Student's Name: **Li Tsz Wing**

Student ID: **19053779D**

Course No.: **AMA4951**

Supervisor Name: **Dr James Huang**

Capstone Project Title: **Logistic Learning Data Analysis Method And Its Applications To Business And Economics.**

Table of Contents

1. Abstract

1.1 Overview

Logistic regression is one of the most statistical machine learning methods for the prediction in classification used in business. Nowadays, many statistical software also provides the logistic regression function with the capability of modeling of more complex logistic regression. This paper will discover the theory behind the logistic regression.

Credit risk also called default risk refers to the probability of loss due to the counterparty who failure to meet contractual debt obligations, such as loan, bond, trad credit. It is determined by two components, first is the probability of default, and the other ones is the loss give default. This paper also will be discovered how to use the logistic regression the define the credit risk of the counterparty.

1.2 Literature Review

Logistic Regression History

Logistic function first appears in early 19th century, published by Pierre François Verhulst who was a Belgian mathematician and lived in Brussels in 1838, under the: "*Correspondance mathématique et physique* ". It only demonstrated some curve agrees well with the actual course of the population of France, Belgium, Essex and Russia, but he did not mention how he fitted the curves. In 1845, Verhulst published a more detailed version under the name "*Proceedings*". It contains the function and the properties of the curves and Verhulst names it logistic. After Verhulst's initial discovery of logistic function, the most notable discoveries were the probit model developed by Chester Ittner Bliss and maximum likelihood estimation developed by Ronald Fisher in 1934 to 1935.

The discovery of Verhulst was repeated by a biologist Raymond Pearl and mathematician Lowell J. Reed in 1920. At that time, Pearl and Reed were unaware of Verhulst's work and they arrived independently at a close variant of the logistic model curve. In 1922, Du Pasquier, Professor of Mathematics at Neuchâtel informed Pearl and Reed of Verhulst's work. In 1943, It was the first known application of logistic model used by Worcester and Wilson in bioassay. In 1944, Joseph Berkson, really took logistic model well and following the probit terminology for many following years. However, logit was not going to be accepted initially, deemed as inferior to the probit model. In 1966 Cox step up for the logistic

regression applications with the logit model, he introduced multinomial logit model. In 1973 McFadden made a connection between multinomial logit model and discrete choice theory. Nowadays, logistic regression become one of the most common and useful machine learning algorithms.

Machine Learning

Machine learning is a subset of artificial intelligence and computer science focusing on the use of data to build the algorithm and imitate the learning process of humans, gradually improving its accuracy.

Zhou, Z.-H. (2021). stated that machine learning is to develop learning algorithms that build models from data. Through feeding the experience data into the learning algorithm to obtain a model that can make predictions on new observations. The machine learning is the subject of learning algorithms.

Credit Risk

Risk assessment the procedure known as screening is the standard for credit application. Nowadays, screening is done by scoring models that rely on historic data (*Altmanet al.2018, Thomas 2009*) especially with large-scale lending. Predictive value of the information that goes into the model is the main determinants of screening quality.

There are already lots of studies and well-established models for credit scoring, such as the (*Altman 1968*) for large corporates and (*Thomas 2009*) for consumers. The most of the different between two group is in the type of information that is used for prediction. Business models mainly rely on the financial accounts and market information data. Consumer models usually rely on personal credit history for examples the FICO score.

Python Programming

Python is a general-purpose programming language. Marvin (2018) mentioned that it is notorious for having a very simple "*pseudocode-like*" syntax that places emphasis on readability and expressiveness. Therefore, people can be easier to write the code and easier to maintain. Additionally, Python has a vast standard library that is augmented by an even larger array of third-party libraries. These are all developed and supported by Python's very active community.

The development is faster in Python rather other programming language as it is an

interpreted language. This means that the instructions are interpreted at runtime. Developer no need to pre-compile the code into machine language instructions making for faster experimentation and prototyping.

1.3 Objective

In this article, we define what is machine learning, define what is logistic regression, explain the theory behind logistic regression, showing others machine learning methods to make the classification decision, explore the application to business by logistic regression, discover the difference between logistic regression and other models.

2. Machine Learning

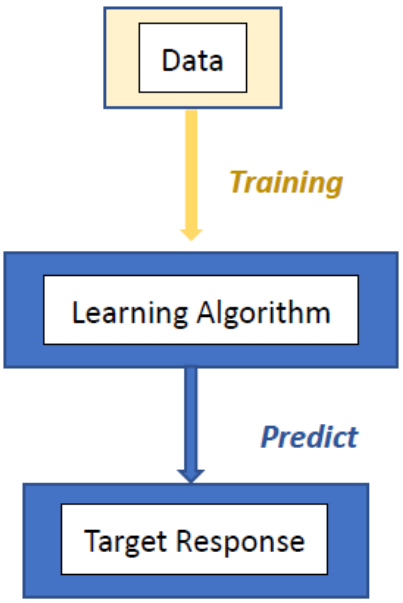
2.0 Background

Machine learning is a branch of computer science and artificial intelligence which focuses on the use of data and algorithms to imitate the way that humans learn and keep improving its accuracy. It through the use of statistical methods, algorithms are trained to make classifications or predictions, and to discover key insights in data mining projects. These insights subsequently drive decision making within applications and businesses.

2.1 Unsupervised Learning And Supervised Learning

2.1.1 Unsupervised Learning

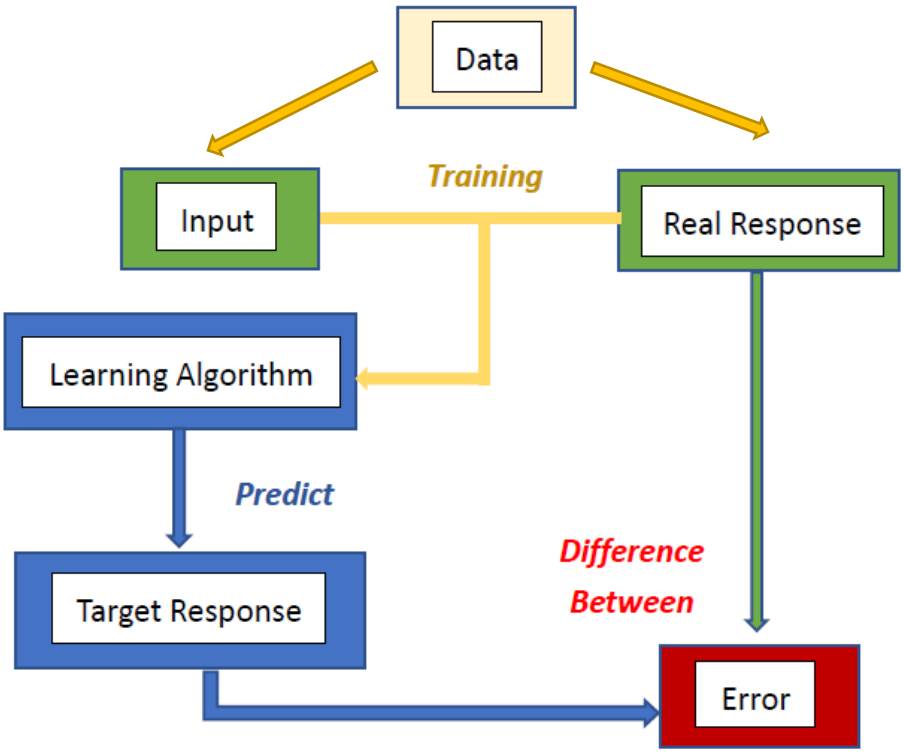
In unsupervised learning, also called clustering or self-organized learning, the class labels of training data are unknown, there are no external sources of information



guiding to build the learning algorithm. Given a set of observations and measurements with the aim of establishing the existence of classes or cluster in the data.

2.1.2 Supervised Learning

In supervised learning, also called classification or active learning, the training data are accompanied by labels indication the class of the observations. New data is classified based on the training set. In this paper, the data of the customer is the input, such as income, age. The assigned credit classes are the output. Supervised learning making the own algorithm to make the classification decisions and as a result to obtain the target responses. There is the difference in our algorithm between the target response and the real response called error. All supervised learning can minimize the error by tuning the algorithm parameters.



2.2 Generative And Discriminative Classifiers

2.2.1 Generative Classifiers

In general, a generative model models the probability distribution of each outcome. These models use the intuition of joint probability in theory: $P(outcomes|features)$. By using probability estimates and maximum likelihood function to model input data and distinguish between different class from the dataset. These models can generate new data instances, but presence of outliers affects these models to a significant extent. It mainly focuses on how the data was generated.

Naïve Bayes

The most common model in generative model is the Bayes Classifier.

Conditional Probability:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} ; P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Bayes' Theorem:

Let $B_1, \dots, B_n$ be a part of the sample space N, for any event A and any x = 1, … n:

$$P(B_x|A) = \frac{P(B_x \cap A)}{P(A)} = \frac{P(B_x)(P(A|B_x)}{P(B_1)(P(A|B_1) + \cdots + P(B_n)P(A|B_n)}$$

Let $X = \{x_1, \dots, x_r\}$ be the observations of attribute, $Y = \{y_1, \dots, y_n\}$ be the outcomes.

For example, x_1 can be the age, x_2 can be the height and $Y = \{Yes, No\}$

Bayes' Classifiers:

$$f(y_1, \dots, y_n) = \arg\max P(Y = y_n | X = (x_1, \dots, x_r))$$

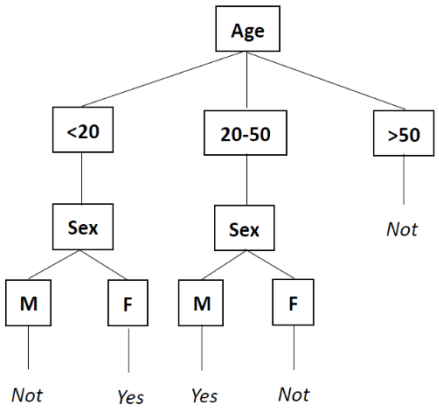
2.2.2 Discriminative Classifiers

Discriminative models, also called conditional models. Through learning the boundary between different classes or labels in the dataset, to make the classification decision, it is different from generative models. Therefore, the goal of discriminative classifiers is to find the decision boundary that can clearly separating one class from another. Discriminative models make no assumption about the data and not capable of generating new data instances. It carries a serious advantage. Comparing with generative models that discriminative models being more robust to outliers. It mainly focuses on predicting labels of the data.

Decision Tree

- A flow-chart like tree structure.
- Internal node denotes a test on an attribute.
- Branch represents an outcome of the test.
- Leaf nodes represent class labels or class distribution.

Decision tree generation consists of two phases. First tree construction, at start all the training examples are at the root and partition examples recursively based on selected attributes. Second, tree pruning, identify and remove branches that reflect noise or outliers. Example of a decision tree, if we need to predict people who like the game or not:



It represents the knowledge in the form of IF-THEN rules. One rule is created for each path form the root to a leaf and each attribute-value pair along a path forms a conjunction. The leaf node holds the class prediction and rules easier for human to understand.

Example:

IF Age > 50, THEN he/she does not like the game

IF Age < 20, AND Sex = F, THEN she likes the game

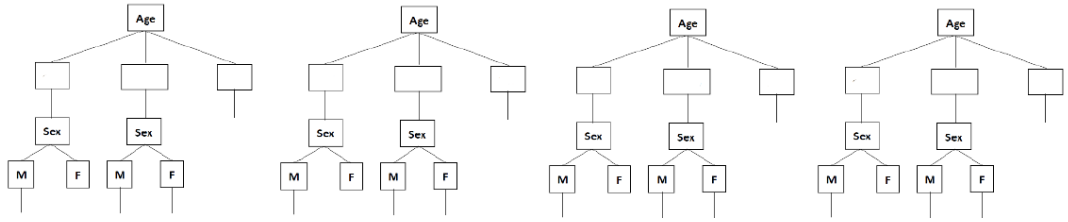
Ensemble Learning

Ensemble learning involves combining multiple prediction models to improve accuracy. A group of weaker models can be combined to form a stronger model with less bias or variance. This technique can be applied to logistic regression.

Bagging-Random Forest

Random Forest is an ensemble learning technique that uses many decision trees. Bagging is used to randomly choose samples of training data to estimate a population parameter. The low correlation between each decision tree is the key to Random Forest's accuracy. It allows for ensemble predictions that are more accurate than individual predictions.

Here are different decision tree models:



Bagging

There are three main approaches for bagging to choose the best outcome:

1. *Max voting*: Each decision tree makes a prediction and votes for each sample. The predicted class with the highest votes is considered being the final prediction. It is mainly used for classification problems.
2. *Average*: Predictions are extracted from each base models and an average among all the predictions are used to make the final prediction. It is mainly used for regression problems.

3. Logistic Regression

Generalized Linear Models

Generalized Linear Models are a generalization of the abilities and concept of regular linear regression models. Logistic regression is one of the transformations of the linear regression model. Therefore, in the logistic regression part below, also will discuss about the theories in linear regression.

3.0. Background

Linear regression is the basic form of the general regression, which is an approach to make the model by the relationship between input variables $x_1, x_2, \dots, x_n$ and continuous variables y.

- $\hat{y}$ is the outcome or called the dependent variable.
- b is the constant term.
- $w_1, w_2, \dots w_n$ are the weight or parameters, also called the effects.
- $x_1, x_2, \dots, x_n$ are the attributes or features of the data.
- ε is the error term.

Sum of weighted features:

$$\hat{y} = b + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + \cdots + w_nx_n + \varepsilon$$

$$\hat{y} = (\sum_{i=1}^{n} w_ix_i) + b$$

Here is the dot product notation, the sum of products of the corresponding numbers of each vector written as $w \cdot x$, so that the function become:

$$\hat{y} = w \cdot x + b$$

Squared-error loss (SSE):

$$\varepsilon(W) = (y - \hat{y})^2$$

$$\varepsilon(W) = \frac{1}{n}\sum_{i=1}^{n}(y_i - b - W_iX_i - \cdots - W_nX_n)^2$$

$$\varepsilon(W) = (Y - XW)^T(Y - XW)$$

SSE is the approach to find the best regression line by minimizing the distances between the observed values from the regression line.

To obtain ordinary least square error by taking the derivative of the loss function:

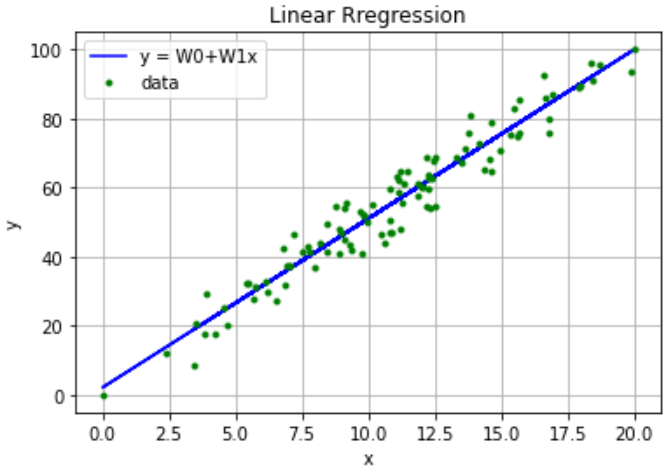
$$\widehat{w} = \arg\min \frac{1}{n}\sum_{i=1}^{n}(y_i - b - W_i X_i - \cdots - W_n X_n)^2$$

$$\frac{\partial \varepsilon(W)}{\partial w} = < Y, Y > -2 < Y, WX > + < WX, WX >$$

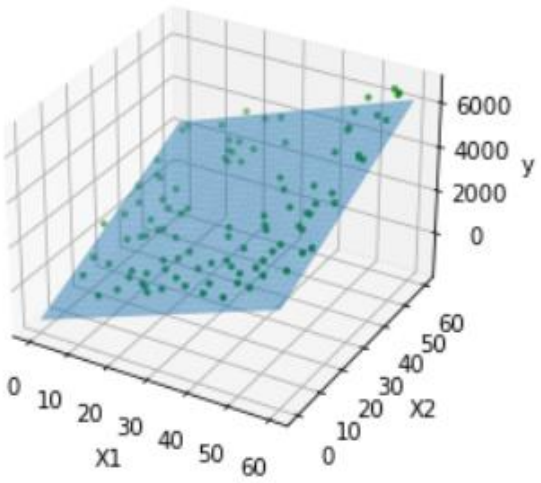
$$0 = 0 - 2X^T Y + 2(X^T X)W$$

The least square estimate of $\widehat{w} = (X^T X)^{-1} X^T Y$

Simulation part in 2D:



Simulation part in 3D:



Probability And Odds

Probability is the ratio between the number of events and the total number of events, which is constrained between zero and one. Odds are the ratio between different probabilities, which is constrained between zero and positive infinity. Odds ratio is the ratio between different odds. A large odds ratio represents a small probability and vice-versa.

Probability:

$$\mathrm{P}(\text{outcome}) = \frac{\text{number of that paricular outcome}}{\text{Total number of all oucome}}$$

Odds:

$$\text{Odds} = \frac{\mathrm{P}(\text{outcome})}{1 - \mathrm{P}(\text{outcome})}$$

Odds ratio, let s be the different outcomes will happen:

$$Odds\ ratio = \frac{\text{Odds}_1}{\text{Odds}_2}$$

Logistic Regression

Logistic regression is similar with linear regression, which is also an approach to make the model by the relationship between input variables $x_1, x_2, \dots, x_n$but discrete variables y.

Log Odds

Log odds is the value which finding the odds ratio and taking the logarithm.

$$\log\left(\frac{\mathrm{P}(\text{outcome})}{1 - \mathrm{P}(\text{outcome})}\right) = \log(\text{Odds})$$

$$\frac{\mathrm{P}(\text{outcome})}{1 - \mathrm{P}(\text{outcome})} = e^{\log(Odds)}$$

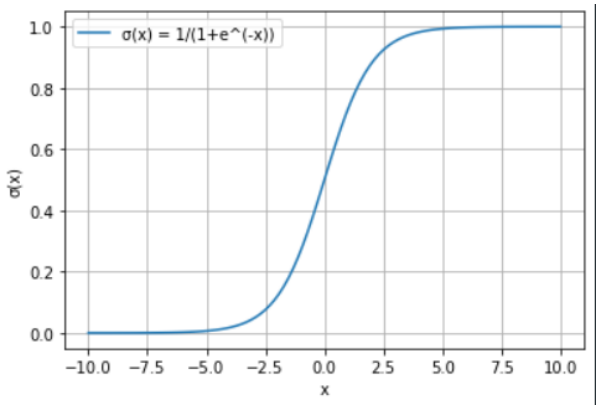
$$\mathrm{P} = \frac{e^{\log(Odds)}}{1 + e^{\log(Odds)}}$$

3.1 Sigmoid Function

Sigmoid classifier can train our observation into binary decision to fulfill our goal for binary logistic regression.

Sigmoid Function (Also called Logistic Function):

$$\sigma(x) = \frac{1}{(1 + e^{-x})}$$



Sigmoid function can change the real-valued number into range [0,1]. It is a likely linear function squashed toward 0 to 1. Also, it has a monotonically increasing properties, so that it can be usable to derive classification rules. By applying sigmoid function to the sum of weighted features of logistic regression, the sum of weighted features can change into a number between 0 to 1.

$$P(y = 1) = \frac{1}{1 + e^{-(w \cdot x + b)}} = \sigma(w \cdot x + b)$$

$$P(y = 0) = \frac{e^{-(w \cdot x + b)}}{1 + e^{-(w \cdot x + b)}} = 1 - \sigma(w \cdot x + b)$$

$$P(y = 1) + P(y = 0) = 1$$

Also, sigmoid function has the property:

$$\sigma(-x) = 1 - \sigma(x)$$

Therefore: $P(y = 0) = \sigma(-(w \cdot x + b))$

3.2 Likelihood Function

In linear regression, sum squared error (SSE) is very popular as a cost function. If we apply SSE into logistic regression:

$$\varepsilon(w) = (y - \frac{1}{1 + e^{-(w \cdot x + b)}})^2$$

The complexity and non-linearity of this loss function make it difficult to find a possible solution. In addition, SSE has many minimums, it also impossible in the next step (Gradient descent). Therefore, in logistic regression, maximum likelihood estimation will be more suitable. Maximum likelihood estimation was proposed by statistician Sir R. A. Fisher between 1912 and 1922, it is still popular statistical methods used to fit a mathematical model.

Probability of a training data:

$$P(y_n|x_n; w) \begin{cases} \sigma(w^T x_n) \ \ if \ y_n = 1 \\ 1 - \sigma(w^T x_n) \ otherwise \end{cases}$$

By using the fact that y_n *is either 1 or 0*

$$P(y_n|x_n; w) = \sigma(w^T x_n)^{y_n}(1 - \sigma(w^T x_n))^{1-y_n}$$

Log-likelihood of the training data S:

$$P(S) = \prod_{i=1}^{n} P(y_n|x_n; w) = \prod_{i=1}^{n} \{\sigma(w^T x_n)^{y_n}(1 - \sigma(w^T x_n))^{1-y_n}\}$$

$$\log P(S) = \sum_{i=1}^{n} y_n \log \sigma(w^T x_n) + \sum_{i=1}^{n} (1 - y_n) \log(1 - \sigma(w^T x_n))$$

After adding up with its negation, it will be more convenient for next steps. And it became the error function of logistic regression, it also called cross-entropy error function:

$$\varepsilon(w) = -\log P(S) = -\sum_{i=1}^{n} \{y_n \log \sigma(w^T x_n) + (1 - y_n) \log(1 - \sigma(w^T x_n))\}$$

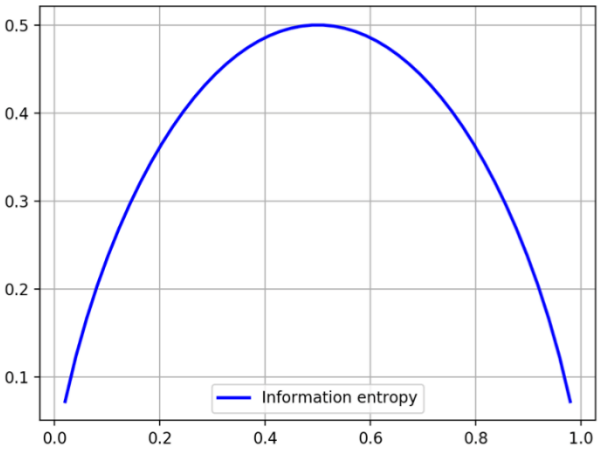
3.3 Cross Entropy Loss

Cross entropy loss is popular loss function for the classification problem, also called log loss or logistic loss. The aim is to minimize the loss during training. Another loss function is the mean squared error for the linear regression problem.

Let k be the number of outcomes and p_i be the probability getting belong the class $i \in \{1, \dots, k\}$:

$$Cross\ Entropy\ Loss = -\sum_{i=1}^{n} p_i log_k p_i$$

The higher the value of entropy, the higher the uncertainty for probability distribution and the smaller the value the less the uncertainty.
Let $k = 2$ and $p_1 = p, p_2 = 1 - p$:



By observing the graph, when $p_1 = p_2 = 0.5$, it will arrive the highest information entropy, it means the least of the uncertainty in this case.

Nowadays, we have lots type of loss function for logistic regression, this is due to the simplification process of cross-entropy loss. Here is one of the changes from original loss function and it will use in one of the optimizations approaches in below.

Recall that our loss function in Logistic regression:

$$\varepsilon(w) = -\sum_{i=1}^{n} \{y_n \log \sigma(w^T x_n) + (1 - y_n) \log(1 - \sigma(w^T x_n))\}$$

Sometimes some researchers will change the cross-entropy loss by the step:

$$\varepsilon(w) = -\sum_{i=1}^{n} \{y_n \log \sigma(w^T x_n) + (1 - y_n) \log(1 - \sigma(w^T x_n))\}$$

$$\varepsilon(w) = -\sum_{i=1}^{n} y_n (\log \sigma(w^T x_n) - \log\big(\sigma(-w^T x_n)\big)) + \log(\sigma(-w^T x_n))$$

$$\varepsilon(w) = -\sum_{i=1}^{n} y_n \log \frac{\sigma(w^T x_n)}{\sigma(-w^T x_n)} + \log(\sigma(-w^T x_n))$$

$$\varepsilon(w) = -\sum_{i=1}^{n} y_n \log \frac{1 + e^{w^T x_n}}{1 + e^{-w^T x_n}} + \log(\sigma(-w^T x_n))$$

$$\varepsilon(w) = -\sum_{i=1}^{n} y_n \log \frac{e^{w^T x_n}(e^{-w^T x_n} + 1)}{1 + e^{-w^T x_n}} + \log(\sigma(-w^T x_n))$$

$$\varepsilon(w) = \sum_{i=1}^{n} -y_n w^T x_n + \log(1 + e^{w^T x_n})$$

If instead we are to use the labels $y = 1$ or $y = -1$:

$$\varepsilon(w) = \sum_{i=1}^{n} -y_n w^T x_n + \log(1 + e^{w^T x_n}) \equiv \sum_{i=1}^{n} \log(1 + e^{-y_n w^T x_n})$$

3.4 Convexity Of Logistic Regression Loss Function

There are 4 ways to prove the function convexity:

(1). ***Definition of a convex function:***

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \; where \; \forall \, x, y \; \in \; \mathbb{R} \; and \; \alpha \in [0,1]$$

(2). ***First order condition of convexity****:*

$$f(y) \geq f(x) + \nabla_x^T f(x)(y-x) \; where \; \forall \, x, y \; \in \; \mathbb{R}$$

(3). ***Second order condition of convexity:***

$$z^T \; \nabla_x^2 f(x) z \geq 0 \; where \; \forall z \; and \; \nabla_x^2 f(x) \; is \; the \; hessian \; matrix$$

(4). ***Linear combination more than one convex function, when*** $\boldsymbol{f(x) and \; g(x)}$ ***are two convex functions:***

$$(\theta_1 f + \theta_2 g)(x) = \theta_1 f(x) + \theta_2 g(x)$$

Recall that our loss function in Logistic regression:

$$\varepsilon(w) = -\sum_{i=1}^{n} \{y_n \log \sigma(w^T x_n) + (1-y_n) \log(1-\sigma(w^T x_n))\}$$

And split it back into two parts:

$$-\sum_{i=1}^{n} \{y_n \log \sigma(w^T x_n)\} \; and \; -\sum_{i=1}^{n} \{(1-y_n) \log(1-\sigma(w^T x_n))\}$$

Because y_n will be always positively. By the linear combination theory of convex function (4), after proving $-\log \sigma(w^T x_n) \; and \; -\log(1-\sigma(w^T x_n))$ are convex, the whole loss function also convex.

3.5 cont.

(1) Proving $-\log \sigma(w^T x_n)$ is convex:

$$-\log \sigma(w^T x_n) = -\log\left(\frac{1}{1+e^{-(w\cdot x+b)}}\right) = \log(1+e^{-(w\cdot x+b)})$$

First order:

$$\nabla_w\{\log(1+e^{-(w\cdot x+b)})\} = x\left(\frac{-e^{-(w\cdot x+b)}}{1+e^{-(w\cdot x+b)}}\right) = x(\sigma(w\cdot x+b)-1)$$

Second order:

$$\begin{aligned}\nabla_w^2\{\log(1+e^{-(w\cdot x+b)})\} &= \nabla_w\{x(\sigma(w\cdot x+b)-1)\}\\ &= \left(\frac{1}{1+e^{-(w\cdot x+b)}}\right)\left(1-\frac{1}{1+e^{-(w\cdot x+b)}}\right)xx^T\\ &= \{\sigma(w\cdot x+b)(1-\sigma(w\cdot x+b)\}xx^T\end{aligned}$$

Based on this hessian matrix, where $\sigma(w\cdot x+b)$ between 0 to 1:

$$\begin{aligned}\forall z: z^T\nabla_w^2\{-\log(1+e^{-(w\cdot x+b)})\}z &= z^T\{\sigma(w\cdot x+b)(1-\sigma(w\cdot x+b)\}xx^Tz\\ &= \{\sigma(w\cdot x+b)(1-\sigma(w\cdot x+b)\}(x^Tz)^2 \geq 0\end{aligned}$$

(2) Proving: $-\log(1-\sigma(w^T x_n))$ is convex:

$$-\log(1-\sigma(w^T x_n)) = -\log\left(\frac{e^{-(w\cdot x+b)}}{1+e^{-(w\cdot x+b)}}\right) = w^Tx + \log(1+e^{-(w\cdot x+b)})$$

First order:

$$\nabla_w\{-\log(1-\sigma(w^T x_n))\} = x + \nabla_w\{\log(1+e^{-(w\cdot x+b)})\}$$

Second order:

$$\begin{aligned}\nabla_w^2\{-\log(1-\sigma(w^T x_n))\} &= \nabla_w\left\{x+\nabla_w\{\log(1+e^{-(w\cdot x+b)})\}\right\}\\ &= \nabla_w^2\log(1+e^{-(w\cdot x+b)})\end{aligned}$$

Using the fact in part (1) hessian matrix part, $\nabla_w^2\{\log 1+e^{-(w\cdot x+b)}\}$ is positive semi-definite. $\nabla_w^2\{-\log(1-\sigma(w^T x_n))\}$ also is positive semi-definite.

As proved $-\log \sigma(w^T x_n)$ and $-\log(1-\sigma(w^T x_n))$ are convex function, so that the cross-entropy loss function of logistic regression is convex.

3.5 Optimization Methods

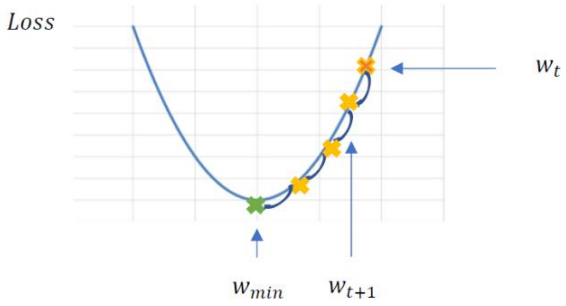
3.5.1 Gradient Descent

Gradient descent is a way to find the optimal weights through minimize the loss function with partial derivatives. This algorithm can be applied to function with these properties:

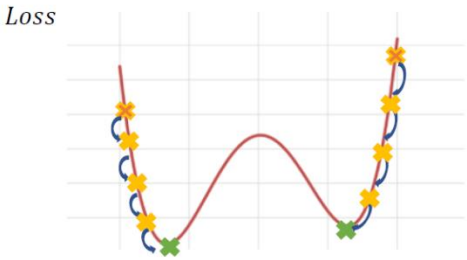
- $f(x)$ is continuous and differentiable
- $f(x)$ is convex function

$$w_{i+1} = w_i - \alpha_i \frac{d\varepsilon}{dw} w_i$$

Gradient descent is a method by figuring out which direction the slope of the function is rising the most steeply and moving in the opposite direction by a certain number α which is called learning rate in machine learning.



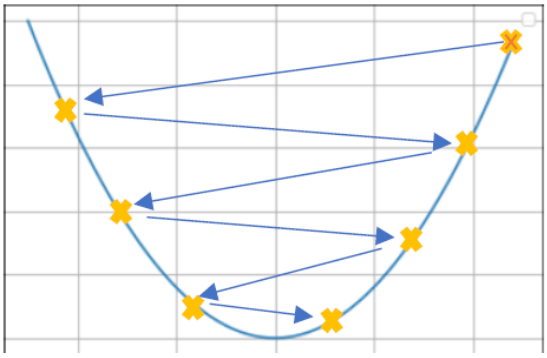
When the function is convex, using the gradient descent can get the global optimum.



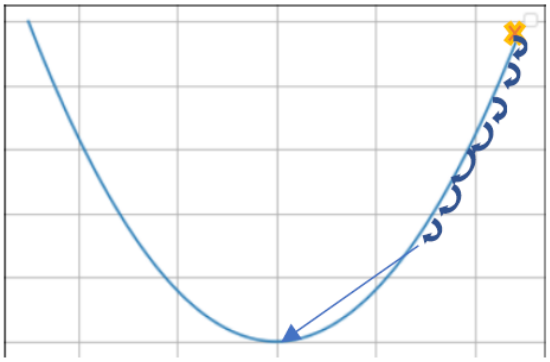
When the function is non-convex, there exist more than one optimum, gradient descent is not possible.

3.5.1 cont.

There should be a proper learning rate before gradient descent.



When there is a large learning rate, the learning path will be changing different direction after one iteration, it may never converge.



When there is a large learning rate, the learning path will be kept one direction and having a better result in the accuracy, but it may use lots of the computational resources.

3.5.1 cont.

Procedure of Gradient Descent:

Recall the loss function (negative log-likelihood):

$$\varepsilon(w) = -\log P(S) = \sum_{i=1}^{n} \{y_n \log \sigma(w^T x_n) + (1 - y_n) \log(1 - \sigma(w^T x_n))\}$$

Computing the gradient for minimizing w:

$$\frac{\partial \varepsilon(w)}{\partial w} = \sum_{i=1}^{n} \left\{ y_n \frac{\sigma(w^T x_n)(1-\sigma(w^T x_n))}{\sigma(w^T x_n)} x_n - (1 - y_n) \frac{\sigma(w^T x_n)(1-\sigma(w^T x_n))}{1-\sigma(w^T x_n)} x_n \right\}$$

$$\frac{\partial \varepsilon(w)}{\partial w} = \sum_{i=1}^{n} \{y_n(1 - \sigma(w^T x_n))x_n - (1 - y_n)\sigma(w^T x_n)x_n\}$$

$$\frac{\partial \varepsilon(w)}{\partial w} = \sum_{i=1}^{n} \{\sigma(w^T x_n) - y_n\}x_n$$

Stochastic gradient descent, by choosing a proper step size $\alpha > 0$, and keep update the parameters until it converges by:

$$w_{t+1} = w_t - \alpha \sum_{i=1}^{n} \{\sigma(w^T x_n) - y_n\}x_n$$

3.5.2 Newton's Method

We proved logistic regression is a convex optimization problem in part **3.4** so that we can also apply Newton's Method. Newton's Method is an iterative optimization method which finding the roots of a polynomial function.

Procedure of Newton's Method:

If we only have one feature: (1 input variable)

$w_{n+1} = w_n + \frac{\varepsilon(w)}{\varepsilon'(w)}$ loop until $w_n - w_{n+1} \approx 0$

If we have more than one feature (> 1 input variable), it would be requiring the Hessian and gradient matrix:

$w_{n+1} = w_n + {H_{\varepsilon(w)}}^{-1} \nabla\varepsilon(w)$ loop until $w_n - w_{n+1} \approx 0$

where $\nabla\varepsilon$ is the gradient of the loss function, its vector of partial derivatives $\begin{bmatrix} \frac{\partial\varepsilon}{\partial w_1} \\ \vdots \\ \frac{\partial\varepsilon}{\partial w_n} \end{bmatrix}$

and H is the Hessian of the loss function, its matrix of second partial derivatives.

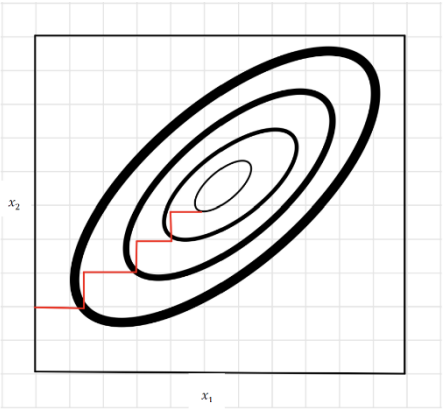
$$H = \begin{pmatrix} \frac{\partial^2\varepsilon}{\partial {w_1}^2} & \frac{\partial^2\varepsilon}{\partial w_1 \partial w_2} & \cdots \\ \frac{\partial^2\varepsilon}{\partial w_2 \partial w_1} & \frac{\partial^2\varepsilon}{\partial {w_2}^2} & \vdots \\ \vdots & \cdots & \ddots \end{pmatrix}$$

3.5.3 Coordinate Descent

To compare gradient descent and coordinate descent gradient descent uses the gradient of the loss function and coordinate descent uses only the original loss function. When there is a loss function $\varepsilon(w)$ with more than one parameter,

$w = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$ then coordinate descent begins at some point ($W_1 \dots W_n$) and then starts to search for a minimum first along the W_1 direction. Then update the parameter W_1, and search for a minimum along the W_2 searches along the W_2 direction. Loop this process until finishes all parameter searching. It is extremely simple to implement.

When we have a loss function $f(x_1, x_2)$, the optimization procedure will be as below.



3.6 Binomial Logistic Regression

3.6.1 Convert Multinomial To Various Binomial

The original logistic regression can only solve the binary prediction problem. There are two common methods can change original logistic regression to many logistic regression models to predict multiple outcomes.

The first common method is **one-vs-rest classifier.** In the training part **one-vs-rest** will build multiple classifiers that separate the data into two different groups, the first group would be our objective outcome and the second would be the remaining outcomes. Therefore, if we have **N** number of the outcome, then we have the number of **N** classifiers. Finally, put all the data into those classifiers and based on the highest accuracy score classifier to predict the outcome.

- If we have outcomes [A, B, C],
- The first classifier A is the first group, and B&C be the second group.
- Second classifier B be the first group, and A&C be the second group.
- Third classifier C is the first group, and A&B be the second group.

The second common method is **multinomial classifier.** It means **many-vs-many** classifiers**.** It is like **one-vs-rest,** but it only selects two outcomes to build multiple classifiers in the training part. Therefore, if we have the **N** number of the outcome, we should have $\boldsymbol{C_2^N}$ classifiers. Finally, put all the data into those classifiers and based on the votes from all the classifiers predict the outcome.

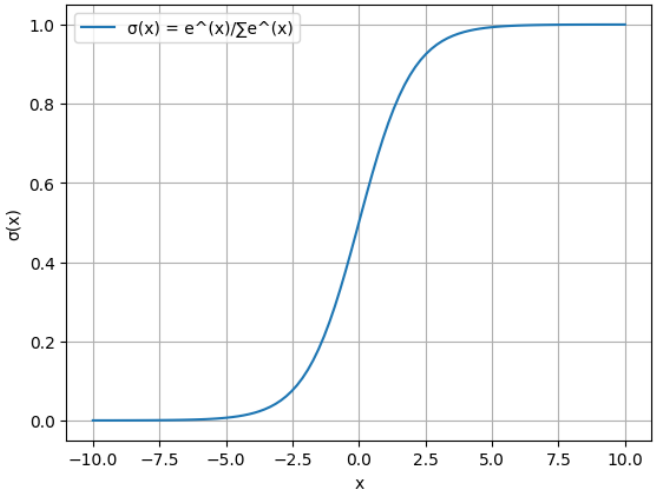
- If we have outcomes [A, B, C, D],
- We would have the classifiers [A&B], [A&C], [A&D], [B&C], [B&D], [C&D]

3.6.2 Softmax Function

Softmax function like the sigmoid function. It is the main element in multinomial logistic regression which can output an input value into a vector of values that follows a probability distribution within [0,1] and total sums up to 1.

Softmax Function:

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$



The output of the softmax function is a vector:

$$softmax(x) = \left[\frac{e^{x_1}}{\sum_{j=1}^{n} e^{x_j}}, \frac{e^{x_2}}{\sum_{j=1}^{n} e^{x_j}}, \frac{e^{x_3}}{\sum_{j=1}^{n} e^{x_j}}, \dots, \frac{e^{x_n}}{\sum_{j=1}^{n} e^{x_j}}\right]$$

By applying sigmoid function to the sum of weighted features of logistic regression:

$$\sigma(w \cdot x + b) = \frac{e^{w \cdot x}}{\sum_{j=1}^{n} e^{w \cdot x_j}}$$

<u>The relationship between sigmoid function and softmax function</u>

If we only have two outcomes of our data:

$$Y_i = \frac{e^{w \cdot x}}{e^{w \cdot x_1} + e^{w \cdot x_2}}, i = 1 \; or \; 2$$

$$Y_1 = \frac{1}{1 + e^{w \cdot x_2}}, Y_2 = \frac{e^{w \cdot x_2}}{1 + e^{w \cdot x_2}}$$

$$P(y = 1) + P(y = 0) = 1$$

Those are same application of sigmoid function in logistic regression.

3.7 Collinearity

Collinearity or called multicollinearity occurred when two or more independent attributes included in the model are highly correlated with each other so that the values of one can be predicted by another variable. Because linear regression and logistic regression are based on linear transformation of the data. Collinearity will affect the standard error and variance of coefficient estimates and reduce the reliability of our model. To solve this problem, there are lots of way to solve this problem, such as **Variance Inflation Factors or Regularization, those will be mentioned below**.

3.8 Variance Inflation Factors (VIF)

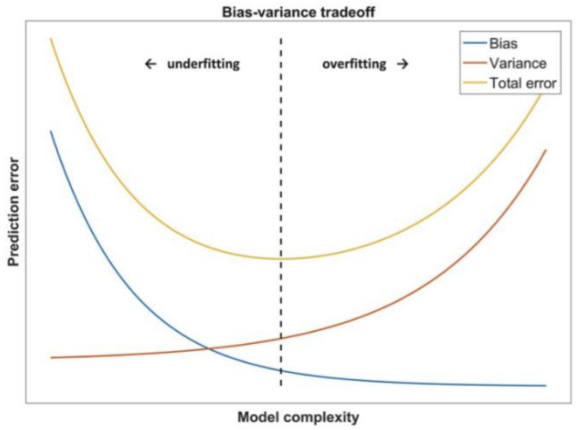
The Variance Inflation Factor (VIF) measures the severity of multicollinearity in regression. This statistical approach indicates the increase in the variance of a regression coefficient because of collinearity.

$$\mathrm{VIF_i} = \frac{1}{1 - \mathrm{R_i}^2}$$

- $\mathrm{R_i}^2$ is the R-squared value, it will be mentioned in the *part 4 Performance of logistic regression*.
- If the VIF = 1, which means that the variable is not correlated with the other.
- If the VIF between 1 and 5, which means variables are moderately correlated.
- If the VIF higher than 5, variables are highly correlated.

After finding the VIF, canceling or combine higher correlated attributes can be solve the multicollinearity problem.

3.9 Bias-Variance Trade-Off



The bias-variance trade-off is the property of the model when increasing the bias in the estimated parameters how can the variance of the parameter estimated across sample can be reduced.

- When there are less samples, then our estimated parameters will be less, it will have a large bias, it leads to underfitting.

- When there are more samples, then there will be more estimated parameters, it will fit our training data well while not fit the new data well, it will have a large variance, it leads to overfitting.

3.10 Regularization In Logistic Regression

Beside the VIF, regularization also can solve the collinearity problem. Moreover, the regression model is not the best before regularization due to the bias-variance trade-off problem. In regularization, λ is called the tuning parameter, which controls the strength of the penalty term.

Recall in linear regression, least square estimate of $\widehat{w}$*:*

$$\widehat{w} = \arg\min \frac{1}{n}\sum_{i=1}^{n}(y_i - b - W_i X_i - \cdots - W_n X_n)^2$$

Ridge linear regression:

$$\widehat{w}^{ridge} = \arg\min_{\beta \epsilon \mathbb{R}^n} \frac{1}{n}\sum_{i=1}^{n}(y_i - b - W_i X_i - \cdots - W_n X_n)^2 + \lambda \sum_{j=1}^{n} {w_j}^2$$

$$= \arg\min_{\beta \epsilon \mathbb{R}^n} \frac{1}{n}\|y - WX\|^2 + \lambda \|W\|^2$$

- $\lambda = 0$ it goes back ordinary least square estimate.
- $\lambda \to \infty$ $\widehat{w}^{ridge} = 0$

Lasso linear regression:

$$\widehat{w}^{lasso} = \arg\min_{\beta \epsilon \mathbb{R}^n} \frac{1}{n}\sum_{i=1}^{n}(y_i - b - W_i X_i - \cdots - W_n X_n)^2 + \lambda \sum_{j=1}^{n} |w_j|$$

$$= \arg\min_{\beta \epsilon \mathbb{R}^n} \frac{1}{n}\|y - WX\|^2 + \lambda \|W\|_1$$

- $\lambda = 0$ it goes back ordinary least square estimate.
- $\lambda \to \infty$ $\widehat{w}^{lasso} = 0$

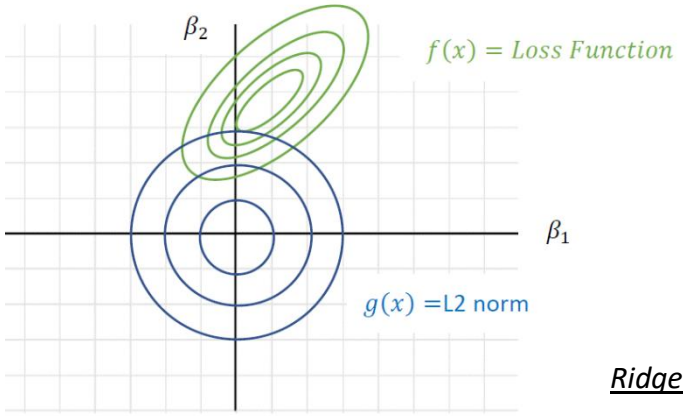
3.10 Cont.

Same trick in logistic regression with loss function plus the penalty

Ridge logistic regression:

$$\widehat{w}^{ridge} = \arg\min_{\beta\epsilon\mathbb{R}^n} \sum_{i=1}^{n} \{y_n \log\sigma(w^T x_n) + (1-y_n)\log(1-\sigma(w^T x_n))\} + \lambda \sum_{j=1}^{n} {w_j}^2$$

- $\lambda = 0$ it goes back original maximum likelihood function estimate.
- $\lambda \to \infty$ $\widehat{w}^{ridge} = 0$

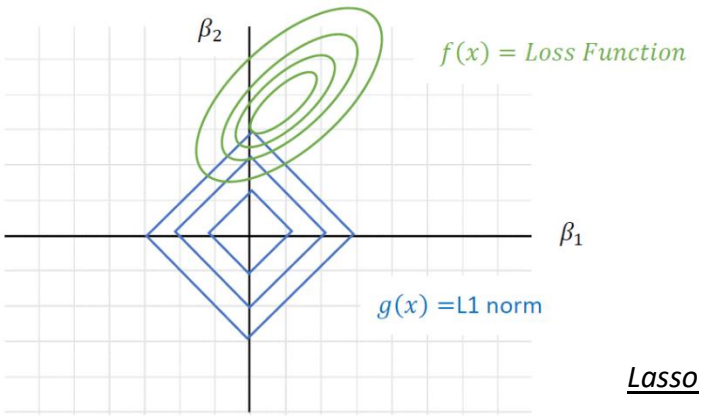


Ridge

Lasso logistic regression:

$$\widehat{w}^{lasso} = \arg\min_{\beta\epsilon\mathbb{R}^n} \sum_{i=1}^{n} \{y_n \log\sigma(w^T x_n) + (1-y_n)\log(1-\sigma(w^T x_n))\} + \lambda \sum_{j=1}^{n} |w_j|$$

- $\lambda = 0$ it goes back original maximum likelihood function estimate.
- $\lambda \to \infty$ $\widehat{w}^{lasso} = 0$



Lasso

3.10 Cont.

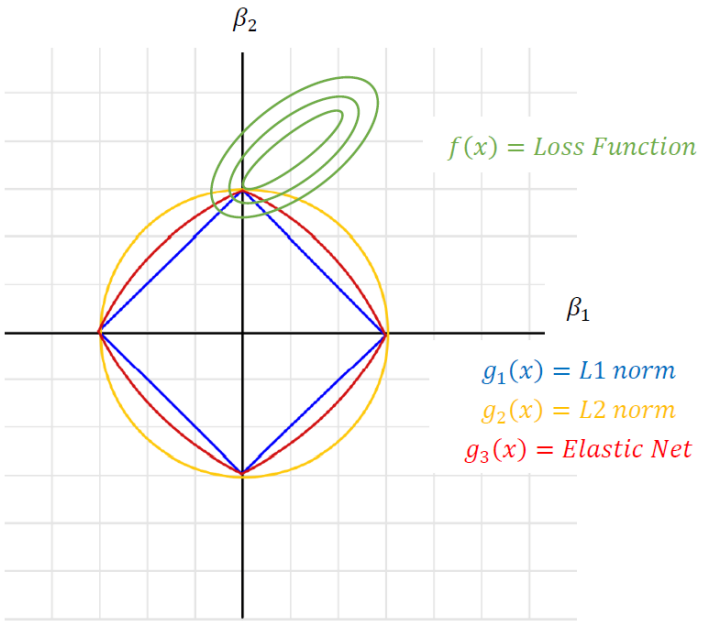
The differences between Ridge and Lasso are that lasso is adding the penalty term which is absolute sum of the coefficients while Ridge is adding the penalty term which is equal to the square of the coefficient. Therefore, compared with Ridge, Lasso tends to make coefficients to zero as compared to ridge, it can help features selection, but it might randomly choose one of the multicollinear variables and eliminate relevant independent variables, so that Lasso does not work very well with multicollinearity. On the other hands, Ridge does not eliminate the attributes in the model even if the variables are irrelevant.

There is the third way to penalize the logistic regression which called elastic net, by adding the l1 norm and l2 norm at the same time:

$$\widehat{w}^{elasic\ net} = \arg\min_{\beta \epsilon \mathbb{R}^n} \sum_{i=1}^{n} \{y_n \log \sigma(w^T x_n) + (1 - y_n) \log(1 - \sigma(w^T x_n))\} + \lambda_1 \sum_{j=1}^{n} {w_j}^2 + \lambda_2 \sum_{j=1}^{n} |w_j|$$

- $\lambda_1, \lambda_2 = 0$ it goes back original maximum likelihood function estimate.
- $\lambda_1 \to \infty$ or $\lambda_2 \to \infty$ then $\widehat{w}^{elasic\ net} = 0$

Elastic Net combines characteristics of both lasso and ridge. it reduces the impact of different attributes while not eliminating all of the features.



Elastic Net

4. Performance Of Logistic Regression

The table of errors, if the outcome belonging with positive or negative:

	Positive	**Negative**	**Total**
Positive	*True Positive*	*False Positive*	*Predicted Positive*
Negative	*False Negative*	*True Negative*	*Predicted Negative*
Total	*Actual Positive*	*Actual Negative*	*Total Population*

True Positive Rate $= \frac{True\ Positive}{Actual\ Positive}$, False Positive Rate $= \frac{False\ Positive}{Actual\ Negative}$

- The larger True Positive Rate: The better Classifier
- The smaller False Positive Rate: The better Classifier

The Receiver Operation Characteristics Curve

Recall that in logistic regression:

$$\sigma(w \cdot x + b) \geq 0.5 \;\; x \;\in Class\ 1$$
$$\sigma(w \cdot x + b) < 0.5 \;\; x \;\in Class\ 2$$

The 0.5 is called the threshold of the classifiers. However, the threshold of the classifier can be changed and not fixed with 0.5.

The criteria can be changed, γ can be any value between 0 to 1:

$$\sigma(w \cdot x + b) \geq \gamma \;\; x \;\in Class\ 1$$
$$\sigma(w \cdot x + b) < \gamma \;\; x \;\in Class\ 2$$

$\gamma = 0$, logistic regression predicts everything as class 1

- True Negative = False Negative = 0 -> True Positive Rate = False Positive Rate = 1

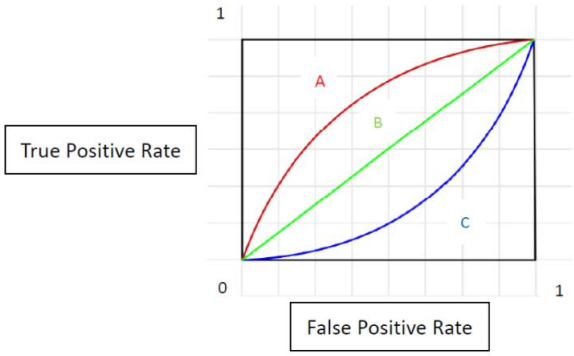
$\gamma = 1$, logistic regression predicts everything as class 0

- True Positive = False Positive = 0 -> True Positive Rate = False Positive Rate = 0

It hints that when threshold is smaller, there are more samples to be predicted as class 1, vice versa.

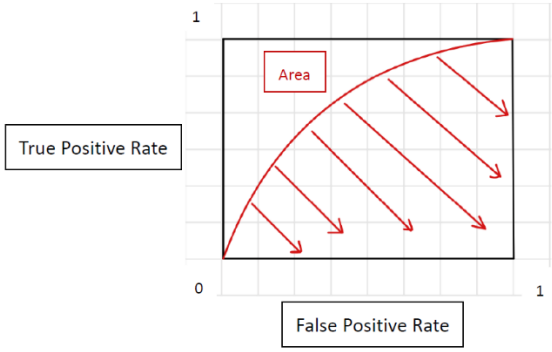
The Receiver Operation Characteristics Curved cont.

Receiver Operation Characteristics Curved (ROC):



ROC curve is (False Positive Rate (γ), True Positive Rate(γ)) for different threshold. It used to compare the performance of the classifier with different threshold or different classifiers. In the plot above, the curve A is the best threshold, and the C is the worst.

The Area under Receiver Operation Characteristics Curved (AUC)



AUC $\in [0,1]$ is defined as the area under the ROC curve, the larger AUC is, the better of the classifier perform.

The error rate can be changed into:

$$\text{True Positive Rate} = \text{P}(Y > \gamma \mid Y \in Actual\ Positive)$$

$$\text{False Posive Rate} = \text{P}(\sigma(w \cdot x + b) > \gamma \mid X \in Actual\ Negative)$$

The AUC formula:

$$\text{AUC} = \int_0^1 \text{True Posive Rate}\ \ \text{d}(\text{False Postive Rate})$$

$$\text{AUC} = -\int_{\infty}^{\infty} \text{P}(Y > \gamma \mid Y \in Actual\ Positive)\text{P}(\sigma(w \cdot x + b) > \gamma \mid X \in Actual\ Negative)\ \text{d}\gamma$$

R-squared for logistic regression

In linear ordinary least square regression, there is statistics measure the amount of variance explained by the regression model, call R^2 value. The R^2 is within from 0 to 1. Higher R-squared indicating the model having a precise accuracy.

$$R^2 = 1 - \frac{Sum\ Squared\ Regression\ (SSR)}{Total\ Sum\ Of\ Squares\ (SST)}$$

$$SSR = \sum (y_i - \hat{y}_i)^2 , SST = \sum (y_i - \bar{y})^2$$

- $\bar{y}$ is the expected value of the observed outcomes.

For logistic regression, there are lots of different way to calculate the R^2, for examples Efron's R^2, McFadden's R^2, McKelvey & Zavoina R^2 or Count R^2. In addition, there is no standard R^2 for logistic regression.

Efron's R^2:

$$R^2 = 1 - \frac{\sum (y_i - \tau_i)^2}{\sum (y_i - \bar{y})^2}$$

- τ is the predicted outcome probability.

McKelvey & Zavoina R^2:

$$R^2 = \frac{Var(\hat{y})}{Var(\hat{y}) + \frac{\tau^2}{3}}$$

Count R^2:

$$R^2 = \frac{Correct}{Total}$$

- Total is the total number of observations.
- Correct is the total number of correctly classified observations.

5. Data Preprocessing

5.1 Data Cleaning

In our real world, most of the data are “Dirty”. It means that the data may be lacking some of the attribute values or contain errors and outliers. If data is incorrect, algorithms become unreliable. Most of the common methods are to due missing data or outliers:

Cancellation

Ignore the attributes which have missing values or outliers. This can reduce the impact of the “Dirty” data, but also waste many data to build up a classifier.

Fill-in

Using mean, mode, class boundaries or IQR to fill in the missing data or outliers. It can maintain the completeness of the data, but it reduced the model's accuracy.

5.2 Data Encoding

One-Hot Code

One-hot encoder converts the categorical feature into numeric data by splitting the column into multiple columns.

	Class
0	Class A
1	Class B
2	Class C
3	Class D

Original Data frame

	Class_Class A	Class_Class B	Class_Class C	Class_Class D
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1

After One-Hot Encode Data frame

Label Encode

Label encoder converts each value in a column into a number.

	Class
0	Class A
1	Class B
2	Class C
3	Class D

Original Data frame

	Class
0	0
1	1
2	2
3	3

After Label Encode Data frame

5.3 Feature Selection

Hypothesis Testing

Different hypothesis tests for different type of data:

	Categorical Data	**Numerical Data**
Categorical Data	Chi-Square Test	ANOVA T-Test
Continuous Data	Regression	Corrleation Test

In logistic regression, mainly using ANOVA T-Test to find out the importance of the feature. It can prevent the collinearity and increase the accuracy of the model.

H0: The input feature is similar with all the outcomes.

H1: The input feature is different with different outcome groups.

Correlation Map

After plotting the correlation map, we can choose the higher covariance features with the outcome and ignore the features that collinear with itself.

5.4 Data Clustering

By using clustering, we can group the data with different clustering approach and create a new feature that can show the data structure for improving the machine learning models accuracy.

Partitioning algorithms:

Construct a partition of a database of **N** objects into a set of **K** clusters

- K-means: Each cluster is represented by the center of the cluster
- K-medoids: Each cluster is represented by one of the objects in the cluster

6. Implement of Logistic Regression In Python

6.1 Python

Python is a computer programming language that contains a corresponding set of software tools and libraries. It was developed by Guido van Rossum in the early 1990s. Compared with another programming, such as R or C++ language, Python was designed to be easy to read and learn. Therefore, Python programs just have a few unnecessary symbols, and they can use in straightforward English instead of complicated syntax.

The Python library is the main feature of this programming language. It contains bundles of code that can be used repeatedly and makes Python to be convenient for the programmer Python libraries play a very important role in the fields of data science, data visualization and machine Learning. Therefore, in this report, we use **Seaborn** and **Sckit-learn** libraries to do the analysis and build up the classifiers for the credit risk datasets.

6.2 Seaborn

Seaborn is a library created by Michael Waskom who is a PHD data scientist for making statistical graphics in Python. It builds on top of Pandas data structures and matplotlib. Seaborn helps users change data on data frames and arrays containing some necessary informative plots.

6.3 Sckit-learn

Scikit-learn is an open-source machine learning library created by Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel of INRIA in 2010 that supports supervised and unsupervised learning. It also provides various tools for, data preprocessing and model tuning so Sckit-learn is one of the most popular libraries for machine learning study nowadays and it will be the main library used in this report.

6.4 Hyper Tuning Parameter Of Logistic Regression

6.4.1 Penalty

There are **3** penalties we can choose in Sckit-learn package:
(More detail in **3.9 Regularization in Logistic Regression**)

'l1' mean we will us Lasso logistic regression.

'l2', mean we will use Ridge logistic regression.

'elasticnet' mean we will use Elastic Net which is combines characteristics of both lasso and ridge.

6.4.2 Multi_class

There are **3 multi_class** we can choose in Sckit-learn package:
(More detail in **4.1 Convert Multinomial to various Binomial**)

'ovr' means the one-vs-rest classifier**.** I

'multinomial' means many-vs-many classifiers**.**

'Auto' will select ***'ovr'*** if the data is binary, or if solver is the ***'liblinear'***, and otherwise selects ***'multinomial'***.

6.4.3 Solver

There are **5** solvers we can choose in Sckit-learn package:
(More detail in **3.6 Optimization Method in Logistic Regression**)

'newton-cholesky', mean using **Newton's Method** to optimize.

- Using **first and second partial derivatives** of the loss function.
- **Computationally expensive** caused by the Hessian Matrix.

'lbfgs', mean using **Newton's Method** to optimize.

- Using **first and second partial derivatives** of the loss function.
- Not compute all the data with l**imited memory.**
- Computationally **less expensive** comparing with original Newton's Method

'liblinear', mean using **Coordinate Descend** to optimize.

- Using the **original** loss function.
- **May get a not stationary point** if the function is not smooth.

'sag' and 'saga' means using **Gradient Descent** to optimize.

- Using the **stochastic gradient** descent of the loss function.
- **Fastest solver** comparing with the other for large data set.
- 'sag' and 'saga' are different from the **penalty term**.

The restriction of the solver:

	Solver				
Multi_class and penalties	*'newton-cholesky'*	*'lbfgs'*	*'liblinear'*	*'sag'*	*'saga'*
None	✓	✓	×	✓	✓
'ovr'+ L1	×	×	✓	×	✓
'multinomial' + L1	×	×	×	×	✓
'ovr'+ L2	✓	✓	✓	✓	✓
'multinomial'+ L2	✓	✓	×	✓	✓
Elastnet	×	×	×	×	✓

6.4.4 Class Weight

This parameter will be useful when the amount of data in each outcome is not equally distributed. It can prevent the model's exceptional accuracy score. For example, if we have 90% of data belong group A. Then the accuracy score must be very high due to the amount of data in group A. However, if we are adding some weighting to each outcome, we can know the real accuracy score of the model.

'balanced' will automatically add the higher weighting into the group of outcomes that only have a few numbers of data and add the lower weighting into the group of outcomes that have more amount of data in the training part.

Or we can manually set the weighting of the outcomes:

- If we have 10 outcomes belonging to group A and 90 outcomes belonging to group B.
- We can set 9:1 weighting into the data.

6.4.5 C

This parameter is used to be set the regularization strength by any number.

- Higher of the number, higher of the regularization strength, vice versa.

6.4.6 Max_iter

This parameter is set the max iterations taken for the solvers to converge.

- If set 100 max_iter, then it will stop in 100^{th} time even if have not converge.

7. Simulation Data Sets For Models Comparison

In this part, we will use two simulated datasets, one is German credit rating and another one is US corporate credit rating dataset. Here, we will discover all the parts of building the classifiers and find out the pros and cons of logistic regression encountering credit rating prediction.

7.1 German Binary Credit Risk Dataset

7.1.1 Data Set Background

This data set is simulated and provided by Professor Dr Hans Hofmann in 2000. Each of the data represents a germen who takes credit or borrowed money from a bank. Each person is classified as good or bad credit rating according to the set of features. Here are the list of **1000** credit ratings issued by major agencies in 1994.
There are 21 attributes for every person. They can be divided in:

Basic Information:

Sex & Marital Status	*Age (years)*	*Telephone*
Foreign Worker	*Duration in Current address*	*No of dependents*

Financial Background:

Account Balance	*Value Savings/Stocks*	*Length of current employment*
Most valuable available asset	*Type of apartment*	*Occupation*

Credit related features:

Creditability	*Duration of Credit (month)*	*Payment Status of Previous Credit*
Purpose	*Credit Amount*	*Instalment per cent*
Guarantors	*Concurrent Credits*	*No of Credits at this Bank*

7.1.2 Data Cleaning

The sample data frame of this data set:

	Creditability	Account Balance	Duration of Credit (month)	Payment Status of Previous Credit	Purpose	Credit Amount	Value Savings/Stocks	Length of current employment	Instalment per cent	Sex & Marital Status	...	Duration in Current address	Most valuable available asset	Age (years)	Concurren Credit
0	1	1	18	4	2	1049	1	2	4	2	...	4	2	21	
1	1	1	9	4	0	2799	1	3	2	3	...	2	1	36	
2	1	2	12	2	9	841	2	4	2	2	...	4	1	23	
3	1	1	12	4	0	2122	1	3	3	3	...	2	1	39	
4	1	1	12	4	0	2171	1	3	4	3	...	4	2	38	

5 rows × 21 columns

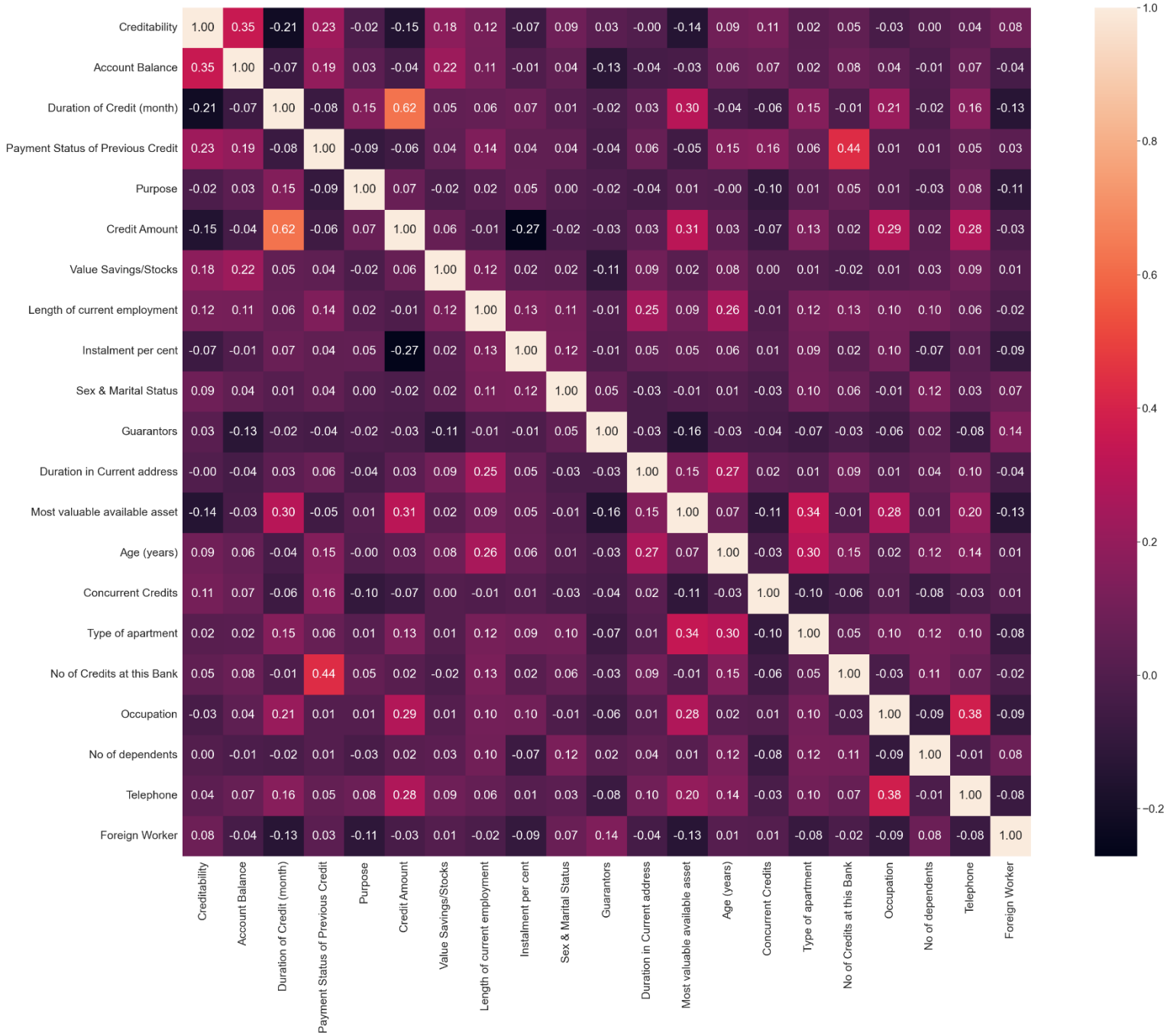
The data type of the data set:

```
Data columns (total 21 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Creditability                    1000 non-null   int64
 1   Account Balance                  1000 non-null   int64
 2   Duration of Credit (month)       1000 non-null   int64
 3   Payment Status of Previous Credit 1000 non-null  int64
 4   Purpose                          1000 non-null   int64
 5   Credit Amount                    1000 non-null   int64
 6   Value Savings/Stocks             1000 non-null   int64
 7   Length of current employment     1000 non-null   int64
 8   Instalment per cent              1000 non-null   int64
 9   Sex & Marital Status             1000 non-null   int64
 10  Guarantors                       1000 non-null   int64
 11  Duration in Current address      1000 non-null   int64
 12  Most valuable available asset    1000 non-null   int64
 13  Age (years)                      1000 non-null   int64
 14  Concurrent Credits               1000 non-null   int64
 15  Type of apartment                1000 non-null   int64
 16  No of Credits at this Bank       1000 non-null   int64
 17  Occupation                       1000 non-null   int64
 18  No of dependents                 1000 non-null   int64
 19  Telephone                        1000 non-null   int64
 20  Foreign Worker                   1000 non-null   int64
dtypes: int64(21)
memory usage: 164.2 KB
```

In this data set, all the data are non-null, so there is not missing value. However, all the features are already encoded with the number by Professor Dr Hans. To assign the data back to its original meaning is part of the process of data cleaning according to the document produced by Dr. Hans.

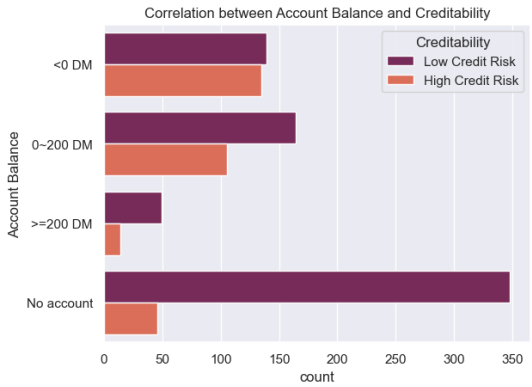
7.1.3 Exploratory Data Analysis (EDA)

Correlation map between all the features before encoding:



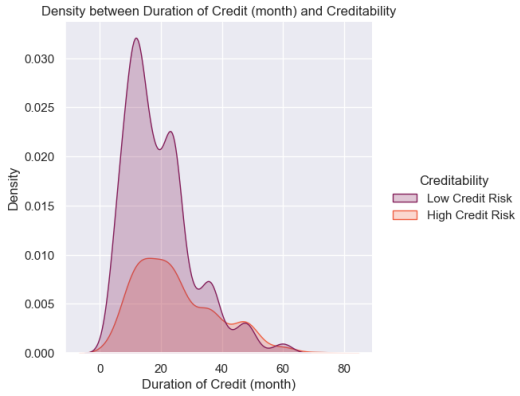
There are lots of **insights** we can get from this correlation map and plot it on the next page.

Correlation between Account Balance and Creditability:



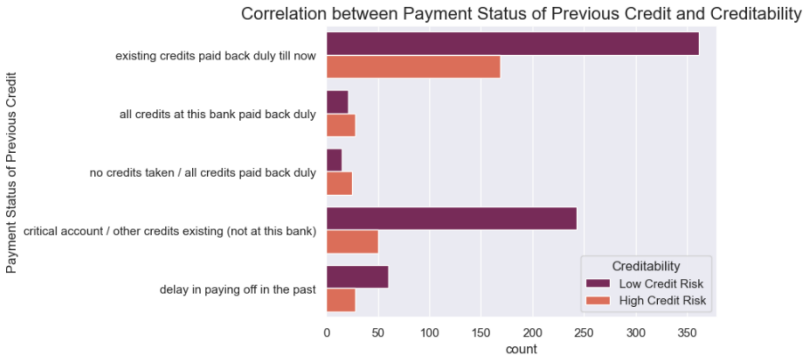
- People having a **higher account balance** or **no account** in this bank will have lower credit risk.

Correlation between Duration of Credit (month) and Creditability:



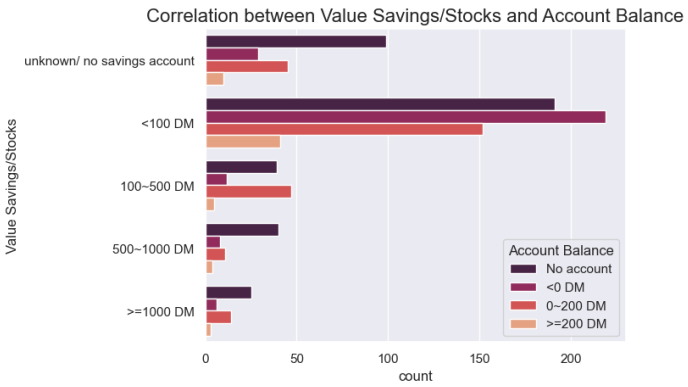
- People having a **shorter time in the duration of credit** have a **lower credit risk**, and vice versa.

Correlation between Payment Status of previous credit and Creditability:



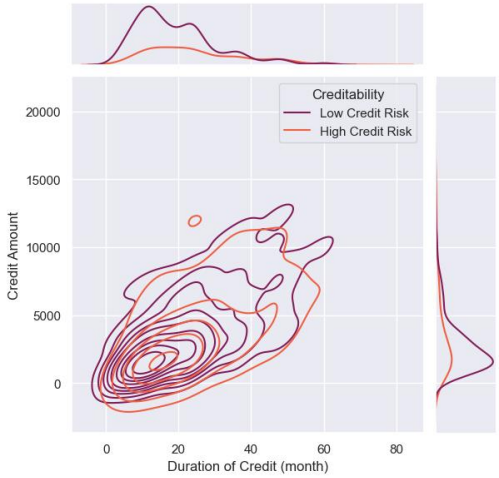
- Most of the previous credit history is **paid back duly** or **existing in other banks.**
- People having the status of **paid back duly**, **existing in other banks** or **delay in paying off in the past** proportionally have a lower credit risk.

Correlation between Value Savings/Stocks and Account Balance:



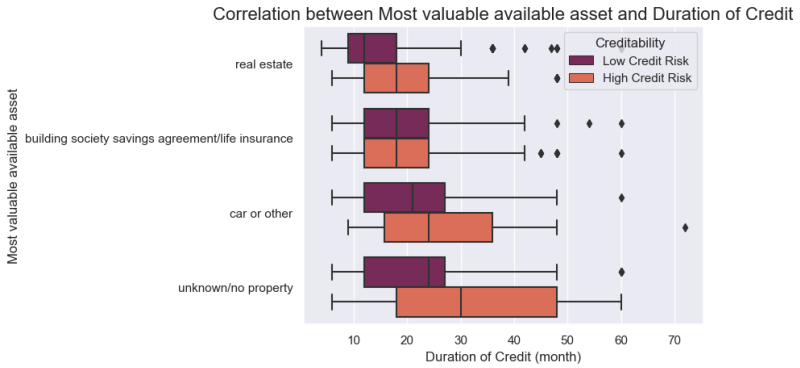
- Most of the people **do not have savings account** or **under 100 Deutsche Mark**.

Correlation between Value Savings/Stocks and Account Balance:



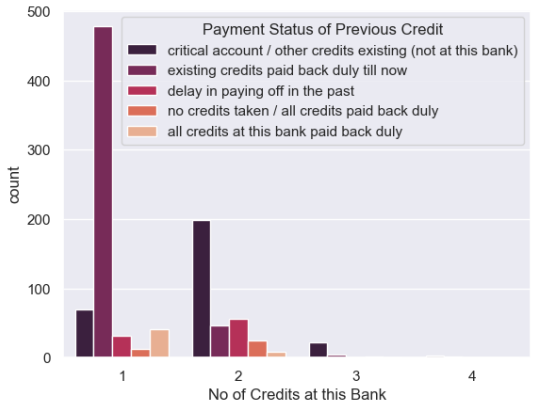
- There is **positive correlation relationship** between **credit amount** and **duration of credit**.

Correlation between Most valuable available asset and Duration of Credit:



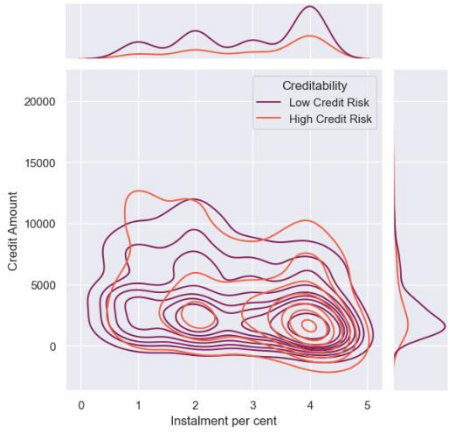
- **High credit risk** people have a **longer duration of credit time**.
- The **lower value of the most valuable asset**, **the longer duration of credit time**.

Correlation between No of Credits and Payment Status of Previous Credit:



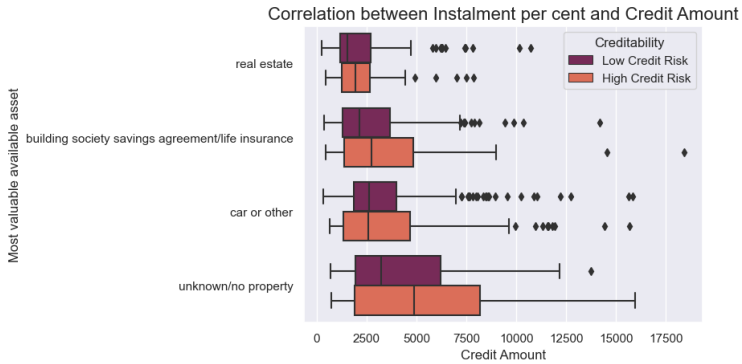
- Most of the people have **1 or 2** credits at this bank

Correlation between Instalment per cent and Credit Amount:



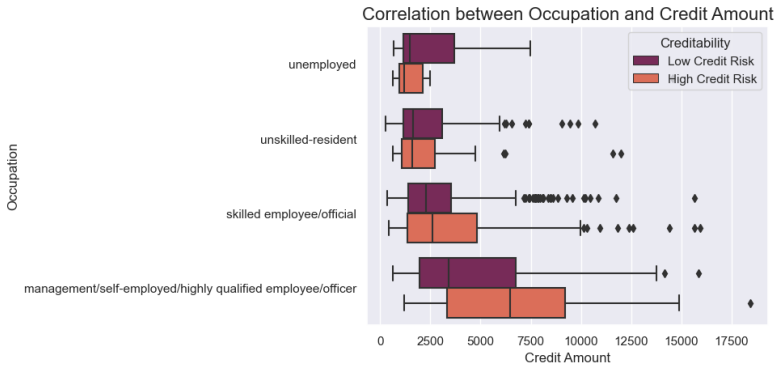
- There is a **slightly negatively correlation relationship** between **instalment per cent** and **credit amount**.

Correlation between Instalment per cent and Credit Amount:



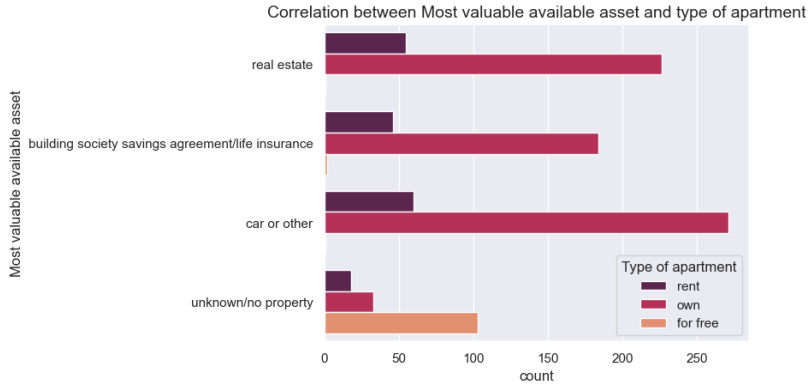
- **High credit risk** people have a **larger credit amount**.

Correlation between Occupation and Credit Amount:



- There is a **positive correlation relationship** between **job skill level** and **credit amount**, people employed with more professional jobs usually having a higher credit amount.

Correlation between Occupation and Credit Amount:



- People who **do not have property**, are mainly living for **free**.

Density between Age (years) and Type of apartment:

- **Ages** mainly belong **20 to 40**.

7.1.4 Data Transformation And Train Test Split

Label encoder:

Label encoder can encode target feature with value between 0 and total class of that feature – 1. Here, **all** column except **Duration of Credit (month), Credit Amount, Instalment per cent Duration in Current address Age (years), No of Credits at this Bank and No of dependents** are categorical data after EDA part, and we need to encode it to certain range of value.

Train Test Split:

Train set:

	Account Balance	Duration of Credit (month)	Payment Status of Previous Credit	Purpose	Credit Amount	Value Savings/Stocks	Length of current employment	Instalment per cent	Sex & Marital Status	Guarantors	Duration in Current address	Most valuable available asset	Age (years)	Concurrent Credits
716	1	48	3	4	7476	2	1	4	3	2	1	3	50	1
351	3	24	3	1	1525	3	1	4	0	2	3	1	34	1
936	0	9	3	5	959	2	0	1	0	2	2	1	29	0
256	1	15	2	5	3643	2	3	1	0	2	4	0	27	1
635	1	24	3	5	3149	2	2	4	3	2	1	3	22	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
106	3	12	1	0	1412	2	0	4	0	1	2	2	29	1
270	3	6	3	2	1236	1	0	2	3	2	4	0	50	1
860	3	48	3	0	3914	4	0	4	2	2	2	2	38	0
435	3	7	2	7	846	4	3	3	3	2	4	3	36	1
102	0	24	2	0	4712	4	0	4	3	2	2	0	34	0

900 rows × 20 columns

Test set:

	Account Balance	Duration of Credit (month)	Payment Status of Previous Credit	Purpose	Credit Amount	Value Savings/Stocks	Length of current employment	Instalment per cent	Sex & Marital Status	Guarantors	Duration in Current address	Most valuable available asset	Age (years)	Concurrent Credits
521	3	36	3	5	10974	2	4	4	0	2	2	1	26	1
737	3	18	1	7	1149	3	0	4	3	2	3	2	46	1
740	3	12	3	5	1736	2	1	3	0	2	4	2	31	1
660	0	8	3	7	1414	2	0	4	3	1	2	2	33	1
411	3	24	2	0	2978	4	0	4	3	2	4	2	32	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
436	3	15	1	4	1532	0	0	4	0	2	3	1	31	1
764	3	24	2	1	2538	2	3	4	3	2	4	1	47	1
88	1	6	1	1	666	3	1	3	0	2	4	2	39	1
63	3	36	2	1	10875	2	3	2	3	2	2	1	45	1
826	0	48	4	0	3844	0	1	4	3	2	4	3	34	1

100 rows × 20 columns

Train test split is an important process to split our data to allocate which data are used to build up the model and which data are used to test the model. In this data set, we set 9:1, 90% Data for training and 10% Data for testing.
(Total: entities **1000**, training entities:**900**, test entities: **100**)

Data Scaling:

Train set after standardization:

	Account Balance	Duration of Credit (month)	Payment Status of Previous Credit	Purpose	Credit Amount	Value Savings/Stocks	Length of current employment	Instalment per cent	Sex & Marital Status	Guarantors	Duration in Current address	Most valuable available asset	Age (years)	Concurr Cre
0	0.666667	1.500000	0.0	0.083333	3.217626	0.0	1.000000	0.5	-1.0	0.0	-0.5	0.0	-0.466667	
1	0.666667	0.000000	-1.0	0.416667	-0.435809	1.0	-0.333333	0.5	0.0	0.0	0.0	1.0	0.866667	
2	0.666667	-0.500000	0.0	0.083333	-0.217533	0.0	0.000000	0.0	-1.0	0.0	0.5	1.0	-0.133333	
3	-0.333333	-0.833333	0.0	0.416667	-0.337269	0.0	-0.333333	0.5	0.0	-1.0	-0.5	1.0	0.000000	
4	0.666667	0.500000	-0.5	-0.750000	0.244306	2.0	-0.333333	0.5	0.0	0.0	0.5	1.0	-0.066667	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
95	0.666667	-0.250000	-1.0	-0.083333	-0.293390	-2.0	-0.333333	0.5	-1.0	0.0	0.0	0.0	-0.133333	
96	0.666667	0.500000	-0.5	-0.583333	0.080692	0.0	0.666667	0.5	0.0	0.0	0.5	0.0	0.933333	
97	0.000000	-1.000000	-1.0	-0.583333	-0.615413	1.0	0.000000	0.0	-1.0	0.0	0.5	1.0	0.400000	
98	0.666667	1.500000	-0.5	-0.583333	3.180812	0.0	0.666667	-0.5	0.0	0.0	-0.5	0.0	0.800000	
99	-0.333333	2.500000	0.5	-0.750000	0.566329	-2.0	0.000000	0.5	0.0	0.0	0.5	2.0	0.066667	

100 rows × 20 columns

Test set after standardization:

	Account Balance	Duration of Credit (month)	Payment Status of Previous Credit	Purpose	Credit Amount	Value Savings/Stocks	Length of current employment	Instalment per cent	Sex & Marital Status	Guarantors	Duration in Current address	Most valuable available asset	Age (years)	Conc
0	0.000000	2.500000	0.0	-0.083333	1.916891	0.0	0.000000	0.5	0.000000	0.0	-1.0	2.0	1.133333	
1	0.666667	0.500000	0.0	-0.583333	-0.295993	1.0	0.000000	0.5	-1.000000	0.0	0.0	0.0	0.066667	
2	-0.333333	-0.750000	0.0	0.083333	-0.506461	0.0	-0.333333	-1.0	-1.000000	0.0	-0.5	0.0	-0.266667	
3	0.000000	-0.250000	-0.5	0.083333	0.491587	0.0	0.666667	-1.0	-1.000000	0.0	0.5	-1.0	-0.400000	
4	0.000000	0.500000	0.0	0.083333	0.307893	0.0	0.333333	0.5	0.000000	0.0	-1.0	2.0	-0.733333	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
895	0.666667	-0.500000	-1.0	-0.750000	-0.338012	0.0	-0.333333	0.5	-1.000000	-1.0	-0.5	1.0	-0.266667	
896	0.666667	-1.000000	0.0	-0.416667	-0.403458	-1.0	-0.333333	-0.5	0.000000	0.0	0.5	-1.0	1.133333	
897	0.666667	2.500000	0.0	-0.750000	0.592358	2.0	-0.333333	0.5	-0.333333	0.0	-0.5	1.0	0.333333	
898	0.666667	-0.916667	-0.5	0.416667	-0.548480	2.0	0.666667	0.0	0.000000	0.0	0.5	2.0	0.200000	
899	-0.333333	0.500000	-0.5	-0.750000	0.889095	2.0	-0.333333	0.5	0.000000	0.0	-0.5	-1.0	0.066667	

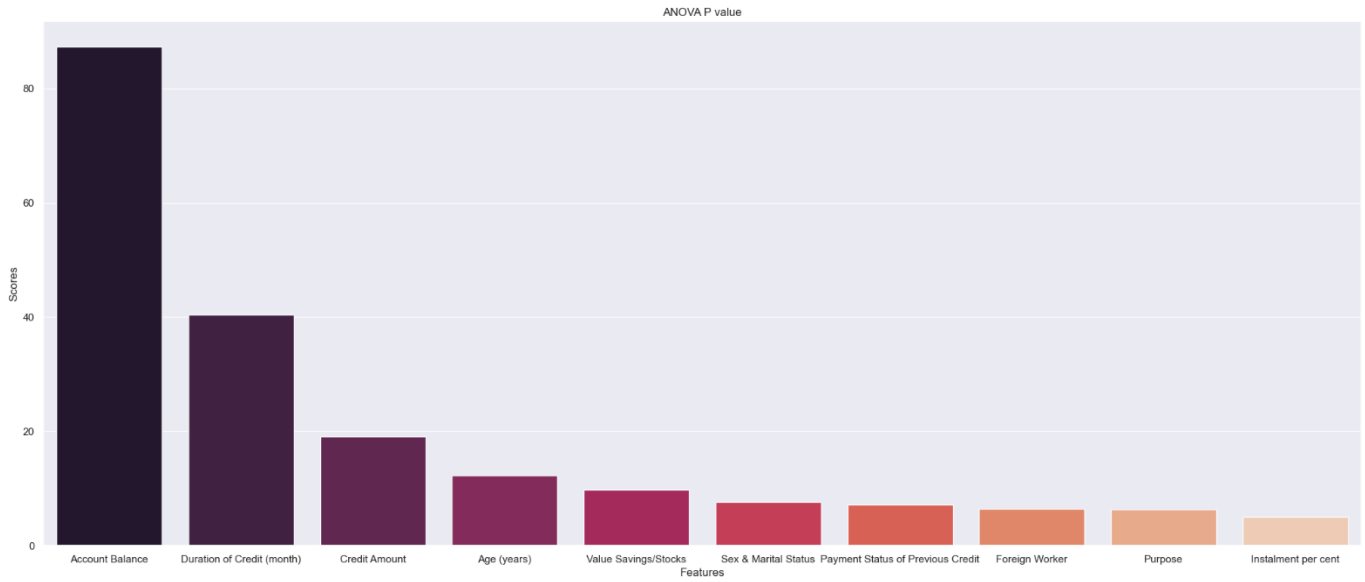
900 rows × 20 columns

Data scaling can shift each of the features into a particular of range, so that it can break the differences in the scales across all the input variables, here we used **Robust Scaler**.

7.1.5 Feature Selection

Input all the features into the machine learning models not only increased the computing time, but it also increased the possibility of collinearly problem.

P value of ANOVA:



H0: Means of all credit rating groups are equal.
H1: At least one mean of credit rating groups are different
($\alpha = 0.05$)

The 10 highest p-value features:
-Account Balance
-Duration of Credit (month)
-Credit Amount
-Age (years)
-Value Savings/Stocks
-Sex & Marital Status
-Payment Status of Previous Credit
-Foreign Worker
-Purpose
-Instalment per cent

7.1.6 Cross-Validation For Hyper Tuning Machine Learning Models

Machine learning models in this report:
-Logistic Regression
-Naïve Bayes Classifier
-Support Vector Machine
-Random Forest
-Gradient Boosting

Cross-validation:
In this report, we cross validate **3** times and using **grid search** setting scoring **"f1"** (Binary Classification) for choosing highest accuracy machine learning models.

Best parameters for different models:

The best parameters for **logistic regression**:
{'C': 0.001, 'class_weight': 'balanced', 'multi_class': 'auto', 'penalty': 'l2', 'solver': 'lbfgs'}

The best parameters for **naïve bayes classifier**:
{'No need tuning parameter'}

The best parameters for **support vector machine**:
{'class_weight': 'balanced', 'decision_function_shape': 'ovo', 'gamma': 'scale', *'kernel': 'rbf'}*

The best parameters for **random forest**:
{'class_weight': 'balanced', 'criterion': 'entropy', 'max_features': 'sqrt', 'n_estimators': 200, 'random_state': 42}

The best parameters for **gradient boosting:**
{'criterion': 'squared_error', 'learning_rate': 0.01, 'loss': 'deviance', 'n_estimators': 200, 'subsample': 0.5}

7.1.7 Implementation

Comparing different models:

	Machine Learning Models	Accuracy	Training Time (s)
0	Logistic Regression	0.77	7.200202
1	Naive Bayes	0.70	0.008404
2	Support Vector Machine	0.74	2.331094
3	Random forest	0.76	310.272295
4	Gradient Boosting	0.74	456.390826

We can observe that **logistic regression** comparatively having the highest accuracy score and **shorter training time** comparing with random forest and gradient boosting those ensemble learning models.

7.2 US Corporate Credit Risk Dataset

7.2.1 Data Set Background

A corporate credit rating means the ability of a firm to repay its debt to creditors. Credit rating companies or agencies are the ones responsible to evaluate a firm repayment ability. When a company in the world wants to issue a new bond, it needs to hire a credit agency to evaluate it. Therefore, investors can know how trustworthy the company is. In 2008, the misleading credit rating of mortgage-related securities leads the credit crisis. Nowadays, to prevent the next financial crisis, the accuracy of credit rating is being emphasized. The data of this data set is based on the financials indicators that can come from the balance sheet, which provided by the credit rating agency company and is just part of the data.

Here are the list of **2029** credit ratings issued by major agencies from 2010 to 2016. There are 30 attributes for every company. They can be divided in:

Basic Information:

Name	*Symbol*	*Rating agency name*
Date	*Sector*	*Rating*

Liquidity Measurement Ratios:

Current ratio	*Quick ratio*	*Cash ratio*	*Days of sales outstanding*

Cash Flow Indicator Ratios:

Operating cash flow per share	*Free cash flow per share*	*Cash per share*
Operating cash flow sales ratio	*Free cash flow operating cash flow ratio*	

Debt Ratios:

Debt ratio	*Debt equity ratio*	*Payables turnover*

Operating Performance Ratios:

Asset turnover	*Enterprise value multiple*

Risk Ratio

Company equity multiplier

Profitability Indicator Ratios:

Gross profit margin	*Operating profit margin*	*Pretax profit margin*
Net profit margin	*Effective tax rate*	*Return on assets*
Return on equity	*Return on capital employed*	*Ebit per revenue*

The sample data frame of this data set:

	Rating	Name	Symbol	Rating Agency Name	Date	Sector	currentRatio	quickRatio	cashRatio	daysOfSalesOutstanding	...
0	A	Whirlpool Corporation	WHR	Egan-Jones Ratings Company	11/27/2015	Consumer Durables	0.945894	0.426395	0.099690	44.203245	...
1	BBB	Whirlpool Corporation	WHR	Egan-Jones Ratings Company	2/13/2014	Consumer Durables	1.033559	0.498234	0.203120	38.991156	...
2	BBB	Whirlpool Corporation	WHR	Fitch Ratings	3/6/2015	Consumer Durables	0.963703	0.451505	0.122099	50.841385	...
3	BBB	Whirlpool Corporation	WHR	Fitch Ratings	6/15/2012	Consumer Durables	1.019851	0.510402	0.176116	41.161738	...
4	BBB	Whirlpool Corporation	WHR	Standard & Poor's Ratings Services	10/24/2016	Consumer Durables	0.957844	0.495432	0.141608	47.761126	...

5 rows × 31 columns

7.2.2 Data Cleaning

The data type of the data set:

```
Data columns (total 31 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Rating                          2029 non-null   object
 1   Name                            2029 non-null   object
 2   Symbol                          2029 non-null   object
 3   Rating Agency Name              2029 non-null   object
 4   Date                            2029 non-null   object
 5   Sector                          2029 non-null   object
 6   currentRatio                    2029 non-null   float64
 7   quickRatio                      2029 non-null   float64
 8   cashRatio                       2029 non-null   float64
 9   daysOfSalesOutstanding          2029 non-null   float64
 10  netProfitMargin                 2029 non-null   float64
 11  pretaxProfitMargin              2029 non-null   float64
 12  grossProfitMargin               2029 non-null   float64
 13  operatingProfitMargin           2029 non-null   float64
 14  returnOnAssets                  2029 non-null   float64
 15  returnOnCapitalEmployed         2029 non-null   float64
 16  returnOnEquity                  2029 non-null   float64
 17  assetTurnover                   2029 non-null   float64
 18  fixedAssetTurnover              2029 non-null   float64
 19  debtEquityRatio                 2029 non-null   float64
...
 29  operatingCashFlowSalesRatio     2029 non-null   float64
 30  payablesTurnover                2029 non-null   float64
```

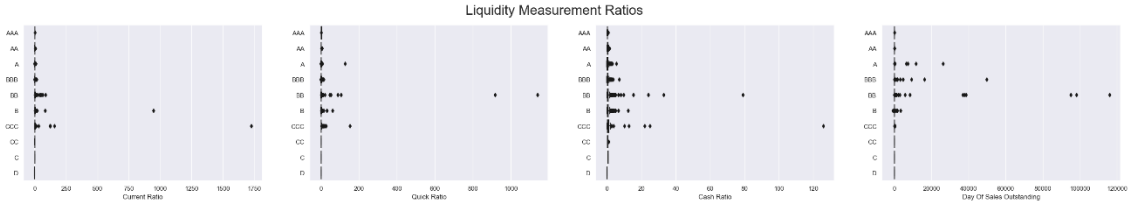
In this data set, all the data are non-null, so that here is not missing value.

The skewness of the data set:

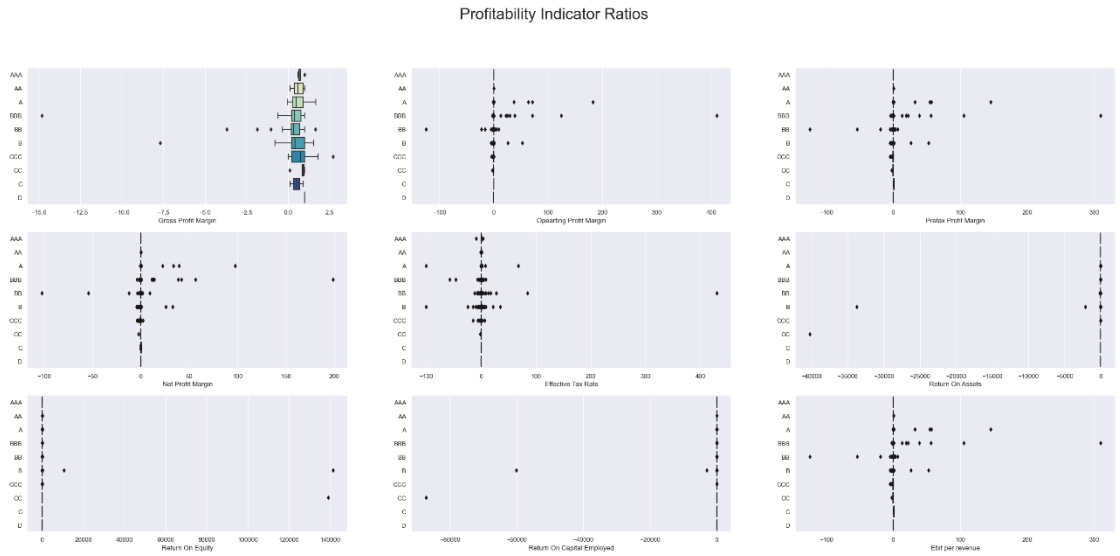
```
currentRatio                          34.338889
quickRatio                            30.925641
cashRatio                             27.098772
daysOfSalesOutstanding                20.399567
netProfitMargin                       17.619910
pretaxProfitMargin                    22.096334
grossProfitMargin                    -14.203446
operatingProfitMargin                 26.493892
returnOnAssets                       -32.112571
returnOnCapitalEmployed              -33.318531
returnOnEquity                        31.702500
assetTurnover                         26.020347
fixedAssetTurnover                    26.120457
debtEquityRatio                        0.268993
debtRatio                              1.285251
effectiveTaxRate                      32.308742
freeCashFlowOperatingCashFlowRatio   -22.913129
freeCashFlowPerShare                  33.677207
cashPerShare                          34.025861
companyEquityMultiplier                0.269093
ebitPerRevenue                        22.099452
enterpriseValueMultiple               13.948016
operatingCashFlowPerShare             30.352915
operatingCashFlowSalesRatio           25.450452
payablesTurnover                      25.919588
```

The data is not evenly distributed, there might outliers in each of the attributes.

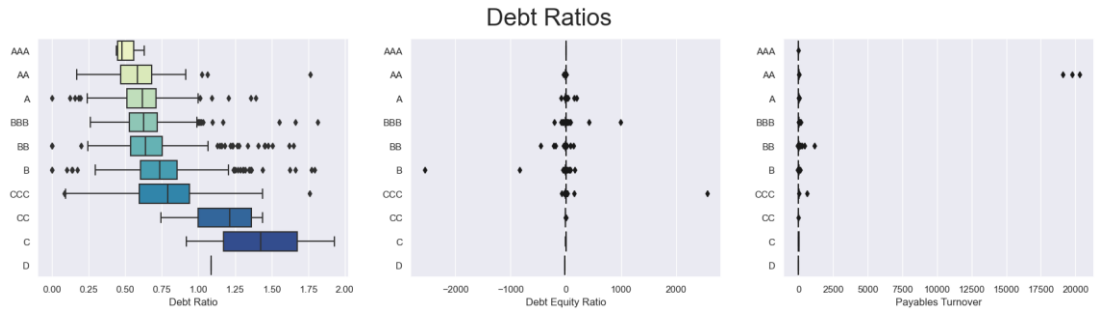
Boxplot of liquidity measurement ratios:



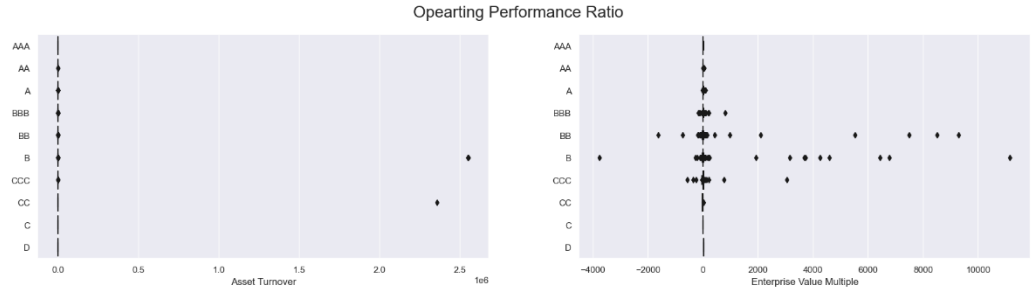
Boxplot of profitability indicator ratios:



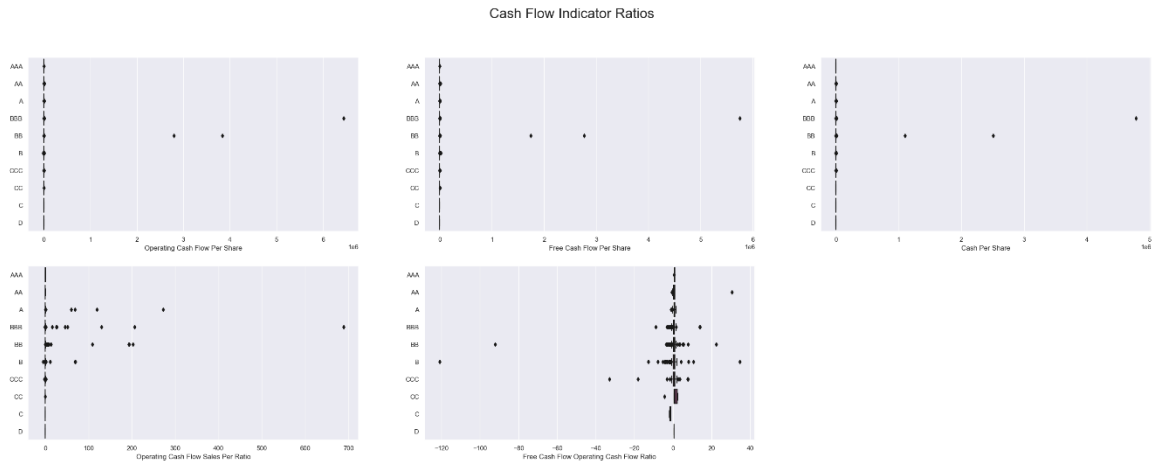
Boxplot of debt ratios:



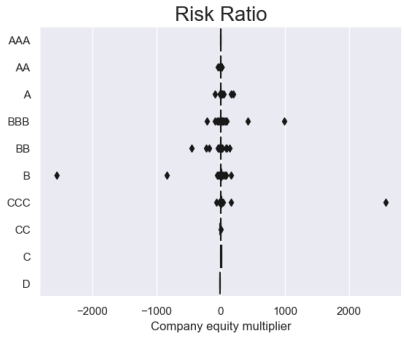
Boxplot of operating performance ratio:



Boxplot of cash flow indicator ratios:

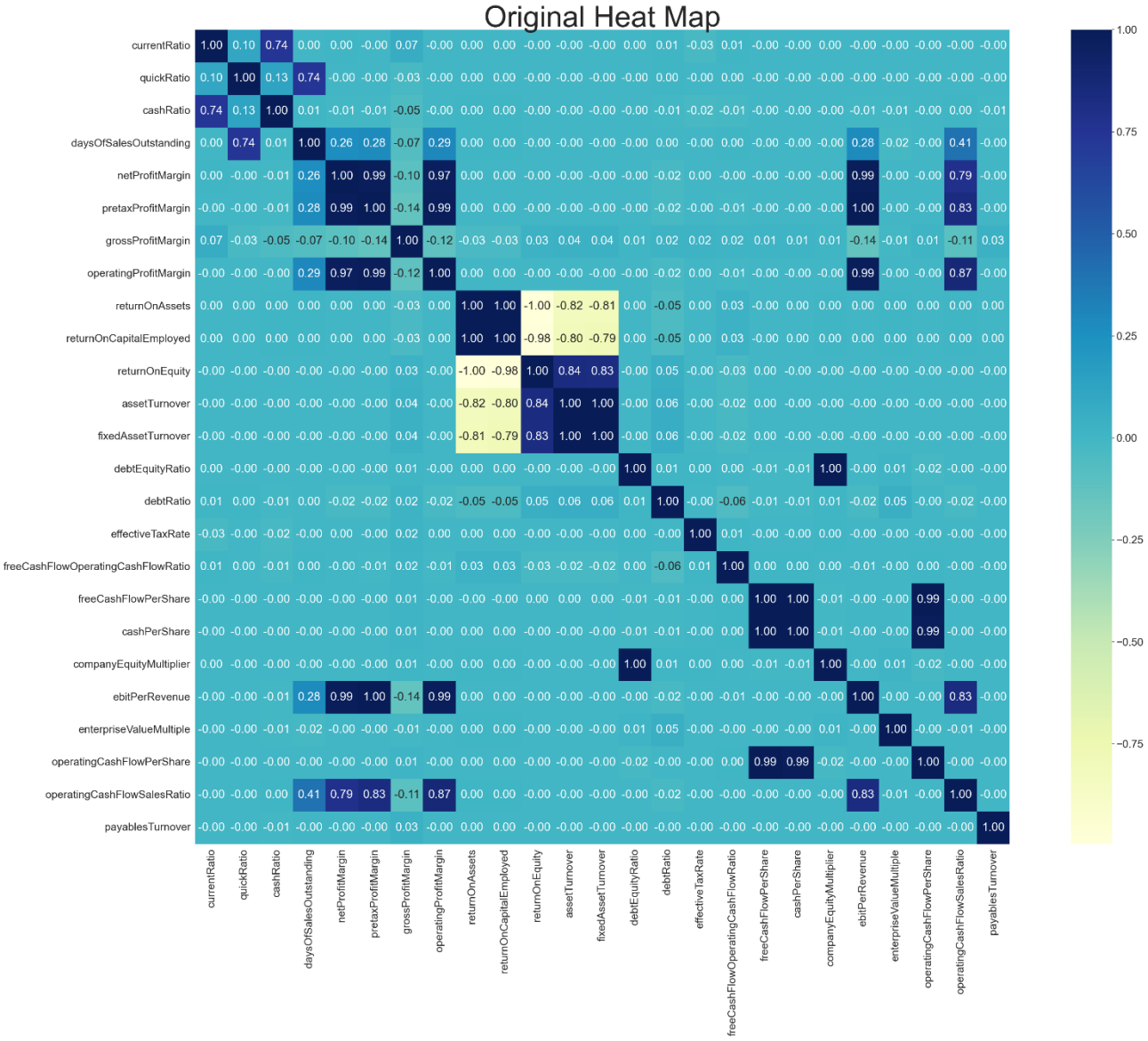


Boxplot of risk indicator ratio:



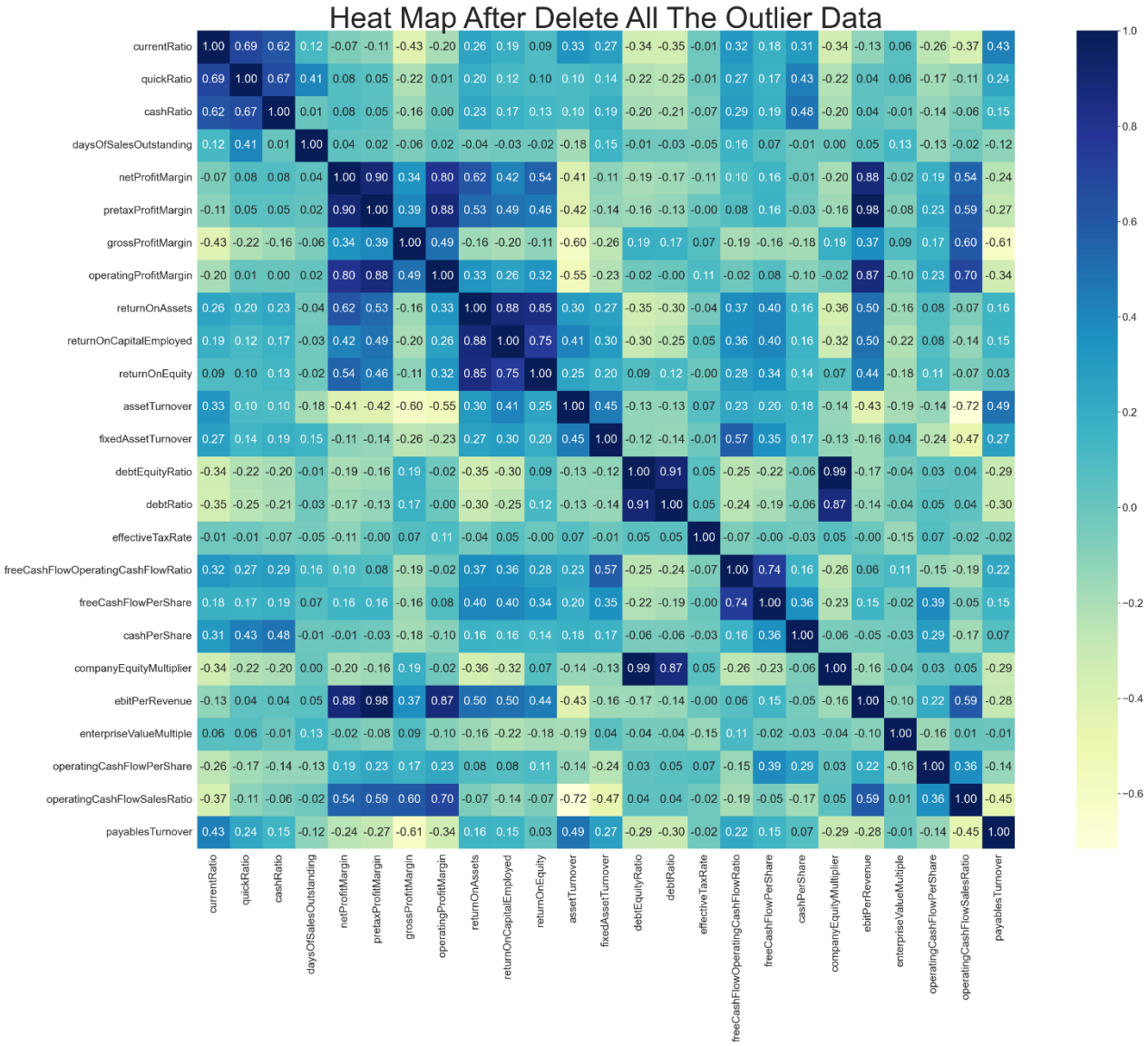
When looking into the boxplot of all the attributes, we can observe that outliers of the data impact the skewness. If we need to get insight and build up the classifiers, cleaning the outliers is a must.

Heat Map of the data without data cleaning:



The outliers of the data cancelled the correlation between each of the attributes.

Heat Map of the data delete all the outlier with IQR:



Here, used IQR x 1.5 rule directly delete the data points which are higher than (Q3+1.5*IQR) or lower than (Q1-1.5*IQR). After the deletion, the correlation of each feature become visible. But the entities of the data from **2029** deleted to **758,** losing about 62.64% of the data.

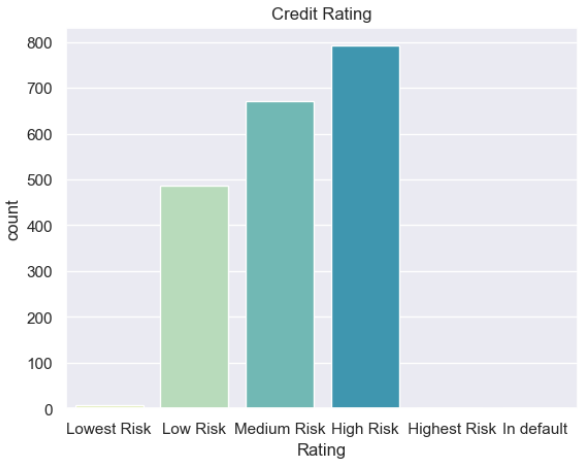
Number of entities of each rating:

```
BBB    671
BB     490
A      398
B      302
AA      89
CCC     64
AAA      7
CC       5
C        2
D        1
Name: Rating, dtype: int64
```

Because the amount of data in each of the ratings is not equally distributed. If we just directly implement those data rating data to do the analysis and build up the models, it leads to misunderstand about the data and not accurate classifiers. Therefore, grouping the credit rating is the proper process before the analysis.

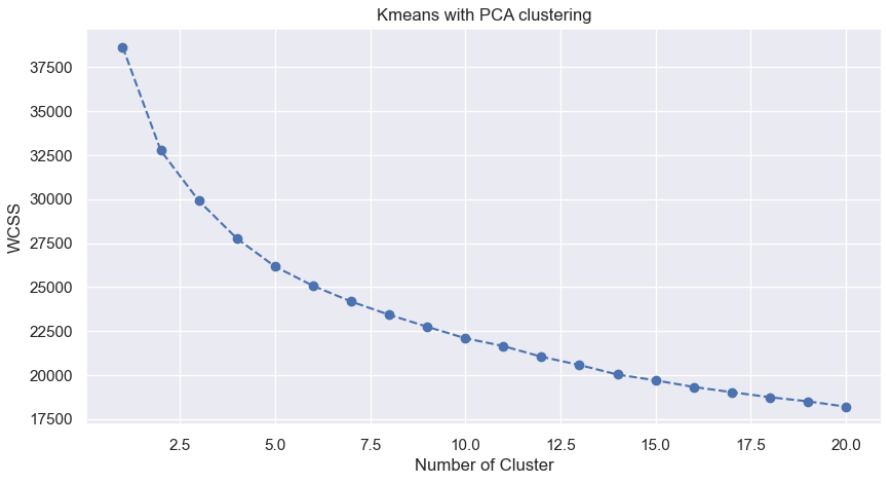
According to major bond ratings of Standard & Poor's (S&P):

Credit Rating	**Risk Level**
AAA	Lowest Risk
AA	Low Risk
A	Low Risk
BBB	Medium Risk
BB	High Risk
B	High Risk
CCC	Highest Risk
CC	Highest Risk
C	Highest Risk
D	In default



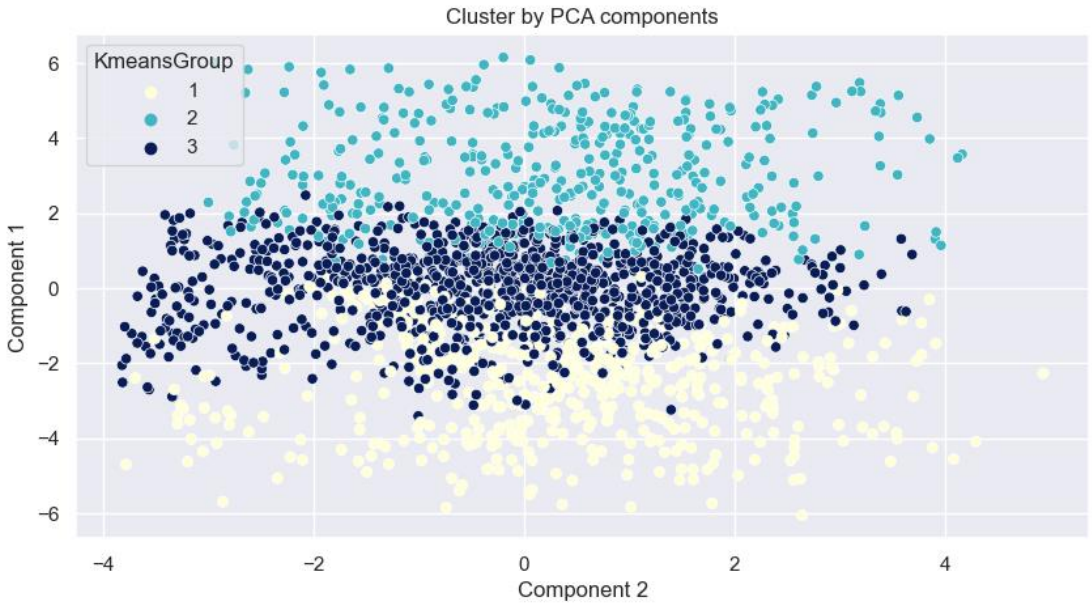
Even if using S&P reallocated the credit rating level, the data also is extremely different in each of the credit ratings. I keep this approach for the grouping part, deleting some of the risk levels will be the only way for the next few parts. However, I want to keep the data completeness of this data set, so I would use K-Means clustering to make the reference grouping the data rather than just deletion.

K-Means Elbow Approach:



- **3-Cluste**r can sufficiently group the data

K-Means PCA Components:



After grouping with K-Means:

Rating	A	AA	AAA	B	BB	BBB	C	CC	CCC	D
KmeansGroup										
1	148.0	43.0	7.0	41.0	105.0	175.0	1.0	0.0	6.0	0.0
2	11.0	1.0	0.0	142.0	135.0	86.0	0.0	5.0	44.0	1.0
3	239.0	45.0	0.0	119.0	250.0	410.0	1.0	0.0	14.0	0.0

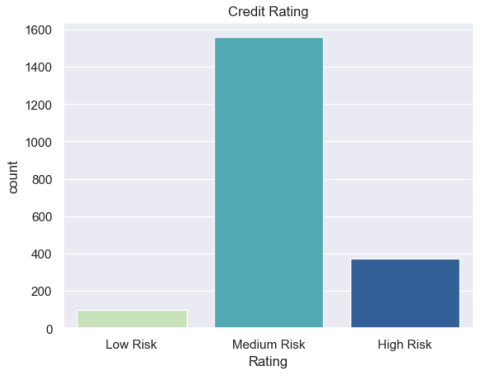
Simply reference of K-Means:

- First Group: AAA, AA
- Second Group: A, BBB, BB
- Third Group: B, CCC, CC, C, D

It would be more proper to reallocate the data groups.

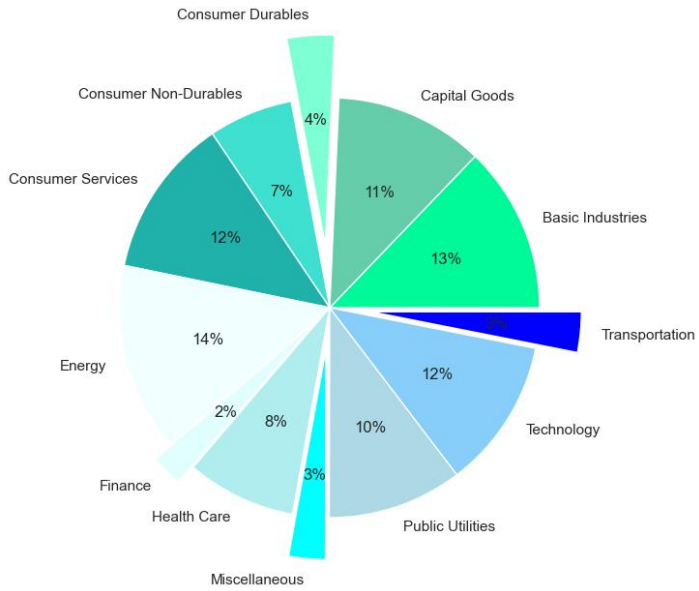
7.2.3 Exploratory Data Analysis (EDA)

Number of credit rating data after cleaning:



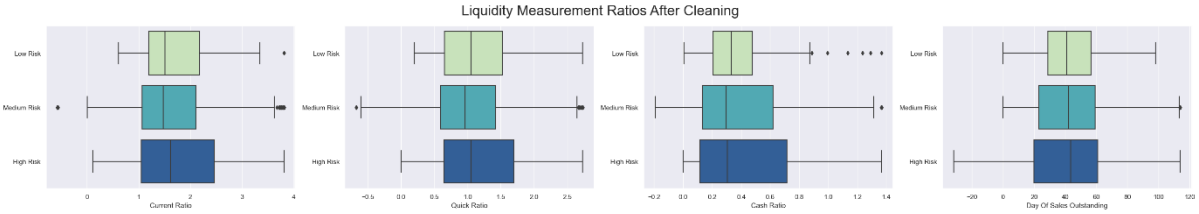
- Although the unbalance situation of credit rating data still exists, it can represent the **insight** of the data due to the amount of the data. Also, there is a **weighting method** for an unbalanced dataset for building the classifier's part in this type of not extreme case.

Number of sectors of the data after cleaning



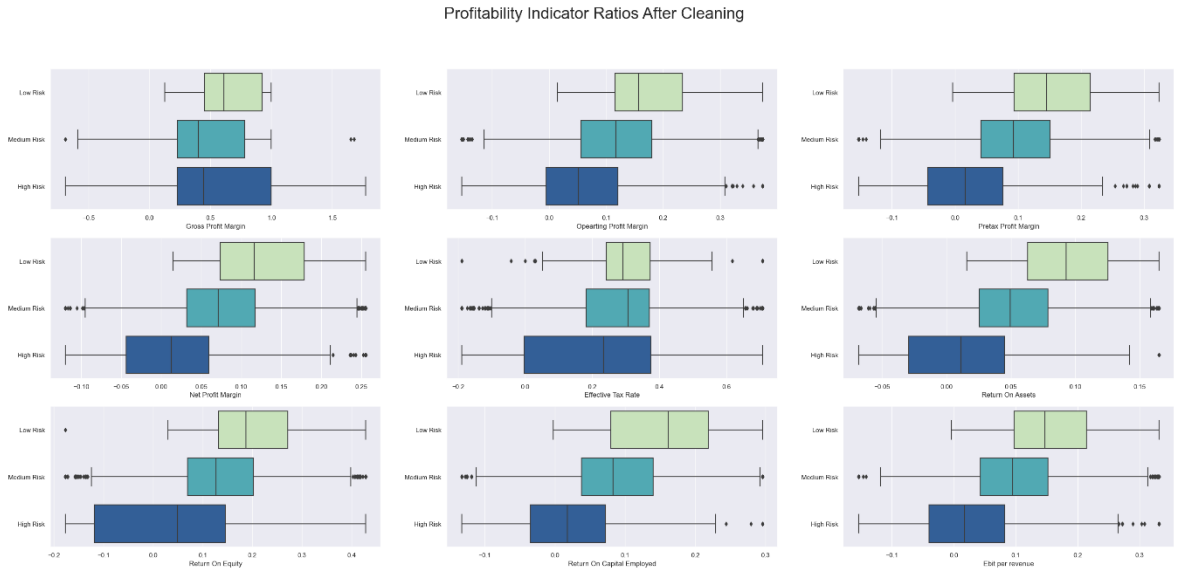
- The sector of the data is **almost equally distributed** in different aspect.
- **Energy** is the most common aspect and **Finance** is the least common.

Liquidity measurement ratios after cleaning:



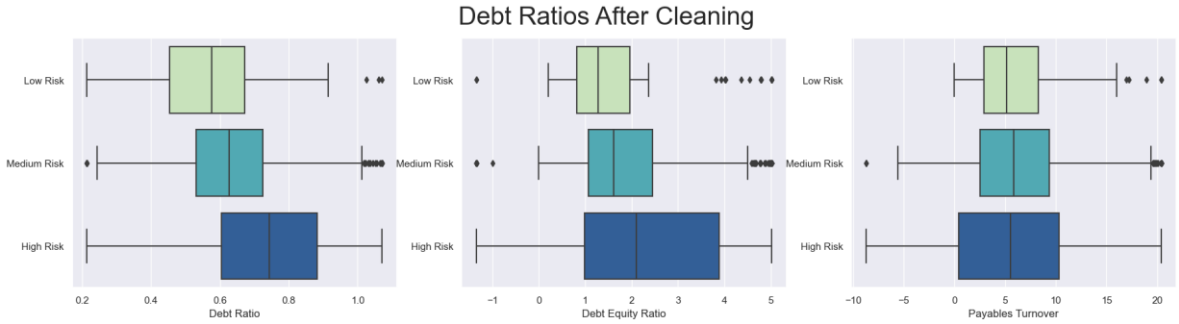
- There is **no obvious relationship** between **liquidity measurement ratios** and credit rating.

*Profitability indicator ratios after cleaning***:**



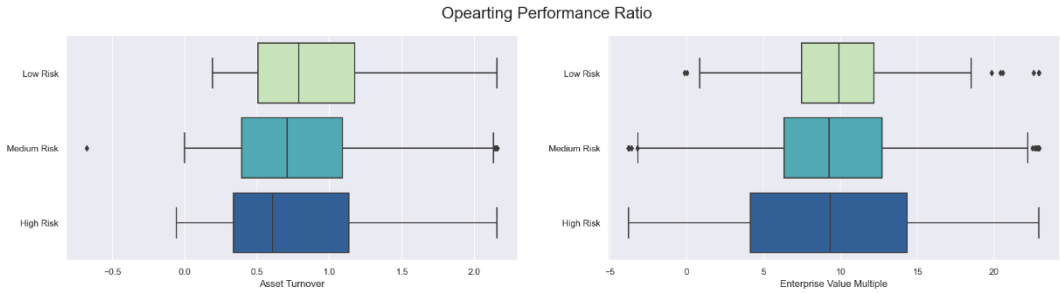
- It shows **positive correlation**, most of the ratios' trend show that higher credit rating company having higher profitability.

Debt ratios after cleaning:



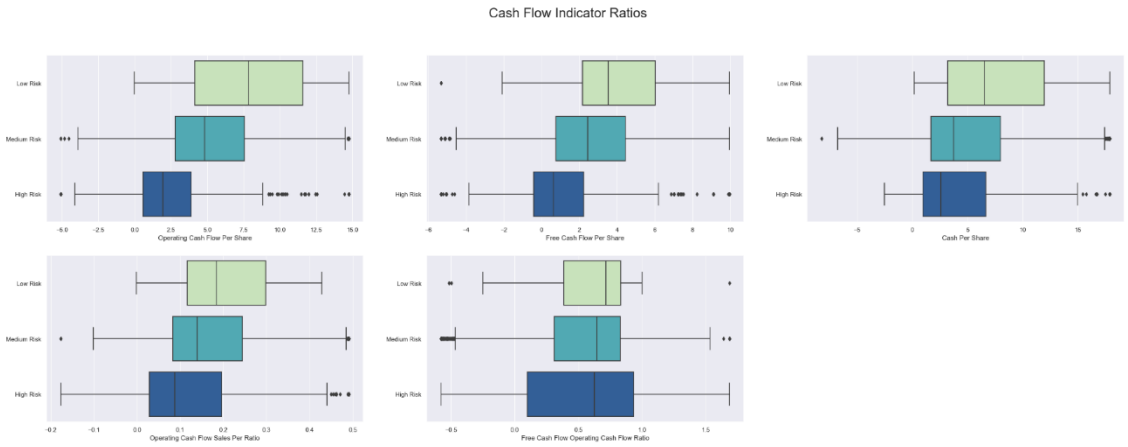
- **Negative correlation** between credit rating and debt ratios, higher debt ratios leading lower credit rating.
- There is **no obvious relationship** between **payables turnover** and credit rating

Operating performance ratio after cleaning:



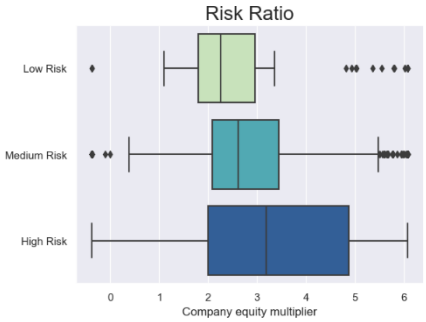
- There is **slightly positive relationship** between operating performance and credit rating, higher asset turnover ratio, higher credit rating.

Cash flow ratios after cleaning:



- Company with higher **operating cash flow per share**, **free cash flow per share, cash per shares** and **operating cash flow sales per ratio** usually having higher credit rating.
- There is **no obvious relationship** between **free cash flow operating cash flow per share** and credit rating.

Risk ratio after cleaning:



- **Negative correlation** between credit rating and **company equity multiplier**.

7.2.4 Data Transformation And Train Test Split

Label encoder:

```
0       2
1       2
2       2
3       2
4       2
       ..
2024    2
2025    2
2026    0
2027    0
2028    0
Name: Rating, Length: 2029, dtype: int32
```

```
0       2
1       2
2       2
3       2
4       2
       ..
2024    1
2025    1
2026    6
2027    6
2028    6
Name: Sector, Length: 2029, dtype: int32
```

Label encoder can encode target feature with value between 0 and total class of that feature – 1. Here, **rating and sector** column are categorical data, and we need to encode it to certain range of value.

Train Test Split:

Train set:

	Sector	currentRatio	quickRatio	cashRatio	daysOfSalesOutstanding	netProfitMargin	pretaxProfitMargin	grossProfitMargin	operatingProfitMargin	return
1782	2	0.998333	0.757117	0.206420	16.958853	0.112542	0.165523	0.653859	0.199369	
212	9	0.880570	0.298924	0.029318	24.885052	0.086875	0.131216	1.000000	0.169269	
2019	4	0.887235	0.864677	0.053052	113.951268	0.255508	0.323434	0.640833	0.373536	
904	3	2.204231	1.627760	0.784066	21.051607	0.076859	0.121536	0.275548	0.136204	
231	0	1.519112	1.326996	0.932311	39.215309	-0.024864	0.003646	0.198398	0.101955	
...	...	...	...	...	...	...	...	...	...	
1130	9	0.576311	0.459033	0.292785	53.112862	0.128957	0.155626	0.837700	0.186058	
1294	9	0.367189	0.292559	0.090749	42.000504	0.074783	0.120232	0.377691	0.179151	
860	3	2.092386	1.264094	0.478273	39.765918	0.101007	0.144763	0.805146	0.149002	
1459	5	1.051136	0.927557	0.332386	41.855212	0.089961	0.094981	0.863320	0.094981	
1126	9	0.692985	0.443754	0.303891	36.669478	0.221722	0.271098	0.800924	0.248441	

1826 rows × 26 columns

Test set:

	Sector	currentRatio	quickRatio	cashRatio	daysOfSalesOutstanding	netProfitMargin	pretaxProfitMargin	grossProfitMargin	operatingProfitMargin	return
1356	0	1.110450	0.688346	0.047782	67.136690	0.084227	0.118931	0.462267	0.136900	
984	5	0.628874	0.551392	0.010943	61.792374	-0.036043	-0.055489	1.000000	-0.013299	
859	3	1.814118	1.132706	0.633882	39.833213	0.088644	0.129826	0.794502	0.134974	
1983	4	3.809332	1.666146	0.119438	97.872737	0.071433	0.100424	0.372569	0.107432	
1293	9	0.589752	0.391544	0.049513	36.913904	0.063232	0.190497	0.613090	0.274128	
...	...	...	...	...	...	...	...	...	...	
344	10	2.160242	0.751758	0.698182	0.000000	0.131158	0.129836	0.203571	0.026011	
1858	10	1.891549	2.478648	0.425263	81.546632	-0.066582	-0.051038	0.930839	0.040621	
567	2	1.120041	0.680746	0.180233	43.916651	0.101267	0.139230	0.342284	0.151664	
570	2	0.892869	0.525056	0.126727	41.137447	0.077368	0.114328	0.338826	0.127814	
307	5	1.420119	0.749720	0.749720	0.000000	-0.119695	-0.153023	1.000000	-0.152745	

203 rows × 26 columns

Train test split is an important process to split our data to allocate which data are used to build up the model and which data are used to test the model. In this data set, we set 9:1, 90% Data for training and 10% Data for testing.
(Total: entities **2029**, training entities:**1826**, test entities: **203**)

Data Scaling:

Train set after standardization:

	Sector	currentRatio	quickRatio	cashRatio	daysOfSalesOutstanding	netProfitMargin	pretaxProfitMargin	grossProfitMargin	operatingProfitMargin	re
0	-0.500000	-0.445909	-0.249650	-0.175423	-0.693759	0.516977	0.686205	0.394670	0.696282	
1	0.666667	-0.553971	-0.786631	-0.548164	-0.473915	0.240909	0.394066	0.965276	0.466418	
2	-0.166667	-0.547855	-0.123594	-0.498213	1.996463	2.054676	2.030898	0.373197	2.026309	
3	-0.333333	0.660655	0.770705	1.040336	-0.580241	0.133178	0.311638	-0.228968	0.213916	
4	-0.833333	0.031972	0.418222	1.352343	-0.076445	-0.960927	-0.692250	-0.356148	-0.047623	
...	...	...	...	...	...	...	...	...	...	
1821	0.666667	-0.833167	-0.598990	0.006348	0.309024	0.693534	0.601927	0.697728	0.594626	
1822	0.666667	-1.025063	-0.794090	-0.418873	0.000807	0.110849	0.300530	-0.060587	0.541888	
1823	-0.333333	0.558023	0.344505	0.396740	-0.061173	0.392912	0.509424	0.644064	0.311650	
1824	0.000000	-0.397455	-0.049902	0.089696	-0.003223	0.274105	0.085506	0.739963	-0.100884	
1825	0.666667	-0.726104	-0.616897	0.029722	-0.147057	1.691282	1.585235	0.637103	1.071015	

1826 rows × 26 columns

Test set after standardization:

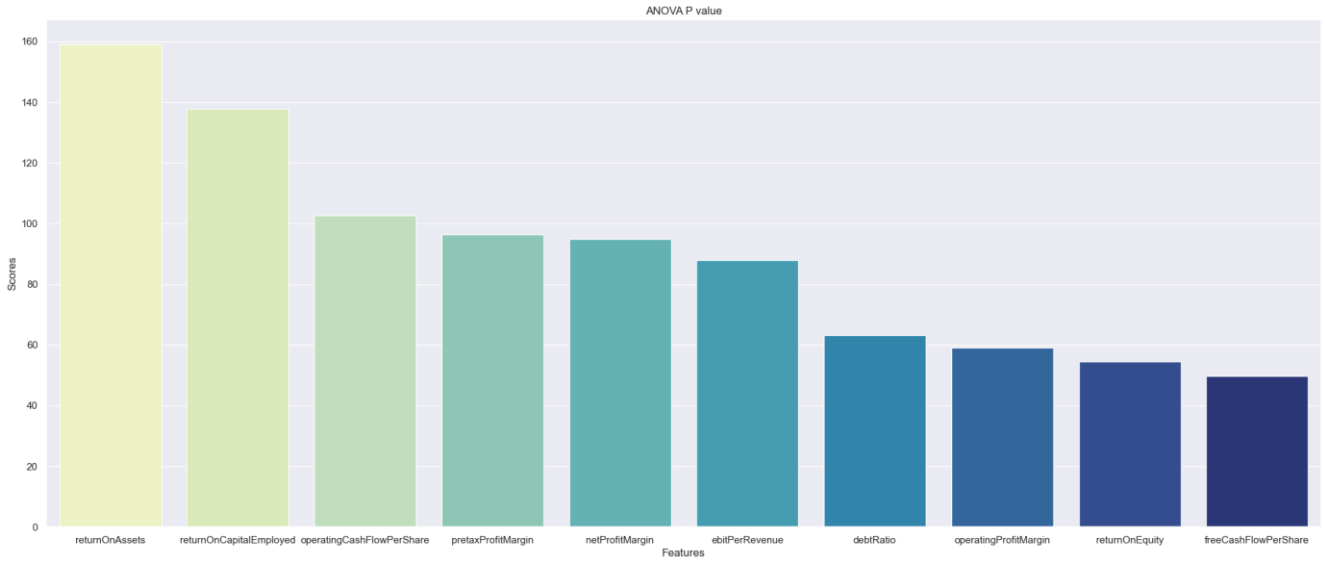
	Sector	currentRatio	quickRatio	cashRatio	daysOfSalesOutstanding	netProfitMargin	pretaxProfitMargin	grossProfitMargin	operatingProfitMargin	retu
0	-0.833333	-0.343027	-0.330246	-0.509304	0.697994	0.212425	0.289458	0.078834	0.219233	
1	0.000000	-0.784934	-0.490750	-0.586838	0.549762	-1.081165	-1.195819	0.965276	-0.927768	
2	-0.333333	0.302677	0.190523	0.724247	-0.059306	0.259936	0.382231	0.626517	0.204524	
3	-0.166667	2.133538	0.815691	-0.358491	1.550502	0.074818	0.131862	-0.069031	-0.005801	
4	0.666667	-0.820833	-0.678085	-0.505661	-0.140278	-0.013390	0.898872	0.327463	1.267178	
...	...	...	...	...	...	...	...	...	...	
198	0.833333	0.620290	-0.255930	0.859577	-1.164137	0.717207	0.382314	-0.347620	-0.627571	
199	0.833333	0.373730	1.767906	0.285172	1.097675	-1.409627	-1.157910	0.851265	-0.516005	
200	-0.500000	-0.334226	-0.339153	-0.230538	0.053954	0.395705	0.462313	-0.118954	0.331980	
201	-0.500000	-0.542685	-0.521614	-0.343151	-0.023132	0.138650	0.250256	-0.124656	0.149847	
202	0.000000	-0.058866	-0.258318	0.968048	-1.164137	-1.980892	-2.026364	0.965276	-1.992646	

Data scaling can shift each of the features into a particular of range, so that it can break the differences in the scales across all the input variables, here we used **Robust Scaler**.

7.2.5 Feature Selection

Input all the features into the machine learning models not only increased the computing time, but it also increased the possibility of collinearly problem.

P value of ANOVA:



H0: Means of all credit rating groups are equal.
H1: At least one mean of credit rating groups are different
($\alpha = 0.05$)

The 10 highest p-value features:
-Return on asset
-Return on capital employed
-Operating cash flow per share
-Pretax profit margin
-Net profit margin
-Ebit per revenue
-Debt ratio
-Operating profit margin
-Rreturn on equity
-Free cash flow per share

7.2.6 Cross-Validation For Hyper Tuning Machine Learning Models

Machine learning models in this report:
-Logistic Regression
-Naïve Bayes Classifier
-Support Vector Machine
-Random Forest
-Gradient Boosting

Cross-validation:
In this report, we cross validate **3** times and using **grid search** setting scoring **"accuracy"** (Binomial Classification) for choosing highest accuracy machine learning models.

Best parameters for different models:

The best parameters for **logistic regression**:
{'C': 100, 'class_weight': 'balanced', 'multi_class': 'auto', 'penalty': 'l2', 'solver': 'liblinear'}

The best parameters for **naïve bayes classifier**:
{'No need tuning parameter'}

The best parameters for **support vector machine**:
{'class_weight': 'balanced', 'decision_function_shape': 'ovo', 'gamma': 'scale', 'kernel': 'rbf'}

The best parameters for **random forest**:
{'class_weight': 'balanced', 'criterion': 'entropy', 'max_features': 'sqrt', 'n_estimators': 200, 'random_state': 42}

The best parameters for **gradient boosting:**
{'criterion': 'friedman_mse', 'learning_rate': 0.01, 'loss': 'log_loss', 'n_estimators': 400, 'subsample': 1}

7.2.7 Implementation

Comparing different models:

	Machine Learning Models	Accuracy	Training Time (s)
0	Logistic Regression	0.743842	19.341565
1	Naive Bayes	0.674877	0.011868
2	Support Vector Machine	0.561576	6.728043
3	Random forest	0.793103	436.240312
4	Gradient Boosting	0.788177	2082.442169

We can observe that **random forest** and **gradient boosting** those ensemble learning models having the **best accuracy score** in this data set. Although **logistic regression** not the best accurate model in this data set, it can have a shorter training time comparing ensemble learning models and a certain accuracy score.

7.3 Result And Findings From Simulated Dataset

Logistic regression is often considered a better choice for **credit risk rating classification tasks** due to several advantages it offers. Firstly, it is simple and easy to interpret. Logistic regression produces a straightforward equation that can be easily understood and interpreted by stakeholders. This is because it is based on a mathematical model, rather than complex ensemble learning models.

Secondly, logistic regression is fast to train and predict. It is computationally efficient and can quickly handle large amounts of data. For instance, in the *7.1 German credit rating data set*, logistic regression achieved the **highest accuracy** in a **shorter training time**. Similarly, in the *7.2 US corporate credit data set*, logistic regression proved to be the most efficient model.

Thirdly, logistic regression is excellent at handling imbalanced datasets. Most credit data sets are imbalanced, as most people or companies have good credit ratings. Logistic regression can handle such datasets by adjusting the misclassification costs.

Finally, logistic regression can handle multiple outcomes, making it suitable for credit risk rating classification problems with many variables. This is because it can handle a wide range of credit rating classifications, from low risk to high-risk, by adjusting the threshold probabilities.

In conclusion, these advantages make logistic regression a popular choice for credit risk rating classification, as it is simple to interpret, fast to train and predict, good at handling imbalanced datasets, and capable of handling multiple outcomes.

8. Lending Club Risk Dataset

In this part, we will follow the process in *part 7* and use logistic regression to build a classifier to predict the real data set.

8.1 Data Set Background

Lending Club is a peer-to-peer lending firm based in the United States that connects individuals seeking to invest with those seeking to borrow. The company facilitates the exchange of funds between investors and borrowers, with the latter returning capital plus interest to the former.

This Lending Club dataset encompasses all loans issued between 2007 and 2015, including the current loan status and the latest payment information. It contains various features. The collections feature indicates whether the borrower has missed one or more payments and the company is attempting to recover the funds. The dataset originally comprises approximately **890,000** rows and **75** variables, but some of them are most of missing data or irrelevant data, such as address details (including zip codes and state). Therefore, after removing the irrelevant or missing data, this data set remains **396,030** rows and **23** variables.

The features:

loan_amnt <- The listed amount of the loan applied for by the borrower.

term <- The number of payments on the loan. Values can be either 36 or 60.

int_rate <- Interest Rate on the loan.
installment <-The monthly payment owed by the borrower.

grade <- Loan grade assigned by Loan Club.

sub_grade <- Loan subgrade assigned by Loan Club

emp_length <- Employment length. (>10 year assigned to 10, <1 year assigned to 0)

home_ownership <- The home ownership status of the borrower.

annual_inc <- The self-reported annual income provided by the borrower.

verification_status <- Indicates if the co-borrowers' joint income was verified.

loan_status <- Fully Paid or Charged Off

purpose <- Borrowers' purpose for the loan request.

Dti <- A ratio calculated using the borrower's total monthly debt payments on the total debt obligations divided by the borrower's self-reported monthly income.

open_acc <- The number of open credit lines in the borrower's credit file.

pub_rec <- Number of derogatory public records

revol_bal <- Total credit revolving balance

revol_util <- Revolving line utilization rate.

total_acc <- The total number of credit lines currently in the borrower's credit file

initial_list_status <-The initial listing status of the loan. Possible values are – W, F

application_type <- Indicates whether the loan is an individual application or a joint application with two co-borrowers.

mort_acc <- Number of mortgage accounts.

pub_rec_bankruptcies <- Number of public record bankruptcies

The sample data frame of this data set:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_length	home_ownership	annual_inc	verification_status	...
0	10000.0	36 months	11.44	329.48	B	B4	10+ years	RENT	117000.0	Not Verified	...
1	8000.0	36 months	11.99	265.68	B	B5	4 years	MORTGAGE	65000.0	Not Verified	...
2	15600.0	36 months	10.49	506.97	B	B3	< 1 year	RENT	43057.0	Source Verified	...
3	7200.0	36 months	6.49	220.65	A	A2	6 years	RENT	54000.0	Not Verified	...
4	24375.0	60 months	17.27	609.33	C	C5	9 years	MORTGAGE	55000.0	Verified	...

8.2.Data Cleaning

The data type of the data set:

```
Data columns (total 23 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  object
 2   int_rate              396030 non-null  float64
 3   installment           396030 non-null  float64
 4   grade                 396030 non-null  object
 5   sub_grade             396030 non-null  object
 6   emp_length            377729 non-null  object
 7   home_ownership        396030 non-null  object
 8   annual_inc            396030 non-null  float64
 9   verification_status   396030 non-null  object
 10  loan_status           396030 non-null  object
 11  purpose               396030 non-null  object
 12  dti                   396030 non-null  float64
 13  earliest_cr_line      396030 non-null  object
 14  open_acc              396030 non-null  float64
 15  pub_rec               396030 non-null  float64
 16  revol_bal             396030 non-null  float64
 17  revol_util            395754 non-null  float64
 18  total_acc             396030 non-null  float64
 19  initial_list_status   396030 non-null  object
...
 21  mort_acc              358235 non-null  float64
 22  pub_rec_bankruptcies  395495 non-null  float64
dtypes: float64(12), object(11)
memory usage: 69.5+ MB
```

In this data set, because we have a large dataset which allow us to remove all the missing data, so that here is not missing value.

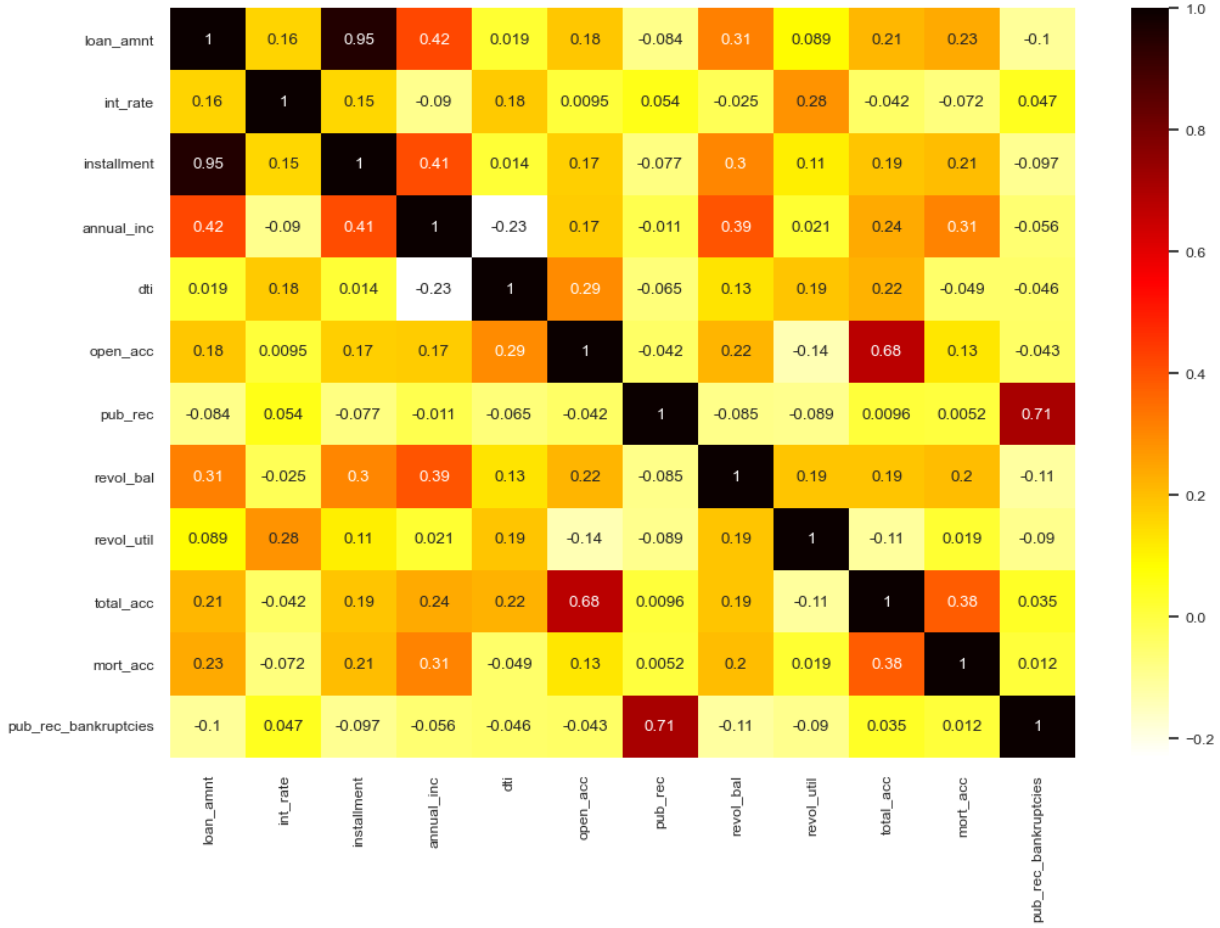
The skewness of the data set:

```
loan_amnt              0.777285
int_rate               0.420669
installment            0.983598
annual_inc            41.042725
dti                  431.051225
open_acc               1.213019
pub_rec               16.576564
revol_bal             11.727515
revol_util            -0.071778
total_acc              0.864328
mort_acc               1.600132
pub_rec_bankruptcies   3.423440
dtype: float64
```

The data is not evenly distributed, there might outliers in each of the attributes.

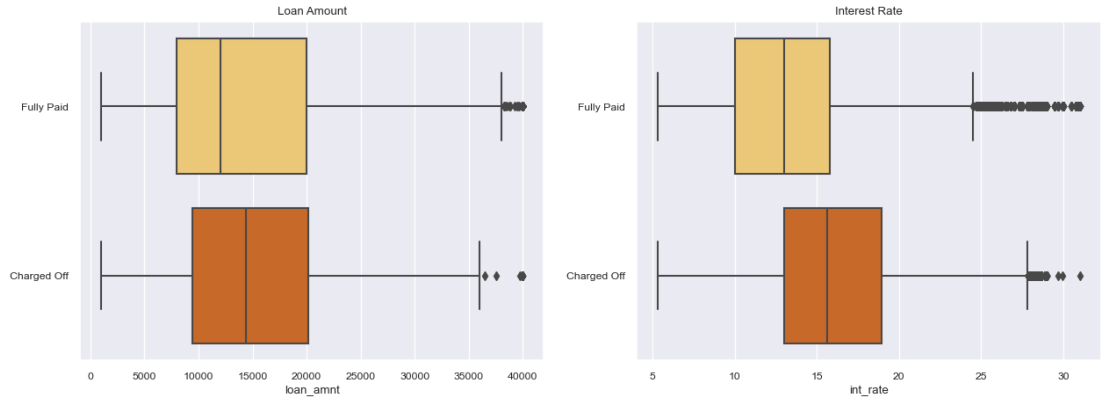
8.3 Exploratory Data Analysis (EDA)

Correlation map between all the features before encoding:



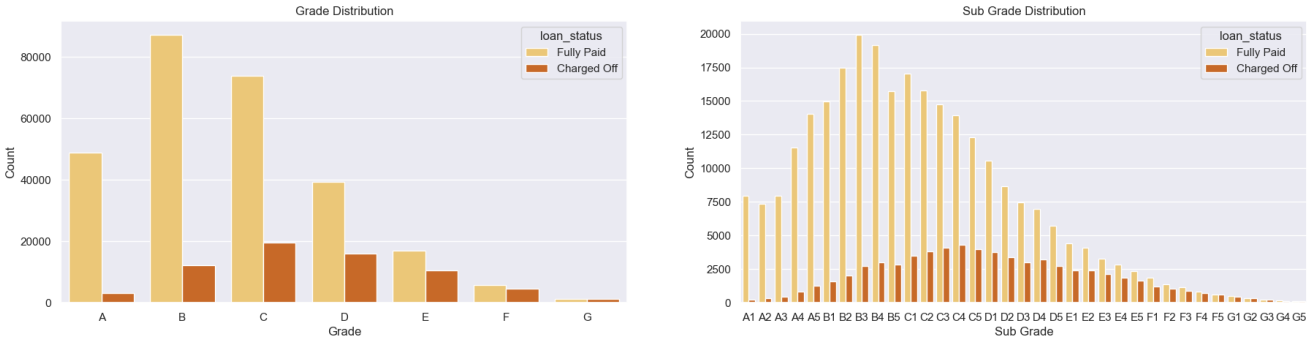
There are some of **insights** we can get from this correlation map and plot it on the next page.

Correlation between loan information and loan status:



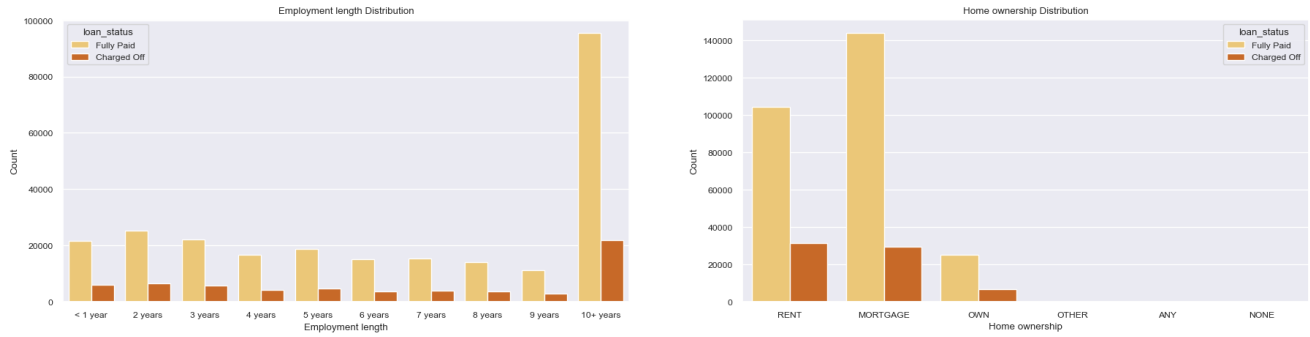
- People having a **higher loan amount** will be **not likely to fully pay the loan**.
- Loan with **lower interest rates** having **more borrowers fully paid**.

Correlation between credit rating and loan status:



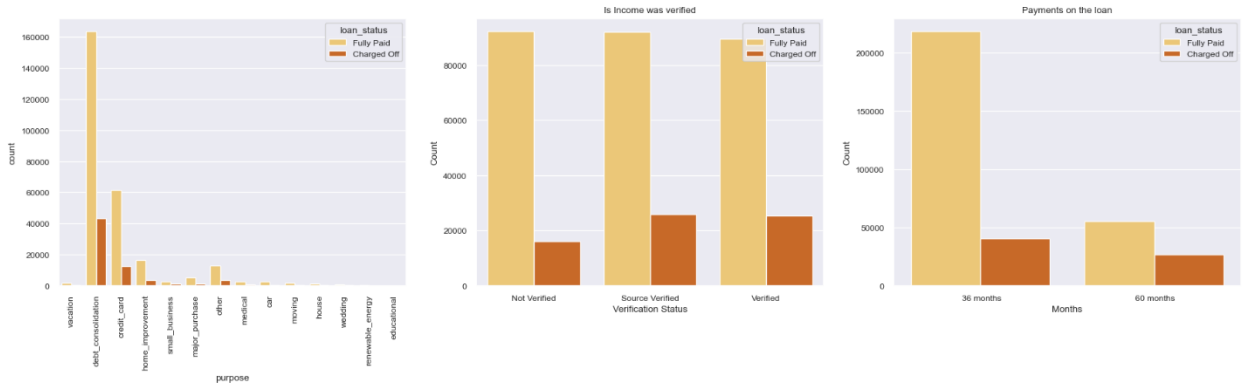
- **Grade and sub-grade** distribution can reflect the **loan status**.
- **Higher grading loans** are more likely to be **fully paid**.

Employment years and home ownership distribution:



- Most ownerships are **rent and mortgage**.
- Most people working **more than 10 years**.

Purposes, verification status and payment months:

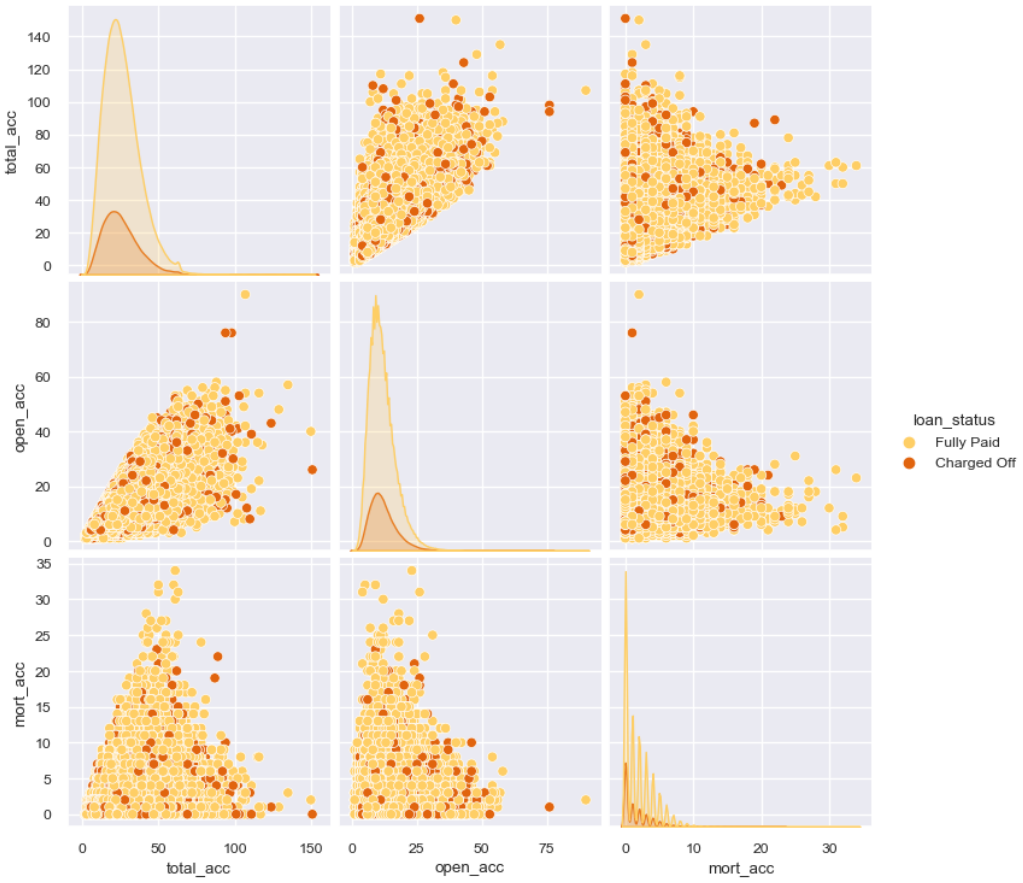


- **Debt consolidation** is the main reason of the loan.
- **Verification statuses** are **equally distributed**.
- **36 Months loans** more likely to be paid on time.

Correlation between credit rating and loan status:

- **Loan amount** is **positive correlated** with **annual income** and **credit revolving balance**.
- **Annual income** and **credit revolving balance** are **right skewed.**

Correlation between credit rating and loan status:



- The number of **mortgage and credit account** is **strong positive correlated**.
- The number of **mortgage and credit account** is **right skewed.**

8.4 Data Transformation And Train Test Split

Data selection:

Due to the size of the dataset, the computation is expensive when training and testing all the data we have. Therefore, removing the sub grade of the data and randomly choosing **20 %** will be more sufficient to build up the model.
(**20%** of the whole dataset: **60000 rows** of the data)

Label encoder:

Label encoder can encode target feature with value between 0 and total class of that feature – 1. Here, **term, sub_grade, emp_length, home_ownership, verification_status, loan_status, purpose,earliest_cr_line,initial_list_status and application_type** column are categorical data, and we need to encode it to certain range of value.

	loan_amnt	term	int_rate	installment	emp_length	home_ownership	annual_inc	verification_status	loan_status	purpose	...
317481	508	0	113	12073	1	1	4342	2	1	2	...
297148	941	1	79	14007	4	1	3586	2	1	1	...
241810	33	0	64	438	8	5	1379	1	1	5	...
144364	229	0	47	3983	2	5	5095	2	1	1	...
77880	587	0	20	12302	6	1	2694	0	1	2	...
...	...	...	...	...	...	...	...	...	...	...	...
90667	508	0	7	10369	1	1	2749	1	1	2	...
42368	269	0	91	5384	1	1	857	1	1	2	...
153822	667	0	64	14643	0	5	3085	2	0	2	...
27399	747	0	4	14910	2	1	4006	1	1	0	...
32232	460	1	90	6212	1	1	4311	1	1	5	...

60000 rows × 21 columns

Train Test Split:

Train set:

	loan_amnt	term	int_rate	installment	emp_length	home_ownership	annual_inc	verification_status	loan_status	purpose	...
115796	41	0	121	627	10	5	549	1	1	6	...
248591	396	1	127	5597	1	1	2189	2	0	2	...
153332	687	0	7	13958	1	1	4118	2	1	2	...
315494	548	0	127	13203	1	5	1743	2	1	2	...
229636	1117	1	137	17448	10	1	4673	1	1	3	...
...	...	...	...	...	...	...	...	...	...	...	...
37042	493	0	172	12756	2	5	891	1	1	2	...
340350	348	0	72	7504	1	5	4736	2	1	8	...
324708	1154	1	153	18065	1	1	4696	1	1	2	...
92797	83	0	111	1367	10	4	525	1	0	5	...
218699	587	1	153	10103	1	4	1176	1	0	5	...

54000 rows × 21 columns

Test set:

	loan_amnt	term	int_rate	installment	emp_length	home_ownership	annual_inc	verification_status	loan_status	purpose	...
334987	388	0	6	7555	10	5	4673	0	1	2	...
123344	747	1	165	13289	1	1	3790	1	0	3	...
216869	667	0	29	13945	1	5	2777	0	0	1	...
365547	0	0	72	30	10	1	613	2	1	8	...
321465	149	0	3	2076	4	5	4946	0	1	0	...
...	...	...	...	...	...	...	...	...	...	...	...
269328	69	0	110	1104	4	5	2777	2	0	2	...
341625	269	0	67	5179	4	5	1012	0	1	1	...
148315	348	0	29	6803	10	1	3187	0	1	3	...
360066	941	1	203	16833	10	5	4006	1	1	2	...
126774	396	0	124	9537	4	1	2903	2	1	8	...

6000 rows × 21 columns

In this data set, we set 9:1, 90% Data for training and 10% Data for testing.
(Total: entities **60000**, training entities:**54000**, test entities: **6000**)

Data Scaling:

Train set after standardization:

	loan_amnt	term	int_rate	installment	emp_length	home_ownership	annual_inc	verification_status	loan_status	purpose	...
0	-0.575314	1.0	0.301370	-0.816478	0.6	1.0	-0.977642	0.5	-1.0	-1.0	...
1	-0.242678	0.0	-0.219178	-0.249004	0.6	1.0	0.736179	0.5	0.0	0.0	...
2	0.209205	0.0	0.328767	0.385030	0.0	0.0	0.467480	0.5	0.0	-1.0	...
3	-0.010460	0.0	-0.246575	0.064405	-0.4	0.0	0.736179	-0.5	0.0	0.0	...
4	-0.209205	0.0	-0.410959	-0.220894	1.6	0.0	-0.568293	0.5	0.0	0.0	...
...	...	...	...	...	...	...	...	...	...	...	...
53995	0.790795	1.0	1.013699	0.540765	1.6	0.0	-0.420325	0.0	-1.0	1.0	...
53996	-0.290795	0.0	0.219178	-0.285299	1.6	1.0	-0.500407	-0.5	-1.0	0.0	...
53997	0.531381	1.0	-0.191781	0.101023	-0.2	1.0	-0.663008	0.0	0.0	0.0	...
53998	-0.209205	0.0	-0.246575	-0.203123	-0.2	1.0	-0.330894	0.5	0.0	0.0	...
53999	-0.692469	0.0	-0.273973	-0.803016	0.2	1.0	-0.302033	-0.5	0.0	0.0	...

54000 rows × 21 columns

Test set after standardization:

	loan_amnt	term	int_rate	installment	emp_length	home_ownership	annual_inc	verification_status	loan_status	purpose	...
0	-0.458159	0.0	-0.424658	-0.562520	0.4	1.0	-0.125610	-0.5	0.0	0.0	...
1	1.667364	1.0	0.506849	0.989230	-0.4	1.0	0.197967	0.5	0.0	-1.0	...
2	-0.625523	0.0	0.369863	-0.716424	-0.2	0.0	-0.961789	0.5	-1.0	0.0	...
3	0.719665	1.0	0.479452	0.370813	-0.2	0.0	-0.302033	0.5	0.0	0.0	...
4	-0.832636	0.0	0.246575	-0.916209	1.6	1.0	-0.467073	-0.5	0.0	5.0	...
...	...	...	...	...	...	...	...	...	...	...	...
5995	0.625523	0.0	-0.630137	0.662789	-0.4	1.0	0.564228	0.0	0.0	0.0	...
5996	1.600418	1.0	1.000000	1.015078	0.6	1.0	0.552846	0.0	-1.0	0.0	...
5997	0.709205	0.0	-0.410959	0.761766	1.2	1.0	-0.258943	-0.5	0.0	3.0	...
5998	1.667364	0.0	0.479452	1.159505	-0.2	0.0	0.719512	0.5	0.0	-1.0	...
5999	0.981172	1.0	0.534247	0.589876	-0.2	0.0	-0.273577	0.5	-1.0	-1.0	...

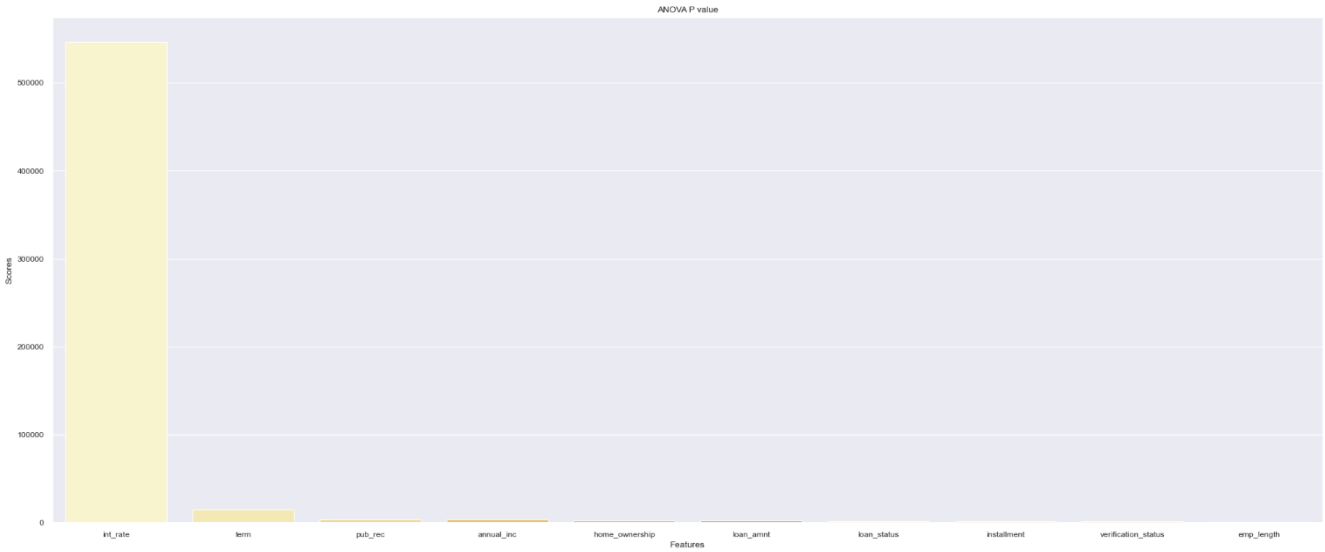
6000 rows × 21 columns

Data scaling can shift each of the features into a particular of range, so that it can break the differences in the scales across all the input variables, here we used **Robust Scaler**.

8.5 Feature Selection

Input all the features into the machine learning models not only increased the computing time, but it also increased the possibility of collinearly problem.

P value of ANOVA:



H0: Means of all credit rating groups are equal.

H1: At least one mean of credit rating groups are different

(α=0.05)

The 10 highest p-value features:

- ***int_rate (Highest P-Value)***
- *term*
- *revol_bal*
- *verification_status*
- *annual_inc*
- *loan_amnt*
- *purpose*
- *installment*
- *loan_status*
- *home_ownership*

8.6 Cross-Validation For Hyper Tuning Logistic Regression

Cross-validation:

In this report, we cross validate **3** times and using **grid search** setting scoring **"accuracy"** (Binomial Classification) for choosing highest accuracy machine learning models.

Best parameters for different models:

The best parameters for **logistic regression**:

{'C': 1, 'class_weight': 'balanced', 'multi_class': 'auto', 'penalty': 'l2', 'solver': ' newton-cg'}

8.7 Result And Findings

	Machine Learning Models	Accuracy	Training Time (s)
0	Logistic Regression	0.796333	609.961489

The Lending Club dataset is a valuable resource for understanding the loan industry and building predictive models to identify factors that influence loan repayment or default. The dataset includes information on loans issued between 2007 and 2015, including loan status, payment history, and borrower characteristics.

After preprocessing the data by removing irrelevant or missing data, we selected the top 10 features using ANOVA. We found that **interest rate had the highest impact on loan default**, followed by loan amount, annual income, credit revolving balance, and the purpose of the loan.

We then used logistic regression, a type of machine learning algorithm that models the probability of a binary outcome, to predict the likelihood of loan default. The model achieved an **accuracy of 79.63%**, which is a good result considering the complexity of the loan repayment process and the many factors that contribute to loan default.

Our analysis also revealed several key insights into the loan industry. For example, we found that borrowers with higher credit grades are more likely to repay their loans in full. We also found that loan amount is positively correlated with annual income and credit revolving balance, which suggests that borrowers with higher income and credit balances are more likely to take out larger loans.

Additionally in US, we found that **debt consolidation** is the primary reason for taking out a loan, which is consistent with other studies that have found debt consolidation to be a popular reason for borrowing. We also found that **36-month loans** are more likely to be

repaid on time than 60-month loans, which may be due to the shorter repayment period and lower interest rates.

Overall, our analysis of the Lending Club dataset provides valuable insights into the loan industry and offers a framework for building predictive models to identify factors that contribute to loan default. By using advanced machine learning techniques and incorporating additional features, our model can be further improved to provide more accurate predictions of loan default.

9. Conclusion

In conclusion, our study on logistic learning data analysis methods and their applications to business and economics, specifically in credit risk rating classification, has revealed several key takeaways.

First, logistic regression stands out as a robust choice for credit risk rating classification tasks due to its simplicity, interpretability, computational efficiency, ability to handle imbalanced datasets, and adaptability to multiple outcomes. Through our analysis of the simulated datasets (7.1 German Credit Rating and 7.2 US Corporate Credit), we demonstrated that logistic regression consistently outperforms other machine learning models, such as random forests, naive bayes, and gradient boosting, in terms of accuracy and efficiency.

Second, our exploration of the real-world Lending Club dataset showcased the practical implications of using logistic regression for credit risk prediction. After preprocessing the data and feature selection, we developed a model with an accuracy of 79.63%, which is impressive considering the complexity of the loan repayment process. Our analysis provides valuable insights into the factors influencing loan defaults, such as interest rate, loan amount, annual income, credit revolving balance, and loan purpose. We also discovered specific trends in the loan industry, such as borrowers with higher credit grades being more likely to repay loans, and 36-month loans showing higher on-time repayment rates compared to 60-month loans.

To enhance the clarity and impact of our research, future studies can focus on improving the logistic regression model by incorporating additional features, exploring alternative preprocessing techniques, and investigating the relationships between variables. Moreover, researchers can consider developing hybrid models that combine logistic regression with other machine learning algorithms to achieve

even higher prediction accuracy. By continuously refining our understanding of credit risk factors and prediction models, we can contribute to the development of more efficient and effective credit risk management strategies in business and economics.

Appendix

Appendix (1) Python Code For Simulation Graph

Simulation part in 2D-Linear Regression

```
Import numpy as np
from sklearn import datasets
import matplotlib.pyplot as plt
x, y, coef = datasets.make_regression(n_samples=100,n_feactures=1,
n_informative=1,noise=10,coef=True,random_state=0)
x = np.interp(x, (x.min(), x.max()), (0,20))
y = np.interp(y, (y.min(), y.max()), (0,100))
model = sklearn.linear_model.LinearRegression()
model.fit(x,y)
y_pred = model.predict(x)
plt.ion()
plt.plot(x,y_pred, color = 'blue', label='y=W0+W1x')
plt.plot(x,y,,'.', label='Data',color='green')
plt.xlabel('x');plt.ylabel('y')
plt.title('Linear Regression')
plt.legend()
plt.grid()
```

Simulation part in 3D-Linear Regression

```
Import matplotlib.pyplot as plt
Import numpy as np
Import sklearn.linear_model
From mpl_toolkits.mplot3d import Axes3D
X_train = np.random.rand(200).reshape(100,2)*60
y_train = (X_train[:,0]**2)+(X_train[:,1]**2)
X_test = np.random.rand(200).reshape(100,2)*60
y_test = (X_test[:,0]**2)+(X_test[:,1]**2)
fig=plt.figure()
ax = fig.add_subplot(111,projection='3d')
ax.scatter(X_train[:,0],X_train[:,1],y_train,marker= ',', color='green')
ax.set_xlabel("X1")
ax.set_xlabel("X2")
ax.set_xlabel("y")
```

```
model = sklearn.linear_model.LinearRegression()
y_pred = model.predict(X_test)
coefs = model.coef
intercept = model.intercept_
xs = np.tile(np.arange(70),(70,1))
ys = np.tile(np.arange(70),(70,1)).T
zs = xs*coefs[0]+ys*coefs[1]+intercept
ax.plot_surface(xs,ys,zx,alpha = 0.5)
plt.show()
```

Appendix (2) Python Code For German Credit Data Set

Importing Packages

```
import pandas as pd
import numpy as np
import time
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import scipy.cluster.hierarchy as shc
from sklearn.cluster import AgglomerativeClustering
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import log_loss
import warnings
warnings.filterwarnings("ignore")
data = pd.read_csv('German-Data.csv')
```

Heat Map

```
plt.figure(figsize=(45,30))
sns.set(font_scale=2)
sns.heatmap(data.corr(),cbar=True, annot=True, square=True ,fmt='.2f',cmap="rocket")
plt.show()
```

Data Structure

```
data.info()
```

Data Cleaning – Mapping the meaning of the dataset.

```
maping_list = {1:'<0 DM',
               2:'0~200 DM',
               3:'>=200 DM',
               4:'No account'
               }
data['Account Balance'] = data['Account Balance'].map(maping_list)


maping_list = {0:'no credits taken / all credits paid back duly',
               1:'all credits at this bank paid back duly',
               2:'existing credits paid back duly till now',
               3:'delay in paying off in the past',
               4:'critical account / other credits existing (not at this bank)'
               }
data['Payment Status of Previous Credit'] = data['Payment Status of Previous
Credit'].map(maping_list)

maping_list = {0:'car (new)',
               1:'car (used)',
               2:'furniture/equipment',
               3:'radio/television',
               4:'domestic appliances',
               5:'repairs',
               6:'education',
               7:'vacation',
               8:'retraining',
               9:'business',
               10:'others',
```

```
                    }
data['Purpose'] = data['Purpose'].map(maping_list)

maping_list = {1:'<100 DM',
                    2:'100~500 DM',
                    3:'500~1000 DM',
                    4:'>=1000 DM',
                    5:'unknown/ no savings account'
                    }
data['Value Savings/Stocks'] = data['Value Savings/Stocks'].map(maping_list)

maping_list = {1:'unemployed',
                    2:'< 1 year',
                    3:'1~4 year',
                    4:'4~7 year',
                    5:'>= 7 year'
                    }
data['Length of current employment'] = data['Length of current employment'].map(maping_list)

maping_list = {1:'male: divorced/separated/married',
                    2:'female: divorced/separated/married',
                    3:'male: single',
                    4:'female: single',
                    }
data['Sex & Marital Status'] = data['Sex & Marital Status'].map(maping_list)

maping_list = {1:'none',
                    2:'co-applicant',
                    3:'guarantor',
                    }
data['Guarantors'] = data['Guarantors'].map(maping_list)

maping_list = {1:'real estate',
                    2:'building society savings agreement/life insurance',
                    3:'car or other',
                    4:'unknown/no property'
                    }
data['Most valuable available asset'] = data['Most valuable available asset'].map(maping_list)
```

```
maping_list = {1:'bank',
               2:'stores',
               3:'none',
               }
data['Concurrent Credits'] = data['Concurrent Credits'].map(maping_list)

maping_list = {1:'rent',
               2:'own',
               3:'for free',
               }
data['Type of apartment'] = data['Type of apartment'].map(maping_list)

maping_list = {1:'unemployed',
               2:'unskilled-resident',
               3:'skilled employee/official',
               4:'management/self-employed/highly qualified employee/officer',
               }
data['Occupation'] = data['Occupation'].map(maping_list)

maping_list = {1:'none',
               2:'yes',
               }
data['Telephone'] = data['Telephone'].map(maping_list)

maping_list = {1:'yes',
               2:'no',
               }
data['Foreign Worker'] = data['Foreign Worker'].map(maping_list)

maping_list = {1:'Low Credit Risk',
               0:'High Credit Risk',
               }
data['Creditability'] = data['Creditability'].map(maping_list)
```

Data visualizations

```
sns.set(font_scale=1)
order =['<0 DM','0~200 DM','>=200 DM','No account']
sns.countplot(data=data, y='Account Balance',hue='Creditability',order=order,palette="rocket")
plt.title('Correlation between Account Balance and Creditability')

sns.set(font_scale=1)
sns.displot(data=data, x="Duration of Credit (month)",
hue="Creditability",kind='kde',fill=True,palette="rocket")
plt.title('Density between Duration of Credit (month) and Creditability')

sns.set(font_scale=1)
order =['existing credits paid back duly till now','all credits at this bank paid back duly',
        'no credits taken / all credits paid back duly','critical account / other credits existing (not
at this bank)',
        'delay in paying off in the past']
sns.countplot(data=data, y='Payment Status of Previous
Credit',hue='Creditability',order=order,palette="rocket")
plt.title('Correlation between Payment Status of Previous Credit and Creditability',fontsize=16)

sns.set(font_scale=1)
order =['unknown/ no savings account','<100 DM','100~500 DM','500~1000 DM','>=1000 DM']
hue_order =['No account','<0 DM','0~200 DM','>=200 DM']
sns.countplot(data=data, y='Value Savings/Stocks',hue='Account
Balance',palette="rocket",order=order,hue_order=hue_order)
plt.title('Correlation between Value Savings/Stocks and Account Balance',fontsize=16)

sns.jointplot(data=data,x="Duration of Credit (month)", y="Credit Amount",
hue="Creditability",kind="kde",palette="rocket")

order=['real estate','building society savings agreement/life insurance','car or other','unknown/no
property']
sns.boxplot(data=data,y='Most valuable available asset',x='Duration of Credit (month)',
hue="Creditability",order=order,palette="rocket")
plt.title('Correlation between Most valuable available asset and Duration of Credit',fontsize=16)

sns.countplot(data=data, x="No of Credits at this Bank", hue="Payment Status of Previous
Credit",palette="rocket")
```

```
plt.title('Correlation between No of Credits at this Bank and Payment Status of Previous Credit',fontsize=16)

sns.jointplot(data=data,x="Instalment per cent", y="Credit Amount", hue="Creditability",kind="kde",palette="rocket")

order=['real estate','building society savings agreement/life insurance','car or other','unknown/no property']
sns.boxplot(data=data,y='Most valuable available asset',x='Credit Amount',hue="Creditability",order=order,palette="rocket")
plt.title('Correlation between Instalment per cent and Credit Amount',fontsize=16)

order=['unemployed','unskilled-resident','skilled employee/official','management/self-employed/highly qualified employee/officer']
sns.boxplot(data=data,y='Occupation',x='Credit Amount',hue="Creditability",order=order,palette="rocket")
plt.title('Correlation between Occupation and Credit Amount',fontsize=16)

sns.set(font_scale=1)
order =['real estate','building society savings agreement/life insurance','car or other','unknown/no property']
sns.countplot(data=data, y='Most valuable available asset',hue='Type of apartment',palette="rocket",order=order)
plt.title('Correlation between Most valuable available asset and type of apartment',fontsize=14)

sns.set(font_scale=1)
order=['unemployed','unskilled-resident','skilled employee/official','management/self-employed/highly qualified employee/officer']
hue_order =['real estate','building society savings agreement/life insurance','car or other','unknown/no property']
sns.countplot(data=data, hue='Most valuable available asset',y='Occupation',palette="rocket",hue_order=hue_order,order=order)
plt.title('Correlation between Value Savings/Stocks and Account Balance')

sns.displot(data=data,hue="Type of apartment", x="Age (years)",kind='kde', fill=True,palette="rocket")
plt.title('Density between Age (years) and Type of apartment')
```

Data Encode

```
corr = data.copy()

le1 = LabelEncoder()
le1.fit(corr['Account Balance'])
corr['Account Balance'] = le1.transform(corr['Account Balance'])

le2 = LabelEncoder()
le2.fit(corr['Payment Status of Previous Credit'])
corr['Payment Status of Previous Credit'] = le2.transform(corr['Payment Status of Previous Credit'])

le3 = LabelEncoder()
le3.fit(corr['Purpose'])
corr['Purpose'] = le3.transform(corr['Purpose'])

le4 = LabelEncoder()
le4.fit(corr['Value Savings/Stocks'])
corr['Value Savings/Stocks'] = le4.transform(corr['Value Savings/Stocks'])

le5 = LabelEncoder()
le5.fit(corr['Length of current employment'])
corr['Length of current employment'] = le5.transform(corr['Length of current employment'])

le6 = LabelEncoder()
le6.fit(corr['Sex & Marital Status'])
corr['Sex & Marital Status'] = le6.transform(corr['Sex & Marital Status'])

le7 = LabelEncoder()
le7.fit(corr['Guarantors'])
corr['Guarantors'] = le7.transform(corr['Guarantors'])

le8 = LabelEncoder()
le8.fit(corr['Most valuable available asset'])
corr['Most valuable available asset'] = le8.transform(corr['Most valuable available asset'])

le9 = LabelEncoder()
le9.fit(corr['Concurrent Credits'])
corr['Concurrent Credits'] = le9.transform(corr['Concurrent Credits'])
```

```
le10 = LabelEncoder()
le10.fit(corr['Type of apartment'])
corr['Type of apartment'] = le10.transform(corr['Type of apartment'])

le11 = LabelEncoder()
le11.fit(corr['Occupation'])
corr['Occupation'] = le11.transform(corr['Occupation'])

le12 = LabelEncoder()
le12.fit(corr['Telephone'])
corr['Telephone'] = le12.transform(corr['Telephone'])

le13 = LabelEncoder()
le13.fit(corr['Foreign Worker'])
corr['Foreign Worker'] = le13.transform(corr['Foreign Worker'])

le14 = LabelEncoder()
le14.fit(corr['Creditability'])
corr['Creditability'] = le14.transform(corr['Creditability'])
```

Train Test data split

```
X=corr.iloc[:,1:]
y=corr['Creditability']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
scaler = RobustScaler()
X_train = pd.DataFrame(
    scaler.fit_transform(X_train),
    columns = X_train.columns
)

X_test = pd.DataFrame(
    scaler.transform(X_test),
    columns = X_test.columns
)
```

Feature selection by ANOVA

```
fs = SelectKBest(score_func=f_classif, k='all')
fs.fit(X_train, y_train)
features_list = []
scores_list = []
for i in range(len(fs.scores_)):
  features_list.append(corr.iloc[:,1:].columns[i])
  scores_list.append(fs.scores_[i])
df_features = pd.DataFrame({'Features': features_list, 'Scores': scores_list})
df_features = df_features.sort_values(by=['Scores'],ascending=False)
sns.set(font_scale=1)
plt.figure(figsize=(25,10))
sns.barplot(x='Features', y='Scores', data=df_features[0:10],palette="rocket")
plt.title("ANOVA P value")

X=corr.loc[:,['Account Balance',
'Duration of Credit (month)','Credit Amount','Age (years)','Value Savings/Stocks','Sex & Marital
Status',
'Payment Status of Previous Credit',
'Foreign Worker','Purpose','Instalment per cent']]
y=corr['Creditability']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

scaler = RobustScaler()
X_train = pd.DataFrame(
      scaler.fit_transform(X_train),
      columns = X_train.columns
)

X_test = pd.DataFrame(
      scaler.transform(X_test),
      columns = X_test.columns
)
Runtime = []
```

Logistic Regression train and test

```
start = time.time()
logr = LogisticRegression()
param_grid_logr = [
      {'penalty':['l2','elasticnet','none'],
      'C':[0.001, 0.01, 0.1,1,10,100],
      'solver':['lbfgs','newton-cg','liblinear','sag','saga'],
      'multi_class':['auto','ovr','multinomial'],
      'class_weight':['balanced']
      }
]
param_logr = GridSearchCV(logr, param_grid = param_grid_logr, cv = 3, verbose=True,scoring='f1')
best_param_logr = param_logr.fit(X_train,y_train)
end = time.time()
Runtime.append(end - start)
print("The best parameters for logistic regression",best_param_logr.best_params_)

logr = LogisticRegression(C =
0.01 ,multi_class='auto',penalty='l2',solver='lbfgs',class_weight='balanced')
logr.fit(X_train,y_train)
logr_Prediction = logr.predict(X_test)
Accuracy_Logistic = accuracy_score(y_test,logr_Prediction)
print("The accuracy score of logisitc regression is:",Accuracy_Logistic)
```

Naive Bayes test

```
start = time.time()
gnb = GaussianNB()
gnb.fit(X_train, y_train)
gnb_Prediction = gnb.predict(X_test)
Accuracy_NaiveBayes = accuracy_score(y_test,gnb_Prediction)
end = time.time()
Runtime.append(end - start)
print("The accuracy score of Naive Bayes Classifier is:",Accuracy_NaiveBayes)
```

Support vector machine train and test

```
start = time.time()
svm = SVC()
param_grid_svm = [
```

```
        {
        'kernel':['rbf','linear','sigmoid'],
        'gamma':['scale', 'auto'],
        'class_weight':['dict' , 'balanced'],
        'decision_function_shape':['ovo', 'ovr'],
        }
]
param_svm = GridSearchCV(svm, param_grid = param_grid_svm, cv = 3, verbose=True,scoring="f1")
best_param_svm = param_svm.fit(X_train,y_train)
end = time.time()
Runtime.append(end - start)
print("The best parameters for support vector machine",best_param_svm.best_params_)

svm = SVC(kernel='rbf',class_weight='balanced',decision_function_shape='ovo',gamma='scale')
svm.fit(X_train, y_train)
svm_Prediction = svm.predict(X_test)
Accuracy_SVM = accuracy_score(y_test,svm_Prediction)
print("The accuracy score of support vector machine is:",Accuracy_SVM)
```

Random forest train and test

```
start = time.time()
rf = RandomForestClassifier()
param_grid_rf = [
        {'n_estimators':[100,200,400,600],
        'criterion':['gini', 'entropy', 'log_loss'],
        'max_features':['sqrt', 'log2', None],
        'class_weight':['balanced', 'balanced_subsample'],
        'random_state':[42],
        }
]
param_rf = GridSearchCV(rf, param_grid = param_grid_rf, cv = 3, verbose=True,scoring="f1")
best_param_rf = param_rf.fit(X_train,y_train)
end = time.time()
Runtime.append(end - start)
print("The best parameters for random forest",best_param_rf.best_params_)
```

```
rf_Model =
RandomForestClassifier(class_weight='balanced',criterion='entropy',max_features='sqrt',n_estimat
ors=200,random_state=42)
rf_Model.fit(X_train, y_train)
rf_Prediction = rf_Model.predict(X_test)
Accuracy_RandomForest = accuracy_score(y_test,rf_Prediction)
print("The accuracy score of random forest is:",Accuracy_RandomForest)
```

Gradient boosting train and test

```
start = time.time()
gb = GradientBoostingClassifier()
param_grid_gb = [
    {
    'loss':['log_loss', 'deviance', 'exponential'],
    'learning_rate':[0.1,0.01,0.001],
    'n_estimators':[100,200,400,600,800],
    'subsample':[0.0,0.5,1],
    'criterion':['friedman_mse', 'squared_error', 'mse'],
    }
]
param_gb = GridSearchCV(gb, param_grid = param_grid_gb, cv = 3, verbose=True,scoring="f1")
best_param_gb = param_gb.fit(X_train,y_train)
end = time.time()
Runtime.append(end - start)
print("The best parameters for Gradient Boosting",best_param_gb.best_params_)

gb_Model = GradientBoostingClassifier(criterion = 'squared_error',learning_rate=0.01,
loss='deviance',n_estimators=200,subsample=0.5)
gb_Model.fit(X_train, y_train)
gb_Prediction = gb_Model.predict(X_test)
Accuracy_gb = accuracy_score(y_test,gb_Prediction)
print("The accuracy score of Gradient Boosting is:",Accuracy_gb)
```

Models' comparison

```
model_list = ['Logistic Regression', 'Naive Bayes','Support Vector Machine','Random
forest','Gradient Boosting']
accuracy_list = [Accuracy_Logistic, Accuracy_NaiveBayes, Accuracy_SVM, Accuracy_RandomForest,
Accuracy_gb]
```

```
df_models = pd.DataFrame({'Machine Learning Models': model_list, 'Accuracy': accuracy_list,
'Training Time (s)': Runtime})
df_models
```

Appendix (3) Python Code For US corporate Credit Data Set

Importing Packages

```
import pandas as pd
import numpy as np
import time
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import log_loss
import warnings
warnings.filterwarnings("ignore")
data = pd.read_csv('Corporate-Data.csv')
```

Data structure

```
data.head()
```

Data visualization before clustering

```
sns.set(font_scale=1) # size of font
ax=sns.countplot(data=data,
x='Rating',order=['AAA','AA','A','BBB','BB','B','CCC','CC','C','D'],palette="YlGnBu")
plt.title('Credit Rating')

fig, axes = plt.subplots(1, 4,figsize=(35,5))
axes = axes.flatten()
#sns.set_theme(style = "whitegrid")
order=['AAA','AA','A','BBB','BB','B','CCC','CC','C','D']
sns.boxplot(ax =
axes[0],x='currentRatio',y='Rating',data=data,palette="ch:.25",order=order).set(xlabel="Current
Ratio",ylabel="")
sns.boxplot(ax =
axes[1],x='quickRatio',y='Rating',data=data,palette="ch:.25",order=order).set(xlabel="Quick
Ratio",ylabel="")
sns.boxplot(ax =
axes[2],x='cashRatio',y='Rating',data=data,palette="ch:.25",order=order).set(xlabel="Cash
Ratio",ylabel="")
sns.boxplot(ax =
axes[3],x='daysOfSalesOutstanding',y='Rating',data=data,palette="ch:.25",order=order).set(
    xlabel="Day Of Sales Outstanding",ylabel="")
fig.suptitle("Liquidity Measurement Ratios",fontsize=24)

fig, axes = plt.subplots(3, 3,figsize=(35,15))
axes = axes.flatten()

order=['AAA','AA','A','BBB','BB','B','CCC','CC','C','D']
sns.boxplot(ax =
axes[0],x='grossProfitMargin',y='Rating',data=data,palette="YlGnBu",order=order).set(
    xlabel="Gross Profit Margin",ylabel="")
sns.boxplot(ax =
axes[1],x='operatingProfitMargin',y='Rating',data=data,palette="ch:.25",order=order).set(
    xlabel="Opearting Profit Margin",ylabel="")
```

```
sns.boxplot(ax =
axes[2],x='pretaxProfitMargin',y='Rating',data=data,palette="ch:.25",order=order).set(
    xlabel="Pretax Profit Margin",ylabel="")
sns.boxplot(ax =
axes[3],x='netProfitMargin',y='Rating',data=data,palette="ch:.25",order=order).set(
    xlabel="Net Profit Margin",ylabel="")
sns.boxplot(ax =
axes[4],x='effectiveTaxRate',y='Rating',data=data,palette="ch:.25",order=order).set(
    xlabel="Effective Tax Rate",ylabel="")
sns.boxplot(ax = axes[5],x='returnOnAssets',y='Rating',data=data,palette="ch:.25",order=order).set(
    xlabel="Return On Assets",ylabel="")
sns.boxplot(ax = axes[6],x='returnOnEquity',y='Rating',data=data,palette="ch:.25",order=order).set(
    xlabel="Return On Equity",ylabel="")
sns.boxplot(ax =
axes[7],x='returnOnCapitalEmployed',y='Rating',data=data,palette="ch:.25",order=order).set(
    xlabel="Return On Capital Employed",ylabel="")
sns.boxplot(ax =
axes[8],x='ebitPerRevenue',y='Rating',data=data,palette="ch:.25",order=order).set(
    xlabel="Ebit per revenue",ylabel="")
fig.suptitle("Profitability Indicator Ratios",fontsize=28)

fig, axes = plt.subplots(1,3,figsize=(22,5))
axes = axes.flatten()

order=['AAA','AA','A','BBB','BB','B','CCC','CC','C','D']
sns.boxplot(ax = axes[0],x='debtRatio',y="Rating",data=data,palette="YlGnBu",order=order).set(
    xlabel="Debt Ratio",ylabel="")
sns.boxplot(ax =
axes[1],x='debtEquityRatio',y="Rating",data=data,palette="ch:.25",order=order).set(
    xlabel="Debt Equity Ratio",ylabel="")
sns.boxplot(ax =
axes[2],x='payablesTurnover',y="Rating",data=data,palette="ch:.25",order=order).set(
    xlabel="Payables Turnover",ylabel="")
fig.suptitle("Debt Ratios",fontsize=28)

fig, axes = plt.subplots(1,2,figsize=(22,5))
axes = axes.flatten()
order=['AAA','AA','A','BBB','BB','B','CCC','CC','C','D']
```

```
sns.boxplot(ax = axes[0],x='assetTurnover',y='Rating',data=data,palette="ch:.25",order=order).set(
      xlabel='Asset Turnover',ylabel="")
sns.boxplot(ax =
axes[1],x='enterpriseValueMultiple',y="Rating",data=data,palette="ch:.25",order=order).set(
      xlabel="Enterprise Value Multiple",ylabel="")
fig.suptitle("Opearting Performance Ratio",fontsize=20)

fig, axes = plt.subplots(2,3,figsize=(35,12))
axes = axes.flatten()
order=['AAA','AA','A','BBB','BB','B','CCC','CC','C','D']
sns.boxplot(ax =
axes[0],x='operatingCashFlowPerShare',y='Rating',data=data,palette="ch:.25",order=order).set(
      xlabel="Operating Cash Flow Per Share",ylabel="")
sns.boxplot(ax =
axes[1],x='freeCashFlowPerShare',y='Rating',data=data,palette="ch:.25",order=order).set(
      xlabel="Free Cash Flow Per Share",ylabel="")
sns.boxplot(ax = axes[2],x='cashPerShare',y='Rating',data=data,palette="ch:.25",order=order).set(
      xlabel="Cash Per Share",ylabel="")
sns.boxplot(ax =
axes[3],x='operatingCashFlowSalesRatio',y='Rating',data=data,palette="ch:.25",order=order).set(
      xlabel="Operating Cash Flow Sales Per Ratio",ylabel="")
sns.boxplot(ax =
axes[4],x='freeCashFlowOperatingCashFlowRatio',y='Rating',data=data,palette="ch:.25",order=orde
r).set(
      xlabel="Free Cash Flow Operating Cash Flow Ratio",ylabel="")
fig.suptitle("Cash Flow Indicator Ratios",fontsize=24)
fig.delaxes(axes[5])

figsize=(22,5)
order=['AAA','AA','A','BBB','BB','B','CCC','CC','C','D']
sns.boxplot(x='companyEquityMultiplier',y='Rating',data=data,palette="ch:.25",order=order).set(
      xlabel='Company equity multiplier',ylabel="")
plt.title("Risk Ratio",fontsize=20)
```

Heat map

```
corr = data.iloc[:,6:31].corr()
plt.figure(figsize=(45,30))
sns.set(font_scale=2)
sns.heatmap(corr,cbar=True, annot=True, square=True ,fmt='.2f',cmap="YlGnBu")
plt.title("Original Heat Map",fontsize=60)
plt.show()
```

Heat map direct delete all the outliers.

```
df_RemoveOutlier = data.copy()
for j in df_RemoveOutlier.columns[6:31]:
    q1 = df_RemoveOutlier[j].quantile(0.25)
    q3 = df_RemoveOutlier[j].quantile(0.75)
    iqr = q3 - q1
    lower_fence    = q1-1.5*iqr
    higher_fence = q3+1.5*iqr
    for i in range(len(df_RemoveOutlier)):
        if data.loc[i,j] < lower_fence or data.loc[i,j] > higher_fence:
            df_RemoveOutlier.loc[i,j] = None
        else:
            df_RemoveOutlier.loc[i,j] = df_RemoveOutlier.loc[i,j]
df_RemoveOutlier = df_RemoveOutlier.dropna()

corr = df_RemoveOutlier.iloc[:,6:31].corr()
plt.figure(figsize=(45,30))
sns.set(font_scale=2)
sns.heatmap(corr,cbar=True, annot=True, square=True ,fmt='.2f',cmap="YlGnBu")
plt.title("Heat Map After Delete All The Outlier Data",fontsize=60)
plt.show()
```

Heat map direct fill in all the outliers by IQR.

```
df_fillin = data.copy()
for j in df_fillin.columns[6:31]:
    q1 = df_fillin[j].quantile(0.25)
    q3 = df_fillin[j].quantile(0.75)
    iqr = q3 - q1
    lower_fence    = q1-1.5*iqr
    higher_fence = q3+1.5*iqr
    for i in range(len(df_fillin)):
        if data.loc[i,j] < lower_fence:
            df_fillin.loc[i,j] = lower_fence
        elif data.loc[i,j] > higher_fence:
            df_fillin.loc[i,j] = higher_fence
        else:
            df_fillin.loc[i,j] = df_fillin.loc[i,j]

corr = df_fillin.iloc[:, 6:31].corr()
plt.figure(figsize=(45,30))
sns.set(font_scale=2)
sns.heatmap(corr,cbar=True, annot=True, square=True ,fmt='.2f',cmap="YlGnBu")
plt.title("Heat Map After Filling In Outlier Data",fontsize=60)
plt.show()
```

K-Means clustering the data.

```
kmeans_data = df_fillin.iloc[:,5:]
le0 = LabelEncoder()
le0.fit(kmeans_data['Sector'])
kmeans_data['Sector'] = le0.transform(kmeans_data['Sector'])
scaler_st = RobustScaler()
X_std = scaler_st.fit_transform(kmeans_data)
pca = PCA()
pca.fit(X_std)
scores_pca=pca.transform(X_std)
wcss=[]
for i in range(1,21):
    kmeans_pca = KMeans(n_clusters= i, init='k-means++',random_state=42)
    kmeans_pca.fit(scores_pca)
    wcss.append(kmeans_pca.inertia_)
```

```
plt.figure(figsize=(10,5))
sns.set(font_scale=1)
plt.plot(range(1,21),wcss,marker='o',linestyle ='--')
plt.xlabel('Number of Cluster')
plt.ylabel('WCSS')
plt.title('Kmeans with PCA clustering')
plt.show()

figsize=(15,5)
kmeans_pca = KMeans(n_clusters=3, init='k-means++',random_state=42)
kmeans_pca.fit(scores_pca)
pcadf = pd.DataFrame(scores_pca)
df_x_pca_kmeans = pd.concat([df_fillin.reset_index(drop=True),pcadf],axis=1)
df_x_pca_kmeans.columns.values[31:35]=['Component 1','Component 2','Component
3','Component 4']
df_x_pca_kmeans['Segment K-means PCA'] = kmeans_pca.labels_
df_x_pca_kmeans['KmeansGroup'] = df_x_pca_kmeans['Segment K-means PCA']
df_x_pca_kmeans['KmeansGroup'] = df_x_pca_kmeans['KmeansGroup'] +1
x_axis = df_x_pca_kmeans['Component 2']
y_axis = df_x_pca_kmeans['Component 1']
plt.figure(figsize=(10,5))
sns.set(font_scale=1)
hue_order=[1,2,3,4]
sns.scatterplot(x=x_axis,y=y_axis, hue =
df_x_pca_kmeans['KmeansGroup'],hue_order=hue_order,palette="YlGnBu")
plt.title('Cluster by PCA components')
plt.show()
```

Mapping the data using K-Means result

```
rating_maping_list = {'AAA':'Lowest Risk',
                      'AA':'Low Risk','A':'Low Risk',
                      'BBB':'Medium Risk',
                      'BB':'High Risk','B':'High Risk',
                      'CCC':'Highest Risk', 'CC':'Highest Risk','C':'Highest Risk',
                      'D':'In Default'}

df_fillin_prelevel['Rating'] = df_fillin_prelevel['Rating'].map(rating_maping_list)
```

Data visualization after clustering

```
fig, axes = plt.subplots(1, 4,figsize=(35,5))
axes = axes.flatten()
order=['Low Risk','Medium Risk','High Risk']
sns.boxplot(ax =
axes[0],x='currentRatio',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Current Ratio",ylabel="")
sns.boxplot(ax = axes[1],x='quickRatio',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Quick Ratio",ylabel="")
sns.boxplot(ax = axes[2],x='cashRatio',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Cash Ratio",ylabel="")
sns.boxplot(ax =
axes[3],x='daysOfSalesOutstanding',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Day Of Sales Outstanding",ylabel="")
fig.suptitle("Liquidity Measurement Ratios After Cleaning",fontsize=24)

fig, axes = plt.subplots(3, 3,figsize=(35,15))
axes = axes.flatten()
order=['Low Risk','Medium Risk','High Risk']
sns.boxplot(ax =
axes[0],x='grossProfitMargin',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Gross Profit Margin",ylabel="")
sns.boxplot(ax =
axes[1],x='operatingProfitMargin',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Opearting Profit Margin",ylabel="")
sns.boxplot(ax =
axes[2],x='pretaxProfitMargin',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Pretax Profit Margin",ylabel="")
sns.boxplot(ax =
axes[3],x='netProfitMargin',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Net Profit Margin",ylabel="")
sns.boxplot(ax =
axes[4],x='effectiveTaxRate',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Effective Tax Rate",ylabel="")
sns.boxplot(ax =
axes[5],x='returnOnAssets',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Return On Assets",ylabel="")
```

```
sns.boxplot(ax =
axes[6],x='returnOnEquity',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Return On Equity",ylabel="")
sns.boxplot(ax =
axes[7],x='returnOnCapitalEmployed',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Return On Capital Employed",ylabel="")
sns.boxplot(ax =
axes[8],x='ebitPerRevenue',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Ebit per revenue",ylabel="")
fig.suptitle("Profitability Indicator Ratios After Cleaning",fontsize=28)

fig, axes = plt.subplots(1,3,figsize=(22,5))
axes = axes.flatten()
order=['Low Risk','Medium Risk','High Risk']
sns.boxplot(ax = axes[0],x='debtRatio',y="Rating",data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Debt Ratio",ylabel="")
sns.boxplot(ax =
axes[1],x='debtEquityRatio',y="Rating",data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Debt Equity Ratio",ylabel="")
sns.boxplot(ax =
axes[2],x='payablesTurnover',y="Rating",data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Payables Turnover",ylabel="")
fig.suptitle("Debt Ratios After Cleaning",fontsize=28)

fig, axes = plt.subplots(1,2,figsize=(22,5))
axes = axes.flatten()
order=['Low Risk','Medium Risk','High Risk']
sns.boxplot(ax =
axes[0],x='assetTurnover',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel='Asset Turnover',ylabel="")
sns.boxplot(ax =
axes[1],x='enterpriseValueMultiple',y="Rating",data=df_fillin,palette="YlGnBu",order=order).set(
    xlabel="Enterprise Value Multiple",ylabel="")
fig.suptitle("Opearting Performance Ratio",fontsize=20)

fig, axes = plt.subplots(2,3,figsize=(35,12))
axes = axes.flatten()
order=['Low Risk','Medium Risk','High Risk']
```

```
sns.boxplot(ax =
axes[0],x='operatingCashFlowPerShare',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
      xlabel="Operating Cash Flow Per Share",ylabel="")
sns.boxplot(ax =
axes[1],x='freeCashFlowPerShare',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
      xlabel="Free Cash Flow Per Share",ylabel="")
sns.boxplot(ax =
axes[2],x='cashPerShare',y='Rating',data=df_fillin,palette="YlGnBu",order=order).set(
      xlabel="Cash Per Share",ylabel="")
sns.boxplot(ax =
axes[3],x='operatingCashFlowSalesRatio',y='Rating',data=df_fillin,palette="YlGnBu",order=order).se
t(
      xlabel="Operating Cash Flow Sales Per Ratio",ylabel="")
sns.boxplot(ax =
axes[4],x='freeCashFlowOperatingCashFlowRatio',y='Rating',data=df_fillin,palette="YlGnBu",order=
order).set(
      xlabel="Free Cash Flow Operating Cash Flow Ratio",ylabel="")
fig.suptitle("Cash Flow Indicator Ratios",fontsize=24)
fig.delaxes(axes[5])

figsize=(22,5)
order=['Low Risk','Medium Risk','High Risk']
sns.boxplot(x='companyEquityMultiplier',y='Rating',data=df_fillin,palette="YlGnBu",order=order).s
et(
      xlabel='Company equity multiplier',ylabel="")
plt.title("Risk Ratio",fontsize=20)

plt.figure(figsize=(7,10))
colors = ['mediumspringgreen','mediumaquamarine','aquamarine','turquoise',
'lightseagreen','azure','lightcyan','paleturquoise','aqua','lightblue','lightskyblue','blue']
explode = (0,0,0.3,0,0,0,0.1,0,0.2,0,0,0.2)
df_fillin['Sector'].str.get_dummies().sum().plot.pie(autopct='%.0f%%',label=False,colors=colors,e
xplode=explode)
plt.ylabel("")
```

Data Encode

```
le1 = LabelEncoder()
le1.fit(df_fillin['Rating'])
df_fillin['Rating'] = le1.transform(df_fillin['Rating'])

le2 = LabelEncoder()
le2.fit(df_fillin['Sector'])
df_fillin['Sector'] = le2.transform(df_fillin['Sector'])

X=df_fillin.iloc[:,5:]
y=df_fillin['Rating']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

scaler = RobustScaler()
X_train = pd.DataFrame(
        scaler.fit_transform(X_train),
        columns = X_train.columns
)

X_test = pd.DataFrame(
        scaler.transform(X_test),
        columns = X_test.columns
)
```

Feature Selection

```
fs = SelectKBest(score_func=f_classif, k='all')
fs.fit(X_train, y_train)
features_list = []
scores_list = []
for i in range(len(fs.scores_)):
  features_list.append(df_fillin.iloc[:,5:].columns[i])
  scores_list.append(fs.scores_[i])
df_features = pd.DataFrame({'Features': features_list, 'Scores': scores_list})
df_features = df_features.sort_values(by=['Scores'],ascending=False)
plt.figure(figsize=(25,10))
sns.barplot(x='Features', y='Scores', data=df_features[0:10],palette="YlGnBu")
plt.title("ANOVA P value")
```

Train Test data split

```
X=df_fillin.loc[:,['returnOnAssets','returnOnCapitalEmployed','operatingCashFlowPerShare','pretaxProfitMargin',
'netProfitMargin','ebitPerRevenue','debtRatio','operatingProfitMargin','returnOnEquity','freeCashFlowPerShare']]
y=df_fillin['Rating']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

scaler = RobustScaler()
X_train = pd.DataFrame(
    scaler.fit_transform(X_train),
    columns = X_train.columns
)

X_test = pd.DataFrame(
    scaler.transform(X_test),
    columns = X_test.columns
)
Runtime = []
```

Logistic Regression train and test

```
start = time.time()

logr = LogisticRegression()
param_grid_logr = [
    {'penalty':['l2','elasticnet','none'],
    'C':[0.001,0.01,0.1,1,10,20,50,100],
    'solver':['lbfgs','newton-cg','liblinear','sag','saga'],
    'multi_class':['auto','ovr','multinomial'],
    'class_weight':['balanced']
    }
]
param_logr = GridSearchCV(logr, param_grid = param_grid_logr, cv = 3,
verbose=True,scoring='accuracy')
best_param_logr = param_logr.fit(X_train,y_train)
end = time.time()
Runtime.append(end - start)
print("The best parameters for logistic regression",best_param_logr.best_params_)
```

```
logr = LogisticRegression(C =
0.01 ,multi_class='auto',penalty='l2',solver='liblinear',class_weight='balanced')
logr.fit(X_train,y_train)
logr_Prediction = logr.predict(X_test)
Accuracy_Logistic = accuracy_score(y_test,logr_Prediction)
print("The accuracy score of logisitc regression is:",Accuracy_Logistic)
```

Naive Bayes test

```
start = time.time()
gnb = GaussianNB()
gnb.fit(X_train, y_train)
gnb_Prediction = gnb.predict(X_test)
Accuracy_NaiveBayes = accuracy_score(y_test,gnb_Prediction)
end = time.time()
Runtime.append(end - start)
print("The accuracy score of naive classifier is:",Accuracy_NaiveBayes)
```

Support vector machine train and test

```
start = time.time()
svm = SVC()
param_grid_svm = [
      {
      'kernel':['rbf','linear','sigmoid'],
      'gamma':['scale', 'auto'],
      'class_weight':['dict' , 'balanced'],
      'decision_function_shape':['ovo', 'ovr'],
      }
]
param_svm = GridSearchCV(svm, param_grid = param_grid_svm, cv = 3,
verbose=True,scoring="accuracy")
best_param_svm = param_svm.fit(X_train,y_train)
end = time.time()
Runtime.append(end - start)
print("The best parameters for support vector machine",best_param_svm.best_params_)

svm = SVC(kernel='rbf',class_weight='balanced',decision_function_shape='ovo',gamma='scale')
svm.fit(X_train, y_train)
```

```
svm_Prediction = svm.predict(X_test)
Accuracy_SVM = accuracy_score(y_test,svm_Prediction)
print("The accuracy score of support vector machine is:",Accuracy_SVM)
```

Random forest train and test

```
start = time.time()
rf = RandomForestClassifier()
param_grid_rf = [
    {'n_estimators':[100,200,400,600],
    'criterion':['gini', 'entropy', 'log_loss'],
    'max_features':['sqrt', 'log2', None],
    'class_weight':['balanced', 'balanced_subsample'],
    'random_state':[42],

    }
]
param_rf = GridSearchCV(rf, param_grid = param_grid_rf, cv = 3, verbose=True,scoring="accuracy")
best_param_rf = param_rf.fit(X_train,y_train)
end = time.time()
Runtime.append(end - start)
print("The best parameters for random forest",best_param_rf.best_params_)

rf_Model =
RandomForestClassifier(class_weight='balanced',criterion='entropy',max_features='sqrt',n_estimat
ors=200,random_state=42)
rf_Model.fit(X_train, y_train)
rf_Prediction = rf_Model.predict(X_test)
Accuracy_RandomForest = accuracy_score(y_test,rf_Prediction)
print("The accuracy score of random forest is:",Accuracy_RandomForest)
```

Gradient boosting train and test

```
start = time.time()
gb = GradientBoostingClassifier()
param_grid_gb = [
    {
    'loss':['log_loss', 'deviance', 'exponential'],
    'learning_rate':[0.1,0.01,0.001],
    'n_estimators':[100,200,400,600,800],
```

```
        'subsample':[0.0,0.5,1],
        'criterion':['friedman_mse', 'squared_error', 'mse'],
        }
]
param_gb = GridSearchCV(gb, param_grid = param_grid_gb, cv = 3,
verbose=True,scoring="accuracy")
best_param_gb = param_gb.fit(X_train,y_train)
end = time.time()
Runtime.append(end - start)
print("The best parameters for Gradient Boosting",best_param_gb.best_params_)

gb_Model = GradientBoostingClassifier(criterion = 'friedman_mse',learning_rate=0.01,
loss='accuracy',n_estimators=400,subsample=1)
gb_Model.fit(X_train, y_train)
gb_Prediction = gb_Model.predict(X_test)
Accuracy_gb = accuracy_score(y_test,gb_Prediction)
print("The accuracy score of Gradient Boosting is:",Accuracy_gb)
```

Models' comparsion

```
model_list = ['Logistic Regression', 'Naive Bayes','Support Vector Machine','Random
forest','Gradient Boosting']
accuracy_list = [Accuracy_Logistic, Accuracy_NaiveBayes, Accuracy_SVM, Accuracy_RandomForest,
Accuracy_gb]
df_models = pd.DataFrame({'Machine Learning Models': model_list, 'Accuracy': accuracy_list,
'Training Time (s)': Runtime})
df_models
```

Appendix (4) Python Code For Loan Club Credit Data Set

Importing Packages

```
import pandas as pd
import numpy as np
import time
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import log_loss
import warnings
warnings.filterwarnings("ignore")
data = pd.read_csv('lending_club_loan_two.csv')
data.drop(['address','title','emp_title','issue_d'],axis=1,inplace=True)
```

Data Structure

```
data.info()
```

Heatmap

```
plt.figure(figsize=(12, 8))
sns.heatmap(data.corr(), annot=True,cmap="hot_r")
```

Data visualization

```
fig, axes = plt.subplots(1,2,figsize=(15,5))
axes = axes.flatten()
sns.boxplot(ax=axes[0],x='loan_amnt',y='loan_status',data=data,palette="YlOrBr").set(
    xlabel='loan_amnt',ylabel="",title="Loan Amount")
sns.boxplot(ax = axes[1],x='int_rate',y='loan_status',data=data,palette="YlOrBr").set(
    xlabel="int_rate",ylabel="",title="Interest Rate")

sns.set(font_scale=1)
fig, axes = plt.subplots(1,2,figsize=(22,5))
axes = axes.flatten()
g_order = ['A','B','C','D','E','F','G']
sub_order=['A1','A2','A3','A4','A5','B1','B2','B3','B4','B5','C1','C2','C3','C4','C5','D1','D2','D3','D4','D5',
'E1','E2','E3','E4','E5','F1','F2','F3','F4','F5','G1','G2','G3','G4','G5']
sns.countplot(ax = axes[0],data=data, hue='loan_status',x = 'grade',order=g_order,
palette="YlOrBr").set(
    xlabel='Grade',ylabel="Count",title="Grade Distribution")
sns.countplot(ax = axes[1],data=data, hue='loan_status',x = 'sub_grade',order=sub_order,
palette="YlOrBr").set(
    xlabel='Sub Grade',ylabel="Count",title="Sub Grade Distribution")

sns.set(font_scale=0.8)
fig, axes = plt.subplots(1,2,figsize=(22,5))
axes = axes.flatten()
y_order = ['< 1 year','2 years','3 years','4 years','5 years','6 years','7 years','8 years','9 years','10+
years']
sns.countplot(ax = axes[0],data=data, x="emp_length",
hue="loan_status",palette="YlOrBr",order=y_order).set(
    xlabel='Employment length',ylabel="Count",title="Employment length Distribution")
sns.countplot(ax = axes[1],data=data, x="home_ownership",
hue="loan_status",palette="YlOrBr").set(
    xlabel='Home ownership',ylabel="Count",title="Home ownership Distribution")

sns.set(font_scale=0.8)
fig, axes = plt.subplots(1,3,figsize=(22,5))
axes = axes.flatten()
sns.countplot(ax = axes[0],data=data, x="purpose",
hue="loan_status",palette="YlOrBr").tick_params(axis='x', labelrotation=90)
```

```
sns.countplot(ax = axes[1],data=data, x="verification_status",
hue="loan_status",palette="YlOrBr").set(
        xlabel='Verification Status',ylabel="Count",title="Is Income was verified")
sns.countplot(ax = axes[2],data=data, x="term", hue="loan_status",palette="YlOrBr").set(
        xlabel='Months',ylabel="Count",title="Payments on the loan")

dataPair = data.loc[:,['loan_status','loan_amnt','annual_inc','revol_bal']]
sns.pairplot(dataPair,hue="loan_status",palette="YlOrBr")

dataPair = data.loc[:,['loan_status','total_acc','open_acc','mort_acc']]
sns.pairplot(dataPair,hue="loan_status",palette="YlOrBr")
```

Feature Selection

```
data = data.sample(n=60000)
test = data.drop(['sub_grade'],axis=1)
X=test.drop(['grade'],axis=1)
y=test['grade']
X = X.apply(LabelEncoder().fit_transform)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
scaler = RobustScaler()
X_train = pd.DataFrame(
        scaler.fit_transform(X_train),
        columns = X_train.columns
)

X_test = pd.DataFrame(
        scaler.transform(X_test),
        columns = X_test.columns)

fs = SelectKBest(score_func=f_classif, k='all')
fs.fit(X_train, y_train)
features_list = []
scores_list = []
for i in range(len(fs.scores_)):
  features_list.append(data.columns[i])
  scores_list.append(fs.scores_[i])
df_features = pd.DataFrame({'Features': features_list, 'Scores': scores_list})
df_features = df_features.sort_values(by=['Scores'],ascending=False)
```

```
plt.figure(figsize=(25,10))
sns.barplot(x='Features', y='Scores', data=df_features[0:10],palette="YlOrBr")
plt.title("ANOVA P value")
```

Implement:

```
X=data.loc[:,['int_rate','term','revol_bal','verification_status','annual_inc','loan_amnt','purpose',
'installment','loan_status','home_ownership']]
y=data['grade']
X = X.apply(LabelEncoder().fit_transform)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

scaler = RobustScaler()
X_train = pd.DataFrame(
    scaler.fit_transform(X_train),
    columns = X_train.columns
)

X_test = pd.DataFrame(
    scaler.transform(X_test),
    columns = X_test.columns
)
Runtime = []

start = time.time()

logr = LogisticRegression()
param_grid_logr = [
    {'penalty':['l2','elasticnet','none'],
    'C':[0.001,0.01,0.1,1,10,100],
    'solver':['lbfgs','newton-cg','liblinear','sag','saga'],
    'multi_class':['auto','ovr','multinomial'],
    'class_weight':['balanced']
    }
]
param_logr = GridSearchCV(logr, param_grid = param_grid_logr, cv = 3,
verbose=True,scoring='accuracy')
best_param_logr = param_logr.fit(X_train,y_train)
end = time.time()
```

```
Runtime.append(end - start)
print("The best parameters for logistic regression",best_param_logr.best_params_)

logr = LogisticRegression(C = 1 ,multi_class='auto',penalty='l2',solver='newton-
cg',class_weight='balanced')
logr.fit(X_train,y_train)
logr_Prediction = logr.predict(X_test)
Accuracy_Logistic = accuracy_score(y_test,logr_Prediction)
print("The accuracy score of logisitc regression for grade is:",Accuracy_Logistic)
model_list = ['Logistic Regression']
accuracy_list = [Accuracy_Logistic]
df_models = pd.DataFrame({'Machine Learning Models': model_list, 'Accuracy': accuracy_list,
'Training Time (s)': Runtime})
df_models
```

Reference List

Ibm Cloud Learn Hub (2020). What are Neural Networks?. https://www.ibm.com/hk-en/cloud/learn/neural-networks

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification Journal of Machine Learning Research 9(2008)

Donaldson, T. (2008). Python (2nd ed.). Peachpit Press.

Zhou, Z.-H. (2021). Machine learning. Springer. https://doi.org/10.1007/978-981-15-1967-3

Brzezinski, J. R. (2000). Logistic regression for classification of text documents. Thesis (Ph.D.)--DePaul University, School of Computer Science, Telecommunications, and Information Systems, 2000.

ANDREEVA, G., & ALTMAN, E. I. (2021). THE VALUE OF PERSONAL CREDIT HISTORY IN RISK SCREENING OF ENTREPRENEURS: EVIDENCE FROM MARKETPLACE LENDING. Journal of Financial Management, Markets and Institutions, 9(1), 2150004–2150004–25.

Khadka, M. S., George, K. M., Park, N., & Sarangan, V. (2008). Parameter estimation of Copula using maximum likelihood estimation (MLE) method. Thesis (M.S.)--Oklahoma State University, 2008.

Kroese, D. P., Botev, Z. I., & Taimre, T. (2020). Data science and machine learning : mathematical and statistical methods. CRC Press.

Brzezinski, J. R. (2000). Logistic regression for classification of text documents. Thesis (Ph.D.)--DePaul University, School of Computer Science, Telecommunications, and Information Systems, 2000.

Tang, D. Y., Titman, S., & Yan, H. (2006). Essays on credit risk. Thesis (Ph.D.)--The University of Texas at Austin, 2006.

Cramer, J. S. (2003). The origins and development of the logit model. In Logit Models

from Economics and Other Fields (pp. 149–157). Cambridge University Press.