στεγαστική κατάσταση του δανειολήπτη τη στιγμή που αιτείται το δάνειο (RENT, OWN, MORTGAGE, OTHER)
- annual_inc: Το ετήσιο εισόδημα που δηλώνει ο δανειολήπτης στην αίτηση.
- verification_status: Δηλώνει αν η LendingClub επαλήθευσε το ετήσιο εισόδημα το δανειολήπτη ή όχι (verified, not verified).
- issue_d: Ο μήνας τον οποίο το δάνειο δόθηκε.
- loan_status: Τρέχουσα κατάσταση του δανείου (fully paid(1), Charged off (0).
- purpose: Ο λόγος για τον οποίο αιτήθηκε το δάνειο.
- title: Ο τίτλος του δανείου που δηλώθηκε από τον δανειολήπτη.
- zip_code: Ο ταχυδρομικός κώδικας του δανειολήπτη.
- addr_state: Η πολιτεία της Αμερικής στην οποία διαμένει ο αιτούμενος.

- dti: Ένας λόγος που υπολογίζεται από τις συνολικές εναπομείναντες δόσεις του δανειολήπτη με βάση το συνολικό δάνειο, εξαιρουμένων των

στεγαστικών δανείων και των αιτούμενων δανείων στην LendingClub, διαιρούμενος με το μηνιαίο εισόδημα που έχει δηλώσει ο δανειολήπτης.

- earliest_cr_line: Ο παλαιότερος λογαριασμός του δανειολήπτη.
- open_acc: Ο αριθμός ανοιχτών λογαριασμών του δανειολήπτη.
- pub_rec: Αν ο δανειολήπτης έχει ποινικό μητρώο.
- revol_bal: Υπόλοιπο ανακυκλούμενου ποσού.
- total_acc: Αριθμός λογαριασμών που έχει ανοιχτούς ο δανειολήπτης.
- Initial_list_status: Η αρχική κατηγοριοποίηση που έλαβε ο δανειολήπτης. Θετικές τιμές W, F.
- application_type: Δηλώνει αν η δήλωση για το δάνειο είναι ατομική ή κοινή με κάποιον άλλο.
- mort_acc: Αριθμός στεγαστικών δανείων.
- pub_rec_bankruptcies: Αριθμός φορών που έχει χρεωκοπήσει ο ενδιαφερόμενος.
- FICO: Το σκορ αξιοπιστίας του δανειολήπτη το οποίο δίνεται από το LendingClub

2.1 Περιγραφική Ανάλυση

Για την ανάγνωση των δεδομένων χρησιμοποιήθηκε το αρχείο '*loan_data.csv*' ενώ, η επεξεργασία και η ανάλυσή τους έγινε σε Python και σε περιβάλλον Jupyter Notebook με τη χρήση των παρακάτω βιβλιοθηκών:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#needed for plotting
%matplotlib inline
```

Αφού πρώτα φορτώσουμε τα δεδομένα, η πρώτη κίνηση είναι να εξερευνήσουμε τον τύπο της κάθε μεταβλητής.

```
data = pd.read_csv('loan_data.csv')
```

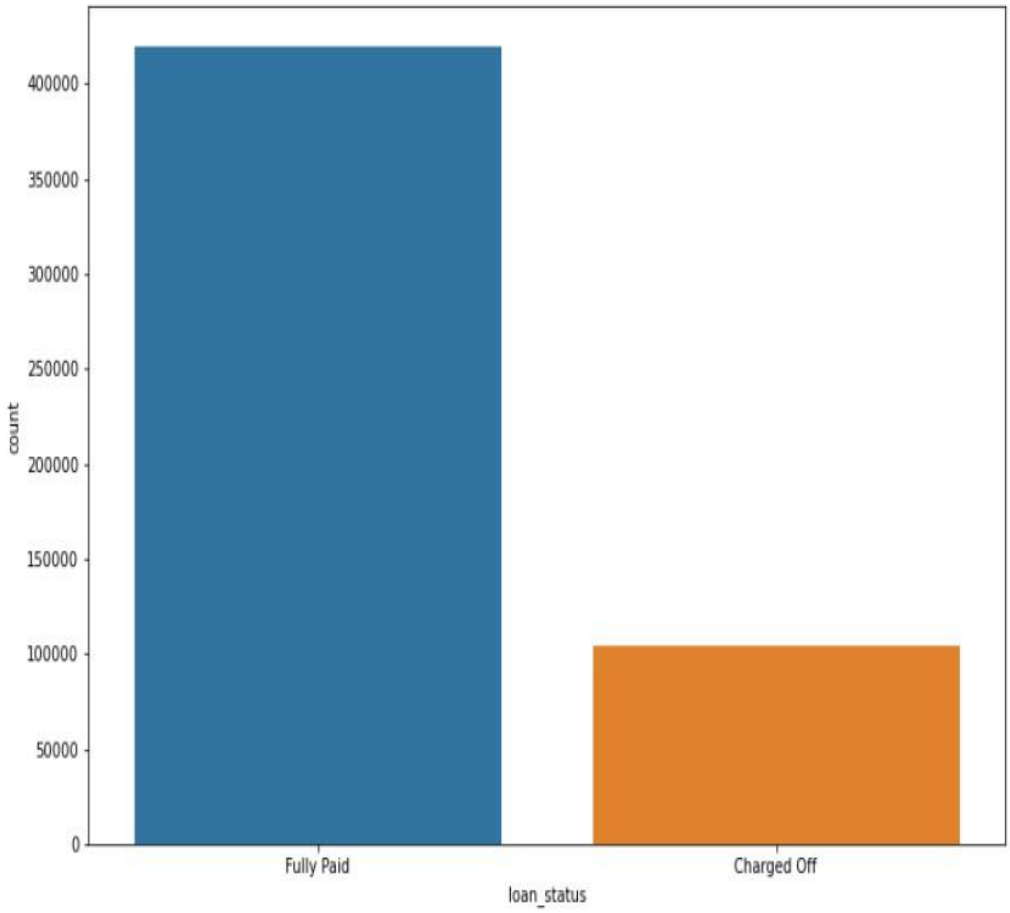
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 524288 entries, 0 to 524287
Data columns (total 29 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   loan_amnt             524288 non-null  int64
 1   term                  524288 non-null  object
 2   int_rate              524288 non-null  float64
 3   installment           524288 non-null  float64
 4   grade                 524288 non-null  object
 5   sub_grade             524288 non-null  object
 6   emp_title             490798 non-null  object
 7   emp_length            492965 non-null  object
 8   home_ownership        524288 non-null  object
 9   annual_inc            524288 non-null  float64
 10  verification_status   524288 non-null  object
 11  issue_d               524288 non-null  object
 12  loan_status           524288 non-null  object
 13  purpose               524288 non-null  object
 14  title                 517562 non-null  object
 15  zip_code              524287 non-null  object
 16  addr_state            524288 non-null  object
 17  dti                   524143 non-null  float64
 18  earliest_cr_line      524288 non-null  object
 19  open_acc              524288 non-null  int64
 20  pub_rec               524288 non-null  int64
 21  revol_bal             524288 non-null  int64
 22  revol_util            523973 non-null  float64
 23  total_acc             524288 non-null  int64
 24  initial_list_status   524288 non-null  object
 25  application_type      524288 non-null  object
 26  mort_acc              504327 non-null  float64
 27  pub_rec_bankruptcies  523922 non-null  float64
 28  FICO Score            524288 non-null  float64
dtypes: float64(8), int64(5), object(16)
memory usage: 116.0+ MB
```

Μιας και ο στόχος αυτής της μεταπτυχιακής διατριβής είναι να κατηγοριοποιήσουμε τους δανειολήπτες σε αυτούς που πιθανόν θα αποπληρώσουν το δάνειο και σε αυτούς που πιθανόν όχι, η πρώτη ερώτηση που έχουμε να απαντήσουμε είναι πόσες παρατηρήσεις πέφτουν σε κάθε κατηγορία.

```
#create a countplot for the loan_status feature
plt.figure(figsize=(12,8))
sns.countplot(x = 'loan_status', data = data)
```

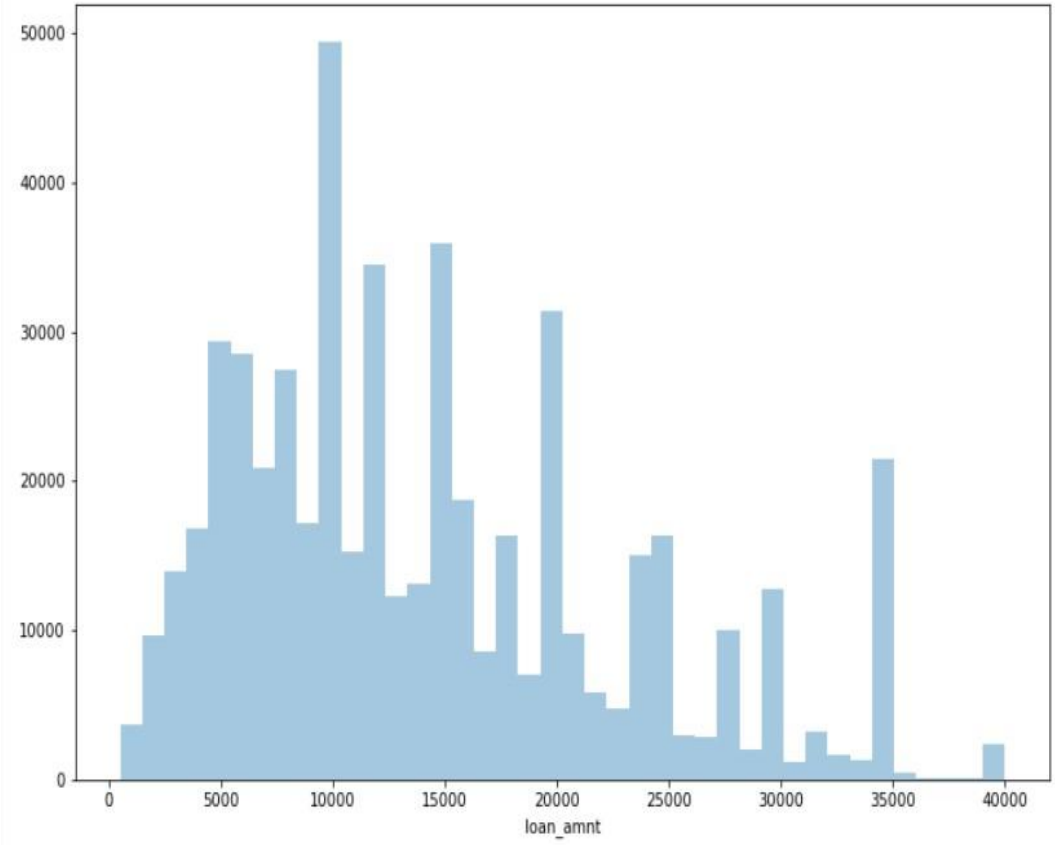
```
<AxesSubplot:xlabel='loan_status', ylabel='count'>
```



Μετά την παραπάνω απεικόνιση των κατηγοριών είναι φυσικό να θέλουμε να ελέγξουμε την κατανομή των ποσών των δανείων.

```
#Create a histogram of the loan_amnt column
plt.figure(figsize=(12,8))
sns.distplot(data['loan_amnt'], kde = False, bins = 40)
```

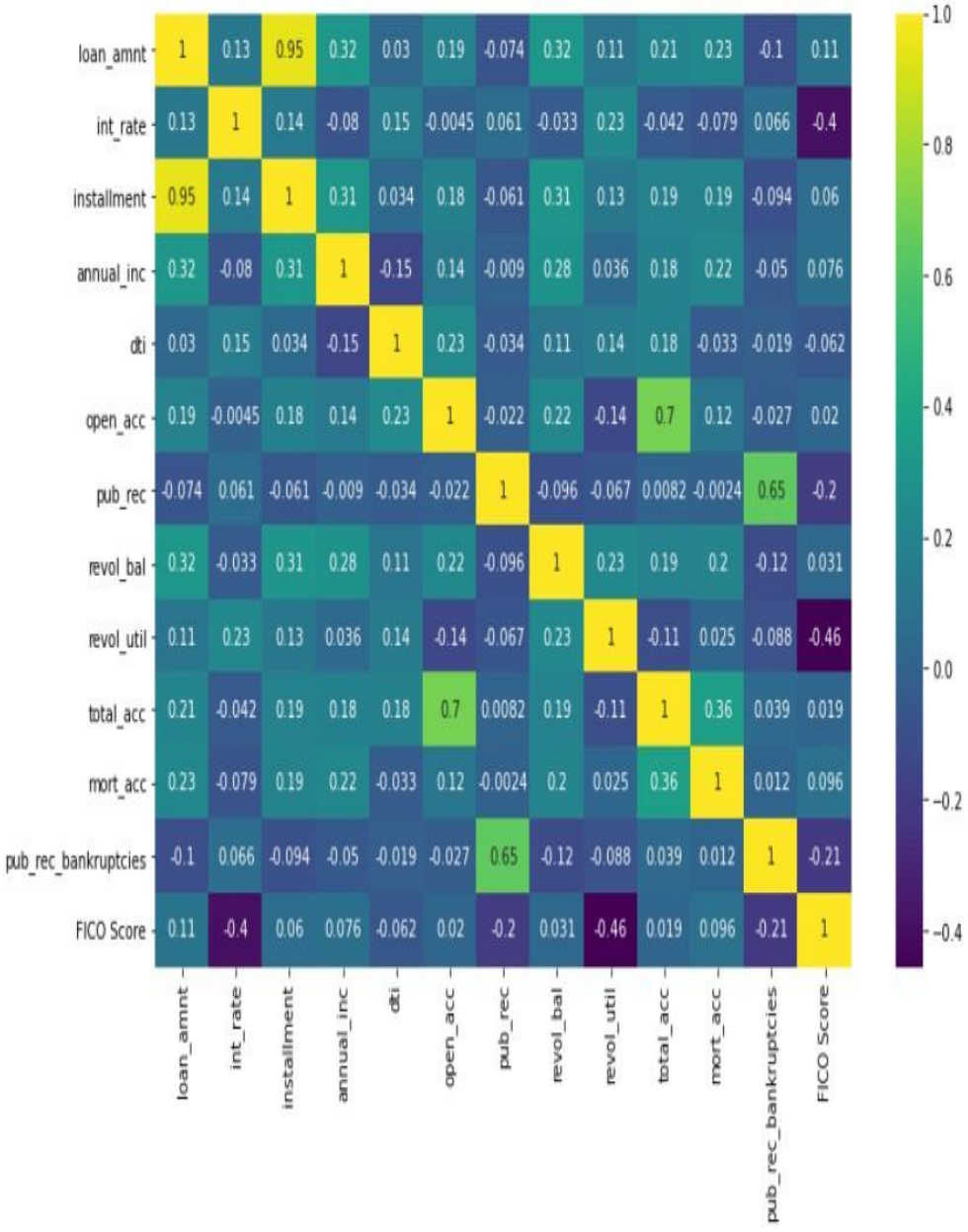
```
<AxesSubplot:xlabel='loan_amnt'>
```



Παρακάτω βλέπουμε ένα διάγραμμα συσχέτισης μεταξύ όλων των αριθμητικών μεταβλητών. Η συσχέτιση δεν έχει μονάδες και παίρνει τιμές μεταξύ -1 (τέλεια αντισυσχέτιση) και 1 (τέλεια συσχέτιση).

```
In [11]: #explore correlation between the continuous feature variables
         plt.figure(figsize=(12,8))
         sns.heatmap(data.corr(), annot = True, cmap="viridis")
```

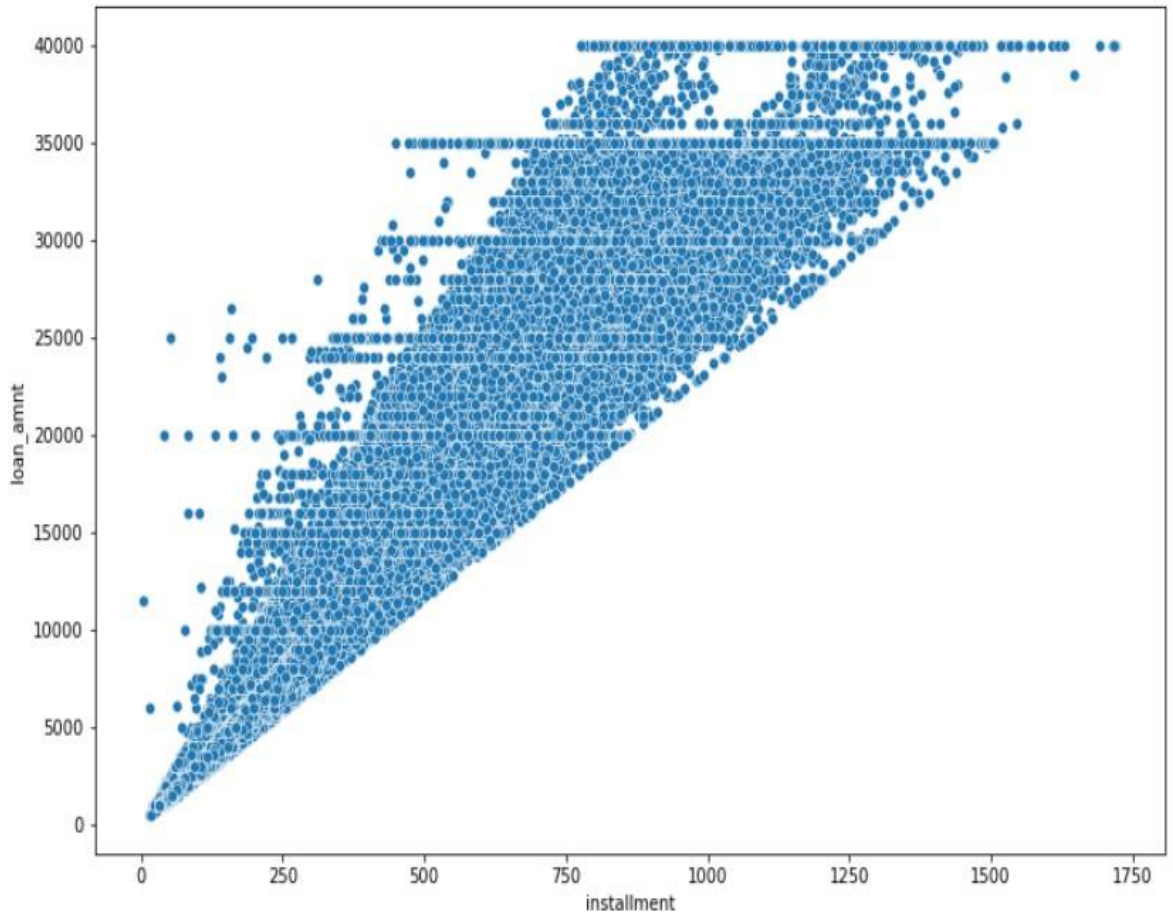
```
Out[11]: <AxesSubplot:>
```



Προφανώς κάθε μεταβλητή έχει τέλεια συσχέτιση με τον εαυτό της, αλλά πέρα από αυτές τις τέλειες συσχετίσεις, μπορούμε εύκολα να διακρίνουμε αρκετά υψηλές συσχετίσεις μεταξύ άλλων μεταβλητών όπως του installment και του loan_amnt.

```
# we can also clearly see some high level correlation in our dataset
#lets start with 'installment' and 'loan_amount'

plt.figure(figsize=(12,8))
sns.scatterplot(x = 'installment', y = 'loan_amnt', data = data)
```

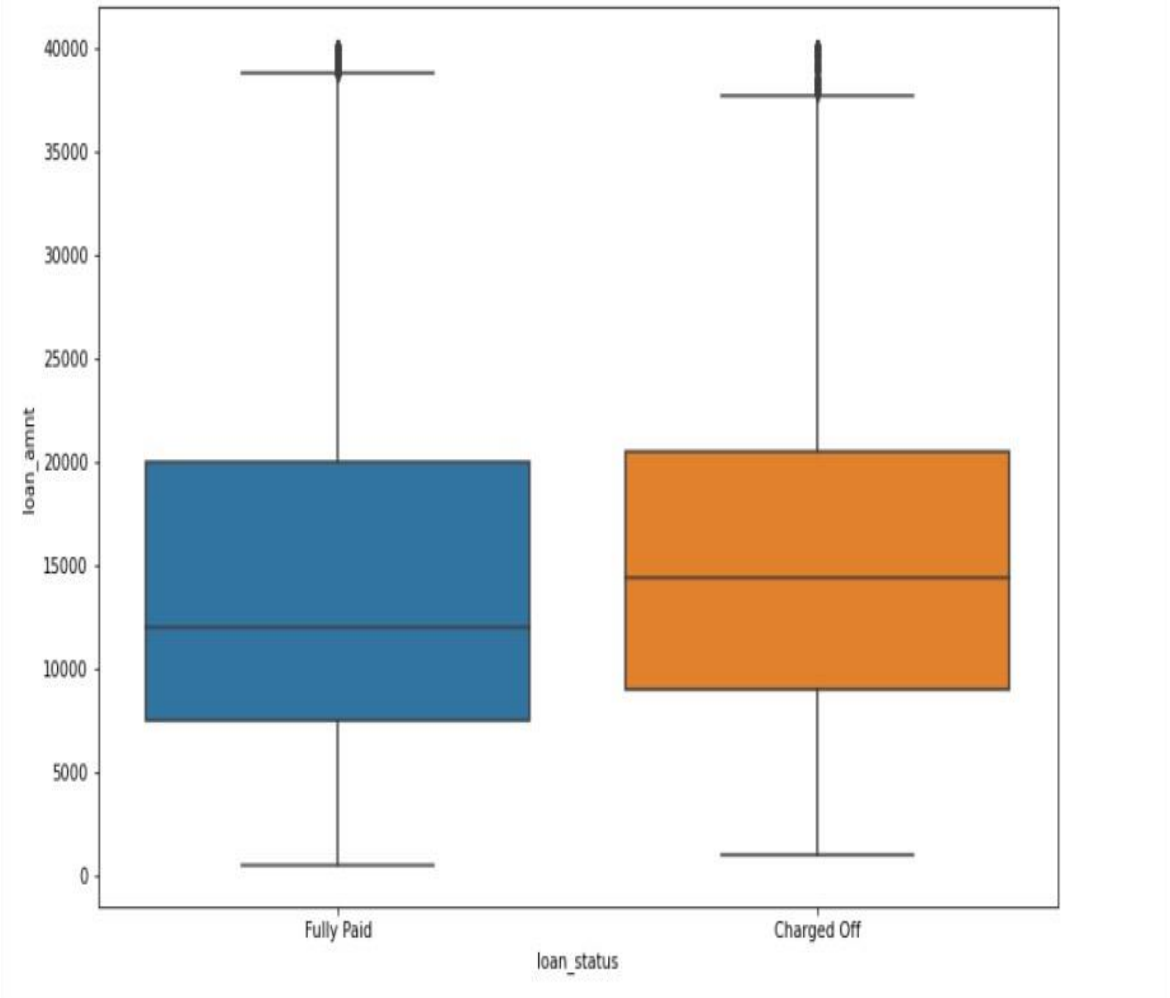
```
<AxesSubplot:xlabel='installment', ylabel='loan_amnt'>
```



Μια ακόμα πολύ ενδιαφέρουσα διερεύνηση θα είναι αυτή της σχέσης μεταξύ loan_amnt και loan_status. Παρακάτω μπορούμε εύκολα να διακρίνουμε ότι οι άνθρωποι που δεν αποπληρώνουν το δάνειό τους φαίνεται να έχουν κατά μέσο όρο λίγο πιο υψηλά δάνεια από εκείνους που αποπληρώνουν.

```
#relationship between the loan_status and the loan_amnt
plt.figure(figsize=(12,8))
sns.boxplot(x = 'loan_status', y = 'loan_amnt', data = data)
```

```
<AxesSubplot:xlabel='loan_status', ylabel='loan_amnt'>
```




```
data.groupby('loan_status')['loan_amnt'].describe()
```

loan_status	count	mean	std	min	25%	50%	75%	max
Charged Off	104445.0	15601.324621	8763.917673	1000.0	9000.0	14400.0	20475.0	40000.0
Fully Paid	419843.0	14205.103217	8698.074117	500.0	7500.0	12000.0	20000.0	40000.0

Όπως είδαμε παραπάνω υπάρχουν κάποιες κατηγοριοποιήσεις και κάποιοι βαθμοί οι οποίοι δίνονται από την LendingClub στους δανειολήπτες. Θα δούμε παρακάτω τις κατανομές αυτών των κατηγοριοποιήσεων αλλά και πώς σχετίζονται με το αν κάποιος πληρώνει ή όχι το εκάστοτε δάνειο.

```
#exploring the Grade and SubGrade attributes
data['grade'].unique()
```

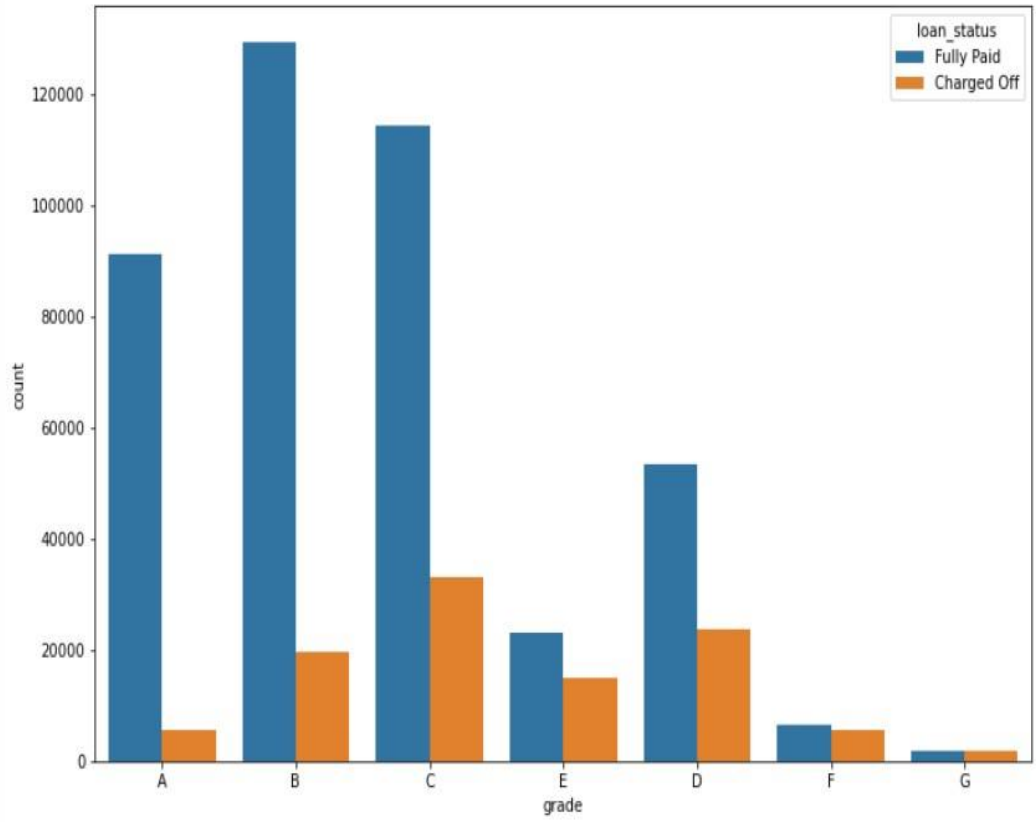
```
array(['A', 'B', 'C', 'E', 'D', 'F', 'G'], dtype=object)
```

```
data['sub_grade'].unique()
```

```
array(['A2', 'B4', 'C3', 'B2', 'C5', 'C1', 'A4', 'C2', 'E5', 'D4', 'F2',
       'B5', 'B3', 'D2', 'B1', 'C4', 'A1', 'A3', 'A5', 'D5', 'D3', 'E1',
       'D1', 'E2', 'G5', 'E3', 'F3', 'F1', 'E4', 'F5', 'G3', 'G2', 'F4',
       'G1', 'G4'], dtype=object)
```

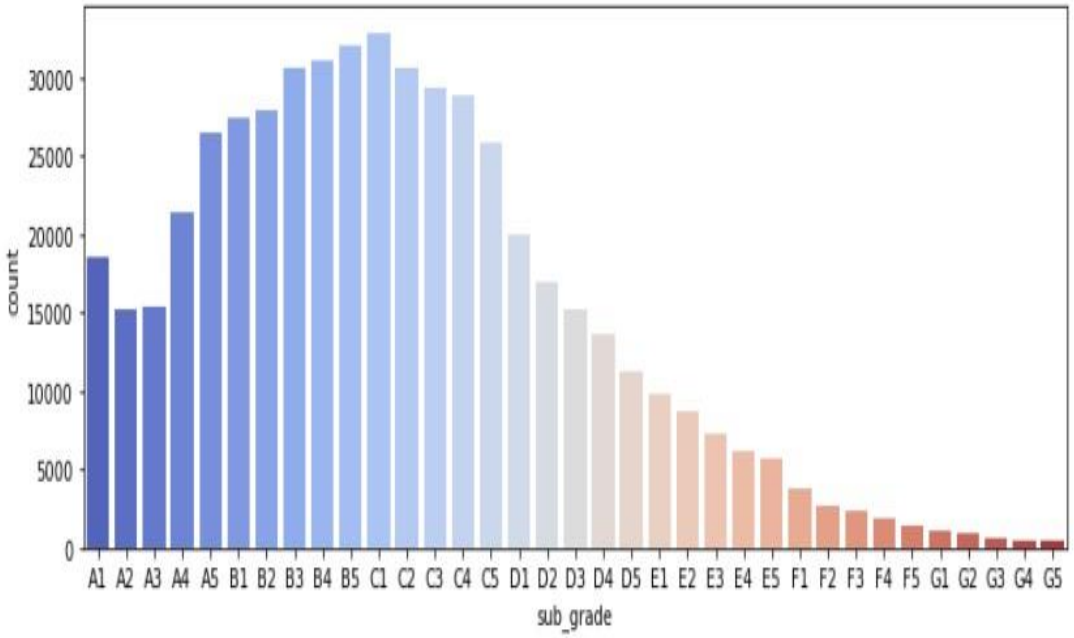
```
#Creating a count plot per grade
plt.figure(figsize=(12,8))
sns.countplot(x = 'grade', hue = 'loan_status',  data = data)
```

```
<AxesSubplot:xlabel='grade', ylabel='count'>
```



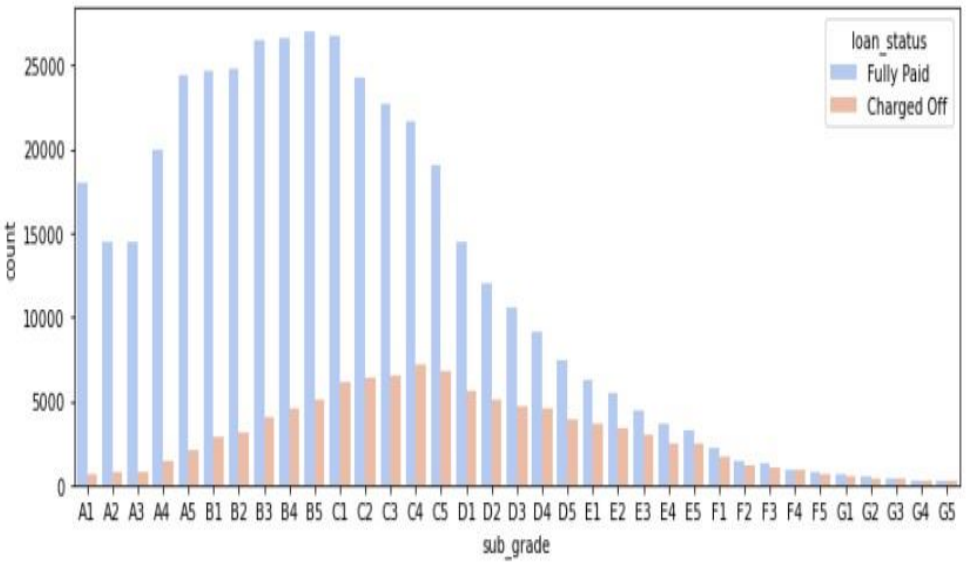
```
#Creating a count plot per Sub_grade
plt.figure(figsize=(12,4))
sns.countplot(x = 'sub_grade', data = data, order = sorted(data['sub_grade'].unique()), palette='coolwarm')
```

```
<AxesSubplot:xlabel='sub_grade', ylabel='count'>
```




```
plt.figure(figsize=(12,4))
sns.countplot(x = 'sub_grade', hue = 'loan_status',data = data, order = sorted(data['sub_grade'].unique()), palette='coolwarm')
```

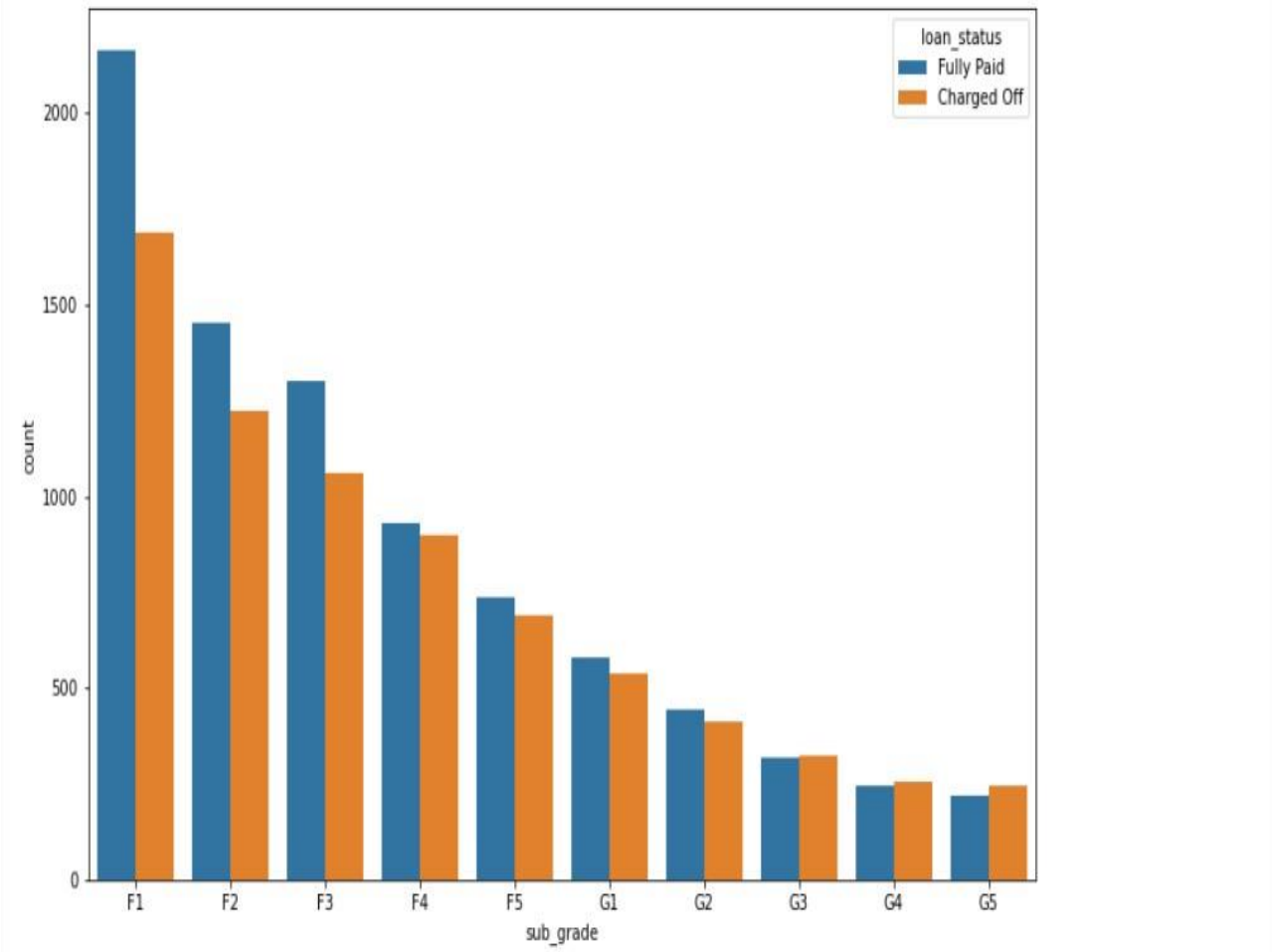
```
<AxesSubplot:xlabel='sub_grade', ylabel='count'>
```



Από το παραπάνω διάγραμμα μπορούμε εύκολα να δούμε ότι τα δάνεια στις κατηγορίες F1 έως G5 δεν φαίνεται να αποπληρώνονται συχνά. Για να το διακρίνουμε λίγο καλύτερα, παρακάτω παρατίθεται μια εικόνα η οποία απεικονίζει αυτές τις κατηγορίες.

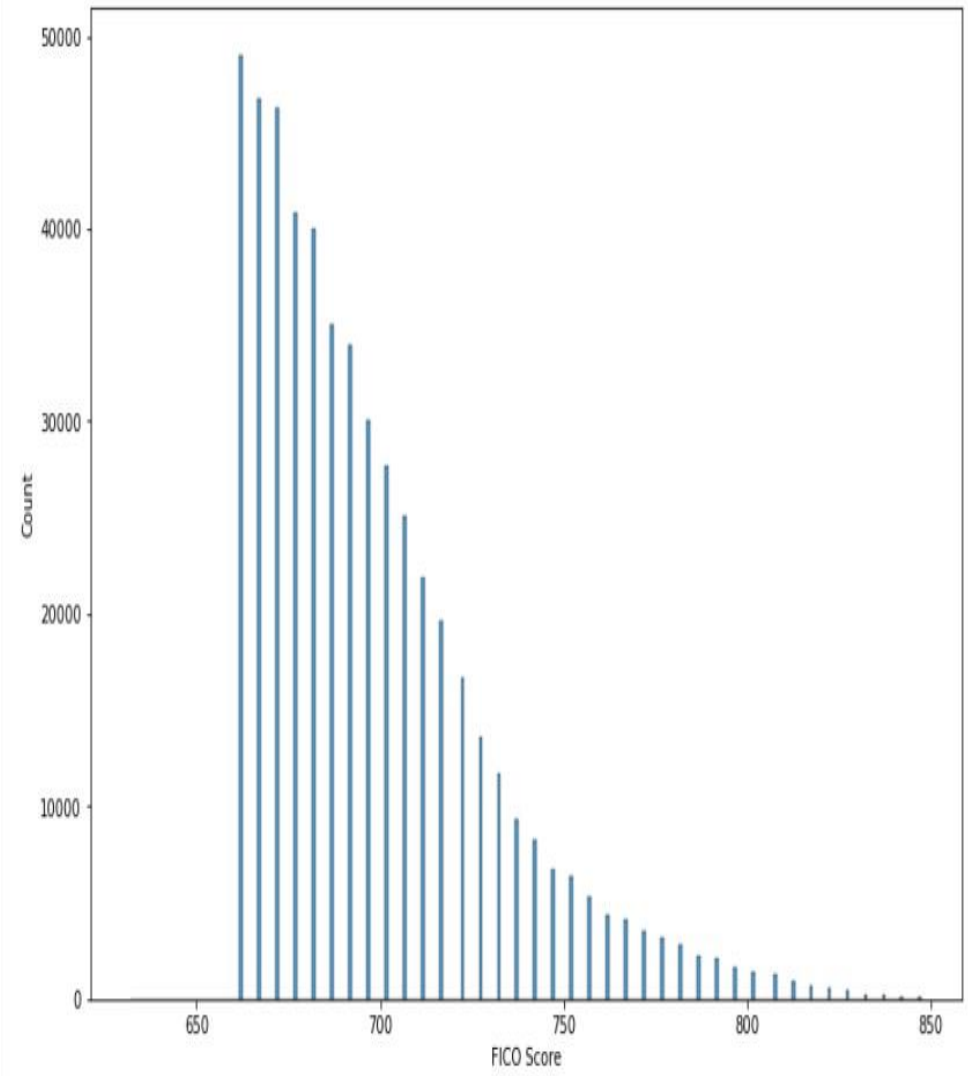
```
#we can clearly see the F and G subgrades don't get paid back that often
f_g = data[(data['grade'] == 'F') | (data['grade'] == 'G')].sort_values(by = 'sub_grade')
plt.figure(figsize=(12,8))
sns.countplot(x = 'sub_grade', hue = 'loan_status' , data = f_g, hue_order = ['Fully Paid', 'Charged Off'])
```

```
<AxesSubplot:xlabel='sub_grade', ylabel='count'>
```

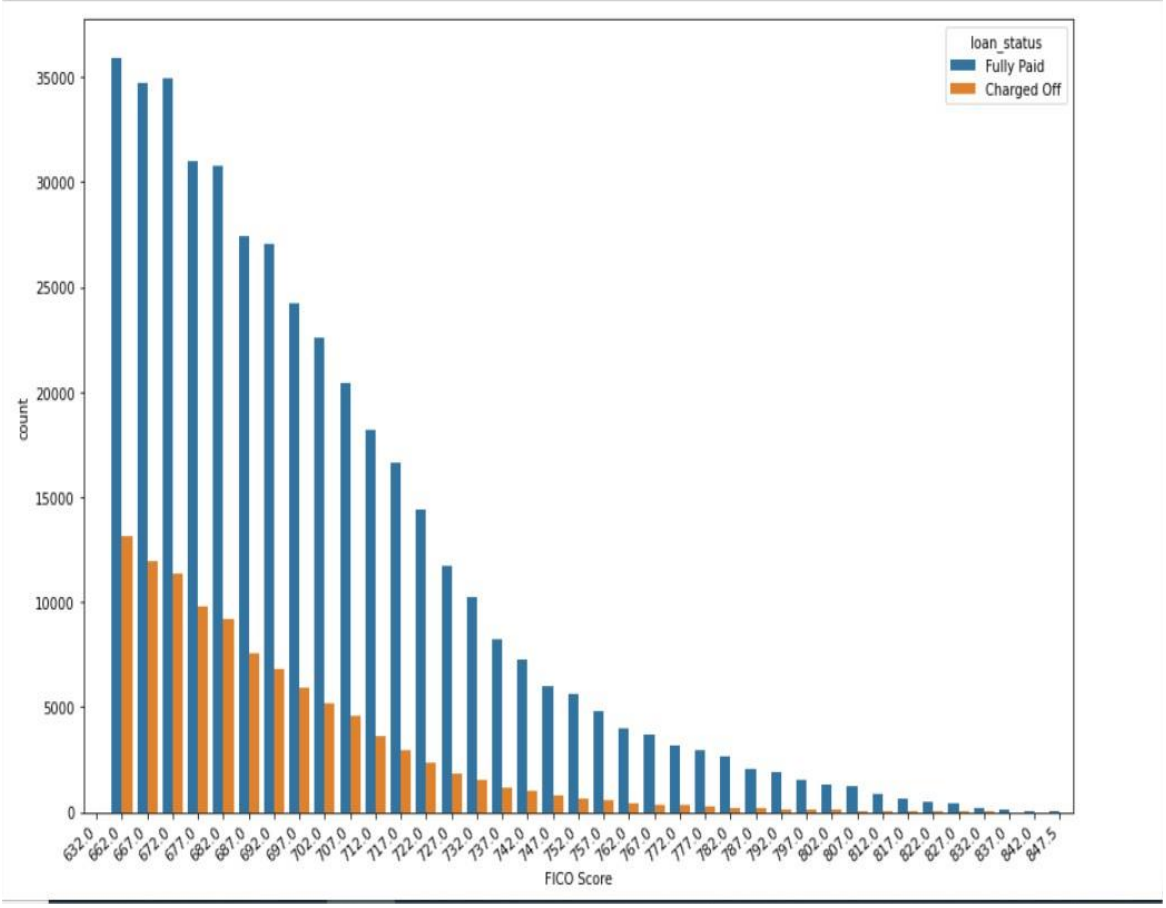


```
plt.figure(figsize=(12,8))
sns.histplot(x = 'FICO Score', data = data)
```

```
<AxesSubplot:xlabel='FICO Score', ylabel='Count'>
```



```
plt.figure(figsize=(12,8))
ax = sns.countplot(x='FICO Score', hue = 'loan_status', data=data)
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
plt.tight_layout()
plt.show()
```



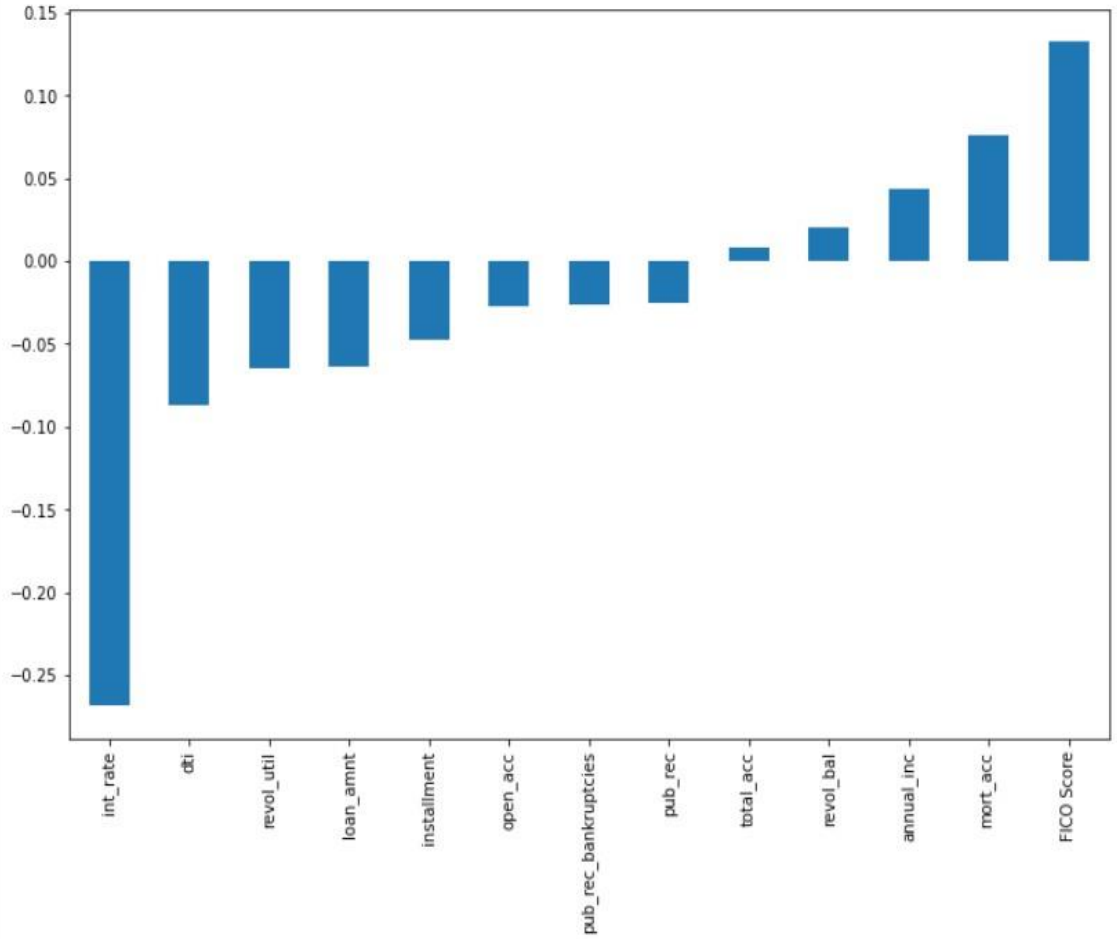
Μπορούμε να συμπεράνουμε από το παραπάνω διάγραμμα ότι τα δάνεια αποπληρώνονται συνήθως ασχέτως με το Fico Score που απονέμεται στους δανειολήπτες. Οπότε, ίσως θα μπορούσαμε να πούμε ότι το Fico δεν είναι ένα μέτρο που όντως μπορεί να διασφαλίσει την ελαχιστοποίηση του πιστωτικού κινδύνου για τους χρηματοοικονομικούς φορείς.

Τέλος στην περιγραφική ανάλυση θα δώσει ένα διάγραμμα συσχέτισης μεταξύ του loan_status και των υπόλοιπων αριθμητικών μεταβλητών. Προτού όμως γίνει αυτό θα πρέπει να δημιουργήσουμε μια νέα στήλη η οποία θα μετατρέψει το loan_status από περιγραφική μεταβλητή σε αριθμητική (1: Το δάνειο αποπληρώθηκε, 0: Το δάνειο δεν αποπληρώθηκε).

```
#Creating a new column that we contain a mapping of loan_status to 1  and 0
data['loan_repaid'] = data['loan_status'].map({'Fully Paid':1, 'Charged Off':0})
```

```
#Creating a bar plot showing the correlation between loan_repaid and the rest of the features
plt.figure(figsize=(12,8))
data.corr()['loan_repaid'].sort_values().drop('loan_repaid').plot(kind='bar')
```

```
<AxesSubplot:>
```



2.2 Επεξεργασία Δεδομένων και επιλογή χαρακτηριστικών

Στη προηγούμενη ενότητα γνωρίσαμε τις μεταβλητές μας. Όπως είδαμε κάποιες από αυτές ήταν αριθμητικές, κάποιες άλλες ήταν κατηγορικές και προσπαθήσαμε να μελετήσουμε κάποιες από τις κατανομές τους, τις συσχετίσεις μεταξύ τους και πολλά άλλα. Όμως, αν ρίξουμε μια πιο προσεκτική ματιά στα δεδομένα μας θα παρατηρήσουμε πολύ εύκολα ότι υπάρχουν αρκετές ελλείψεις στις μεταβλητές.

Εδώ βλέπουμε το ποσοστό κενών τιμών σε κάθε στήλη:

```
data.isnull().sum()/len(data)*100
```

```
loan_amnt               0.000000
term                    0.000000
int_rate                0.000000
installment             0.000000
grade                   0.000000
sub_grade               0.000000
emp_title               6.387711
emp_length              5.974388
home_ownership          0.000000
annual_inc              0.000000
verification_status     0.000000
issue_d                 0.000000
loan_status             0.000000
purpose                 0.000000
title                   1.282883
zip_code                0.000191
addr_state              0.000000
dti                     0.027657
earliest_cr_line        0.000000
open_acc                0.000000
pub_rec                 0.000000
revol_bal               0.000000
revol_util              0.060081
total_acc               0.000000
initial_list_status     0.000000
application_type        0.000000
mort_acc                3.807259
pub_rec_bankruptcies    0.069809
FICO Score              0.000000
loan_repaid             0.000000
dtype: float64
```

Αν μελετήσουμε την μεταβλητή η οποία αφορά το επάγγελμα του κάθε δανειολήπτη θα δούμε εύκολα ότι υπάρχουν 152.568 διαφορετικά επαγγέλματα με το επάγγελμα του δασκάλου να έρχεται πρώτο στις αιτήσεις.

```
print(data['emp_title'].nunique())
print("-----------------------------------------------------------")
print(data['emp_title'].value_counts())
```

```
152568
-----------------------------------------------------------
Teacher                    9176
Manager                    8344
Owner                      4390
Registered Nurse           3850
RN                         3811
                           ...
Telephonics Corp.             1
MAINTENANCE SHOP LEADER       1
Controller/Accountant         1
expidter                      1
civilen Instructor            1
Name: emp_title, Length: 152568, dtype: int64
```

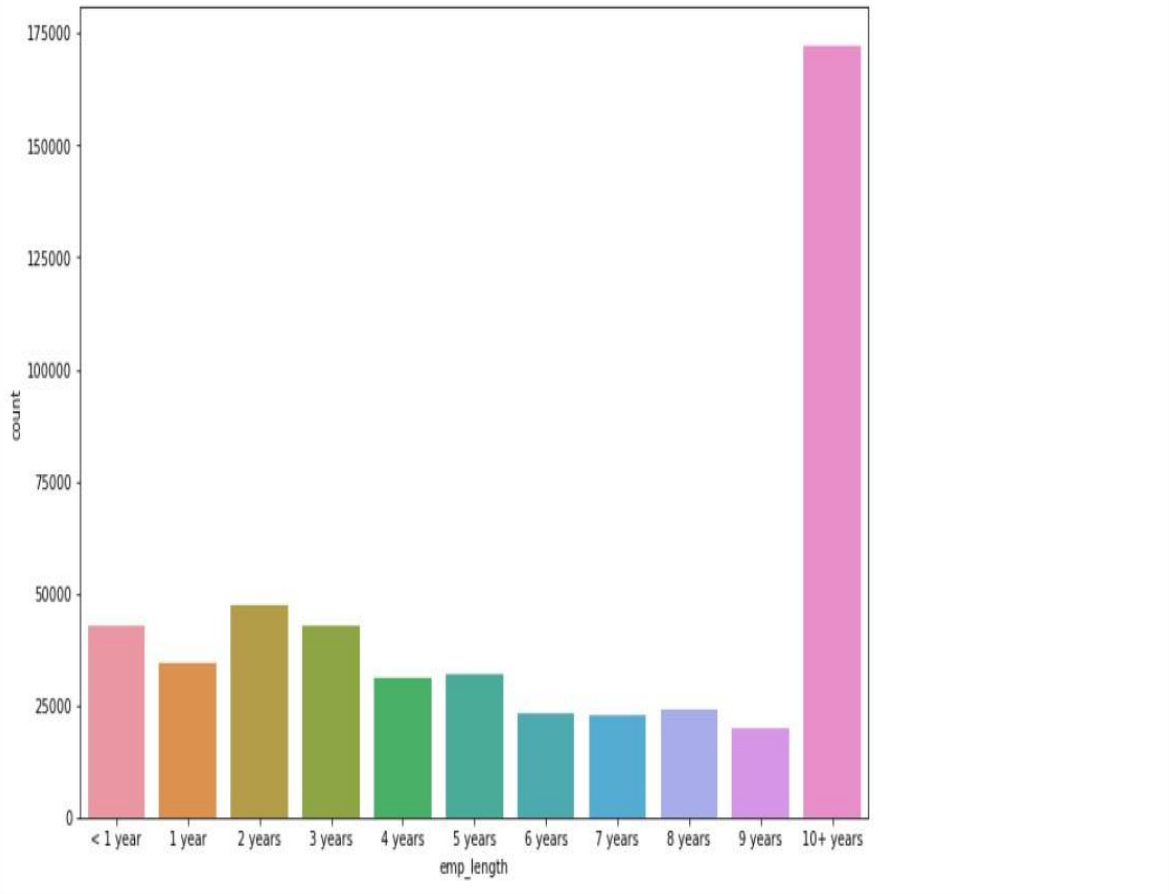
Βλέπουμε ότι τα επαγγέλματα είναι πάρα πολλά ώστε να προσπαθήσουμε να τα μετατρέψουμε σε one hot encoding και υπάρχουν πάρα πολλά αρχικά, συντομογραφίες, κομμένες λέξεις αλλά και σύμβολα ενδιάμεσα στις λέξεις ώστε να μπορέσουμε να εφαρμόσουμε κάποιον αλγόριθμο φυσικής επεξεργασίας γλώσσας (NLP) οπότε την συγκεκριμένη στήλη θα την αφαιρέσουμε.

```
data = data.drop('emp_title', axis = 1)
```

Επόμενη μεταβλητή που θα μελετήσουμε αφορά την χρονική διάρκεια την οποία δουλεύει ο εκάστοτε δανειολήπτης.

```
#Creating a count plot for the emp_length feature
year_order = ['< 1 year', '1 year','2 years','3 years','4 years','5 years','6 years','7 years', '8 years', '9 years','10+ years']
plt.figure(figsize=(12,8))
sns.countplot(x = 'emp_length', data = data, order = year_order)
```

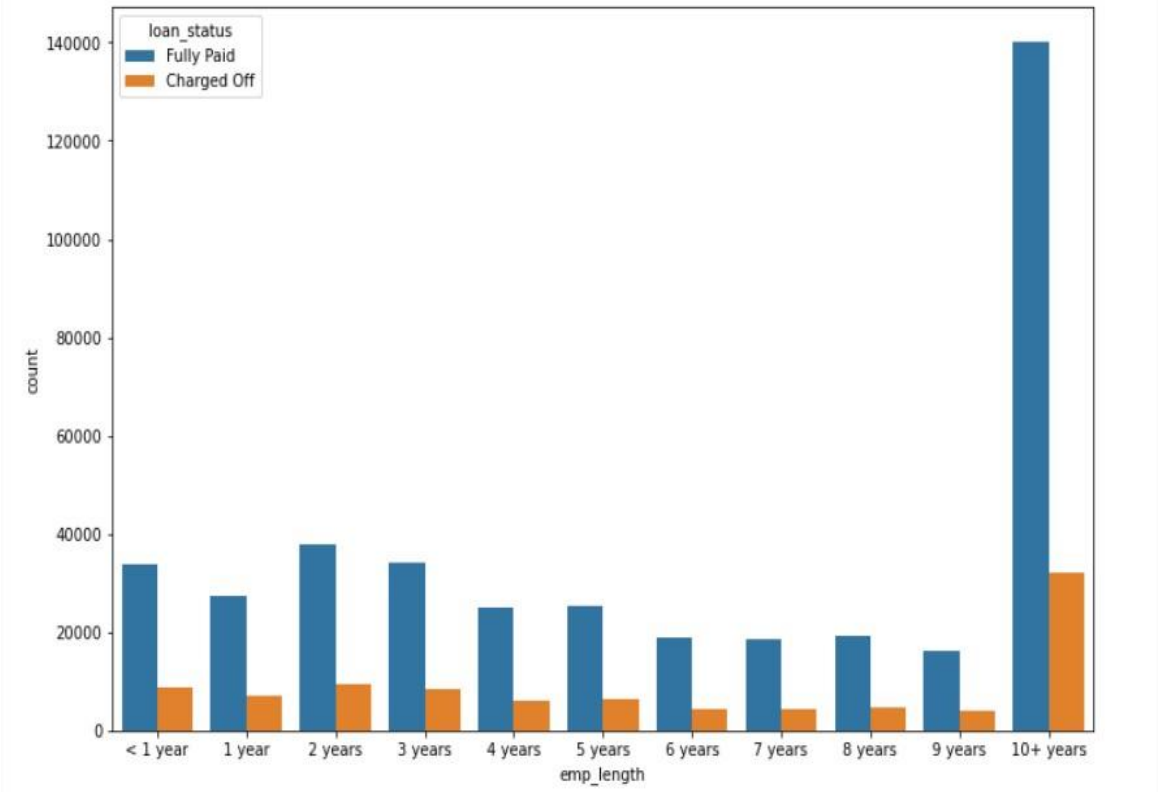
```
<AxesSubplot:xlabel='emp_length', ylabel='count'>
```



Παρατηρούμε ότι οι περισσότεροι από τους δανειολήπτες της εταιρείας δουλεύουν για πάνω από 10 χρόνια. Ας μελετήσουμε τώρα, στα πόσα χρόνια εργασίας φαίνεται να αποπληρώνουν ευκολότερα το δάνειο.

```
plt.figure(figsize=(12,8))
sns.countplot(x='emp_length', hue = 'loan_status',data = data, order = year_order)
```

```
<AxesSubplot:xlabel='emp_length', ylabel='count'>
```



Με το παραπάνω ραβδόγραμμα δεν μπορούμε να καταλάβουμε αν υπάρχει κάποια ισχυρή σχέση μεταξύ των χρόνων εργασίας και της αποπληρωμής του δανείου. Αυτό που πρέπει να μελετήσουμε είναι τι ποσοστό αποπληρώνει και τι όχι σε κάθε κατηγορία.

```
charged_off_count = data[data['loan_repaid'] == 0].groupby('emp_length')['loan_repaid'].count()
fully_paid_count = data[data['loan_repaid'] == 1].groupby('emp_length')['loan_repaid'].count()
```

```
perc_co = charged_off_count/(charged_off_count + fully_paid_count)
```

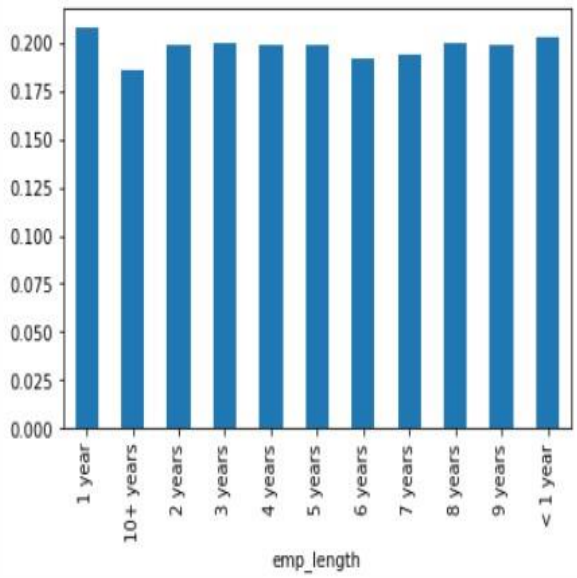
```
perc_co
```

```
emp_length
1 year       0.207976
10+ years    0.186267
2 years      0.199182
3 years      0.199817
4 years      0.198832
5 years      0.199227
6 years      0.191728
7 years      0.194271
8 years      0.199735
9 years      0.198862
< 1 year     0.202679
Name: loan_repaid, dtype: float64
```

Φαίνεται ότι τα ποσοστά αποπληρωμής κάθε κατηγορίας να είναι πολύ κοντά. Για να γίνει πιο ξεκάθαρο αυτό θα το οπτικοποιήσουμε μέσω ενός ραβδογράμματος.

```
perc_co.plot(kind='bar')
#Charge off rates are extremely similar across all employment lengths so we can drop the column
```

```
<AxesSubplot:xlabel='emp_length'>
```



Οπότε είναι ξεκάθαρο ότι μπορούμε να διαγράψουμε την μεταβλητή που σχετίζεται με τα χρόνια εργασίας των ενδιαφερόμενων .

```
data = data.drop('emp_length', axis =1)
```

Η επόμενη μεταβλητή που μπορούμε να μελετήσουμε είναι αυτή του ταχυδρομικού κώδικα (zip code). Απλά, βλέποντας τις τιμές της μεταβλητής αυτής καταλαβαίνουμε ότι πολύ δύσκολα θα πάρουμε κάποια χρήσιμη πληροφορία οπότε θα την διαγράψουμε.

```
#Clearly we cannot get any inference from the zip_code since it is given in this format
data['zip_code'].unique()
```

```
array(['720xx', '300xx', '104xx', '891xx', '940xx', '750xx', '180xx',
       '087xx', '609xx', '115xx', '070xx', '112xx', '100xx', '208xx',
       '330xx', '864xx', '890xx', '913xx', '432xx', '189xx', '600xx',
       '972xx', '331xx', '452xx', '738xx', '480xx', '956xx', '980xx',
       '900xx', '273xx', '029xx', '303xx', '968xx', '190xx', '816xx',
       '212xx', '997xx', '853xx', '201xx', '125xx', '761xx', '800xx',
       '088xx', '770xx', '939xx', '495xx', '410xx', '809xx', '850xx',
       '906xx', '941xx', '481xx', '930xx', '207xx', '178xx', '350xx',
       '320xx', '471xx', '453xx', '782xx', '735xx', '346xx', '060xx',
       '234xx', '946xx', '441xx', '322xx', '945xx', '450xx', '902xx',
       '916xx', '951xx', '217xx', '751xx', '254xx', '146xx', '111xx',
       '960xx', '109xx', '840xx', '559xx', '961xx', '236xx', '318xx',
       '145xx', '210xx', '627xx', '917xx', '342xx', '019xx', '064xx',
       '856xx', '292xx', '014xx', '955xx', '925xx', '605xx', '923xx',
       '021xx', '314xx', '787xx', '774xx', '641xx', '934xx', '458xx',
       '089xx', '953xx', '996xx', '114xx', '277xx', '431xx', '557xx',
       '970xx', '871xx', '327xx', '275xx', '719xx', '113xx', '302xx',
       '209xx', '015xx', '765xx', '286xx', '801xx', '983xx', '534xx',
       '852xx', '773xx', '775xx', '633xx', '606xx', '760xx', '298xx',
```

```
#Therefore we drop the column
data = data.drop('zip_code', axis = 1)
```

Επόμενη μεταβλητή στην σειρά είναι αυτή του τίτλου του δανείου. Φαίνεται ότι μερικοί από τους δανειολήπτες δεν δηλώνουν τον τίτλο του δανείου. Όμως, παρατηρώντας τις μεταβλητές 'title', που αφορά τον τίτλο του δανείου, και την μεταβλητή 'purpose', που αφορά τον λόγο ζήτησης του δανείου, παρατηρούμε ότι η πληροφορία που παίρνουμε είναι η ίδια και δεδομένου ότι η μεταβλητή 'title' έχει κενές τιμές μπορούμε να την αφαιρέσουμε.

```
print(data['title'].head(15))
```

```
0                   Vacation
1         Debt consolidation
2         Debt consolidation
3    Credit card refinancing
4                      Other
5         Debt consolidation
6         Debt consolidation
7         Debt consolidation
8         Debt consolidation
9             Major purchase
10        Debt consolidation
11        Debt consolidation
12   Credit card refinancing
13        Debt consolidation
14        Debt consolidation
Name: title, dtype: object
```

```
print(data['purpose'].head(15))
```

```
0               vacation
1     debt_consolidation
2     debt_consolidation
3            credit_card
4                  other
5     debt_consolidation
6     debt_consolidation
7     debt_consolidation
8     debt_consolidation
9         major_purchase
10    debt_consolidation
11    debt_consolidation
12           credit_card
13    debt_consolidation
14    debt_consolidation
Name: purpose, dtype: object
```

```
#Given the above two columns we can see that the title column is just the same as the purpose column
#So we can drop the 'tilte' column
data = data.drop('title', axis=1)
```

Σειρά έχει η μεταβλητή η οποία αφορά των αριθμό στεγαστικών δανείων, ‘mort_acc’, η οποία φαίνεται να έχει αρκετές κενές τιμές.

```
data.isnull().sum()
```

```
loan_amnt                 0
term                      0
int_rate                  0
installment               0
grade                     0
sub_grade                 0
home_ownership            0
annual_inc                0
verification_status       0
issue_d                   0
loan_status               0
purpose                   0
addr_state                0
dti                     145
earliest_cr_line          0
open_acc                  0
pub_rec                   0
revol_bal                 0
revol_util              315
total_acc                 0
initial_list_status       0
application_type          0
mort_acc              19961
pub_rec_bankruptcies    366
FICO Score                0
loan_repaid               0
dtype: int64
```

```
#number of mortgage accounts
data['mort_acc'].value_counts()
```

```
0.0     201925
1.0      87262
2.0      74169
3.0      55081
4.0      37404
5.0      22583
6.0      12720
7.0       6465
8.0       3200
9.0       1596
10.0       795
11.0       459
12.0       245
13.0       121
14.0       102
15.0        60
16.0        40
17.0        26
18.0        16
19.0        10
20.0         9
24.0         8
22.0         7
27.0         6
25.0         3
23.0         3
21.0         2
26.0         2
34.0         2
37.0         2
36.0         1
28.0         1
35.0         1
32.0         1
Name: mort_acc, dtype: int64
```

```
print(data.corr()['mort_acc'].sort_values())
```

```
int_rate               -0.078817
dti                    -0.033071
pub_rec                -0.002435
pub_rec_bankruptcies    0.012132
revol_util              0.025265
loan_repaid             0.075553
FICO Score              0.095997
open_acc                0.118765
installment             0.194852
revol_bal               0.200657
annual_inc              0.217583
loan_amnt               0.226282
total_acc               0.362744
mort_acc                1.000000
Name: mort_acc, dtype: float64
```

Η συγκεκριμένη μεταβλητή φαίνεται να είναι αρκετά σημαντική και παρουσιάζει τη μεγαλύτερη συσχέτιση με τη μεταβλητή που αφορά τον συνολικό αριθμό λογαριασμών που έχει ανοιχτούς ο ενδιαφερόμενος, οπότε θα επιχειρήσουμε να γεμίσουμε της κενές τιμές με την μέση τιμή των στεγαστικών λογαριασμών ανά τον συνολικό αριθμό λογαριασμών.

```
total_account_mean = data.groupby('total_acc')['mort_acc'].mean()
print(total_account_mean.head())
```

```
total_acc
2    0.000000
3    0.030387
4    0.081081
5    0.122665
6    0.175948
Name: mort_acc, dtype: float64
```

```
#This function will replace the null values with the mean
def fill_mort_acc(total_acc, mort_acc):
    if np.isnan(mort_acc) == True:
        return total_account_mean[total_acc]
    else:
        return total_acc
```

```
data['mort_acc'] = data.apply(lambda x : fill_mort_acc(x["total_acc"], x['mort_acc']), axis =1)
```

```
data.isnull().sum()
```

```
loan_amnt                0
term                     0
int_rate                 0
installment              0
grade                    0
sub_grade                0
home_ownership           0
annual_inc               0
verification_status      0
issue_d                  0
loan_status              0
purpose                  0
addr_state               0
dti                    145
earliest_cr_line         0
open_acc                 0
pub_rec                  0
revol_bal                0
revol_util             315
total_acc                0
initial_list_status      0
application_type         0
mort_acc                 0
pub_rec_bankruptcies   366
FICO Score               0
loan_repaid              0
dtype: int64
```

Οι υπόλοιπες κενές τιμές είναι λιγότερο από το 0.5% των δεδομένων οπότε θα τις διαγράψουμε.

```
#The rest have null values less the .5% of the whole dataset so we can drop the null raws withput any hesitation
data = data.dropna()
```

Στη μεταβλητή 'term', η οποία αφορά τον αριθμό των δόσεων, θα γίνει μια μετατροπή ώστε να κρατήσουμε μόνο το αριθμητικό κομμάτι και θα διώξουμε το λεκτικό.

```
data['term'].head()
```

```
0     36 months
1     36 months
2     36 months
3     36 months
4     36 months
Name: term, dtype: object
```

```
#We would like to keep only the numerical value this column
data['term'] = data['term'].apply(lambda x: int(x[:3]))
```

```
data['term'].unique()
```

```
array([36, 60], dtype=int64)
```

Επίσης, μελετώντας την μεταβλητή που αφορά την ιδιοκτησία σπιτιού, home_ownership, θα δούμε ότι έχει τις εξής τιμές.

```
print(data['home_ownership'].unique())
```

```
['RENT' 'MORTGAGE' 'OWN' 'ANY' 'OTHER' 'NONE']
```

Θα μετατρέψουμε τις τιμές ‘ANY’ και ‘NONE’ σε ‘OTHER’ γιατί διαφορετικά δεν μας παρέχουν σημαντικές πληροφορίες.

```
data['home_ownership'] = data['home_ownership'].replace(['NONE', 'ANY'], 'OTHER')
```

```
print(data['home_ownership'].unique())
```

```
['RENT' 'MORTGAGE' 'OWN' 'OTHER']
```

Έπειτα, η μεταβλητή ‘grade’ φαίνεται να είναι ένα υπερσύνολο της μεταβλητής ‘sub_grade’ οπότε, μπορούμε να την διαγράψουμε.

```
#grade seems to be a super-category of sub_grade
data = data.drop('grade', axis =1)
```

Επίσης, θα διαγράψουμε την μεταβλητή ‘issue_d’ αφού μας υποδηλώνει την ημερομηνία την οποία εγκρίθηκε το δάνειο και σε ένα νέο σύνολο δεδομένων κάτι τέτοιο θα ήταν μεροληπτικό να το γνωρίζουμε.

```
data = data.drop('issue_d', axis = 1)
```

Τέλος, θα διαγράψουμε τις κολώνες ‘loan_status’ και ‘earliest_cr_line’, αφού δεν μας παρέχουν σημαντικές πληροφορίες, και θα εξάγουμε το νέο csv αρχείο το οποίο, μετά την επεξεργασία που έχει γίνει, θα είναι έτοιμο να εφαρμοστούν οι αλγόριθμοι μηχανικής μάθησης.

```
data = data.drop(['earliest_cr_line', 'loan_status'], axis =1 )
```

```
data.to_csv('data_ready_for_embeddings.csv', index = False)
```

3. Επεξεργασία Φυσικής Γλώσσας

Η επεξεργασία φυσικής γλώσσας (natural language processing – NLP) αναφέρεται σε υπολογιστικές τεχνικές που αφορούν τη γλώσσα. Είναι ένα ευρύ πεδίο αλλά στην συγκεκριμένη διπλωματική θα επικεντρωθούμε στην βιβλιοθήκη 'spacy'.

3.1 Η Βιβλιοθήκη spaCy

Η spacy είναι μια σχετικά καινούρια αλλά και δημοφιλής open source βιβλιοθήκη στην Python η οποία χρησιμοποιείται για επεξεργασία φυσικής γλώσσας. Χρησιμοποιεί προπαιδευμένα pipelines σε πάνω από 66 γλώσσες ώστε να μπορεί να μετατρέψει λέξεις και φράσεις σε διανύσματα. Στην συγκεκριμένη περίπτωση θα χρησιμοποιήσουμε ένα pipeline το οποίο έχει εκτεθεί σε έναν μεγάλο όγκο γλωσσικών μοτίβων το οποίο φορτώνεται με το string ΄en_core_web_lg'.

Η spacy αφού εκτεθεί στις λέξεις και στις φράσεις τις οποίες θέλουμε να επεξεργαστούμε θα μας επιστρέψει ένα διάνυσμα με 300 διαστάσεις. Δεδομένου ότι έχουμε αρκετές στήλες στα δεδομένα μας οι οποίες περιέχουν λεκτικές πληροφορίες δεν θα μπορούσαμε να κρατήσουμε για κάθε μια μεταβλητή τόσες πολλές διαστάσεις γιατί θα καταλήγαμε με ένα σύνολο δεδομένων το οποίο η κάθε παρατήρηση θα είχε πάνω από 1.000 διαστάσεις, πράγμα ανεπιθύμητο.

Για να μειώσουμε αυτές τις τεράστιες διαστάσεις θα χρησιμοποιήσουμε τον αλγόριθμο PCA (Principal Component Analysis) ο οποίος θα δούμε πώς λειτουργεί στο επόμενο κεφάλαιο.

Παρακάτω θα δούμε την διαδικασία με την οποία μετατράπηκαν οι κατηγορικές μεταβλητές σε διανύσματα.

Πρώτα φορτώνουμε την βιβλιοθήκη και το προπαιδευμένο pipeline.

```
import spacy
import pandas as pd
```

```
nlp = spacy.load('en_core_web_lg')
```

Ύστερα ελέγχουμε να δούμε ποιες στήλες περιέχουν λέξεις αντί για αριθμούς και τι πληροφορίες μας δίνουν.

```
data = pd.read_csv('data_ready_for_embeddings.csv')
data.select_dtypes('object')
```

	sub_grade	home_ownership	verification_status	purpose	addr_state	initial_list_status	application_type
0	A2	RENT	Not Verified	vacation	AR	w	Individual
1	B4	MORTGAGE	Not Verified	debt_consolidation	GA	f	Individual
2	C3	RENT	Verified	debt_consolidation	NY	w	Individual
3	B2	OWN	Not Verified	credit_card	NV	w	Individual
4	C5	RENT	Verified	other	CA	f	Individual
...	...	...	...	...	...	...	...
523457	B4	RENT	Source Verified	debt_consolidation	NY	w	Individual
523458	D1	MORTGAGE	Source Verified	debt_consolidation	GA	f	Individual
523459	A2	RENT	Not Verified	credit_card	VA	w	Individual
523460	C1	RENT	Source Verified	credit_card	MD	w	Individual
523461	B3	RENT	Source Verified	credit_card	MA	w	Individual

523462 rows × 7 columns

Εύκολα παρατηρούμε ότι η στήλη ‘sub_grade’ και ‘initial_list_status’ δεν έχουν κάποιο λεκτικό νόημα για τον αλγόριθμο οπότε μπορούμε να τις διαγράψουμε.

```
data = data.drop(['sub_grade', 'initial_list_status'], axis = 1)
```

Επίσης θα χρειαστεί να αντικαταστήσουμε τις κάτω παύλες, ’_’, μέσα στις λέξεις με κενά, ‘ ’ στην στήλη purpose.

```
data['purpose'] = data['purpose'].replace('_', ' ', regex = True)
```

Τέλος, παρατηρούμε ότι η μεταβλητή ‘addr_state’ περιέχει τις συντομογραφίες από κάθε μια από τις 21 πολιτείες τις Αμερικής. Επειδή για τον αλγόριθμο οι συντομογραφίες δεν σημαίνουν τίποτα θα τις αντικαταστήσουμε με τα κανονικά ονόματα των πολιτειών αυτών.

```
# Turning the postal observation into country name
data['addr_state'] = data['addr_state'].map({'AR':'Arkansas', 'GA':'Georgia', 'NY':'New York', 'NV':'Nevada', 'CA':'California',
                        'TX':'Texas', 'PA':'Pennsylvania', 'NJ':'New Jersey', 'IL':'Illinois', 'MD':'Maryland', 'FL':'Florida
                       'AZ':'Arizona', 'OH':'Ohio', 'OR':'Oregon', 'OK':'Oklahoma', 'MI':'Michigan', 'WA':'Washington',
                        'NC':'North Carolina', 'RI':'Rhode Island', 'HI':'Hawaii', 'CO':'Colorado', 'AK':'Alaska',
                       'VA':'Virginia', 'KY':'Kentucky', 'AL':'Alabama', 'IN':'Indiana', 'CT':'Connecticut', 'WV':'West Virgi
                        'UT':'Utah', 'MN':'Minnesota', 'MA':' Massachusetts', 'SC':'South Carolina', 'MO':'Missouri',
                       'NM':'New Mexico', 'WI':'Wisconsin', 'TN':'Tennessee', 'KS':'Kansas', 'ME':'Maine', 'MS':'Mississippi',
                        'SD':'South Dakota', 'LA':'Louisiana', 'NE':'Nebraska', 'DC':'District of Columbia',
                       'DE':'Delaware', 'WY':'Wyoming', 'MT':'Montana', 'ID':'Idaho', 'VT':'Vermont', 'ND':'North Dakota', 'IA
```

Αφού τελειώσαμε με την επεξεργασία των κατηγορικών μεταβλητών μπορούμε να προχωρήσουμε στην μετατροπή τους σε διανύσματα.

```
home_ownership = data['home_ownership'].apply(lambda x:nlp(x).vector)

verification_status = data['verification_status'].apply(lambda x:nlp(x).vector)

purpose = data['purpose'].apply(lambda x:nlp(x).vector)

addr_state = data['addr_state'].apply(lambda x:nlp(x).vector)

application_type = data['application_type'].apply(lambda x:nlp(x).vector)

home_ownership_df = pd.DataFrame(home_ownership.to_list())

verification_status_df = pd.DataFrame(verification_status.to_list())

purpose_df = pd.DataFrame(purpose.to_list())

addr_state_df = pd.DataFrame(addr_state.to_list())

application_type_df = pd.DataFrame(application_type.to_list())
```

Αφού τώρα έχουμε τις κατηγορηματικές μεταβλητές σε διανύσματα σειρά έχει η μείωση διάστασης με εφαρμογή του αλγορίθμου PCA.

3.2 Ο Αλγόριθμος PCA

Η μέθοδος PCA (Ανάλυση Κύριων Συνιστωσών), αποτελεί μία γραμμική μέθοδο συμπίεσης Δεδομένων η οποία συνίσταται από τον επαναπροσδιορισμό των συντεταγμένων ενός συνόλου δεδομένων, σε ένα άλλο σύστημα συντεταγμένων το οποίο θα είναι καταλληλότερο στην επικείμενη ανάλυση δεδομένων. Αυτές οι νέες συντεταγμένες είναι το αποτέλεσμα ενός γραμμικού συνδυασμού προερχόμενου από τις αρχικές μεταβλητές και εκπροσωπούνται σε ορθογώνιο άξονα, ενώ τα επικείμενα σημεία διατηρούν μια φθίνουσα σειρά όσον αφορά στην τιμή της διακύμανσής τους. Για τον λόγο αυτό, το πρώτο κύριο συστατικό διατηρεί περισσότερες πληροφορίες δεδομένων σε σύγκριση με το δεύτερο το οποίο δεν διατηρεί πληροφορίες οι οποίες έχουν εισέλθει νωρίτερα (στο πρώτο συστατικό).

Έστω ένα σύνολο δεδομένων,

$$x_1 = (x_{11}, x_{21}, \dots, x_{n1}), \dots, x_n = (x_{1n}, x_{2n}, \dots, x_{nn})$$

n: η διάσταση του διανύσματος

- ***Βήμα 1ο:*** Αφαίρεση του μέσου όρου από κάθε μια από τις διαστάσεις του πίνακα δεδομένων.
$$\widehat{x_1} = (x_{11} - \overline{x_1}, \dots, x_{n1} - \overline{x_1}), \dots, \widehat{x_n} = (x_{1n} - \overline{x_n}, \dots, x_{nn} - \overline{x_n})$$
- ***Βήμα 2ο:*** Υπολογισμός του πίνακα συνδιακύμανσης
$$cov(X,Y) = \frac{\sum_{i=1}^{n} x_i - \bar{x}}{n-1}$$
- ***Βήμα 3ο:*** Υπολογισμός ιδιοδιανυσμάτων και των ιδιοτιμών του πίνακα συνδιακύμανσης
$$cov(X,Y) - \lambda I = 0,$$
I: ο μοναδιαίος πίνακας
λ: η ζητούμενη ιδιοτιμή
αφού βρούμε τις ιδιοτιμές λ για κάθε μια από αυτές βρίσκουμε τα ιδιοδιανύσματα
$$(cov(X,Y) - \lambda I)\vec{v} = 0,$$
$$v\text{: άγνωστο ιδιοδιάνυσμα}$$
- **Βήμα 4ο:** Επιλογή στοιχείων που θα αποτελούν το διάνυσμα χαρακτηριστικών.
    - Σε αυτό το σημείο έρχεται η έννοια της συμπίεσης στοιχείων και της μείωσης των διαστάσεων. Αν λάβουμε υπόψιν τα ιδιοδιανύσματα και τις ιδιοτιμές του προηγούμενου βήματος, θα δούμε ότι οι ιδιοτιμές είναι τελείως διαφορετικές τιμές. Στην πραγματικότητα αποδεικνύεται ότι το ιδιοδιάνυσμα με την υψηλότερη ιδιοτιμή είναι η κύρια συνιστώσα (principal component) του συνόλου των στοιχείων.

- το επόμενο βήμα είναι η τοποθέτησή των ιδιοδιανυσμάτων σε σειρά σύμφωνα με τις αντίστοιχες τιμές των ιδιοτιμών από το μεγαλύτερο στο μικρότερο. Αυτό μας δίνει όλα τα συστατικά αυτά στοιχεία σε σειρά σπουδαιότητας. Στο σημείο αυτό μπορούμε να αγνοήσουμε τα λιγότερο σημαντικά στοιχεία γιατί μπορεί να χάνουμε κάποιες πληροφορίες, όταν όμως οι ιδιοτιμές είναι μικρού μεγέθους, τότε δεν χάνουμε τόσα πολλά σε πληροφορίες-δεδομένα.

$$FeatureVector = (eig_1, eig_2, \dots, eig_n)$$

- ***Βήμα 5ο:*** *Συλλογή νέων δεδομένων*
  $FinalData = (RowFeatureVector)^T * RowDataAdjust,$

  $RowFeatureVector$: πίνακας με ιδιοδιανύσματα σε στήλες ανάστροφος

  RowDataAdjust: αντιστοιχεί στα μέσα δεδομένα αντιμεταθετιμένα, δηλαδή τα στοιχεία των δεδομένων βρίσκονται σε κάθε στήλη, με κάθε γραμμή να έχει ξεχωριστή διάσταση
  [4].

Για να χρησιμοποιήσουμε τον αλγόριθμο PCA θα πρέπει πρώτα να κάνουμε Import το αντικείμενο από την βιβλιοθήκη Sklearn.

```
from sklearn.decomposition import PCA
```

Έπειτα πρέπει να δώσουμε σαν όρισμα σε πόσες συνιστώσες θα μειωθεί το αρχικό μας διάνυσμα.

```
pca = PCA(n_components=3)
```

Τέλος, θα κάνουμε fit & transform τα αρχικά διανύσματα ώστε να καταλήξουμε στα τελικά τριών διαστάσεων και να μπορέσουμε να τα συγχωνέψουμε στο αρχικό μας DataFrame.

```
home_ownership_pca = pca.fit_transform(home_ownership_df)

verification_status_pca = pca.fit_transform(verification_status_df)

purpose_pca = pca.fit_transform(purpose_df)

addr_state_pca = pca.fit_transform(addr_state_df)

application_type_pca = pca.fit_transform(application_type_df)

home_ownership_pca_df = pd.DataFrame(home_ownership_pca, columns = ['home_ownership_x', 'home_ownership_y', 'home_ownership_z'])

verification_status_pca_df = pd.DataFrame(verification_status_pca,
                                          columns=['verification_status_x', 'verification_status_y', 'verification_status_z'])

purpose_pca_df = pd.DataFrame(purpose_pca, columns=['purpose_x', 'purpose_y', 'purpose_z'])

addr_state_pca_df = pd.DataFrame(addr_state_pca, columns=['addr_state_x', 'addr_state_y', 'addr_state_z'])

application_type_pca_df = pd.DataFrame(application_type_pca,
                                       columns=['application_type_x', 'application_type_y', 'application_type_z'])
```

Ένα DataFrame μιας μεταβλητής μειωμένη σε τρείς διαστάσεις θα μοιάζει κάπως έτσι:

application_type_pca_df

	application_type_x	application_type_y	application_type_z
0	-1.925789	0.000010	1.634741e-04
1	-0.809987	-0.000006	1.000786e-06
2	-0.810009	-0.000006	-5.648689e-07
3	-0.810600	-0.000005	-1.197172e-06
4	-0.810409	-0.000005	-5.251210e-07
...	...	...	...
523457	-0.809844	-0.000006	5.080765e-10
523458	-0.809844	-0.000006	5.080765e-10
523459	-0.809844	-0.000006	5.080765e-10
523460	-0.809844	-0.000006	5.080765e-10
523461	-0.809844	-0.000006	5.080765e-10

523462 rows × 3 columns

Ενώ το τελικό DataFrame θα είναι ως εξής:

```
embed_data = pd.concat([data_safe, home_ownership_pca_df, verification_status_pca_df, purpose_pca_df, addr_state_pca_df, applicat
```

```
embed_data
```

	loan_amnt	term	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_util	...	verification_status_z	purpose_x	purpose_y	pur
0	2400	36	7.07	74.19	30000.0	25.32	4	0	723	8.9	...	1.975797e-06	-0.109314	20.489895	12.
1	10000	36	11.67	330.57	89000.0	10.85	9	0	20060	54.5	...	1.813835e-05	-15.446892	-6.767229	-1.
2	19200	36	13.67	653.14	67000.0	43.42	17	0	2666	6.1	...	-2.653020e-06	-15.446817	-6.764812	-1.
3	18475	36	9.16	588.88	100000.0	28.90	14	1	19052	75.9	...	-2.192268e-06	36.162899	-10.016985	-3.
4	5000	36	14.99	173.31	16800.0	10.43	3	0	5244	64.7	...	-6.348833e-07	2.496497	51.561733	-24.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
523457	20000	36	10.99	654.68	75000.0	16.75	15	0	10472	81.2	...	4.130980e-06	-15.446823	-6.764812	-1.
523458	13200	36	15.61	461.54	34000.0	19.73	8	0	15929	102.1	...	4.130980e-06	-15.446823	-6.764812	-1.
523459	17000	36	6.49	520.96	80000.0	12.93	4	0	20644	55.2	...	4.131051e-06	36.162899	-10.016988	-3.
523460	20000	60	12.39	448.85	45000.0	28.45	11	0	34966	90.1	...	4.130980e-06	36.162899	-10.016988	-3.
523461	6025	36	11.39	198.37	40000.0	6.51	6	0	6022	37.6	...	4.130980e-06	36.162899	-10.016988	-3.

523462 rows × 30 columns

```
embed_data.to_csv('data_with_embeddings_final.csv', index = False)
```

```
embed_data.shape
```

```
(523462, 30)
```

4. Μέθοδοι μηχανικής μάθησης και η συμβολή τους στην μείωση πιστωτικού κινδύνου

Όπως είδαμε παραπάνω τον κάθε δανειολήπτη τον χαρακτηρίζουν κάποια δεδομένα τα οποία μέσω κατάλληλης επεξεργασίας μπορούν να συμβάλουν στην πρόβλεψη πιστωτικού κινδύνου, δηλαδή αν ο πελάτης είναι φερέγγυος για την αποπληρωμή του δανείου.

Οι χρηματοπιστωτικές εταιρίες εν γένει και φυσικά οι τράπεζες, προφανώς κατέχουν τέτοια δεδομένα και στην εποχή που ζούμε είναι φυσικό ο όγκος αυτών να είναι γιγαντιαίος, με αποτέλεσμα οι διαχείρισή τους να καθίσταται υπερβολικά δύσκολη εάν όχι ακατόρθωτη.

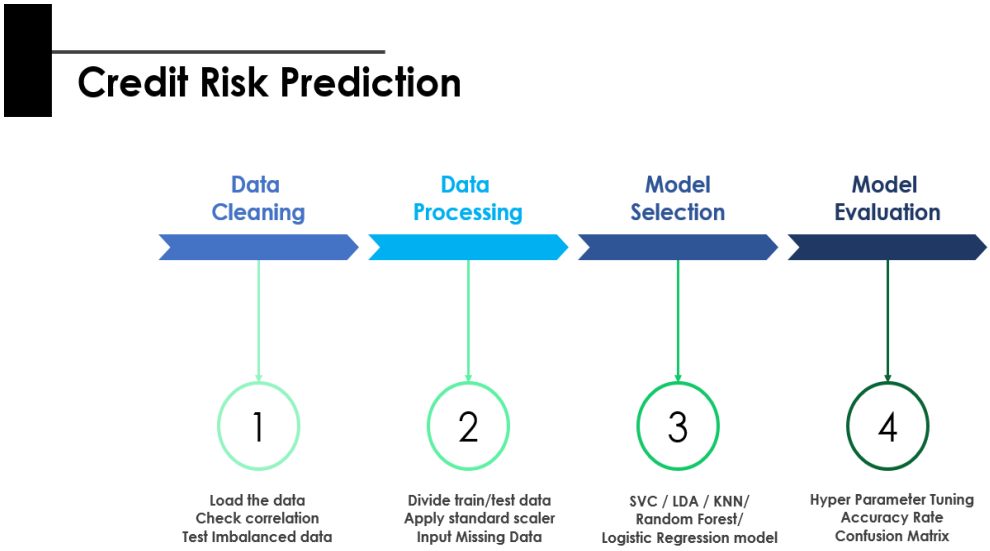
Στο σημείο αυτό μπαίνουν στο προσκήνιο τα μοντέλα μηχανικής μάθησης. Τα μοντέλα μηχανικής μάθησης είναι δομημένα ώστε να μπορούν να χειριστούν κατάλληλα αυτόν τον μεγάλο όγκο δεδομένων και να εξάγουν χρήσιμα συμπεράσματα με στόχο, στην προκειμένη περίπτωση, την ελαχιστοποίηση του πιστωτικού κινδύνου.

Προτού όμως αναφερθούμε στην μηχανική μάθηση αυτή καθ’ αυτή, θα πρέπει να ορίσουμε τι εννοούμε με τον όρο μοντέλο. Με τον όρο μοντέλο εννοούμε απλώς την περιγραφή μιας μαθηματικής (ή πιθανοτικής) σχέσης που υφίσταται μεταξύ διαφορετικών μεταβλητών [1].

Σε πολλές βιβλιογραφίες συναντάμε διαφορετικούς ορισμούς για την μηχανική μάθηση. Στην συγκεκριμένη διπλωματική εργασία θα χρησιμοποιήσουμε τον όρο μηχανική μάθηση (machine learning) για να αναφερόμαστε στη δημιουργία και την χρήση μοντέλων τα οποία μαθαίνουν από τα δεδομένα. Σε άλλα πλαίσια αυτό μπορεί να λέγεται προβλεπτική μοντελοποίηση (predictive modeling) ή εξόρυξη δεδομένων (data mining). Συνήθως ο στόχος μας είναι η χρήση προϋπάρχοντων δεδομένων για την ανάπτυξη μοντέλων με τα οποία να μπορούμε να προβλέψουμε διάφορα αποτελέσματα για νέα δεδομένα, εν προκειμένω αν κάποιος θα μπορέσει να αποπληρώσει το δάνειό του ή όχι [1].

Τα μοντέλα που θα χρησιμοποιήσουμε για την πρόβλεψη είναι μοντέλα επιβλεπόμενης μάθησης (Supervised Learning) δηλαδή, τα μοντέλα εκπαιδεύονται σε σύνολα δεδομένων των οποίων ήδη ξέρουμε τις απαντήσεις [1].

Σε προηγούμενα κεφάλαια κάναμε μια εκκαθάριση των δεδομένων η οποία αποτελούσε το πρώτο βήμα της διαδικασίας πρόβλεψης φερεγγυότητας του δανειολήπτη. Παρακάτω φαίνεται η συνολική εικόνα της διαδικασίας.



Εικόνα 1: Διαδικασία Πρόβλεψης φερεγγυότητας του δανειολήπτη [5].

Ένας συνήθης κίνδυνος στην μηχανική μάθηση είναι το overfitting, δηλαδή να δημιουργήσεις ένα μοντέλο το οποίο αποδίδει καλά στα δεδομένα στα οποία το εκπαιδεύουμε, αλλά να έχει μέτρια έως κακή απόδοση σε οποιαδήποτε νέα δεδομένα. Αυτό θα μπορούσε να οφείλεται στην εκμάθηση θορύβου από τα δεδομένα ή θα μπορούσε να αφορά την εκμάθηση συγκεκριμένων μοτίβων από τα δεδομένα [1].

Η άλλη όψη είναι το underfitting, δηλαδή η δημιουργία ενός μοντέλου το οποίο δεν αποδίδει καλά ούτε στα δεδομένα εκπαίδευσης, ούτε στα νέα δεδομένα. Αν και συνήθως όταν συμβαίνει αυτό αποφασίζουμε ότι το μοντέλο μας δεν είναι αρκετά καλό και αναζητούμε κάποιο καινούριο [1].

Για να αποφευχθούν τα παραπάνω δύο πιθανά σενάρια είναι να χωρίσουμε τα δεδομένα σε ένα σύνολο εκπαίδευσης, για παράδειγμα δύο τρίτα των δεδομένων, και σε ένα σύνολο ελέγχου το οποίο θα αξιολογεί την απόδοση του μοντέλου.

```
x = data.drop(['loan_repaid'], axis = 1).values#we need np.array values cause keras works better with np.arrays
y = data['loan_repaid'].values
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= .3)
```

Τα μοντέλα μηχανικής μάθησης που θα φτιάξουμε παρακάτω έχουν δυαδικό χαρακτήρα, δηλαδή θα προβλέπουν δύο πιθανά ενδεχόμενα, αν ο δανειολήπτης θα πληρώσει ή όχι. Άρα κάθε σημείο δεδομένων θα εμπίπτει σε μια από τις εξής τέσσερις κατηγορίες σφάλματος [1].

- *Αληθώς Θετικό:*

  <<Το μοντέλο προβλέπει ότι ο δανειολήπτης θα αποπληρώσει και όντως αποπληρώνει.>>

- *Ψευδώς Θετικό:*

  <<Το μοντέλο προβλέπει ότι ο δανειολήπτης ΔΕΝ θα αποπληρώσει, αλλά τελικά αποπληρώνει.>>

- *Ψευδώς Αρνητικό:*

  <<Το μοντέλο προβλέπει ότι ο δανειολήπτης θα αποπληρώσει, αλλά δεν αποπληρώνει.>>

- *Αληθώς Αρνητικό*:

  <<Το μοντέλο προβλέπει ότι ο δανειολήπτης ΔΕΝ θα αποπληρώσει και όντως δεν αποπληρώνει.>>

Τα παραπάνω μπορούμε να τα συνοψίσουμε σε ένα πίνακα σύγχυσης (Confusion Matrix).

		Actual values: 1	Actual values: 0
Predicted values	1	TP	FP
	0	FN	TN

Εικόνα 2: Πίνακας Σύγχυσης (Confusion Matrix) [6].

Μπορούμε να χρησιμοποιήσουμε τον πίνακα σύγχυσης για να υπολογίσουμε διάφορα στατιστικά μεγέθη σχετικά με τις επιδόσεις του μοντέλου. Για παράδειγμα, η συνολική ακρίβεια (accuracy) ορίζεται ως το ποσοστό των σωστών προβλέψεων [1].

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Επίσης, είναι σύνηθες να εξετάζουμε τον συνδυασμό ακρίβειας θετικών προβλέψεων (Precision) και ανάκλησης (Recall). Η ακρίβεια θετικών προβλέψεων μετράει το ποσοστό επιτυχίας των θετικών μας προβλέψεων [1].

$$Precision = \frac{TP}{TP + FP}$$

Ενώ η ανάκληση μετράει ποιο ποσοστό των θετικών δειγμάτων αναγνωρίζεται σωστά από το μοντέλο μας [1].

$$Recall = \frac{TP}{TP + FN}$$

Μερικές φορές η ακρίβεια θετικών προβλέψεων και η ανάκληση συνδυάζονται στο σκορ F1, που ορίζεται ως εξής:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Τελευταίο πράγμα που πρέπει να γνωρίζουμε πριν αρχίσουμε να συστήνουμε τα μοντέλα μας είναι η αναπροσαρμογή κλίμακας (feature scaling) των δεδομένων μας.

Μια από τις πιο σημαντικές μετατροπές που γίνεται στα δεδομένα μας είναι το feature scaling. Με μερικές ελάχιστες εξαιρέσεις τα μοντέλα μηχανικής μάθησης δεν αποδίδουν καλά με δεδομένα τα οποία έχουν μεγάλη διαφορά στην κλίμακά τους. Υπάρχουν δύο τρόποι για να φέρουμε τα δεδομένα μας σε παρόμοια κλίμακα, το min-max scaling και το standardization [7].

Στα συγκεκριμένα μοντέλα το feature scaling έχει γίνει με την χρήση του Min-Max Scaler. Το Min-Max Scaling (πολλοί το αποκαλούν normalization) είναι αρκετά απλό. Οι τιμές τον δεδομένων μεταφέρονται και αναπροσαρμόζονται ώστε η μικρότερη τιμή να είναι το 0 και η μεγαλύτερη το 1. Αυτό γίνεται μέσω του παρακάτω τύπου.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Ενώ σε κώδικα γίνεται στις παρακάτω γραμμές με την χρήση της βιβλιοθήκης Sklearn.

```
#Scalling our features between 0 and 1
scaler = MinMaxScaler()

x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

4.1 Ο Αλγόριθμος SMOTE (Synthetic Minority Oversampling Technique)

Όπως είδαμε παραπάνω ο πληθυσμός των δεδομένων μας δεν είναι ισορροπημένος. Είναι προφανές ότι η κλάση των δανειοληπτών οι οποίοι έχουν καταφέρει να αποπληρώσουν πλήρως το δάνειο που έλαβαν είναι πολύ μεγαλύτερη από την κλάση με τους δανειολήπτες που δεν κατάφεραν να επιστρέψουν τα χρήματα που πήραν. Αυτό έχει ως αποτέλεσμα ο κάθε αλγόριθμος που θα κατασκευάσουμε να εμφανίζει μια μεροληψία προς την κλάση Fully Paid (κλάση 1). Προσπαθώντας να αποφύγουμε αυτή την μεροληψία χρησιμοποιούμε τον αλγόριθμο SMOTE.

Ο αλγόριθμος SMOTE στοχεύει στην εξισορρόπηση των δεδομένων μεγαλώνοντας την κλάση με τις λιγότερες παρατηρήσεις. Μπορεί να θεωρηθεί σε μια βελτιωμένη έκδοση oversampling. Βελτιωμένη διότι, ενώ με μια κοινή τεχνική oversampling θα δημιουργούσαμε διπλές και τριπλές εγγραφές μέσα στο νέο σύνολο δεδομένων, ο αλγόριθμος SMOTE δημιουργεί νέα συνθετικά δεδομένα τα οποία βασίζονται στα ήδη υπάρχοντα αλλά διαφέρουν [14].

- **Βήμα 1ο:** Θεωρούμε την κλάση σε μειοψηφία *A*, $\forall\, x \in A$, βρίσκουμε τους k – κοντινότερου γείτονες του χ, υπολογίζοντας την Ευκλείδεια απόσταση μεταξύ του *x* και όλων των υπόλοιπων σημείων στο σύνολο ***A.***
- ***Βήμα 2ο:*** Τα νέα δεδομένα μεγέθους *N* ορίζονται με βάση το πόσο μεγάλη είναι η διαφορά των κλάσεων. Για κάθε $x \in A$, *N* παρατηρήσεις διαλέγονται τυχαία από τους k – κοντινότερους γείτονες, και έτσι κατασκευάζεται το σύνολο A_1.
- ***Βήμα 3ο:*** Για κάθε παρατήρηση $x_k \in A_1$ (k = 1,2,3,…N) χρησιμοποιείται ο παρακάτω τύπος για να δημιουργηθεί ένα καινούριο σημείο.

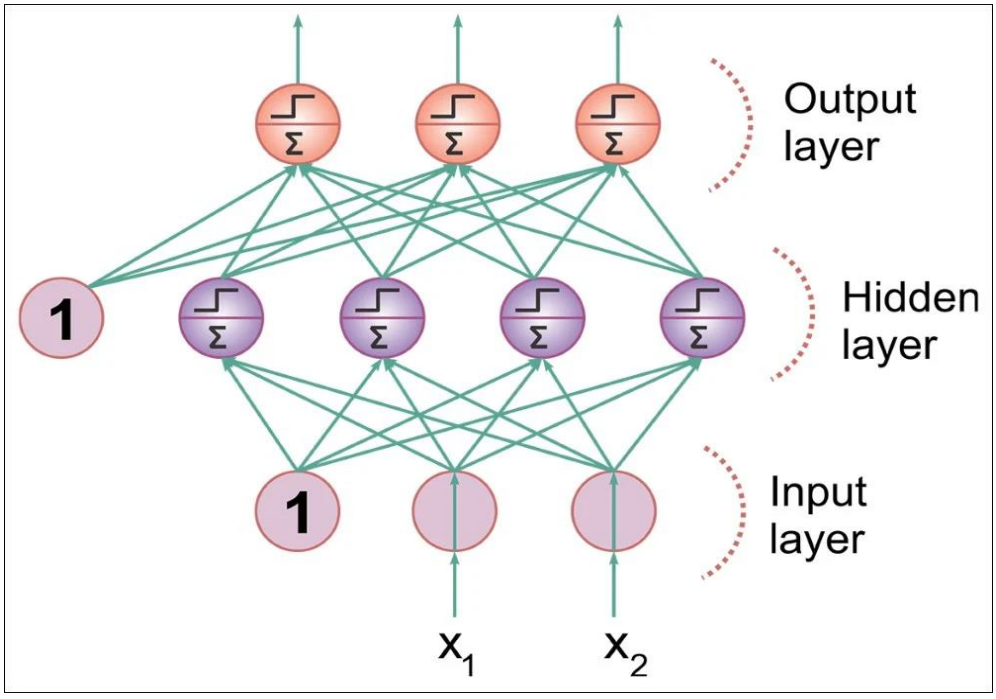
$$x' = x + rand(0,1) * |x - x_k| \text{ [14]}$$

Παρακάτω φαίνεται ο αλγόριθμος σε κώδικα.

```
smote = SMOTE()
x_train_smote, y_train_smote = smote.fit_resample(x_train, y_train)
```

4.2 Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks)

Ένα πολυστρωματικό νευρωνικό δίκτυο αποτελείται από ένα στρώμα εισόδου, ένα στρώμα εξόδου και ενδιάμεσα περιέχονται ένα ή περισσότερα στρώματα τα οποία αποκαλούνται κρυφά. Κάθε στρώμα νευρώνων περιέχει επιπρόσθετα έναν νευρώνα μεροληψίας (Bias neuron) και κάθε επίπεδο συνδέεται πλήρως με το επόμενο.



Εικόνα 8: Πολυστρωματικό Νευρωνικό Δίκτυο [7].

Για πολλά χρόνια οι επιστήμονες προσπαθούσαν, χωρίς επιτυχία, να βρουν έναν αλγόριθμο ο οποίος θα εκπαιδεύει ένα νευρωνικό δίκτυο. Το 1986 οι David Rumelhart, Geoffrey Hinton και Ronald Williams έκδοσαν μια μελέτη η οποία εισήγαγε τον αλγόριθμο εκπαίδευσης οπισθοδιάδοσης (Backpropagation training algorithm) [9].

Ο αλγόριθμος backpropagation συνίσταται από τα εξής βήματα.

- ***Βήμα 1ο:*** Πέρασμα προς τα εμπρός
  - Για κάθε νευρώνα-εισόδου θέτουμε ως y_i την είσοδο που παράγει.
  - Για κάθε νευρώνα j , j = 1,2,…N το σύνολο των νευρώνων (κρυφό ή εξόδου)
    - Υπολογίζεται το δυναμικό
      $v_j = \sum_{i=0}^{n} w_{ij} * y_i$,
      n: είναι ο συνολικός αριθμός εισόδων του νευρώνα j
      w_{ij}: το βάρος της ακμής από τον κόμβο i στον κόμβο j
    - Υπολογίζεται η έξοδος από την συνάρτηση ενεργοποίησης
      $y_i = \varphi(v_j)$
  - Υπολογίζουμε το σφάλμα για κάθε νευρώνα εξόδου. Επειδή η συνάρτηση κόστους είναι η binary cross entropy έχουμε τον παρακάτω τύπο για το σφάλμα [20].

$$E = -\frac{1}{N}\sum_{i=1}^{N} y_{true,i} * \log(p_i) + \left(1 - y_{true,i}\right) * \log\left(1 - p_i\right)$$ [19],

$y_{true,i}$: η πραγματική τιμή του προτύπου
p_i: η πιθανότητα η παρατήρηση να ανήκει στην κλάση 1

- ***Βήμα 2ο:*** Προς τα πίσω πέρασμα (οπισθοδρόμηση λάθους)
  - Υπολογίζεται η τοπική κλίση δ για κάθε νευρώνα.
    - Για τους νευρώνες εξόδου $\delta_j(n) = e_j * \varphi'_j(v_j)$
    - Για τους νευρώνες κρυφού επιπέδου $\delta_j(n) = \varphi'_j(v_j)\sum_k[\delta_\kappa(n) * w_{jk}(n)]$
    - Για τους νευρώνες εισόδου δεν γίνεται υπολογισμός τοπικής κλίσης [20].
- ***Βήμα 3ο:*** Διορθώσεις στα βάρη των ακμών (optimization με χρήση momentum)
  - Υπολογίζονται οι διορθώσεις στα βάρη των ακμών

$$\Delta W_{ij}(n) = \eta * \delta_j(n) * y_i(n) + a * \Delta w_{ji}(n-1),$$

η: ρυθμός μάθησης, 0<η<1
α: momentum, 0<α<1
  - *Υπολογίζονται τα νέα βάρη*

$$W_{ij}(n+1) = W_{ij}(n) + \Delta W_{ij}(n)$$ *[20]*

- **Κριτήριο Τερματισμού:**
  - Το δίκτυο παράγει τις επιθυμητές εξόδους ή έχουν ένα σφάλμα μικρότερο από το κριτήριο που έχουμε θέσει.
  - Το σφάλμα παραμένει ίδιο σε μερικούς διαδοχικούς κύκλους εκπαίδευσης.
  - Ο αλγόριθμος εκτελέστηκε για συγκεκριμένο αριθμό βημάτων [20].

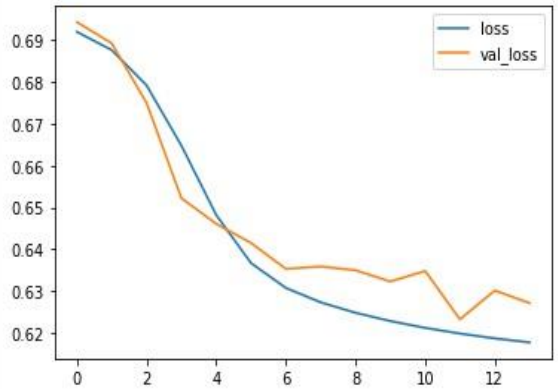
Παρακάτω βλέπουμε σε μερικές γραμμές κώδικα πώς φτιάχνεται ένα νευρωνικό δίκτυο το οποίο χρησιμοποιήσαμε για να προβλέψουμε αν κάποιος δανειολήπτης θα αποπληρώσει το δάνειο του ή όχι.

```
#Building our NN
model = Sequential()

model.add(Dense(80, activation = 'sigmoid'))
model.add(Dense(40, activation = 'sigmoid'))
model.add(Dense(20, activation = 'sigmoid'))


model.add(Dense(1, activation = 'sigmoid'))

learning_rate = 0.00001

opt = optimizers.Adam(learning_rate)

model.compile(loss = 'binary_crossentropy', optimizer = opt)

early_stop = EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1, patience = 2)
```

```
#creating a dataFrame for loss function among validation and training data
loss_history = pd.DataFrame(model.history.history)
```

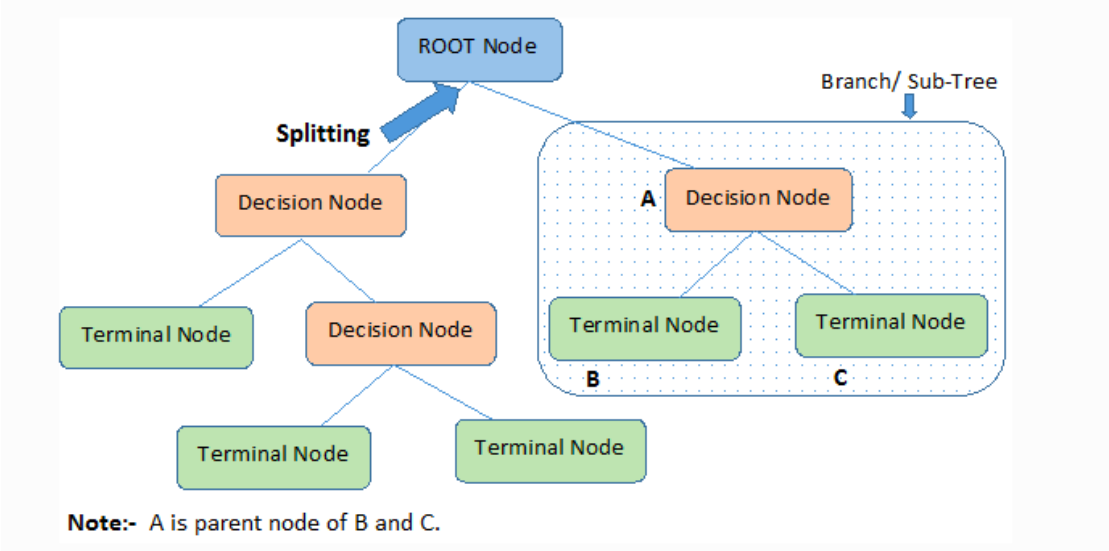
```
#plotting the loss function path among validation and training data
plt.figure(figsize=(12,8))
loss_history.plot()
```

```
<AxesSubplot:>

<Figure size 864x576 with 0 Axes>
```



4.3 Ο αλγόριθμος XGBoost

Ένα δέντρο απόφασης χρησιμοποιεί μια δενδρική δομή για να αναπαραστήσει ένα πλήθος δυνατών μονοπατιών απόφασης και ένα αποτέλεσμα για κάθε μονοπάτι.



Εικόνα 9: Δέντρο απόφασης (Decision Tree) [10].

Υπάρχουν πολλοί λόγοι να συστήνουμε τη χρήση δέντρων απόφασης. Είναι πολύ εύκολο να τα κατανοήσουμε και να τα ερμηνεύσουμε και η διαδικασία με την οποία καταλήγουν σε μια πρόγνωση είναι εντελώς διαφανής.

Το Boosting αναφέρεται σε οποιοδήποτε σύνολο κόμβων σε ένα δέντρο απόφασης το οποίο δυσκολεύεται να ‘μάθει’ αλλά συνδυαστικά μπορούν να φτιάξουν έναν ισχυρό κόμβο ο οποίος θα μαθαίνει ευκολότερα.

Ένας από του δημοφιλέστερους αλγορίθμους boosting είναι ο gradient boosting. Όπως όλοι οι αλγόριθμοι boosting έτσι και αυτός προσθέτει σειριακά κι άλλους προβλεπτές (κόμβους) από τους οποίους ο καθένας προσπαθεί να διορθώσει τους γονείς προβλεπτές. Ωστόσο η διαφορά του με τους υπόλοιπους boosting αλγορίθμους είναι πως αντί να προσπαθεί να προβλέψει την σωστή απάντηση για κάθε παρατήρηση προσπαθεί να προβλέψει το σφάλμα από την σωστή απάντηση [7].

Ο αλγόριθμος που χρησιμοποιείται σε αυτή την διπλωματική είναι ο XGBoost (Extreme Gradient Boosting) ο οποίος κατά βάση είναι ένας κατανεμημένος αλγόριθμος gradient boosting σε δένδρα αποφάσεων. Δηλαδή αντί να προσπαθεί να προσθέσει προβλεπτές σειριακά όπως είπαμε παραπάνω, το κάνει παράλληλα [11].

Έστω ότι έχουμε ένα σύνολο παρατηρήσεων το οποίο δίνεται ως εξής,

$$(x_{11}, x_{12}, \dots, x_{1k}, y_1), \dots, (x_{n1}, x_{n2}, \dots, x_{nk}, y_n),$$

n: ο αριθμός των παρατηρήσεων

x_{ik}: i=1,2,…,n, οι συνιστώσες του διανύσματος χαρακτηριστικών

y_i: i=1,2,…,n οι απαντήσεις των διανυσμάτων χαρακτηριστικών

Τότε η προσεγγιστική συνάρτηση δίνεται από τον τύπο,

$$\hat{y}_i = \emptyset(x_i) = \sum_{m=1}^{M} f_m(x_i), f_m \in F$$

Η διαδικασία επαναλήψεων για τον αλγόριθμο XGBoost ορίζεται ως εξής,

$$για\ 1 \leq m \leq M: F_{m+1}(x) = F_m(x) + f_m(i)$$

Με αντικειμενική συνάρτηση,

$$L(\emptyset) = \sum_i l(\hat{y}_i, y_i) + \sum_m \Omega(f_m)$$

Όπου,

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2$$

Κι έτσι παίρνουμε,

$$L(\emptyset) = \sum_i l(\hat{y}^{(m-1)} + f_m(x_i), y_i) + \sum_m \Omega(f_m) + c$$

Σύμφωνα με το πολυώνυμο 2ου βαθμού του Taylor η αντικειμενική συνάρτηση γίνεται,

$$\sum_{j=1}^{M} \left[\left(\sum_{i \in Ij} g_i \right) \omega_j + \frac{1}{2} \left(\sum_{i \in Ij} h_i + \lambda \right) \omega_j^2 \right] + \lambda T,$$

όπου, g_i και h_i είναι οι παράγωγοι 1ης και 2ης τάξης αντίστοιχα. Οι οποίες γίνονται σταθερές στη t επανάληψη [18].

XG-BOOST

```
xgb = XGBClassifier(use_label_encoder=False,
                    max_depth = 3,
                    subsample = .8,
                    n_estimators = 200,
                    learning_rate = .005,
                    min_child_weight = 1,
                   )


xgb.fit(x_train_smote, y_train_smote)
predictions_xgb = xgb.predict(x_test)
```

4.4 Λογιστική Παλινδρόμηση (Logistic Regression)

Κάποιοι παλινδρομικοί αλγόριθμοι μπορεί να χρησιμοποιηθούν σαν ταξινομητές. Η λογιστική παλινδρόμηση χρησιμοποιείται για τον υπολογισμό της πιθανότητας μιας παρατήρησης να ανήκει σε μια συγκεκριμένη κλάση. Αν η πιθανότητα είναι μεγαλύτερη ή ίση από 50% τότε η παρατήρηση αυτή ανήκει στην συγκριμένη κλάση, αλλιώς η παρατήρηση δεν ανήκει σε αυτή την κλάση. Τέλος, πρέπει να αναφερθεί ότι η λογιστική παλινδρόμηση είναι ένας δυαδικός ταξινομητής [7].

Έστω ότι έχουμε ένα σύνολο παρατηρήσεων το οποίο δίνεται ως εξής,

$$(x_{11}, x_{12}, \dots, x_{1k}, y_1), \dots, (x_{n1}, x_{n2}, \dots, x_{nk}, y_n),$$

n: ο αριθμός των παρατηρήσεων

x_{ik}: i=1,2,…,n, οι συνιστώσες του διανύσματος χαρακτηριστικών

y_i: i=1,2,…,n οι απαντήσεις των διανυσμάτων χαρακτηριστικών

Οι πιθανότητες για την κατηγοριοποίηση των παρατηρήσεων δίνεται από τους παρακάτω τύπους,

$$P(Y = 1|x) = \frac{\exp\,(b_0 + b_1x_1 + b_2x_2 + \cdots + b_kx_k)}{1 + \exp\,(b_0 + b_1x_1 + b_2x_2 + \cdots + b_kx_k}$$

$$P(Y = 0|x) = \frac{1}{1 + \exp\,(b_0 + b_1x_1 + b_2x_2 + \cdots + b_kx_k}$$

Γράφουμε,

$$p = \frac{\exp\,(z)}{1+\exp\,(z)},$$

Όπου $z = b_0 + b_1x_1 + b_2x_2 + \cdots + b_kx_k$

Κι έτσι έχουμε $\ln\left(\frac{p}{1-p}\right) = z$, [18].

Η συνάρτηση στο αριστερό μέλος της εξίσωσης ονομάζεται logit, ο λόγος p/1-p αντιπροσωπεύει τα odds που δεν είναι τίποτα παραπάνω από τον λόγο της πιθανότητας να συμβεί ένα γεγονός προς την πιθανότητα να μην συμβεί. Έτσι η λογιστική παλινδρόμηση επιστρέφει έναν αριθμό στο διάστημα [0,1], ο οποίος αριθμός αντιστοιχεί στην πιθανότητα ένα γεγονός να συμβεί [18].

Logistic Reggression

```
lg = LogisticRegression(solver = 'saga')
lg.fit(x_train_smote, y_train_smote)
predictions_lg = lg.predict(x_test)
```

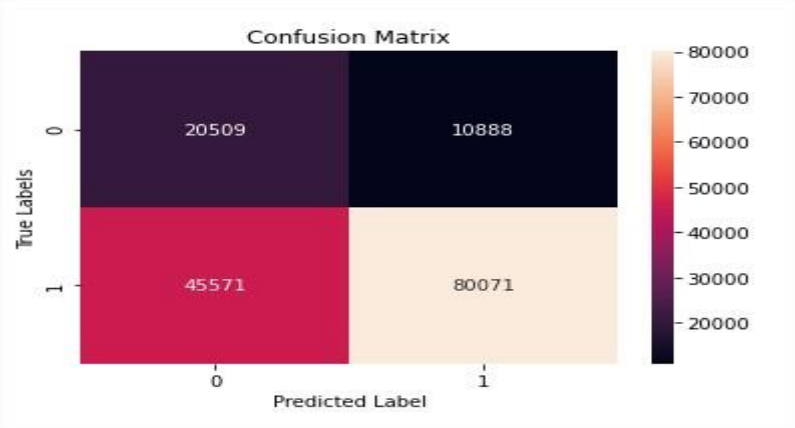
5. Αποτελέσματα

Παραπάνω είδαμε τους αλγορίθμους ταξινόμησης που χρησιμοποιήθηκαν σε αυτή την διπλωματική εργασία και σε ένα αρκετά υψηλό επίπεδο πώς λειτουργούν. Σε αυτό το κεφάλαιο θα δούμε τα αποτελέσματα που μας δίνουν αν τελικά είναι αρκετά ισχυροί ώστε να μπορούν να βοηθήσουν έναν χρηματοπιστωτικό οργανισμό να προβλέψει την πιθανότητα να αποπληρωθεί ένα ζητούμενο δάνειο και τελικά να μπορέσει να ελαχιστοποιήσει τον κίνδυνο ζημίας.

Για να καταλάβουμε λοιπόν αν οι αλγόριθμοι οι οποίοι αναπτύξαμε είναι αρκετά καλοί, θα πρέπει πρώτα να καταλάβουμε τι είναι το πρόβλημα το οποίο ζητάμε να μας επιλύσουν. Στόχος των αλγορίθμων είναι να μειώσουν τον πιστωτικό κίνδυνο, δηλαδή, να μπορούν να προβλέψουν με όσο το δυνατόν μεγαλύτερη ακρίβεια ποιοι από τους ενδιαφερόμενους είναι κατάλληλοι και ικανοί να αποπληρώσουν το δάνειο που αιτούνται. Δεδομένου ότι οι αλγόριθμοι αποσκοπούν στο να μειώσουν τον κίνδυνο κακού δανεισμού σε έναν χρηματοπιστωτικό οργανισμό και όχι να αυξήσουν το κέρδος του απαραίτητα, το πρόβλημα ανάγεται στη μείωση των false positive αποτελεσμάτων. Τα false positive αποτελέσματα όπως είδαμε παραπάνω είναι οι ενδιαφερόμενοι δανειολήπτες οι οποίοι κρίνονται ικανοί για την αποπληρωμή του δανείου αλλά τελικά οδηγούν τον χρηματοπιστωτικό οργανισμό σε απώλεια κεφαλαίου. Τα false negative αποτελέσματα είναι οι ενδιαφερόμενοι οι οποίοι κρίθηκαν ακατάλληλοι για χορήγηση δανείου αλλά τελικά ήταν μια καλή επένδυση κεφαλαίου. Τα μετρήσιμα τα οποία επηρεάζονται από τα FP και τα FN αντίστοιχα είναι το precision και το recall, στα οποία θα συγκεντρωθούμε παρακάτω.

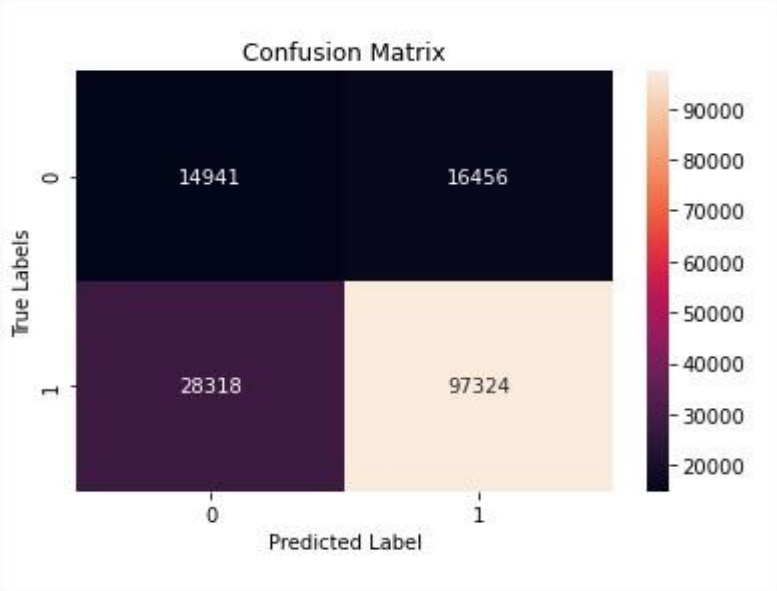
Παρακάτω παρατίθενται πίνακες με τα αποτελέσματα των αλγορίθμων συγκεντρωμένα αλλά και οι πίνακες σύγχυσης του εκάστοτε αλγόριθμου ώστε να καταλάβουμε πώς κατανέμουν τις παρατηρήσεις.

Algorithm	Class	Precision	Recall	f1 Score	Accuracy
ANN	0	0.31	0.65	0.42	64.05%
	1	0.88	0.64	0.74	



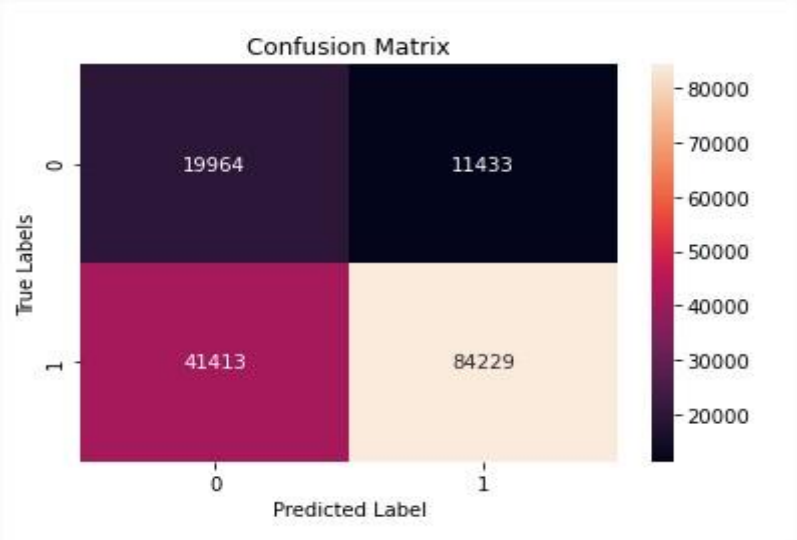
Πίνακας Σύγχυσης Τεχνητού Νευρωνικού Δικτύου.

Algorithm	Class	Precision	Recall	f1 Score	Accuracy
XG-BOOST	0	0.35	0.48	0.40	71.49%
	1	0.86	0.77	0.81	



Πίνακας Σύγχυσης XG-Boost.

Algorithm	Class	Precision	Recall	f1 Score	Accuracy
Logistic Regression	0	0.33	0.64	0.43	66.35%
	1	0.88	0.67	0.76	



Πίνακας Σύγχυσης Λογιστικής Παλινδρόμησης.

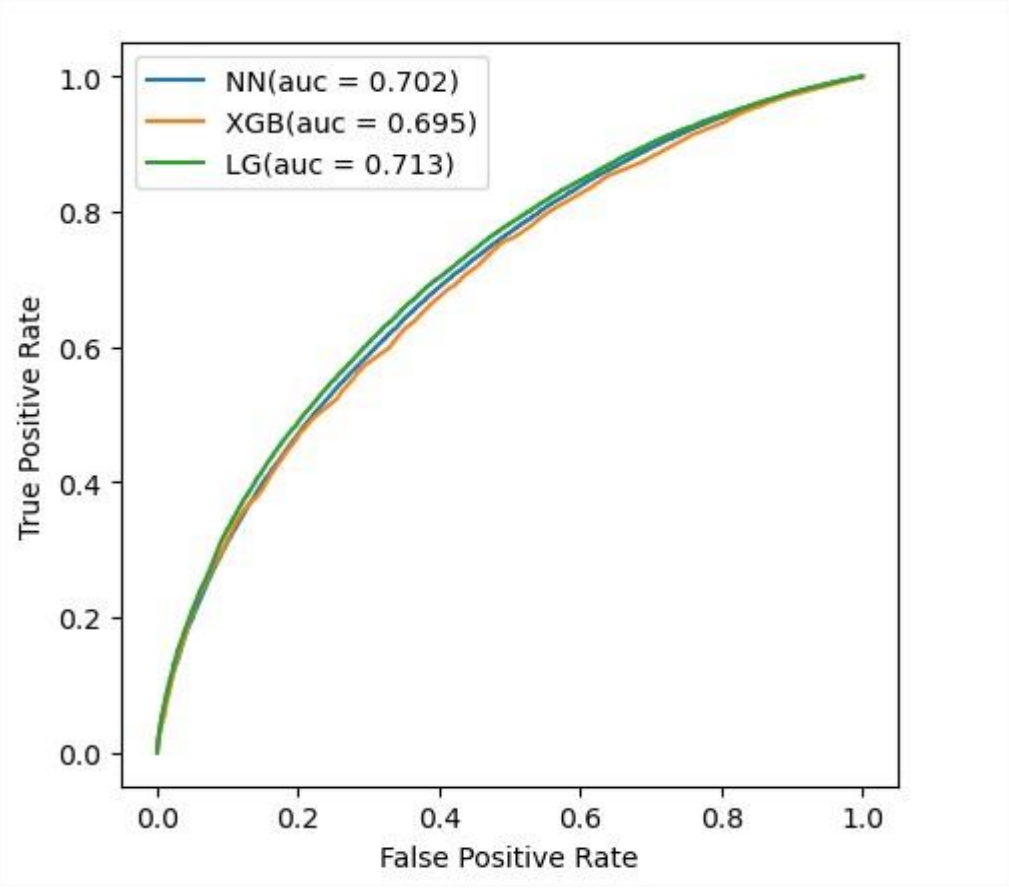
Για να έχουμε μια πλήρη αλλά και συγκεντρωτική εικόνα για το ποιο μοντέλο αποδίδει καλύτερα θα χρησιμοποιήσουμε την καμπύλη ROC (Receiver Operating Characteristic) και την έννοια AUC (Area Under the Curve).

Η καμπύλη ROC χρησιμοποιείται πολύ συχνά στους δυαδικούς κατανεμητές. Στον οριζόντιο άξονα απεικονίζεται το true positive rate (TPR, εναλλακτικό όνομα για το recall) ενώ, στον κάθετο άξονα απεικονίζεται το false positive rate. Το FPR ορίζεται ως η αναλογία των αρνητικών παρατηρήσεων οι οποίες λανθασμένα έχουν ταυτοποιηθεί ως θετικές [7].

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

Ένας πολύ γνωστός τρόπος να συγκριθούν οι αλγόριθμοι (κατανεμητές) είναι το εμβαδόν κάτω από την καμπύλη ROC. Ένας τέλειος κατανεμητής θα είχε AUC = 1 ενώ, ένας τελείως τυχαίος κατανεμητής θα είχε AUC = 0.5. Παρακάτω φαίνονται οι καμπύλες ROC και τα αντίστοιχα AUC που αφορούν τους κατανεμητές που χρησιμοποιούνται στην παρούσα διπλωματική εργασία.



ROC & AUC των μοντέλων.

Τα παραπάνω μοντέλα φαίνεται να είναι πολύ κοντά στο πόσο καλά αποδίδουν. Η λογιστική παλινδρόμηση όμως φαίνεται ξεκάθαρα να αποδίδει καλύτερα από τους υπόλοιπους αλγορίθμους.

Συμπεράσματα & Μελλοντική Έρευνα

Η παρούσα διπλωματική εργασία αναφέρεται στο πώς η εξέλιξη της τεχνολογίας και της επιστήμης των υπολογιστών μπορούν να συμβάλουν σημαντικά στη μείωση πιστωτικού κινδύνου χρηματοοικονομικών οργανισμών αλλά και όχι μόνο. Έγινε αναφορά στον πιστωτικό κίνδυνο και τι επιπτώσεις μπορεί να έχει σε εταιρείες όπως η Lending Club εάν δεν ληφθούν τα κατάλληλα μέτρα για τη μείωσή του. Επίσης αφιερώνεται μεγάλο κομμάτι στο data cleaning αλλά και στο feature engineering αφού και τα δύο είναι τα προπύλαια ώστε να δημιουργηθούν μοντέλα μηχανικής μάθησης τα οποία θα μας δίνουν τα επιθυμητά αποτελέσματα. Δόθηκαν αφ' υψηλού οι ορισμοί για τους αλγορίθμους που χρησιμοποιήσαμε αλλά και πώς αυτοί λειτουργούν.

Για την υλοποίηση των αλγορίθμων χρησιμοποιήθηκαν τέσσερα μοντέλα μηχανικής μάθησης, νευρωνικό δίκτυο, XGBoost και λογιστική παλινδρόμηση. Τα αποτελέσματα από κάθε μοντέλο ήταν παραπλήσια αλλά και αρκετά ικανοποιητικά ώστε να φανούν χρήσιμα σε μια επιχείρηση και να καταφέρει να μειώσει τον κίνδυνο άστοχων επενδύσεων. Τέλος, όπως είδαμε παραπάνω στο σύνολο των δεδομένων μας είχε εξ ορισμού μεγαλύτερη πιθανότητα κάποιος να αποπληρώσει το δάνειο το οποίο ζητούσε, πράγμα που σημαίνει ότι το σύνολό μας είχε κάποια μεροληψία εγκατεστημένη. Έχοντας στο μυαλό μας αυτό ίσως θα μπορούσε να δημιουργηθεί ένα αμερόληπτο μοντέλο προγνωστικής το οποίο δεν θα λάμβανε υπ' όψη του αυτή την μεροληψία του συνόλου ώστε να αποδίδει ακόμα καλύτερα αποτελέσματα για τη μείωση του πιστωτικού κινδύνου.

Βιβλιογραφία

[1] J. Grus (2019), ***'Data Science from Scratch'***

[2] *https://en.wikipedia.org/wiki/LendingClub*, ***'Πληροφορίες για το LendingClub'***

[3] *https://www.kaggle.com/code/faressayah/lending-club-loan-defaulters-prediction*, ***'Πληροφορίες για το σύνολο δεδομένων'***

*[4]*https://eclass.uoa.gr/modules/document/file.php/DI367/%CE%A5%CE%BB%CE%B9%CE%BA%CF%8C/PCA_method.pdf, ***'Πληροφορίες για το αλγόριθμο (PCA)'***

[5] *https://ann-chung-porfolio.webflow.io/work/predictive-analysis*, ***'Εικόνα credit Risk Prediction'***

[6] *https://www.analyticsvidhya.com/blog/2021/05/in-depth-understanding-of-confusion-matrix/*, ***'Πληροφορίες για τον πίνακα σύγχυσης'***

[7] A. Géron (2019), ***'Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow'***

[8] *https://androidkt.com/how-to-scale-data-to-range-using-minmax-normalization/*, ***'Πληροφορίες για τον minmax Scaler'***

[9] *https://apps.dtic.mil/sti/citations/ADA164453*, ***'Αρχική δημοσίευση πάνω στον αλγόριθμο Backpropagation '***

[10] *https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html*, ***'Πληροφορίες για τα δένδρα αποφάσεων'***

[11] *https://www.nvidia.com/en-us/glossary/data-science/xgboost/*, ***'Πληροφορίες για το αλγόριθμο XGBOOST'***

[12] Κ. Ζοπουνίδης (2001), ***'Ανάλυση Χρηματοοικονομικών Αποφάσεων με Πολλαπλά Κριτήρια'***

[13] Φ. Καλφάογλου, ***'Υπόδειγμα Μέτρησης Πιστωτικού Κινδύνου'***

[14] *https://www.geeksforgeeks.org/ml-handling-imbalanced-data-with-smote-and-near-miss-algorithm-in-python/*, ***'Πληροφορίες για τον αλγόριθμο SMOTE'***

[15] A. Namvar, M. Siami F. Rabhi, M. Naderpour, ***'Credit Risk Prediction in an Imbalanced Social Lending Environment'***

[16] Y. Guo, W. Zhou, C. Luo, C. Liu, H. Xiong, ***'Instance-based credit risk assessment for investment decisions in P2P lending'***

[17] M. Malekipirbazari, V. Aksakalli, ***'Risk assessment in social lending via random forests'***

[18] Y. Li, ***'Credit Risk Prediction Based on Machine Learning'***

[19] ***https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/*, 'Binary Cross Entropy Loss Function'**

[20] https://github.com/psounis/legacy/blob/main/plh31/plh31.3.4.card.pdf, ***'Σημειώσεις κ. Ψούνη πάνω στο Backpropagation'***