**DOKUZ EYLÜL UNIVERSITY**

**ENGINEERING FACULTY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CME 4403-MACHINE LEARNING**

**TERM PROJECT**

**By**

**AKİF ÇAKAR**

**ULAŞ YURT**

**GÖKTUĞ KAYA**

**DECEMBER, 2019**

**İZMİR**

1. **Introduction**

Nowadays, banking systems can be cited as one of the areas where machine learning models are widely used. Within these banking systems, the area where machine learning models shine the most is to analyse the financial situation of the customers and to determine whether or not the desired loan is given to the customers, i.e. the credit assessment.

Credit assessment involves predicting applicant reliability and profitability. However, this prediction requires a detailed and rapid examination of many parameters, and cannot be done in a healthy way with the human eye and speed. At this point where human efforts are insufficient, necessary predictions can be made by creating successful and efficient credit assessment systems by using machine learning models.

In the term project, which is the subject of this paper, studies have been carried out in the field of credit assessment, where machine learning models are most commonly used, and studies have been conducted with three selected machine learning models on a sample data set. The selected machine learning models for the sample data set are as follows: Instance-based Algorithm (k-Nearest Neighbour), Decision Tree Algorithm (Iterative Dichotomiser 3), Bayesian Algorithm (Naive Bayes).

The aim of this term project is to adopt the machine learning models used and to understand the importance and necessity of machine learning by conducting studies in an area where these models are used extensively.

1. **Data Set Description**

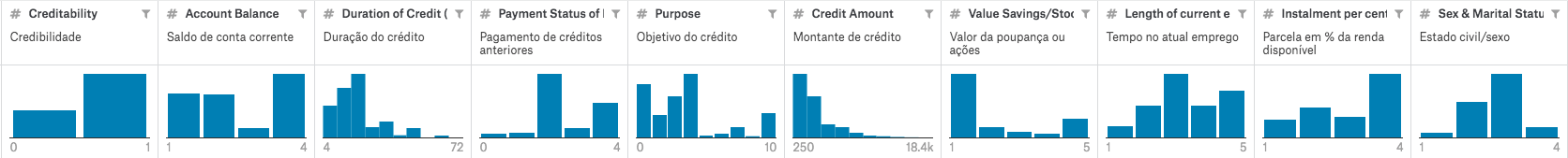
As a sample data set to be used and analysed in the term project, it was decided to use a data set containing the information of customers applying for a loan to a German bank.

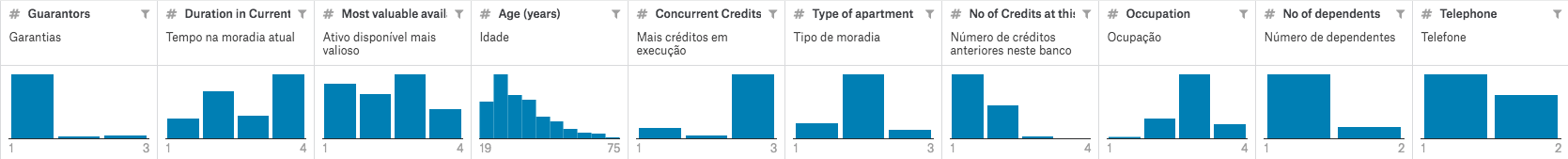
When a bank receives a loan application, based on the applicant's profile, the bank has to decide whether or not to approve the loan. Two types of risks are associated with the bank's decision:

* If the applicant has a good credit risk, i.e. he is likely to repay the loan, approving the loan to the person will not result in loss of business for the bank;
* If the applicant is a bad credit risk, i.e. he is not likely to repay the loan, approving the loan to the person will result in a financial loss to the bank.

In order to minimize loss from the bank's point of view, the bank needs a decision rule about who should approve the loan and who should not. An applicant's demographic and socioeconomic profiles are considered by loan managers before a decision is made regarding their loan application.

The dataset that we will build our decision rule on it, contains information on 20 variables and the rating of whether an applicant is considered a Good or Bad credit risk in 1000 loan applicants from a German Bank. Within this dataset, the 20 variables that we will use to determine whether or not a customer is risky to lend a loan are as shown in Figure 2.1.





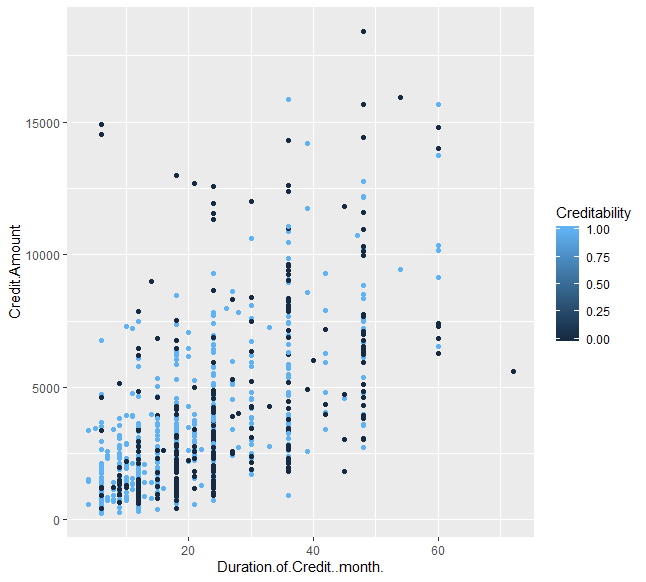
**Figure 2.1:** Sample Dataset Columns

In the German bank data set with the target class creditability, the detailed information of each variable and the effects of these variables on the target class are as shown in the Table 2.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Categories** | **Score** | **rel. frequency in % for** | |
| **good credits** | **bad credits** |
| kredit | Creditability: 1 : credit-worthy 0 : not credit-worthy | | | | |
| laufkont | Balance of current account | no balance or debit | 2 | 35.00 | 23.43 |
| 0 <= ... < 200 DM | 3 | 4.67 | 7.00 |
| ... >= 200 DM or checking account for at least 1 year | 4 | 15.33 | 49.71 |
| no running account | 1 | 45.00 | 19.86 |
| laufzeit | Duration in months (metric) | | | | |
| dlaufzeit | Duration in months (categorized) | <=6 | 10 | 3.00 | 10.43 |
| 6 < ... <= 12 | 9 | 22.33 | 30.00 |
| 12 < ... <= 18 | 8 | 18.67 | 18.71 |
| 18 < ... <= 24 | 7 | 22.00 | 22.57 |
| 24 < ... <= 30 | 6 | 6.33 | 5.43 |
| 30 < ... <= 36 | 5 | 12.67 | 6.86 |
| 36 < ... <= 42 | 4 | 1.67 | 1.71 |
| 42 < ... <= 48 | 3 | 10.67 | 3.14 |
| 48 < ... <= 54 | 2 | 0.33 | 0.14 |
| > 54 | 1 | 2.33 | 1.00 |
| moral | Payment of previous credits | no previous credits / paid back all previous credits | 2 | 56.33 | 51.57 |
| paid back previous credits at this bank | 4 | 16.67 | 34.71 |
| no problems with current credits at this bank | 3 | 9.33 | 8.57 |
| hesitant payment of previous credits | 0 | 8.33 | 2.14 |
| problematic running account / there are further credits running but at other banks | 1 | 9.33 | 3.00 |
| verw | Purpose of credit | new car | 1 | 5.67 | 12.29 |
| used car | 2 | 19.33 | 17.57 |
| items of furniture | 3 | 20.67 | 31.14 |
| radio / television | 4 | 1.33 | 1.14 |
| household appliances | 5 | 2.67 | 2.00 |
| repair | 6 | 7.33 | 4.00 |
| education | 7 | 0.00 | 0.00 |
| vacation | 8 | 0.33 | 1.14 |
| retraining | 9 | 11.33 | 9.00 |
| business | 10 | 1.67 | 1.00 |
| other | 0 | 29.67 | 20.71 |
| **Hoehe** **(Credit)** | Amount of credit in "Deutsche Mark" (metric) | | | | |
| dhoehe | Amount of credit in DM (categorized) | <=500 | 10 | 1.00 | 2.14 |
| 500 < ... <= 1000 | 9 | 11.33 | 9.14 |
| 1000 < ... <= 1500 | 8 | 17.00 | 19.86 |
| 1500 < ... <= 2500 | 7 | 19.67 | 24.57 |
| 2500 < ... <= 5000 | 6 | 25.00 | 28.57 |
| 5000 < ... <= 7500 | 5 | 11.33 | 9.71 |
| 7500 < ... <= 10000 | 4 | 6.67 | 3.71 |
| 10000 < ... <= 15000 | 3 | 7.00 | 2.00 |
| 15000 < ... <= 20000 | 2 | 1.00 | 0.29 |
| > 20000 | 1 | 0.00 | 0.00 |
| sparkont | Value of savings or stocks | < 100,- DM | 2 | 11.33 | 9.86 |
| 100,- <= ... < 500,- DM | 3 | 3.67 | 7.43 |
| 500,- <= ... < 1000,- DM | 4 | 2.00 | 6.00 |
| >= 1000,- DM | 5 | 10.67 | 21.57 |
| not available / no savings | 1 | 72.33 | 55.14 |
| beszeit | Has been employed by current employer for | unemployed | 1 | 7.67 | 5.57 |
| <= 1 year | 2 | 23.33 | 14.57 |
| 1 <= ... < 4 years | 3 | 34.67 | 33.57 |
| 4 <= ... < 7 years | 4 | 13.00 | 19.29 |
| >= 7 years | 5 | 21.33 | 27.00 |
| rate | Instalment in % of available income | >= 35 | 1 | 11.33 | 14.57 |
| 25 <= ... < 35 | 2 | 20.67 | 24.14 |
| 20 <= ... < 25 | 3 | 15.00 | 16.00 |
| < 20 | 4 | 53.00 | 45.29 |
| famges | Marital Status / Sex | male: divorced / living apart | 1 | 6.67 | 4.29 |
| male: single | 2 | 36.33 | 28.72 |
| male: married / widowed | 3 | 48.67 | 57.43 |
| female: | 4 | 8.33 | 9.57 |
| buerge | Further debtors / Guarantors | none | 1 | 90.67 | 90.71 |
| Co-Applicant | 2 | 6.00 | 3.29 |
| Guarantor | 3 | 3.33 | 6.00 |
| wohnzeit | Living in current household for | < 1 year | 1 | 12.00 | 13.43 |
| 1 <= ... < 4 years | 2 | 32.33 | 30.14 |
| 4 <= ... < 7 years | 3 | 14.33 | 15.14 |
| >= 7 years | 4 | 41.33 | 41.29 |
| verm | Most valuable available assets | Ownership of house or land | 4 | 22.33 | 12.43 |
| Savings contract with a building society / Life insurance | 3 | 34.00 | 32.86 |
| Car / Other | 2 | 23.67 | 23.00 |
| not available / no assets | 1 | 20.00 | 31.71 |
| alter | Age in years (metric) | | | | |
| dalter | Age in years (categorized) | 0 <= ... <= 25 | 1 | 26.67 | 15.71 |
| 26 <= ... <= 39 | 2 | 47.33 | 52.72 |
| 40 <= ... <= 59 | 3 | 21.67 | 26.14 |
| 60 <= ... <= 64 | 5 | 2.33 | 3.00 |
| >= 65 | 4 | 2.00 | 2.43 |
| weitkred | Further running credits | at other banks | 1 | 19.00 | 11.71 |
| at department store or mail order house | 2 | 6.33 | 4.00 |
| no further running credits | 3 | 74.67 | 84.29 |
| wohn | Type of apartment | rented flat | 2 | 62.00 | 75.43 |
| owner-occupied flat | 3 | 14.67 | 9.14 |
| free apartment | 1 | 23.33 | 15.57 |
| bishkred | Number of previous credits at this bank (including the running one) | one | 1 | 66.67 | 61.86 |
| two or three | 2 | 30.67 | 34.43 |
| four or five | 3 | 2.00 | 3.14 |
| six or more | 4 | 0.67 | 0.57 |
| beruf | Occupation | unemployed / unskilled with no permanent residence | 1 | 2.33 | 2.14 |
| unskilled with permanent residence | 2 | 18.67 | 20.57 |
| skilled worker / skilled employee / minor civil servant | 3 | 62.00 | 63.43 |
| executive / self-employed / higher civil servant | 4 | 17.00 | 13.86 |
| pers | Number of persons entitled to maintenance | 0 to 2 | 2 | 84.67 | 84.43 |
| 3 and more | 1 | 15.33 | 15.57 |
| telef | Telephone | no | 1 | 62.33 | 58.43 |
| yes | 2 | 37.67 | 41.57 |

**Table 2.1:** Dataset description table

One of the important points in the German bank data set is that the target class is a numerical variable, which means that as a result of the machine learning models we will apply, it is aimed to determine the eligibility ratio of the customer applying for a loan. This can be better expressed on the graph as shown in Figure 2.2, an example of the relationship between the two columns of the dataset.



**Figure 2.2:** Creditability graph according to two variables

1. **Pre-Processing**

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Pre-processing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values, therefore to execute random forest algorithm null values have to be managed from the original raw data set. Another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in one data set, and best out of them is chosen.

The German Bank data set used in this project was not subjected to intensive pre-processing because it is compatible with the machine learning models to be applied and there is no missing data. The data set was normalized with "center" and "scale" methods and made ready for use.

* library(ggplot2)
* library(gmodels)
* library(GGally)
* library(caret)

* data <- read.csv("german\_credit.csv")
* ####################### Pre-Process-Normalized
* preObj <- preProcess(data[, -21], method=c("center", "scale"))
* newData <- predict(preObj, data[, -21])
* #######################-visualization of the data set
* summary(data)
* ggpairs(data)
* ggplot(data = data, aes(x = Duration.of.Credit..month.,y = Credit.Amount, col = Creditability)) +
* geom\_point()

* ggplot(data = data, aes(x = Credit.Amount,y = Age..years., col = Creditability)) +
* geom\_point()
* #######################

Figure 3.1 Implementation of Pre-processing

Continuous data in Figure 3.1 data were normalized and preliminary preparation was provided. At the same time, the Figure 2.2 is shown on the graph.

1. **Machine Learning Models**

In this project, three different machine learning methods were used. These methods are k Nearest Neighbour Algorithm, Decision Tree Algorithm and Naïve Bayes Classifier. In this section, implementation and analysis of these models are going to be explained clearly.

* 1. **k-Nearest Neighbour**

k-Nearest Neighbour Algorithm is a method for classification and regression used on a data set. The instances which have the minimum distance value to the target feature are listed in ascending order and for a numerical target feature, mean of the first k instances; for a categorical target feature value, the majority of the first k instances are considered.

The calculation for this model is simply as mentioned above. But the real problem is, to find the optimum “k” value which can be used for analysing the data.

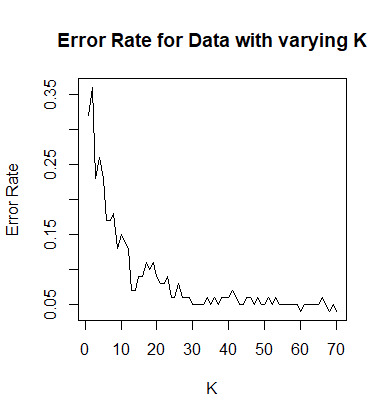


Figure 4.1.1 Error Rate with Varying K

By using the above graph (in Figure 4.1.1) and some calculations which are shown below(in Figure 4.1.2), we have calculated the optimal k value.

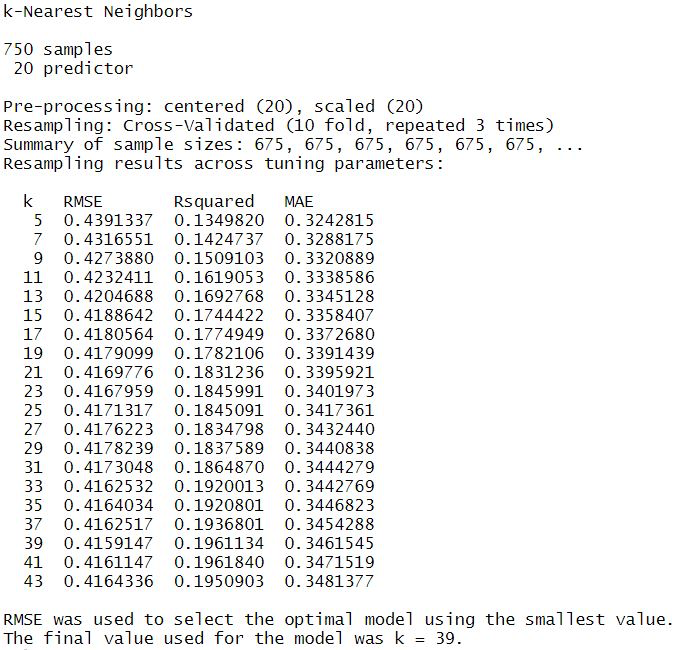


Figure 4.1.2 (Calculations with varying K value)

In Figure 4.1.2, k value is getting increased and RMSE (Root-Mean-Squared Error), R-Squared MAE (Mean Absolute Error) values are also changed. These values are used as a metric for expressing the error of a machine leaning model. Both them might have a value in range 0-∞.

You can see the calculation formulas for RMSE and MAE values below figures.

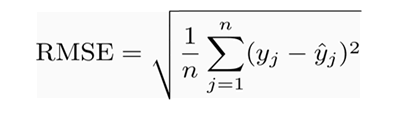


Figure 4.1.3 RMSE Formula

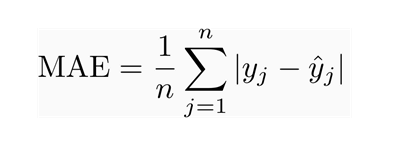


Figure 4.1.4 MAE Formula

In Figure 4.1.3 and Figure 4.1.4, y^ represents the predicted value and y represents the observed value of a feature. N is the total number of predictions made. The closer the result is to zero, the more successful predictions are.

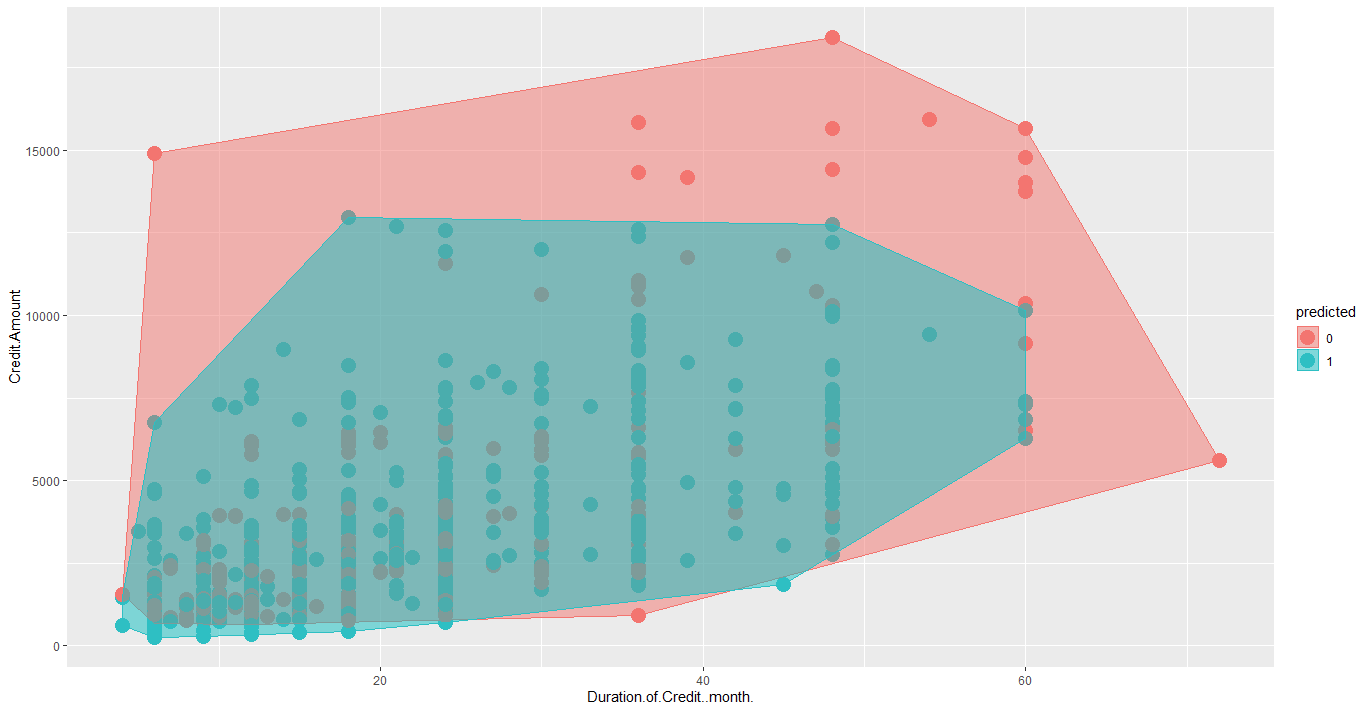


Figure 4.1.5 k-NN Map

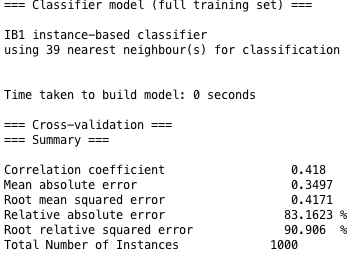


Figure 4.1.6 Error Calculation Results for k=39

As you can see in the above figure (Figure 4.1.6), results of calculations for the error values with k = 39, are displayed.

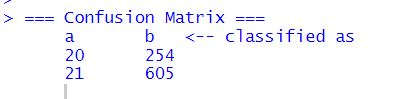


Figure 4.1.7 Confusion Matrix for k-NN

Now, we have all the values needed to calculate Accuracy, Precision and Recall. Formulas for the calculation of these values are explained in below figures.

|  |  |
| --- | --- |
| NEGATIVE (Predicted) | POSITIVE (Predicted) |
| True Negative | False Positive | NEGATIVE (Actual) |
| False Negative | True Positive | POSITIVE (Actual) |

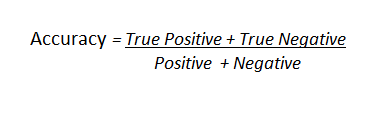


Figure 4.1.8 Accuracy Formula

By using the formula in the Figure 4.1.8;

Accuracy value for the k-NN Model

= (605 + 20) / (20+21+254+605) = 625 / 900

Accuracy = 0.694

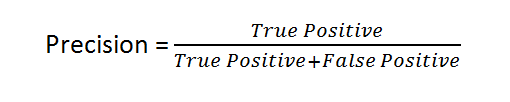


Figure 4.1.9 Precision Formula

By using the formula in the Figure 4.1.9;

Precision value for k-NN Model

= 605 / (605+254) = 14 / 45

Precision = 0.704

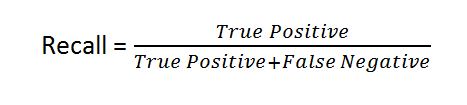


Figure 4.1.10 Recall Formula

By using the formula in the Figure 4.1.10;

Recall value for the k-NN Model

= 605 / (605+21) = 14 / 37

Recall = 0.966

k-NN Model for the dataset is explained above. You can see the implementation of this model in R-Programming Language in below figure. (Figure 4.1.11)

* library(**class**) # Contains the "knn" function
* library(plyr)
* library(ggplot2)
* library(ISLR)
* library(caret)
* setwd('C:\\Users\\akifc\\Desktop')
* data <- read.csv("german\_credit.csv")
* ##########################-Cross-Validation
* #Create 10 equally size folds
* folds <- cut(seq(1,nrow(data)),breaks=10,labels=FALSE)
* ##########################-Find Optimum k Value
* testIndexes <- which(folds==1,arr.ind=TRUE)
* testData <- data[testIndexes, ]
* trainData <- data[-testIndexes, ]
* #First try to determine the right K-value
* acc <- numeric() #holding variable

* highest\_accuracy <- -99
* highest\_k\_accuracy <- -1
* **for**(i **in** 1:70){
* predict <- knn(train=trainData, test=testData, cl=trainData$Creditability, k=i)
* acc <- c(acc, mean(predict==testData$Creditability))
* **if**(highest\_accuracy < acc[i]){
* #To find the value which has the highest accuracy.
* highest\_accuracy <- acc[i]
* highest\_k\_accuracy <- i
* }
* }
* #Plot error rates for k=1 to 30
* plot(1-acc, type="l", ylab="Error Rate", xlab="K", main="Error Rate for Data with varying K")
* #highest accuracy
* highest\_accuracy
* #k value has highest accuracy
* highest\_k\_accuracy
* ##############################################-Apply knn Algorithm || Cross Validation
* acc <- numeric() #reset acc list
* #Perform 10 fold cross validation
* **for**(i **in** 1:(10/2)){
* #Segement your data by fold using the which() function
* testIndexes <- which(folds==i,arr.ind=TRUE)
* testData <- data[testIndexes, ]
* trainData <- data[-testIndexes, ]
* predict <- knn(train=trainData, test=testData, cl=trainData$Creditability, k=highest\_k\_accuracy)
* acc <- c(acc, mean(predict==testData$Creditability))
* }
* mean(acc)
* summary(predict)
* ##############################################-Ploting
* # First use Convex hull to determine boundary points of each cluster
* plot.df1 = data.frame(x = plot.df$Duration.of.Credit..month.,
* y = plot.df$Credit.Amount,
* predicted = plot.df$predicted)
* find\_hull = function(df) df[chull(df$x, df$y), ]
* boundary = ddply(plot.df1, .variables = "Creditability", .fun = find\_hull)
* ggplot(plot.df, aes(Duration.of.Credit..month., Credit.Amount, color = predicted, fill = predicted)) +
* geom\_point(size = 5) +
* geom\_polygon(data = boundary, aes(x,y), alpha = 0.5)

Figure 4.1.11 Implementation of k-NN Model

* 1. **Decision Tree**

As the second Machine Learning Model, we have used Decision Tree (DT) Algorithm.

The algorithm chooses the most relevant feature for the prediction of a target. You can see the result tree of this model in the below figure. (Figure 4.2.1)

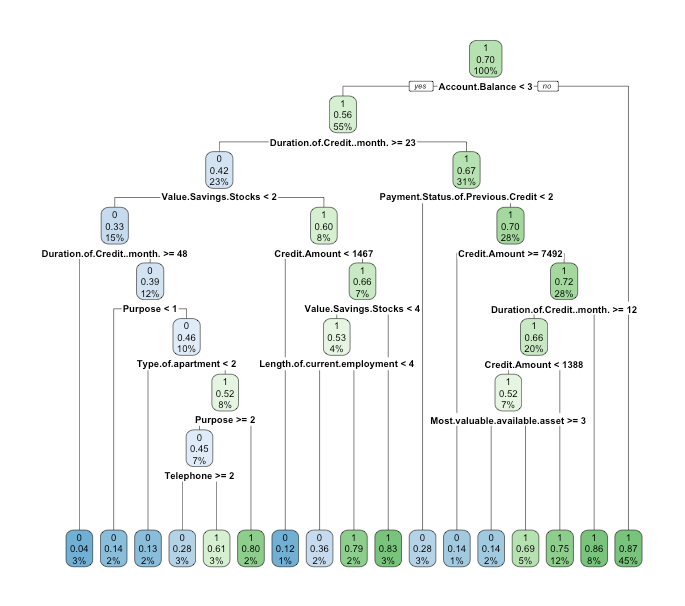


Figure 4.2.1 Final Decision Tree

You can see the error calculation results and the confusion matrix below. The calculations were done by using the formulas in Figure 4.1.3 and Figure 4.1.4.

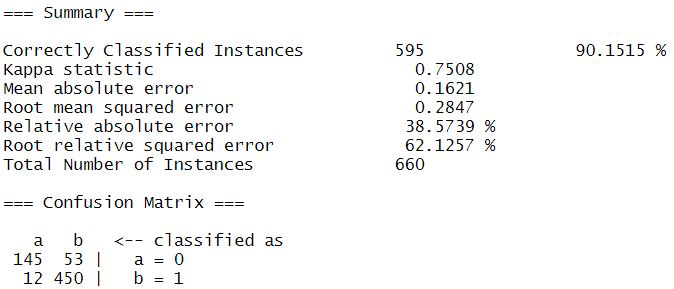


Figure 4.2.2 Error Calculations and Confusion Matrix for Decision Tree Model

Now, we have all the values needed to calculate Accuracy, Precision and Recall. Formulas for the calculation of these values are explained in below figures.

|  |  |
| --- | --- |
| NEGATIVE (Predicted) | POSITIVE (Predicted) |
| True Negative | False Positive | NEGATIVE (Actual) |
| False Negative | True Positive | POSITIVE (Actual) |

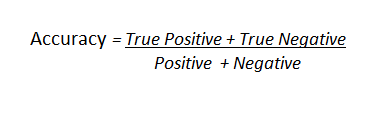


Figure 4.2.3 Accuracy Formula

By using the formula in the Figure 4.3.3;

Accuracy value for the DT Model

= (450 + 145) / (450 + 145 + 53 + 12) = 157 / 503

Accuracy = 0.901

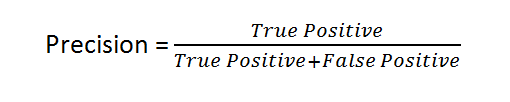


Figure 4.2.4 Precision Formula

By using the formula in the Figure 4.3.4;

Precision value for DT Model = 450 / (450+53) = 450 / 503

Precision = 0.894

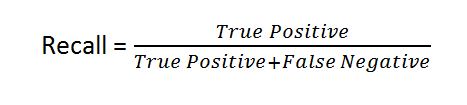


Figure 4.2.5 Recall Formula

By using the formula in the Figure 4.3.6;

Recall value for the DT Model = 450 / (450+12) = 450 / 462

Recall = 0.974

The calculations of metrics and the error values (Mean Absolute Error, Root-Mean Squared Error etc.) are shown in the above figures.

You can see the implementation of Decision Tree Model which contains tree creation and calculations. (Figure 4.2.5)

* library(ggplot2) # Data visualization
* library(rpart.plot)
* library(rpart)
* library(randomForest)
* data <- read.csv("german\_credit.csv")

* ################## Set random seed. Don't remove this line.
* **set**.seed(1)
* ################## Shuffle the dataset; build train and test
* n <- nrow(data)
* shuffled <- data[sample(n),]
* train <- shuffled[1:round(0.7 \* n),]
* test <- shuffled[(round(0.7 \* n) + 1):n,]
* ##################Fill in the model that has been learned.
* tree <- rpart(Creditability ~ ., data=train, method = "class")

* ################# cross validation
* ################## Set random seed
* **set**.seed(2)
* ################## Initialize the accs vector
* accs <- rep(0,10)
* ################## dividing data in 10 times (10 different train and test data) and --
* **for** (i **in** 1:10) {
* # These indices indicate the interval of the test set
* indices <- (((i-1) \* round((1/10)\*nrow(shuffled))) + 1):((i\*round((1/10) \* nrow(shuffled))))
* # Exclude them from the train set
* train <- shuffled[-indices,]
* # Include them in the test set
* test <- shuffled[indices,]
* # A model is learned using each training set
* tree <- rpart(Creditability ~ ., train, method = "class")
* # Make a prediction on the test set using tree
* pred <- predict(tree,test,type='class')
* # Assign the confusion matrix to conf
* conf <- table(test$Creditability,pred)
* # Assign the accuracy of this model to the ith index in accs
* accs[i] <- sum(diag(conf))/sum(conf)
* }
* ################## Print out the mean of accs
* mean(accs)
* summary(tree)
* ################### visualization
* rpart.plot(tree, extra = 106)

Figure 4.2.5 Implementation of Decision Tree

* 1. **Naïve Bayes**

As the third machine learning model, we have used the Naïve Bayes Classifier Algorithm. As an important point, in this method, it is necessary to use a dataset which has a categorical valued target feature. For classification, some probability calculations are done to implement this model and by using these probability values, the given target feature might be predicted as a categorical value. The larger the dataset, the more accurate predictions we can receive by using this model.

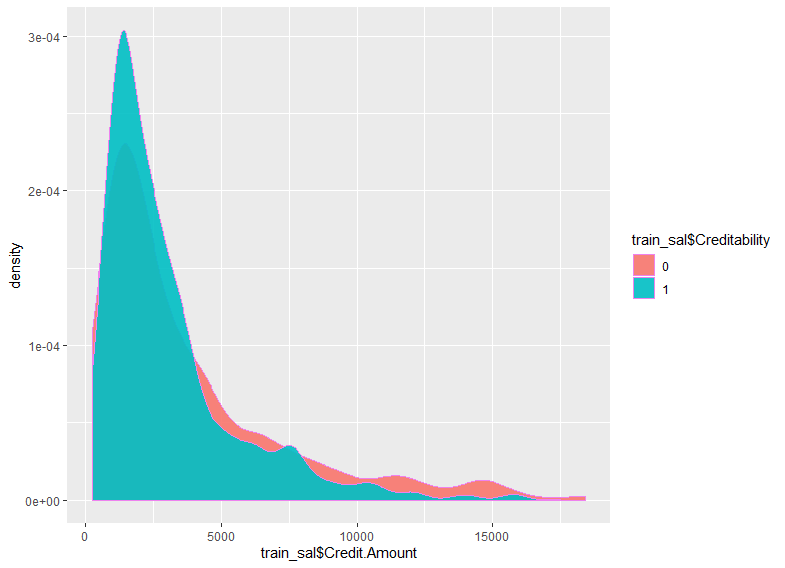
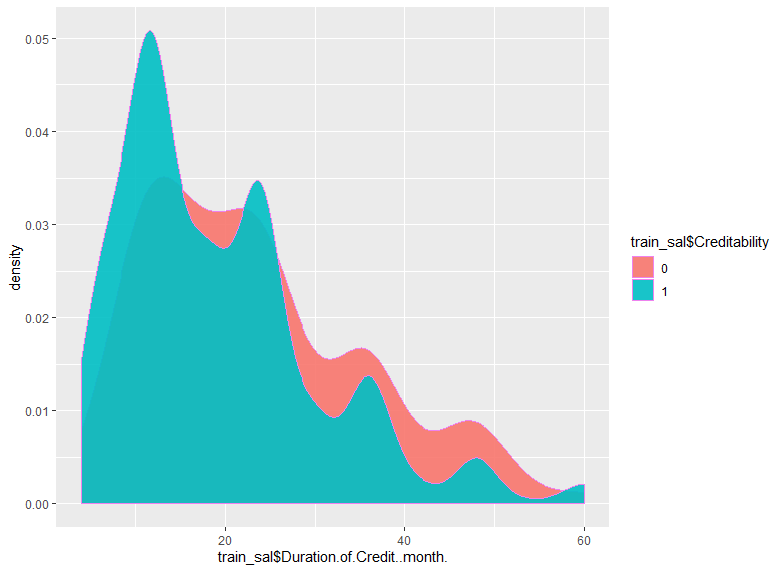


Figure 4.3.1 Density Graph For Attributes

As mentioned above, this model requires a categorical target feature for best results. So, we have transformed the target feature values from 1-0 to True-False.

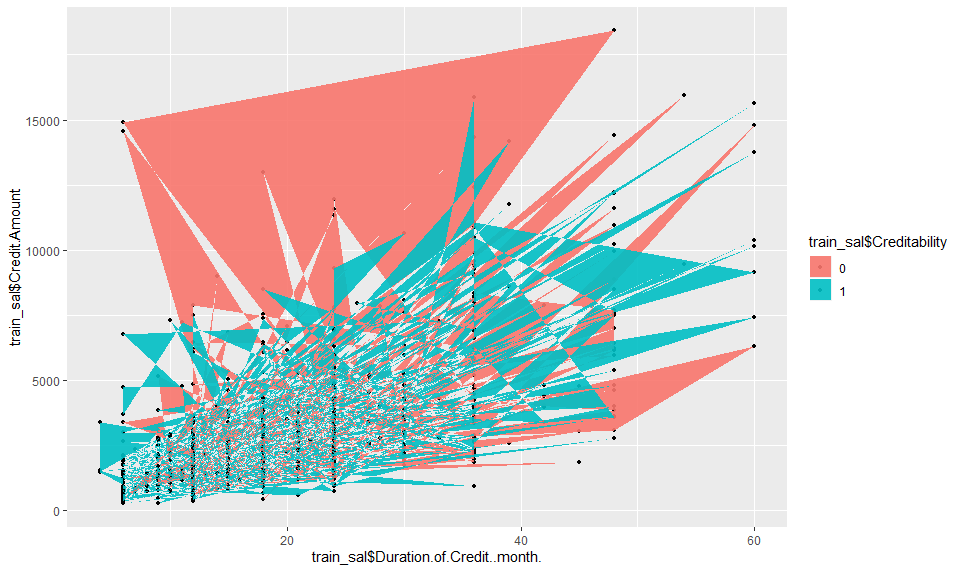


Figure 4.3.2 Naïve Bayes Map

As we have used the metrics for the correctness of previous models, we have also calculated them for Naive Bayes Classifier. The results obtained are displayed in the figure below.

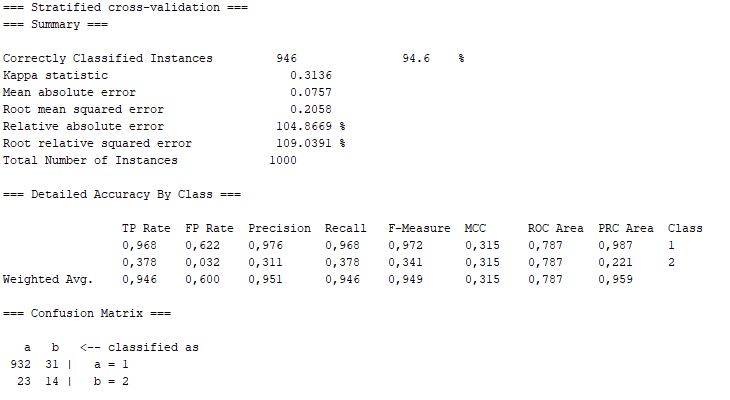
****

Figure 4.3.3 Error Calculations & Confusion Matrix for Naïve Bayes Model

As it was shown in the Figure 4.3.3, results of the error calculations are smaller values than the results of k-Nearest Neighbour Model. Mean Absolute Error and Root-Mean-Squared Error values are much closer to zero than the values obtained from k-Nearest Neighbour Model.

The confusion matrix is located at the bottom of the above figure. The elements of the matrix are named as above:

|  |  |
| --- | --- |
| NEGATIVE (Predicted) | POSITIVE (Predicted) |
| True Negative | False Positive | NEGATIVE (Actual) |
| False Negative | True Positive | POSITIVE (Actual) |

Now, we have all the values needed to calculate Accuracy, Precision and Recall. Formulas for the calculation of these values are explained in below figures.

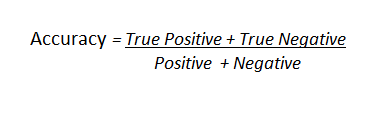


Figure 4.3.4 Accuracy Formula

By using the formula in the Figure 4.3.4;

Accuracy value for the Naïve Bayes Model

= (932 + 14) / (932+23+14+31) = 946 / 955

Accuracy = 0.946

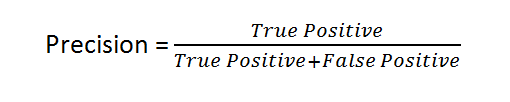


Figure 4.3.5 Precision Formula

By using the formula in the Figure 4.3.5;

Precision value for the Naïve Bayes Model

= 14 / (14+31) = 14 / 45

Precision = 0.311

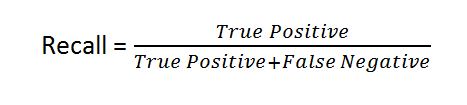


Figure 4.3.6 Recall Formula

By using the formula in the Figure 4.3.6;

Recall value for the Naïve Bayes Model

= 14 / (14+23) = 14 / 37

Recall = 0.378

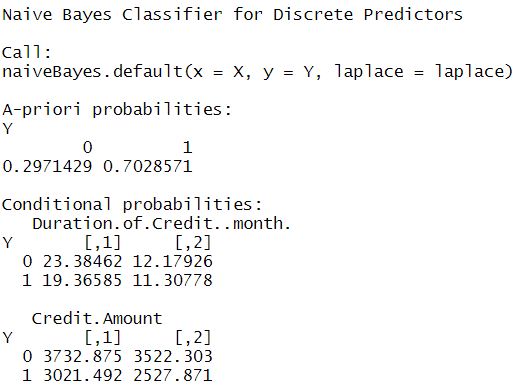
****

Figure 4.3.7 Test Result Of Naïve Bayes Classifier

You can see the implementation of Naïve Bayes Classifier Model using the R programming language below. (Figure 4.3.8)

1. # Libraries
2. library(naivebayes)
3. library(ggplot2)
4. library(caret)
5. library(psych)
6. library(e1071)
8. mydata <- read.csv("german\_credit.csv")
10. # Data(Train) # Data(Test)
11. **set**.seed(7267166)
12. trainIndex=createDataPartition(mydata$Creditability, p=0.7)$Resample1
13. train\_sal=mydata[trainIndex, ]
14. test\_sal=mydata[-trainIndex, ]
16. View(train\_sal)
17. train\_sal$Creditability <- **as**.factor(train\_sal$Creditability)
18. **class**(train\_sal)
20. View(test\_sal)
21. test\_sal$Creditability <- **as**.factor(test\_sal$Creditability)
22. **class**(test\_sal)


26. #Naive Bayes
27. data=naiveBayes(Creditability~Duration.of.Credit..month.+Credit.Amount, data=train)

30. #Density Plot
32. ggplot(data=train\_sal,aes(x = train\_sal$Duration.of.Credit..month., fill = train\_sal$Creditability)) +
33. geom\_density(alpha = 0.9, color = 'Violet')
35. ggplot(data=train\_sal,aes(x = train\_sal$Credit.Amount, fill = train\_sal$Creditability)) +
36. geom\_density(alpha = 0.9, color = 'Violet')

39. #Visualization
40. # Plot and ggplot
41. ggplot(data=train\_sal,aes(x=train\_sal$Duration.of.Credit..month., y = train\_sal$Credit.Amount, fill = train\_sal$Creditability)) +
42. geom\_point(size = 1) +
43. geom\_polygon(data = train\_sal, aes(train\_sal$Duration.of.Credit..month.,train\_sal$Credit.Amount), alpha = 0.9)

Figure 4.3.8 Naïve Bayes Model Implementation