

# P2P Lending Club

Data Analysis

by

Hans Alcahya Tjong (570795)

Ananda Eraz Irfananto (547048)

Winter semester 21/22

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Methodology</b>	<b>5</b>
2.1	Linear Regression . . . . .	5
2.1.1	Simple Linear Regression . . . . .	5
2.1.2	Multiple Linear Regression . . . . .	7
2.2	Logistic Regression . . . . .	9
2.2.1	Logistic Model . . . . .	9
2.2.2	Multiple Logistic Regression . . . . .	11
2.3	Support Vector Machines . . . . .	11
2.3.1	Maximal Margin Classifier . . . . .	11
2.3.2	Soft Margin Classifier . . . . .	16
2.3.3	Support Vector Machines . . . . .	17
2.4	Confusion Matrix . . . . .	18
<b>3</b>	<b>Empirical Results and Conclusion</b>	<b>20</b>
3.1	Data Preparation . . . . .	20
3.2	Multiple Linear Regression . . . . .	22
3.3	Multiple Logistic Regression . . . . .	25
3.4	Support Vector Machines . . . . .	28
3.5	Conclusion . . . . .	31
<b>4</b>	<b>References</b>	<b>32</b>

## Abstract

The customer's risk profile is very important for both banks and *Peer-to-Peer* (P2P) lending platforms to decide whether a customer could get a loan. In this paper we want to predict the FICO score of customers using one of supervised learning methods, namely *linear regression*. We build this model with significant features only, that are developed by a stepwise algorithm. Last, we create a model for predicting probability of meeting the given credit underwriting criteria using *logistic regression*. As a comparison we utilize one of the most commonly used of machine learning algorithms *support vector machines*. We find that the result of support vector machines with radial basis kernel has the greatest accuracy level. The codes corresponding to the paper are available on GitHub<sup>1</sup>.

## 1 Introduction

Banks are used to be the only option when it comes to get a loan. Banks usually require their customers an excellent credit score, otherwise the customers loan application will get rejected. When a customer asks for a large loan, the bank will offer a secured loan. That means that the customer has to provide collateral, such as property. When the customer fails to default the loan, the customer could lose the collateral. On the other hand, P2P lending is a process of obtaining financing from other individuals, as opposed to a financial intermediary like banks.

P2P lenders give opportunity for customers who typically have been excluded from getting loans from the bank. It includes customers with lower -or even no- credit scores as well as zero assets. Banks are bricks and mortar institutions, even if they have online banking. They will have higher overhead which will impact the fees, interests, etc. Instead, P2P lending uses online marketplace. P2P lending websites connect borrowers to lenders or investors. The website settles rates and terms as well as enables the transaction.

---

<sup>1</sup>[https://github.com/hansalca1403/wissArbeiten/blob/main/P2P\\_Lending\\_Club.r](https://github.com/hansalca1403/wissArbeiten/blob/main/P2P_Lending_Club.r)

Some P2P platform will not allow anyone to invest easily, as they may prefer an accredited investor. From the borrower's perspective, they are labeled to certain risk categories set by the lender. The risk categories are assessed based on requested amount, public records, credit purposes, and income, which will help the lenders on selecting which borrowers they want to lend to. The borrower's benefit is typically more lenient credit requirements than the one from banks. On the other side, P2P lending creates potential for lenders to earn higher returns from their investment than other instruments like stock market or real estate.

## 2 Methodology

### 2.1 Linear Regression

This chapter is about linear regression, a straightforward approach for supervised learning. Linear regression is useful for predicting quantitative response from a set of predictor variables. Moreover, it determines which variables in particular are significant predictors of the outcome variables and in what way do they *indicated by the beta estimates* impact the outcome variable. These regression estimates are used to explain the relationship between predictor variable  $X$  and response variable  $Y$ .

#### 2.1.1 Simple Linear Regression

*Simple linear regression* is an approach for predicting a quantitative response  $Y$  on the basis of a single predictor variable  $X$ . Assumed there is approximately a linear relationship between  $X$  and  $Y$ . The linear relationship will be mathematically written as

$$Y \approx \beta_0 + \beta_1 X$$

$\beta_0$  and  $\beta_1$  are unknown constants that designate the *intercept* and *slope* in the linear model and together they are known as the model *parameters*. Then, after we have estimated  $\hat{\beta}_0$  and  $\hat{\beta}_1$  for the model parameters by using the training data, we can predict the response variable  $Y$  by computing

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x,$$

with  $\hat{y}$  designates the prediction of  $Y$  on basis  $X = x$ .

##### 2.1.1.1 Estimating the Parameters

As said before,  $\beta_0$  and  $\beta_1$  are unknown. Before we make predictions, we have to

use the data to estimate the parameters. Let

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$$

designate  $n$  observation pairs, each of which consists of a measurement of  $X$  and  $Y$ . The main objective here is to get parameter estimates  $\hat{\beta}_0$  and  $\hat{\beta}_1$  that fit the linear model, so that

$$\hat{y}_i \approx \hat{\beta}_0 + \hat{\beta}_1 x_i$$

$$i = 1, \dots, n.$$

Here we want to acquire parameters  $\hat{\beta}_0$  and  $\hat{\beta}_1$  that will make the regression line as close as possible to the  $n$  data points. We will take the *least squares* approach to acquire these parameters. Figure 1 represents the simple linear regression model.

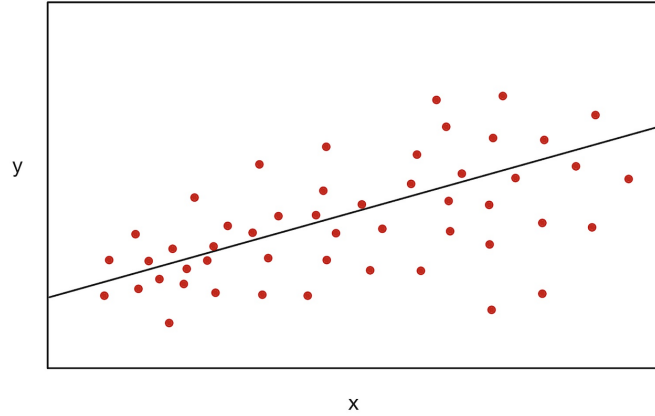


Figure 1: Simple linear regression model, the red dots represent data points

Let  $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$  be the prediction for  $Y$  based on  $x_i$  ( $i$ th value of  $X$ ). Then  $e_i = y_i - \hat{y}_i$  designates the  $i$ th *residual*. *Residual* is the difference between  $i$ th actual response value and the  $i$ th predicted response value from our linear model. Next we define the *residual sum of squares* (RSS) as

$$RSS = e_1^2 + e_2^2 + \dots + e_n^2,$$

and furthermore

$$RSS = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \cdots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2.$$

To obtain optimum  $\hat{\beta}_0$  and  $\hat{\beta}_1$ , we have to minimize the RSS. The *least squares parameter estimates* will be written as

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2},$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$

with  $\bar{y} \equiv \frac{1}{n} \sum_{i=1}^n y_i$  and  $\bar{x} \equiv \frac{1}{n} \sum_{i=1}^n x_i$  as the sample means.

#### 2.1.1.2 $R^2$ Statistics

The  $R^2$  statistic provides the measure of fit. It is the proportion of variability in  $Y$  that can be explained by using  $X$ , has the value between 0 and 1, and also independent of the scale of  $Y$ . To calculate  $R^2$ , we use the formula

$$R^2 = 1 - \frac{RSS}{TSS},$$

where  $TSS = \sum (y_i - \bar{y})^2$  is the *total sum of squares*. TSS measures the total variance in  $Y$ .  $R^2$  value near 0 indicates that the model did not explain much of the variability in the response, meaning that the regression model could be wrong.

#### 2.1.2 Multiple Linear Regression

As explained before, simple linear regression is a practical approach to predict a response on the basis of a single predictor variable. Unfortunately, in reality we often come up with more than one predictor variable. How do we integrate these extra predictors to make our analysis? One solution would be by making separate simple linear regressions, each of them using different predictor. This solution, however, is not that effective, because we will have different regression equation

for each predictor so that a single prediction would be hard to conclude. Another thing is that by using simple linear regression, each of the equation will exclude the other predictors in forming estimates for the parameters. Instead, we will extend the simple linear regression by giving each predictor its own separate slope in a single model. Assumed that we have  $i$  predictors:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_i X_i + \epsilon,$$

where  $X_i$  designates the  $i$ th predictor and  $\beta_i$  is the slope for each  $i$ th predictor and is interpreted as the average effect on  $Y$  of a one unit increase in  $X_i$ , while other predictors fixed.  $\epsilon$  designates the residual of the regression.

### 2.1.2.1 Estimating the Parameters

As we have shown in the simple linear regression, the regression parameters  $\beta_0, \beta_1, \dots, \beta_i$  are also unknown and therefore must be estimated. Given estimates  $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_i$ , we can make predictions using the formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_i x_i.$$

The parameters are going to be estimated by using the least square approach, like it was the case in simple linear regression. We choose  $\beta_0, \beta_1, \dots, \beta_i$  to minimize the sum of squared residuals

$$\begin{aligned} RSS &= \sum_{j=1}^n (y_j - \hat{y}_j)^2 \\ &= \sum_{j=1}^n (y_j - \hat{\beta}_0 - \hat{\beta}_1 x_{j1} - \hat{\beta}_2 x_{j2} - \cdots - \hat{\beta}_i x_{ji})^2 \end{aligned}$$

### 2.1.2.2 Selecting Significant Variables

On many cases, it is possible that only some of the predictors are related with the response. To determine which predictors are related to the response, so that we



can make a single model only for those predictors, is called *variable selection*. In our case, we will be using *Akaike Information Criterion*(AIC) to determine which variables are significant.

## 2.2 Logistic Regression

In linear regression model, the response variable  $Y$  is assumed quantitative. But in other situations, the response variable is instead qualitative or also referred as categorical. The task is now we predict the probability of each categories of a qualitative variable, as the base for making our final prediction. For example, if someone took a loan, then it's either they *have* or *have not* paid it back. Then we could code these qualitative response as:

$$Y = \begin{cases} 0 & \text{if not default} \\ 1 & \text{if default} \end{cases}$$

After that we could make a linear regression to this binary response, and predict **paid** if  $Y > 0.5$  and **unpaid** otherwise. But rather than making a linear model to this response, logistic regression models the probability that  $Y$  belongs to a certain category. For example, the probability of default can be written as

$$p(X) = Pr(Y = 1|X)$$

### 2.2.1 Logistic Model

As mentioned before, the response variable in logistic regression is qualitative. Therefore we cannot model the probability by using linear regression model. One of the reasons is that the predicted probability value would not fall between 0 and 1 if we use linear model. We must then model  $p(X)$  using a function that gives results between 0 and 1 for all values of  $X$ . In logistic regression, we use the *logistic function*,

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}.$$

After a bit of manipulation, we got

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X}.$$

The left-hand side of the equation is called the *odds* and have value between 0 and  $\infty$ . Then by giving both sides the logarithm, we will have

$$\log \left( \frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X.$$

The left-hand side is now called the *log-odds* or *logit*.

### 2.2.1.1 Estimating the Parameters

The parameters  $\beta_0$  and  $\beta_1$  are also unknown like the case in linear regression and they need to be estimated based on training data. The preferred approach is the *maximum likelihood*. The idea is that we look for  $\hat{\beta}_0$  and  $\hat{\beta}_1$  by plugging these estimates into the model for  $p(X)$  yields a number close to one for all  $Y = 1|X$ , and a number close to zero for all  $Y = 0|X$ . This can be formalized using *likelihood function*:

$$L(\beta_0, \beta_1) = \prod_{i: y_i=1} p(x_i) \prod_{i': y_{i'}=0} (1 - p(x_{i'})).$$

The estimates  $\hat{\beta}_0$  and  $\hat{\beta}_1$  are chosen to maximize the likelihood function. For our topic we will not go deep into the mathematical details of maximum likelihood as it can be easily fit using **R** function that we will discuss more later on. Once the parameters have been estimated, we can put them in our model equation:

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X}}.$$

### 2.2.2 Multiple Logistic Regression

Assumed now that we have multiple predictors. Just like on linear regression, we can extend the formula that we have as follows:

$$\log \left( \frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X + \dots + \beta_p X_p,$$

where  $X = (X_1, \dots, X_p)$  are  $p$  predictors. The equation can be rewritten as

$$p(X) = \frac{e^{\beta_0 + \beta_1 X + \dots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X + \dots + \beta_p X_p}}.$$

We also use the maximum likelihood to estimate  $\beta_0, \beta_1, \dots, \beta_p$ .

## 2.3 Support Vector Machines

One of the most commonly used of machine learning algorithms is support vector machine. It was developed in the 1990s by Vladimir Vapnik and colleagues and that has become more popular since then. The support vector machine is a supervised learning approach for classification, which is generally suited for binary classification. There are further extensions of support vector machines to accommodate cases of more than two classes, but it is outside of the scope of our seminar project.

### 2.3.1 Maximal Margin Classifier

It all started with a simple classifier called the maximal margin classifier. We will firstly define the concept of an hyperplane.

#### 2.3.1.1 Hyperplane

In a  $n$ -dimensional space, a flat affine subspace of dimension  $n - 1$  is called a hyperplane, e.g. a line is a flat one-dimensional subspace of two-dimensional space

for that a line is a hyperplane of two-dimensional space. Another example is, in three dimensional space, a plane is a flat two-dimensional subspace that is also called hyperplane.

The definition of a  $n$ -dimensional hyperplane is defined by the equation:

$$h(X) := \beta_0 + \beta^T X = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n = 0,$$

$$\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^n.$$

If a point  $X = (X_1, X_2, \dots, X_n)^T$  satisfies the hyperplane equation, then  $X$  lies on the hyperplane.

Suppose that  $X$  has the following property:

$$h(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n < 0$$

or

$$h(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n > 0.$$

Since we have inequality instead of equality, it follows that  $X$  lies on one side of the hyperplane.

### 2.3.1.2 Classification Using a Hyperplane

Given a data set of  $m$  training observations in  $n$ -dimensional space:

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1n} \end{pmatrix}, \dots, x_m = \begin{pmatrix} x_{m1} \\ \vdots \\ x_{mn} \end{pmatrix},$$

where the dimension of  $n$  corresponds to size of features and we define two classes where these observations fall into:  $\{-1, 1\} \ni y_1, \dots, y_m$ . One can say that each observation  $x_i$  has a property of  $y_i$  that is either  $-1$  or  $1$ .

We will now construct some hyperplanes that separate the training observations

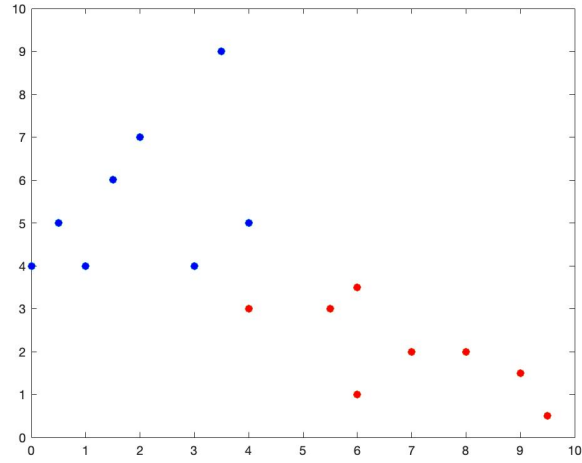


Figure 2: Two classes of observations, shown in blue and in red

perfectly. Three separating hyperplanes are shown by figure 3.

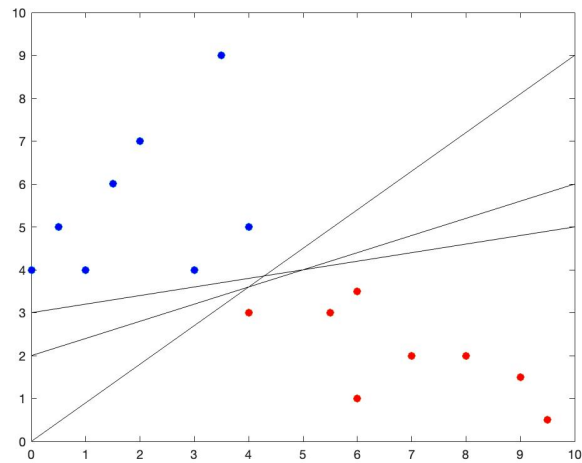


Figure 3: Separating hyperplanes

Suppose the observations from the blue class have the property of  $y_i = 1$  and those from the red class  $y_i = -1$ . Then all separating hyperplanes have the property:

$$h(x_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_n x_{in} < 0 \text{ if } y_i = -1$$

and

$$h(x_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_n x_{in} > 0 \text{ if } y_i = 1.$$

It is equivalent to:

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_n x_{in}) > 0$$

for all  $i = 1, \dots, n$ .

### 2.3.1.3 Maximal Margin Classifier

If the data sets can be perfectly separated, then there will be an infinite number of such hyperplanes. Imagine we tilt the separating hyperplane by a little bit, there exists many hyperplanes that do not touch the observations. Therefore we have to decide which hyperplane to use. One of the reasonable way is using the maximal margin classifier as method to decide which is the optimal separating hyperplane. Optimal separating hyperplane is a hyperplane that is farthest from the training observations. The smallest distance between the observation to the separating hyperplane is known as the margin and is given by:

$$M = \frac{1}{\|\beta\|_2} + \frac{1}{\|\beta\|_2} = \frac{2}{\|\beta\|_2}.$$

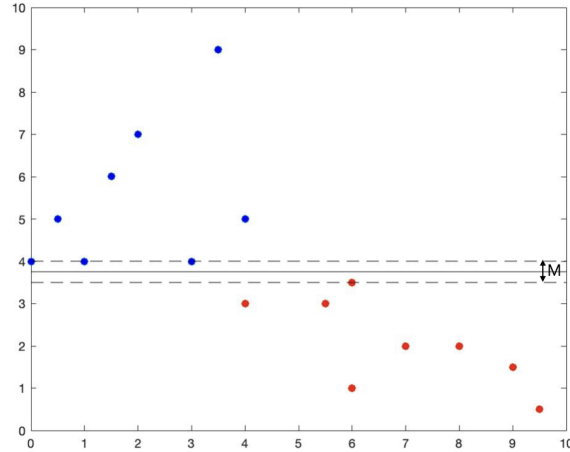


Figure 4: Separating hyperplanes and its margin

Distance between any observation  $x_i = (x_{i1}, x_{i2}, \dots, x_{in})^T$ ,  $i = 1, \dots, m$  and the hyperplane  $\beta_0 + \beta^T X = 0$  is defined by:

$$d(x_i) := \frac{|\beta_0 + \beta^T X|}{\|\beta\|_2}.$$

Figure 4 shows a separating hyperplane and its margin. However, this hyperplane is not optimal, since we can find another separating hyperplane with greater margin. It could be said that optimal separating hyperplane is the separating hyperplane, which has largest margin.

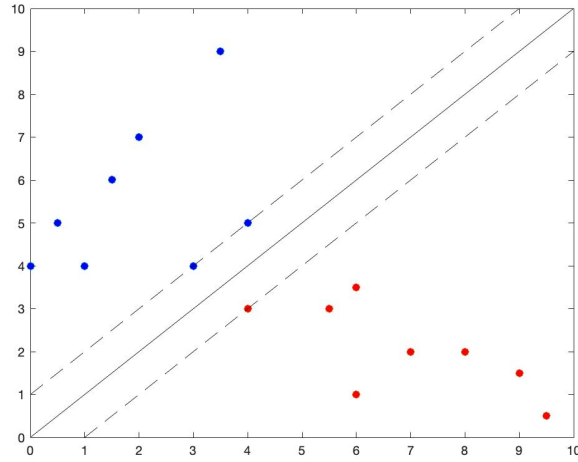


Figure 5: Optimal separating hyperplanes

This can be considered as an optimization problem. The optimal separating hyperplane is the solution of the following optimization problem:

$$\begin{aligned} \min_{\beta_0, \beta} \quad & \frac{\|\beta\|^2}{2} \\ \text{s.t.} \quad & y_i(\beta_0 + \beta^T X_i) \geq 1, \quad \forall i \in \{1, \dots, m\}. \end{aligned}$$

The maximal margin classifier can be applied when training observations are linearly separable, or it can be said, if a separating hyperplane exists. Unfortunately most data sets are linearly non-separable, that means the maximal margin classifier is not applicable, since the classes should be separable by a linear boundary.

### 2.3.2 Soft Margin Classifier

Soft Margin Classifier is an extension of maximal margin classifier. The difference between soft margin classifier and maximal margin classifier is, that in soft margin classifier some observations are allowed to be on the incorrect side of the margin or even on the wrong side of hyperplane. We can say, that the margin can be violated by a small subset of observations.

In the optimization problem we introduce slack variables  $\epsilon_i$ , which measure and penalise the degree of misclassification of  $x_i$ . Then the hyperplane is the solution to the optimization problem:

$$\begin{aligned} \min_{\beta_0, \beta, \epsilon_i} \quad & \frac{\|\beta\|^2}{2} + C \sum_{i=1}^m \epsilon_i \\ \text{s.t.} \quad & y_i(\beta_0 + \beta^T X_i) \geq 1 - \epsilon_i, \quad \epsilon_i \geq 0, \quad \forall i \in \{1, \dots, m\}, \\ & \sum_{i=1}^m \epsilon_i \leq C, \end{aligned}$$

where  $C$  is a tuning parameter, that is nonnegative. The slack variable  $\epsilon_i$  describes the position of a given point  $x_i$ . If  $\epsilon_i = 0$  then there is no misclassification and it follows that  $x_i$  is on the correct side. If  $\epsilon_i \in (0, 1)$  then  $x_i$  lies between the support hyperplane and the separating hyperplane. If  $\epsilon_i > 1$  then  $x_i$  is on the wrong side of the separating hyperplane. The tuning parameter  $C$  has a role as upper bound of the sum of  $\epsilon_i$ 's, so it determines how much we will tolerate.

As the parameter  $C$  increases, the margin will widen and it implies that more violation are allowed to the margin. When  $C$  is small, the margins are narrow and rarely violated. We generally choose the tuning parameter  $C$  via cross-validation. The maximal margin classifier and the soft margin classifier deal only with linear separable data sets. However as we have hinted, most problems are linearly non-separable. That means, there exists no hyperplane that can separate the data sets. Therefore we apply the support vector machines.



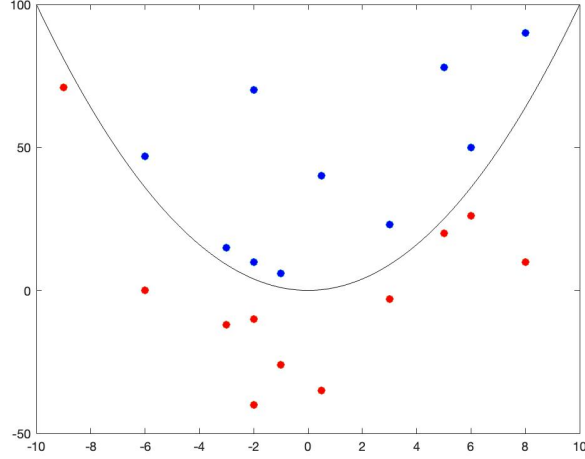


Figure 6: Non-linear problem

### 2.3.3 Support Vector Machines

Support vector machine is a generalisation of soft margin classifier. Since the problem is not linearly separable in input space, we enlarge the space using kernels. First of all we define the generalized definition of hyperplane. Let  $V$  be a vector space with inner product  $\Phi : \mathbb{R}^n \rightarrow V$ , then the hyperplane can be written as:

$$h(X) = \beta_0 + \beta^T \phi(X).$$

The function  $K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ ,

$$K(X_i, X_j) = \phi(X_i)^T \phi(X_j),$$

is referred to as kernel function.

Then the modified optimization is defined by:

$$\begin{aligned} \min_{\beta_0, \beta, \epsilon_i} \quad & \frac{\|\beta\|^2}{2} + C \sum_{i=1}^m \epsilon_i \\ \text{s.t.} \quad & y_i(\beta_0 + \beta^T \Phi(X_i)) \geq 1 - \epsilon_i, \quad \epsilon_i \geq 0, \quad \forall i \in \{1, \dots, m\}, \end{aligned}$$

$$\sum_{i=1}^m \epsilon_i \leq C.$$

There are some popular kernel functions. For instance, we simply take:

$$K(X_i, X_j) = X_i^T X_j,$$

that is known as a linear kernel. The linear kernel leads us back to the soft margin classifier, because the soft margin classifier is also linear in the input space. Some other kernel functions are:

1. Polynomial kernel of degree d

$$K(X_i, X_j) = (1 + X_i^T X_j)^d,$$

2. Sigmoid kernel

$$K(X_i, X_j) = \tanh(\alpha X_i^T X_j + \gamma),$$

3. Gaussian radial basis kernel

$$K(X_i, X_j) = \exp(-\gamma \|X_i - X_j\|^2).$$

## 2.4 Confusion Matrix

A confusion matrix is a table used to describe the performance of a classification model. It compares the predictions with the actual value. The table consists of 4 different combinations of predicted and actual value.

- **True Negative (TN)**: We predicted negative and it is actually negative.
- **True Positive (TP)**: We predicted positive and it is actually positive.
- **False Positive (FP)**: We predicted positive but it is actually negative.  
This is also called as “Type 1 Error”
- **False Negative (FN)**: We predicted negative but it is actually positive.  
This is also called as “Type 2 Error”

		Actual	
		Negative (0)	Positive (1)
Prediction	Negative (0)	TN	FN
	Positive (1)	FP	TP

Figure 7: Confusion matrix table

The performance metrics for confusion matrix are *accuracy*, *sensitivity* and *specificity*, which are calculated on the basis of classifier above.

**Accuracy** represents the ratio of correctly classified points to the total number of points.

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

**Sensitivity** represents the ratio of correctly predicted positive points to all actual positives.

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

**Specificity** represents the ratio of correctly predicted negative points to all actual negatives.

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

## 3 Empirical Results and Conclusion

### 3.1 Data Preparation

In this chapter we will discuss the application of our methods in R. We use the data *lendingclub.csv*<sup>2</sup>. The data itself is collected from Lending Club, one of the largest P2P lending platform, and it is a historical data from 2007 to 2015.

Table 1: Lending Club Data Set

Sample size	Numerical features	Categorical features
9578	days.with.cr.line	credit.policy
	revol.bal	purpose
	revol.util	inq.last.6mths
	int.rate	delinq.2yrs
	dti	pub.rec
	installment	not.fully.paid
	fico	
	log.annual.inc	

Table 2: Numerical features

Numerical features	Definition
days.with.cr.line	The number of days the borrower has had a credit line
revol.bal	The borrower's revolving balance (amount unpaid at the end of the credit card billing cycle)
revol.util	The borrower's revolving line utilization rate (the amount of the credit line used relative to total credit available)
int.rate	The interest rate of the loan

<sup>2</sup><https://www.kaggle.com/urstrulyvikas/lending-club-loan-data-analysis>

Numerical features	Definition
dti	The debt-to-income ratio of the borrower (amount of debt divided by annual income)
installment	The monthly installment
fico	FICO credit score of the borrower
log.annual.inc	The natural log of the self-reported annual income of the borrower

Table 3: Categorical features

Categorical features	Definition
credit.policy	1 if the customer meets the credit underwriting criteria of LendingClub.com, and 0 otherwise
purpose	The purpose of the loan (takes values “creditcard”, “debtconsolidation”, “educational”, “majorpurchase”, “smallbusiness”, and “all_other”)
inq.last.6mths	The borrower’s number of inquiries by creditors in the last 6 months
delinq.2yrs	The number of times the borrower had been 30+ days past due on a payment in the past 2 years
pub.rec	The borrower’s number of derogatory public records (bankruptcy filings, tax liens, or judgments)
not.fully.paid	The status of loan default

On multiple linear regression, the response variable  $y$  is *fico*. We are going to predict the fico score of the applicant based on the selected features that we are going to discuss later. Meanwhile for the multiple logistic regression and support vector machine, the response variable  $y$  is the *credit.policy*. This is a binary response on whether the applicant meets the credit underwriting criteria

of Lending Club ( $y = 1$ ) or not ( $y = 0$ ). The R packages that we are going to use on this project are:

- **caret**
- **LogicReg**
- **e1071**
- **MASS**
- **tibble**
- **ggplot2**

After we installed the packages, the lending club data set has to be read by using function `read.csv()`. Then we check if the data set has missing values. If indeed there are missing values, then we have to eliminate them first.

Next, we are going to split the data into training (70%) and test (30%) sets. In order to have a consistent and random split, we use the function `set.seed()`.

```
set.seed(1234)
indizes = createDataPartition(y = data_00$credit.policy,
                              p = 0.70, list = F)
data_train <- data_00[indizes,]
data_test <- data_00[-indizes,]
```

## 3.2 Multiple Linear Regression

We will start by using the `lm()` function to fit a multiple linear regression model, with *fico* as the response variable  $y$  and the rest of the features as the predictor variable  $x$ <sup>3</sup>. We fit the training data using the `lm()` function. The `summary()` function is used in order to gather some information about the fit.

---

<sup>3</sup>The code is available at: [https://github.com/hansalca1403/wissArbeiten/blob/main/P2P\\_Lending\\_Club.r](https://github.com/hansalca1403/wissArbeiten/blob/main/P2P_Lending_Club.r)

```
lm_fit = lm(fico~., data = data_train)
summary(lm_fit)
```

The result of the `summary()` function shows us the value of  $\hat{\beta}_0$  and  $\hat{\beta}_i$  for every features, the p-values of each features as well as their significance level, and also the  $R^2$  statistic. In chapter 2 it is said that only some of the predictors are related with the response. To determine that, we are using the function `stepAIC()` on the previous model by performing stepwise model selection by AIC.

```
AIC(lm_fit)
step1m = stepAIC(lm_fit, direction = "both")
step1m$anova
```

This process will give us the final model which include:

Table 4: Final model based on AIC

Response variable	Predictor variable
fico	credit.policy
	purpose
	int.rate
	installment
	dti
	days.with.cr.line
	revol.bal
	revol.util
	delinq.2yrs
	pub.rec
	not.fully.paid

Now we can fit the final linear model using once again the function `lm()`.

```
lm_fit1 = lm(fico~credit.policy + purpose + int.rate + installment +
             log.annual.inc + dti + days.with.cr.line + revol.bal
             + revol.util + delinq.2yrs +
             pub.rec + not.fully.paid, data = data_train)
summary(lm_fit1)

# Predict on test set
lm_prob = predict(lm_fit1, newdata = data_test, type = "response")
lm_prob1 = enframe(lm_prob, name = "applicants", value = "fico")
```

By applying the `predict()` function to the data test, we will get the predictions of test set using this model. After that we make the regression plot. It shows us the predicted fico score versus the actual fico score. These bullets are the sample from the test set and the red line we call it the *regression line*. Then we also calculate the residual by subtracting the actual value with the predicted value as mentioned in the chapter 2.

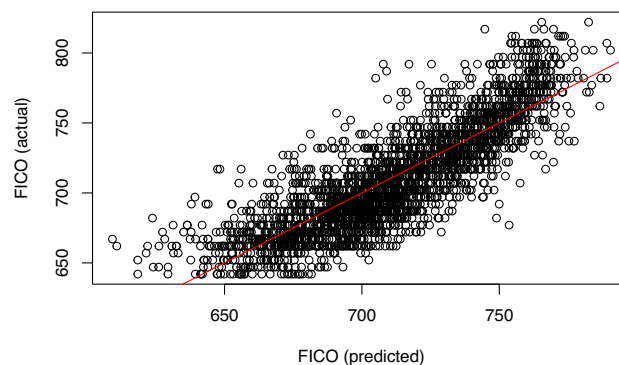


Figure 8: Multiple linear regression plot



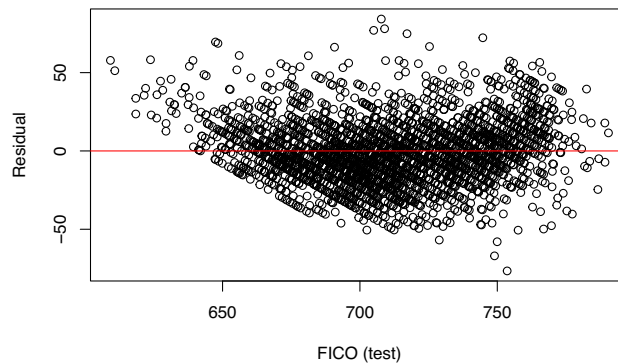


Figure 9: Residual plot

### 3.3 Multiple Logistic Regression

As explained in chapter 2, logistic regression predicts the probability of each categories of a categorical variable. We use the `glm()` function to fit *generalized linear models*, a class of models that includes logistic regression. The syntax of the `glm()` function is the same like `lm()` function, except that we have to put the argument `family=binomial` so that the R knows we want to run a logistic regression. The steps are also the same like what we did with linear regression, we fit the training data using `glm()` function and then select the significant features with `stepAIC()` function.

```
glm_fit = glm(credit.policy ~ ., data = data_train, family = binomial)
summary(glm_fit)
```

```
step = stepAIC(glm_fit, direction = "both")
step$anova
```

This process will give us the final model which include:

Table 5: Final model based on AIC

Response variable	Predictor variable
credit.policy	installment
	log.annual.inc
	fico
	days.with.cr.line
	revol.bal
	revol.util
	delinq.2yrs
	inq.last.6mths
	not.fully.paid

Now we can fit the final linear model using once again the function `glm()`.

```
fit.log1 <- glm(credit.policy ~ installment + log.annual.inc + fico
               + days.with.cr.line
               + revol.bal + delinq.2yrs + not.fully.paid
               + revol.util + inq.last.6mths,
               data = data_train, family=binomial)
summary(fit.log1)

# Predict on test set
glm.prob1 = predict(fit.log1, data_test, type = "response")
glm_prob1frame = enframe(glm.prob1, name = "applicants",
                          value = "probability")

# Probability > 0.5 --> 1, otherwise 0
glm.pred1 = factor(ifelse(glm.prob1>0.5,1,0))
```

After applying the `predict()` function to the data test, we now have the pre-

dictions of test set. We can evaluate the multiple logistic regression model using `confusionMatrix()` function. Following is the confusion matrix of the multiple logistic regression as well as its performance metrics:

Table 6: Confusion matrix for multiple logistic regression

	0	1
Negative	376	81
Positive	184	2232

Table 7: Performance metrics

	Value
Accuracy	0.9078
Sensitivity	0.6714
Specificity	0.9650

### 3.4 Support Vector Machines

We use the `e1071` package to implement support vector machine in R. This package contains functions for probabilistic and statistic algorithms and the `svm()` function is one of them. The function `svm()` is used to train a support vector machine. As we have discussed in the previous chapter there are four kernel functions, that we use in the support vector machines for our project. We have to define which function in the argument `kernel`, for instance `kernel="radial"` when the radial kernel function is used.

Other three values for the argument `kernel` are "linear", "polynomial", and "sigmoid". A `cost` argument specifies how much will the margin be violated. If the `cost` argument is large, then the margin will be narrow and only few support vectors that lie on the margin or violate the margin. Conversely, when the `cost` argument is small, then the margin will be wide and there will be many support vectors on the margin or violating the margin.

In case that `kernel="radial"` is used we also define the `gamma` argument in order to specify a value of  $\gamma$  for the radial basis kernel. If `kernel="polynomial"` is used, then we use the `degree` argument to specify a degree for the polynomial kernel. We fit the training data using the `svm()` function with four different kernel functions and parameters. This is an example of support vector machines with radial basis kernel:

```
svmfit = svm(credit.policy ~ installment + log.annual.inc + fico
             + days.with.cr.line + revol.bal + revol.util
             + inq.last.6mths + delinq.2yrs + not.fully.paid,
             data = data_train, kernel="radial", gamma=1, cost=1)
```

The `summary` function is used in order to obtain some information about the fit. For instance for the support vector machines fit with radial basis kernel:

```
summary(svmfit)
```

The result of the `summary` function tells us number of support vectors, number of classes, and also how many support vectors belong to each class. In chapter 2 it is said that we generally choose the parameters via cross-validation. We can perform this using `tune.svm()`, which selects the best parameters through 10-fold cross validation. Best parameters are parameters that result in the lowest cross-validation error rate. The best parameters for support vector machines with radial basis kernel are defined by:

```
tune.svm(credit.policy ~ installment + log.annual.inc + fico +
  days.with.cr.line + revol.bal + revol.util + inq.last.6mths +
  delinq.2yrs + not.fully.paid, data = data_train, kernel="radial",
  cost=c(0.1,1,10,100,1000), gamma=c(0.5,1,2,3,4))
```

For all other kernels we have to change the value of the argument `kernel`. After the code has been evaluated we obtain the best parameters for our support vector machines.

Table 8: Best parameters

Kernel	Cost	Gamma	Degree
Radial	10	0.5	-
Linear	0.01	-	-
Polynomial	0.5	1	2
Polynomial	1	1	3

Therefore, the best parameters for support vector machines with radial basis kernel involves `cost=10` and `gamma=0.5`. We then fit the training data again using its best parameters. Here is an example for support vector machines with radial basis kernel:

```
svmfit_b = svm(credit.policy ~ installment + log.annual.inc + fico
+ days.with.cr.line + revol.bal + revol.util
+ inq.last.6mths + delinq.2yrs + not.fully.paid,
data = data_train, kernel="radial", gamma=0.5, cost=10)
```

By applying the `predict()` function to the data test we obtain the predictions of test set using this model. We evaluate the support vector machines models using `confusionMatrix()` function. Following is the confusion matrix of support vector machines with radial kernel function:

Table 9: Support vector machines with radial basis kernel

	0	1
Negative	442	62
Positive	118	2251

We then calculate the accuracy, sensitivity, and specificity for all models.

Table 10: Accuracy, Sensitivity, and Specificity

	Radial	Linear	Polynomial $d = 2$	Polynomial $d = 3$
Accuracy	0.9373	0.9092	0.8824	0.9304
Sensitivity	0.7893	0.6125	0.4482	0.7375
Specificity	0.9732	0.9810	0.9875	0.9771

Table 10 provides measurement of performance of confusion matrix. The support vector machines with radial basis kernel has an accuracy of 0.9373. The accuracy, which indicates the success rate, is defined by dividing the sum of true positive and true negative with the number of observation.

### 3.5 Conclusion

Based on the `summary(lm_fit1)`, the multiple  $R^2$  for the linear model is 0.7071. This value means that the model explains the variability in the response quite well. We can also say that the selected features correlate with the predicted FICO score.

As for the binary response variable `credit.policy` in multiple logistic regression, the performance shown by confusion matrix says that the model classifies the sample very well with accuracy reaching 90,78%.

From table 7 and table 10 we can conclude that the support vector machines with radial basis kernel has the greatest accuracy between all models, this includes the logistic regression. Although this is one of advantages support vector machines have, the support vector machines do not calculate the probability of default. It can be said that support vector machines are suited for classification, but it need further smoothing to obtain the probability of default.

## 4 References


- James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). An Introduction to Statistical Learning with Applications in R. New York, USA: Springer Texts in Statistics.
- Murty, M.N., Raghava, R. (2016). Support Vector Machines and Perceptrons. Springer Briefs in Computer Science.
- Suzuki, J. (2020). Statistical Learning with Math and R. Springer.
- Awad, M., Khanna, R. (2015). Efficient Learning Machines. Apress.



## Declaration of Authorship

We hereby confirm that we have authored this Seminar paper independently and without use of others than the indicated sources. All passages (and codes) which are literally or in general matter taken out of publications or other sources are marked as such.

Berlin, 27 February 2022



Hans Alcahya Tjong



Ananda Eraz Irfananto