Empirical Results and Conclusion

Data Preparation

In this chapter we will discuss the application of our methods in R. We use the data *lendingclub.csv*[1]. 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
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

[1]https://www.kaggle.com/urstrulyvikas/lending-club-loan-data-analysis

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,]
```

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 [2]. We fit the training data using the `lm()` function. The `summary()` function is used in order to gather some information about the fit.

```
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)
steplm = stepAIC(lm_fit, direction = "both")
steplm$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
```

[2]The code is available at: https://github.com/hansalca1403/wissArbeiten/blob/main/P2P_Lending_Club.r

```
                + 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.

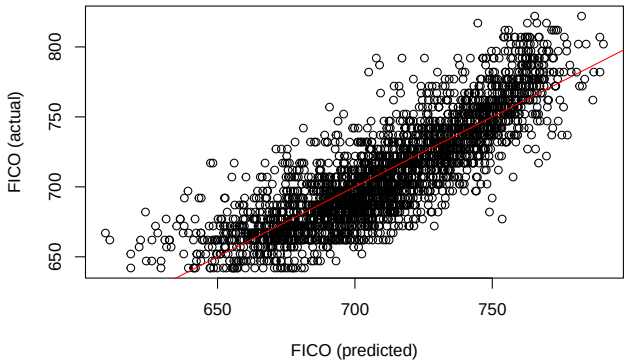

Figure 1: Multiple linear regression plot

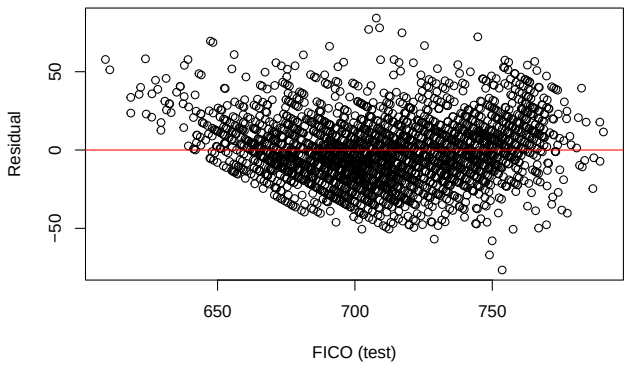

Figure 2: Residual plot

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 predictions 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 peformance 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

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 γ 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 kernal 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.

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 quiet 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.