DOKUZ EYLUL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CME4403 INTRODUCTION TO MACHINE LEARNING PROJECT REPORT

AIRLINE PASSENGER SATISFACTION PREDICTION

by

Yiğit Can Akçay

Fatih Semirgin

Ahmet Başbuğ

Lecturer

Assoc.Dr. Zerrin Işık

IZMIR

30.12.2023

CONTENTS

INTRODUCTION

1.1 Problem Definition

Today, air travel has become an indispensable mode of transportation for many people. However, the air travel experience can vary depending on many factors, which can affect the overall satisfaction level of passengers. In this paper, we consider an air passenger satisfaction forecasting problem.

Our dataset includes a variety of characteristics such as passengers' gender, customer type, age, travel purpose, class, flight distance and satisfaction levels with a range of services. Our main objective is to determine a traveler's satisfaction level with air travel based on these attributes.

This classification task is divided into two main classes: "Satisfied" and "Neutral or Dissatisfied". Using classification algorithms, the satisfaction level of a passenger can be predicted based on the characteristics in our dataset. This prediction can provide important information for airlines to improve service quality and enhance the passenger experience.

This study aims to provide important insights using data mining and machine learning techniques, focusing on customer satisfaction in the air travel industry. The machine learning algorithms used include K-Nearest Neighbors, Random Forest, Naive Bayes, Decision Tree, Support Vector Machines and Neural Network.

1.2 Dataset Description

This comprehensive dataset includes a large-scale survey to assess airline passenger satisfaction. It contains a total of 130,000 data samples, of which 26,000 are reserved for evaluation in the testing phase, while the remaining 104,000 samples are designed to be used in the training phase of the model.

In total, there are 25 different columns in this rich dataset, each containing a variety of attributes carefully selected to understand and predict the air travel experience. The names of the attributes in the dataset, their descriptions and the range of values they can take are given in Table 1.

Feature Name	Description	Values
X		Numeric (0 – 26.000)
Id	Unique identifier for each entry	Numeric (17 – 130.000)
Gender	Gender of the passengers	Female, Male
Customer Type	The customer type	Loyal customer, Disloyal customer
Age	The actual age of the passengers	Numeric (7 – 85)
Type of Travel	Purpose of the flight of the passengers	Personal Travel, Business Travel
Class	Travel class in the plane of the passengers	Business, Eco, Other
Flight Distance	The flight distance of this journey	Numeric (31 – 4983)
Inflight Wifi Service	Satisfaction level of the inflight wifi service	Numeric (0 – 5)
Departure/Arrival Time Convenient	Satisfaction level of Departure/Arrival time convenient	Numeric (0 – 5)
Ease of Online Booking	Satisfaction level of online booking	Numeric (0 – 5)
Gate Location	Satisfaction level of Gate location	Numeric (0 – 5)
Food and Drink	Satisfaction level of Food and drink	Numeric (0 – 5)
Online Boarding	Satisfaction level of online boarding	Numeric (0 – 5)
Seat Comfort	Satisfaction level of Seat comfort	Numeric (1 – 5)
Inflight Entertainment	Satisfaction level of inflight entertainment	Numeric (0 – 5)
On-board Service	Satisfaction level of On-board service	Numeric (0 – 5)
Leg Room Service	Satisfaction level of Leg room service	Numeric (0 – 5)
Baggage Handling	Satisfaction level of baggage handling	Numeric (1 – 5)
Check-in Service	Satisfaction level of Check-in service	Numeric (1 – 5)
Inflight Service	Satisfaction level of inflight service	Numeric (0 – 5)
Cleanliness	Satisfaction level of Cleanliness	Numeric (0 – 5)
Departure Delay in Minutes	Minutes delayed when departure	Numeric (0 – 1128)
Arrival Delay in Minutes	Minutes delayed when Arrival	Numeric (0 – 1.110)
Satisfaction	Airline satisfaction level	Satisfaction, Neutral or Dissatisfaction

Table 1 Dataset Features

1.3 Data Exploration and Preprocessing

In a context where the training and test data of the dataset were kept separately, the two datasets were merged and then blended. This method was applied to ensure that the dataset had a homogeneous distribution when evaluating the overall performance of the model.

```
train_dataset <- read.csv("train.csv")
test_dataset <- read.csv("test.csv")
combined_dataset <-rbind(train_dataset, test_dataset)
shuffled_data <-
combined_dataset[sample(nrow(combined_dataset)), ]
```

The "id" and "X" (row number) columns in the data set were removed from the data set as they were determined to be meaningless or unnecessary for the analysis process.

```
dataset <- dataset[, !(names(dataset) %in% c("id", "X"))]
```

Duplicate data in the dataset was checked and no duplicate rows were detected.

```
duplicate_rows <- dataset[duplicated(dataset), ]
cat("Number of duplicated rows : ",sum(duplicate_rows))

->Number of duplicated rows :  0
```

The data quality matrix of the attributes in the dataset is as shown in Table 2. As we can see, 'Age', 'Flight Distance', 'Inflight Service', 'Departure Arrival Time Conveniet', 'Ease of Online Booking', 'Gate Location', 'Food and Drink', 'Onlie Booking', 'Seat Comfort', 'Inflight Entartainment', 'Onboard Service', 'Leg Room Service', 'Baggage handling', 'Checking Service', 'Inflight Service', 'Cleanliness', and 'Departure Delay in Minutes' have 1 missing value each. Apart from that, 'Arrival Delay in Minutes' has 355 missing values. Even though there are various imputation techniques to fill these missing values, since we have 130 thousand data, we removed these instances from the dataset instead of imputation. These steps were taken to ensure the integrity of the dataset and its accuracy in the analysis process.

```
rows_with_missing_values <- dataset[apply(dataset, 1, function(x) any(is.na(x))), ]
dataset <- na.omit(dataset)
```

Feature	Type	Missing Values	Unique Values	Total Values	Mean Value	Standard Deviation	Min Value	Max Value	Median	1st Quartile	3rd Quartile
X	Numeric	0	89399	115375	37559.340	26536.128	0	89398	31711	14421.5	60554.5
id	Numeric	0	115375	115375	64905.480	37481.831	2	129880	64894	32442.5	97364.5
Age	Numeric	1	76	115375	39.430	15.106	7	85	40	27	51
Flight.Distance	Numeric	1	3809	115375	1189.060	995.755	31	4983	844	414	1744
Inflight.wifi.service	Numeric	1	7	115375	2.729	1.331	0	5	3	2	4
Departure.Arrival.time.convenient	Numeric	1	7	115375	3.057	1.528	0	5	3	2	4
Ease.of.Online.booking	Numeric	1	7	115375	2.755	1.402	0	5	3	2	4
Gate.location	Numeric	1	7	115375	2.975	1.278	0	5	3	2	4
Food.and.drink	Numeric	1	7	115375	3.205	1.331	0	5	3	2	4
Online.boarding	Numeric	1	7	115375	3.253	1.350	0	5	3	2	4
Seat.comfort	Numeric	1	7	115375	3.442	1.321	0	5	4	2	5
Inflight.entertainment	Numeric	1	7	115375	3.356	1.335	0	5	4	2	4
On.board.service	Numeric	1	7	115375	3.382	1.288	0	5	4	2	4
Leg.room.service	Numeric	1	7	115375	3.349	1.316	0	5	4	2	4
Baggage.handling	Numeric	1	6	115375	3.631	1.180	1	5	4	3	5
Checkin.service	Numeric	1	7	115375	3.307	1.266	0	5	3	3	4
Inflight.service	Numeric	1	7	115375	3.641	1.178	0	5	4	3	5
Cleanliness	Numeric	1	7	115375	3.286	1.314	0	5	3	2	4
Departure.Delay.in.Minutes	Numeric	1	453	115375	14.682	38.052	0	1592	0	0	12
Arrival.Delay.in.Minutes	Numeric	355	459	115375	15.044	38.429	0	1584	0	0	13
Gender	Categorical	0	3	115375							
Customer.Type	Categorical	0	3	115375							
Type.of.Travel	Categorical	0	3	115375							
Class	Categorical	0	4	115375							

Table 2 Data Quality Matrix

In the graphs below, the distribution of the classes of categorical data is shown by means of barplot. As can be seen in Figure 1, our target variable satisfaction is not balanced. Therefore, after outlier handling, the dataset was balanced by undersampling.

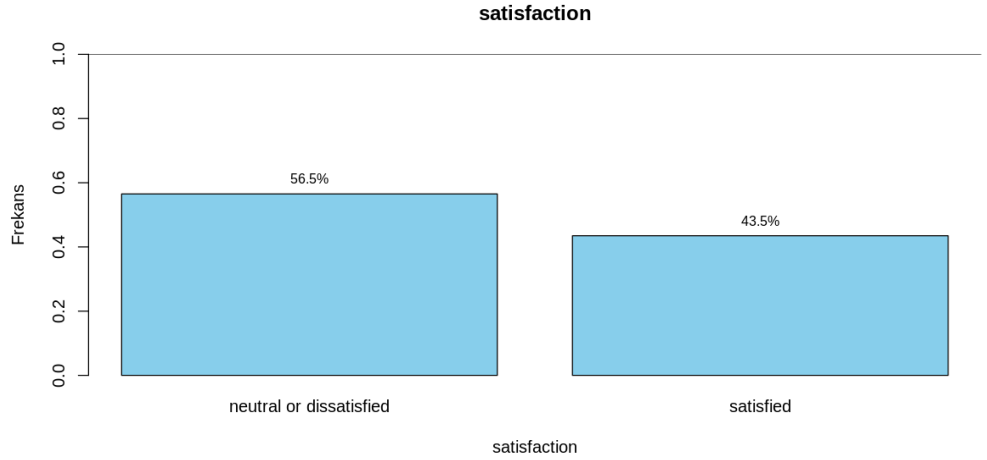

Figure 1 Satisfaction Barplot

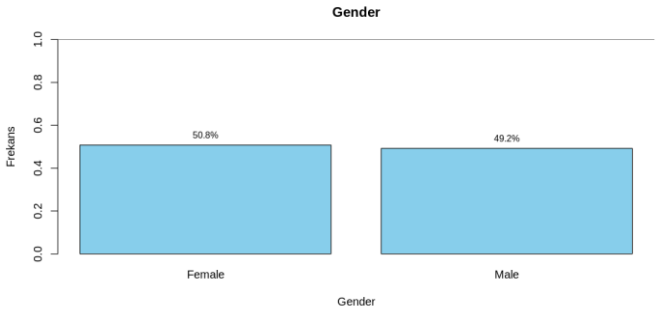

Figure 2 Gender Barplot

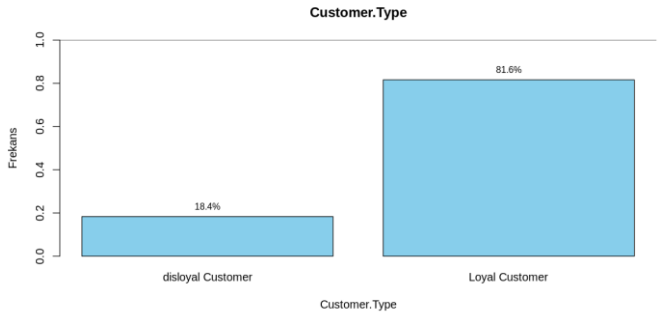

Figure 3 Customer Type Barplot

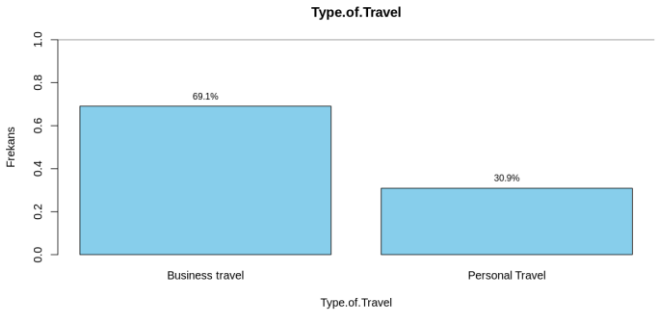

Figure 4 Type of Travel Barplot

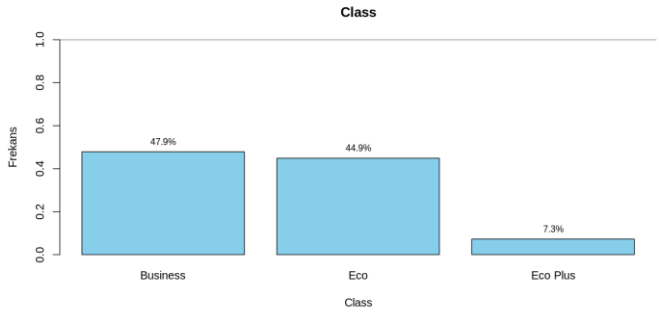

Figure 5 Class Barplot

Another operation is equal-width division of the feature named age. The interval was set as 10 as can be seen in Figure 6. As a result of binning, a normal distribution was obtained. This was done to make the data more understandable and to bring it into a form that fits the model better.

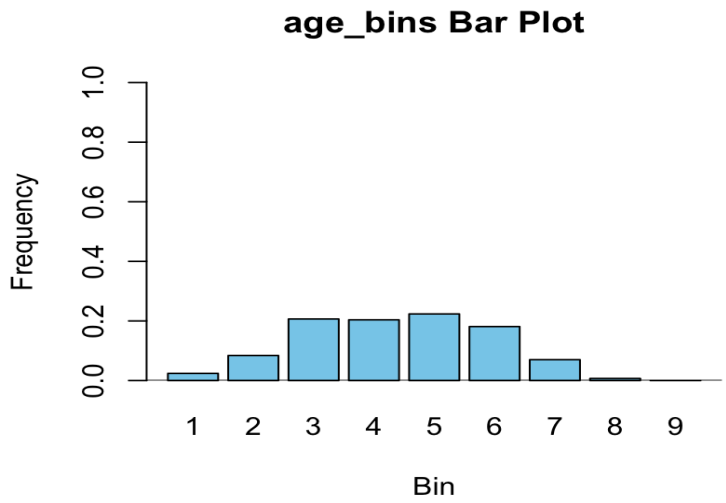

Figure 6 Age Bins Bar Plot

The graphs below show the histogram and boxplot plots of the numeric variables. With the boxplot, the outliers in the features can be visually recognized. As can be seen in the graph in Figure 7, when we look at the distribution of the feature named Flight Distance, we see a right skewed distribution, except for the values outside the boundaries of the boxplot, indicating the presence of outliers.

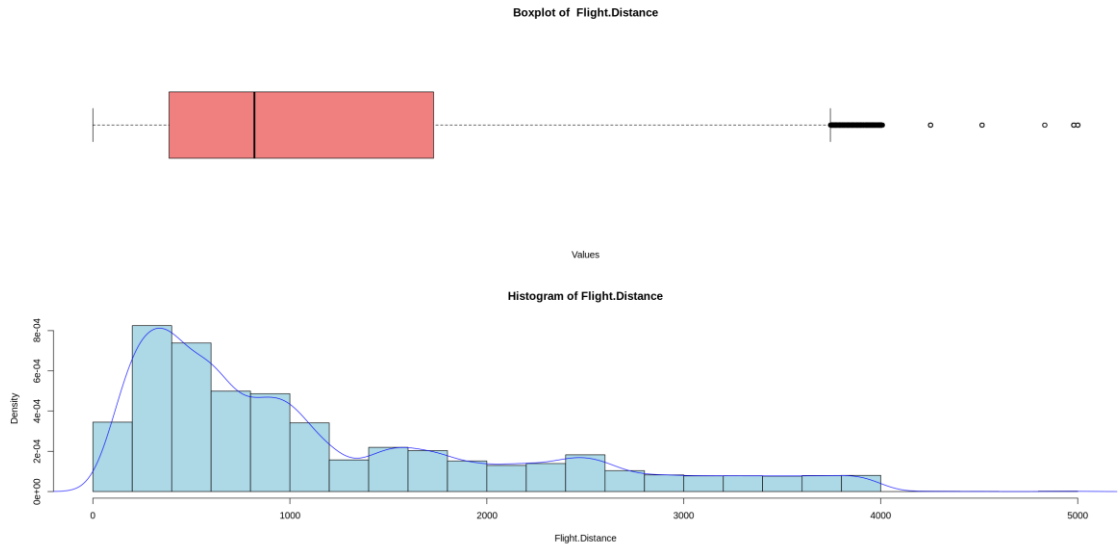

Figure 7 Boxplot and Histogram of Flight Distance

The distribution in Figure 8 is also a right skewed distribution, again outliers can be observed visually through the boxplot graph.

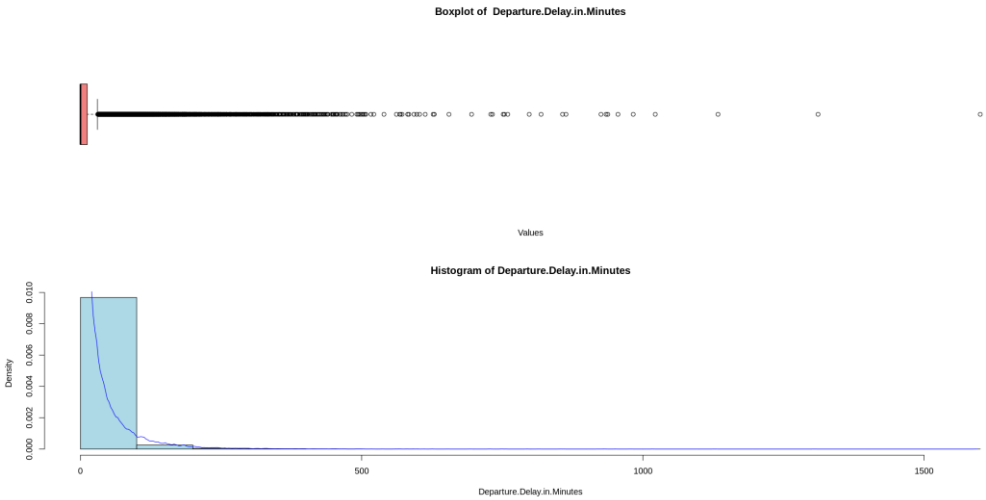

Figure 8 Boxplot and Histogram of Departure Delay In Minutes

Another outlier is observed in the feature called Arrival Delay in Minutes in Figure 9. Again, there is a right skewed distribution.

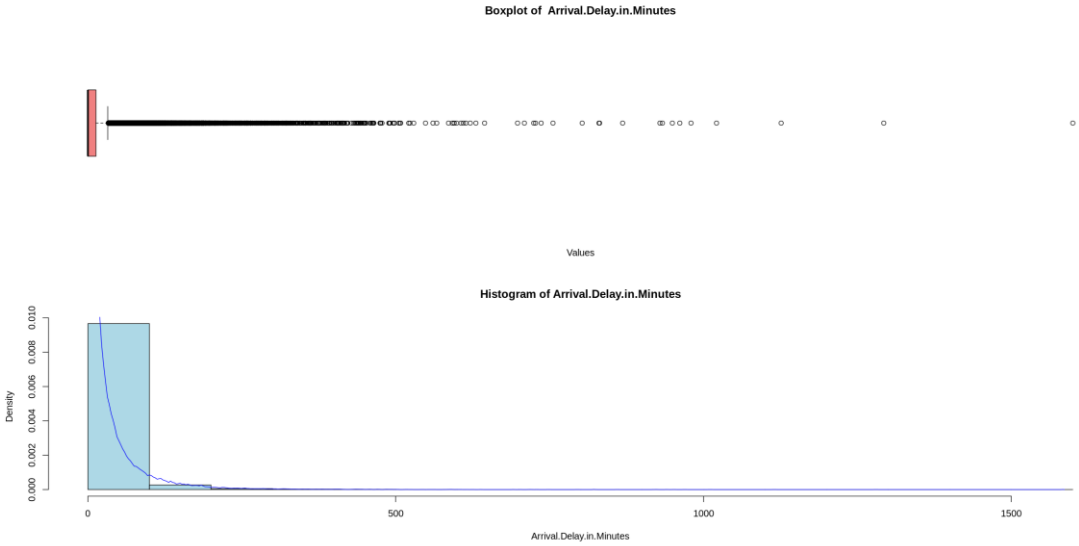

Figure 9 Boxplot and Histogram of Arrival Delay in Minutes

Finally, the attribute Checkin Service is the last one where we observe an outlier in Figure 10. Boxplot and histogram plots of other features are given on the other figures respectively.

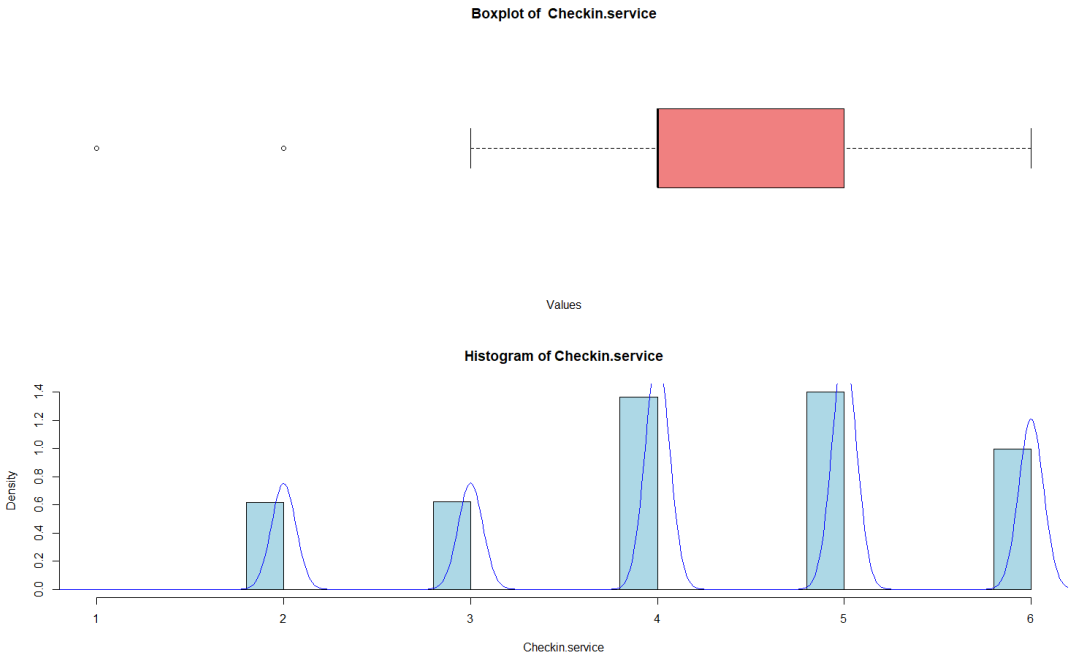

Figure 10 Boxplot and Histogram of Checkin Service

We used the "clamp" method, which is a method used to identify and organize outliers in the dataset. The "clamp_outlier" function we have defined basically determines the lower and upper bounds by using the quartiles (Q1 and Q3) and IQR (Interquartile Range) of the values in a numeric column of the dataset. It handles anomalies in the dataset by limiting the values outside the boundaries according to a set threshold value. The function provides information to the user by calculating the number and proportion of outliers before and after the process.

```
clamp_outlier <- function(x, threshold, column_name) {
  Q1 <- quantile(x, 0.25)
  Q3 <- quantile(x, 0.75)
  IQR_value <- Q3 - Q1
  lower_limit <- Q1 - threshold * IQR_value
  upper_limit <- Q3 + threshold * IQR_value
  outlier_count <- sum(x < lower_limit | x > upper_limit)
  outlier_ratio <- outlier_count / length(x)
  cat("Before Outlier Rate for ",column_name," --> ", outlier_ratio, "\n")
  x[x < lower_limit] <- lower_limit
  x[x > upper_limit] <- upper_limit
  outlier_count <- sum(x < lower_limit | x > upper_limit)
  outlier_ratio <- outlier_count / length(x)
  cat("After Outlier Rate for ",column_name," --> ", outlier_ratio, "\n")
  return(x)
}
threshold <- 1.5
numeric_columns <- names(dataset)[sapply(dataset, is.numeric)]
for (column_name in numeric_columns) {
  dataset[[column_name]] <- clamp_outlier(dataset[[column_name]], threshold,
column_name)
 }
```

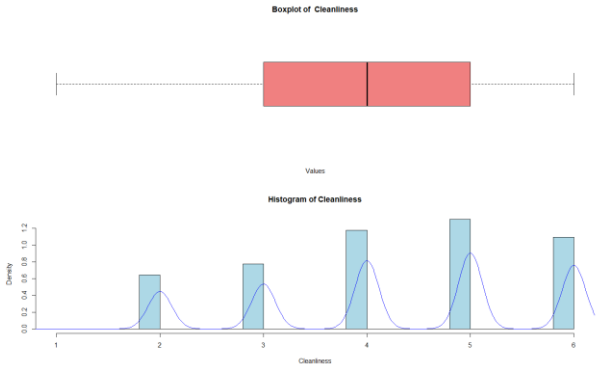

Figure 11 Boxplot and Histogram of Cleanliness

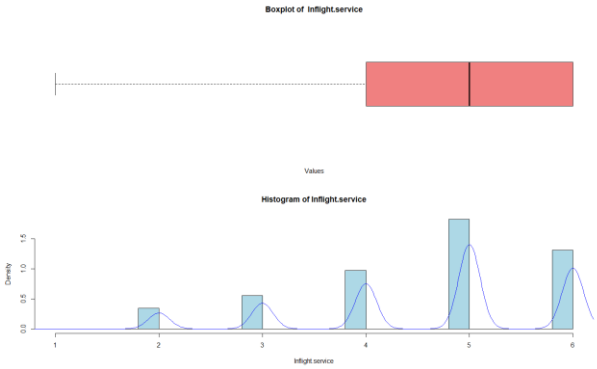

Figure 12 Boxplot and Histogram of Inflight Service

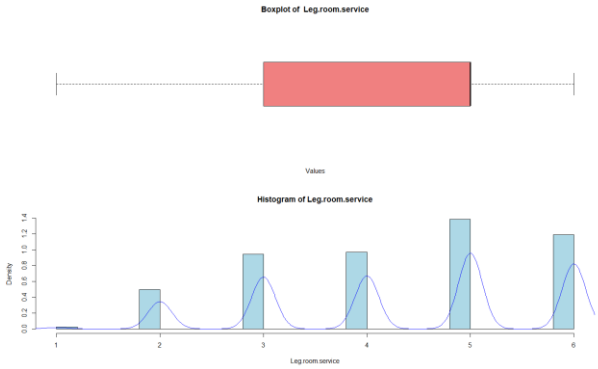

Figure 13 Boxplot and Histogram of Leg Room Service

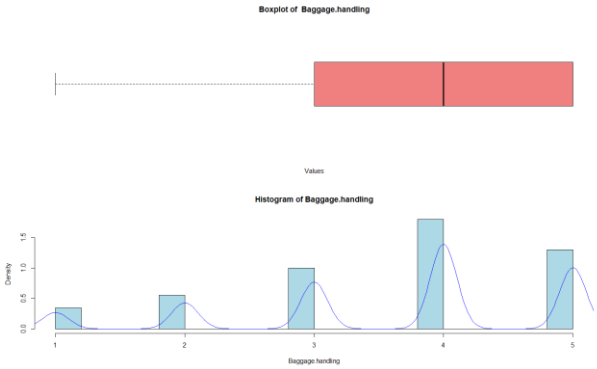

Figure 14 Boxplot and Histogram of Baggage Handling

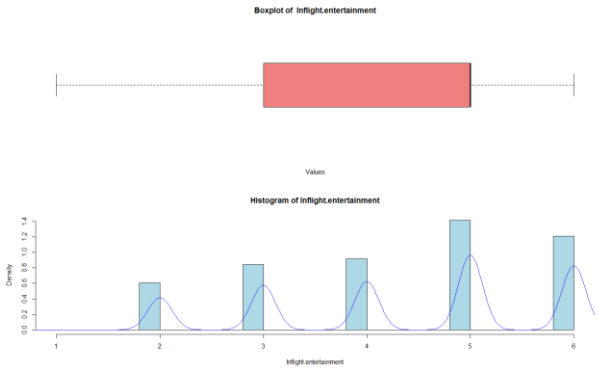

Figure 15 Boxplot and Histogram of Inflight Entertainment

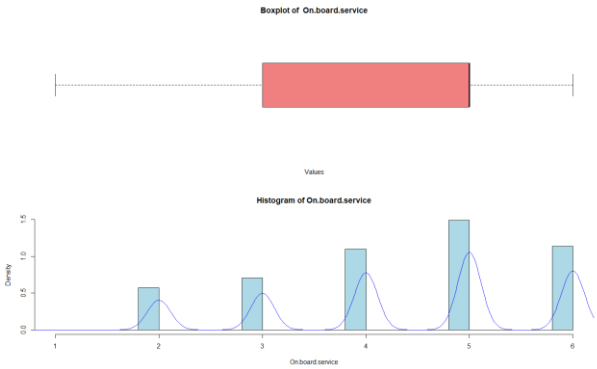

Figure 16 Boxplot of On Board Service

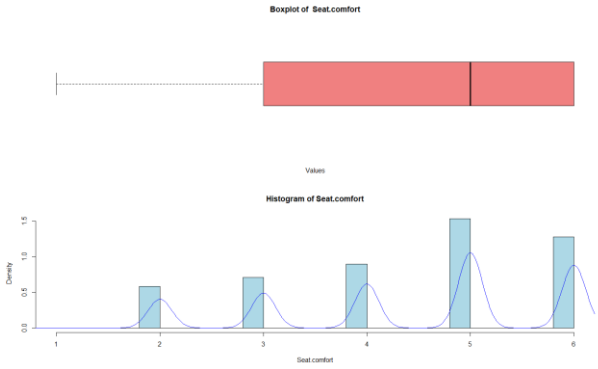

Figure 17 Boxplot and Histogram of Seat Comfort

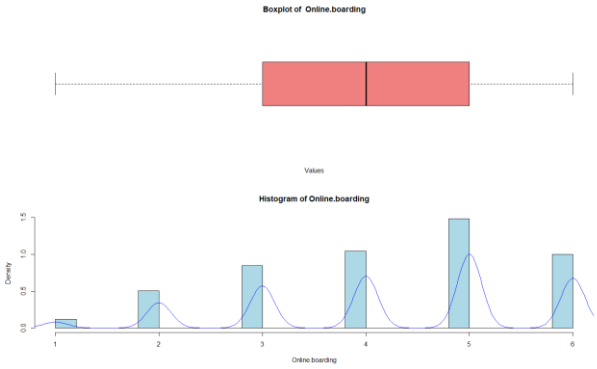

Figure 18 Boxplot and Histogram of Online Boarding

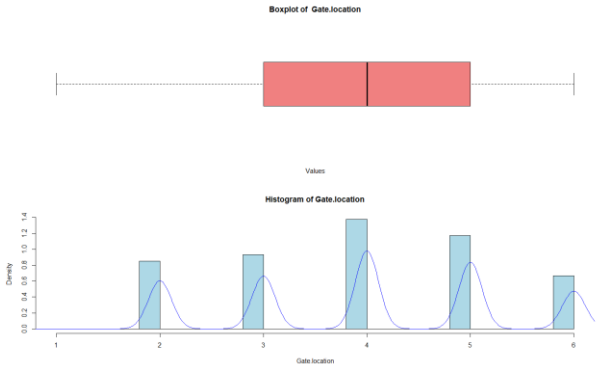

Figure 19 Boxplot and Histogram of Gate Location

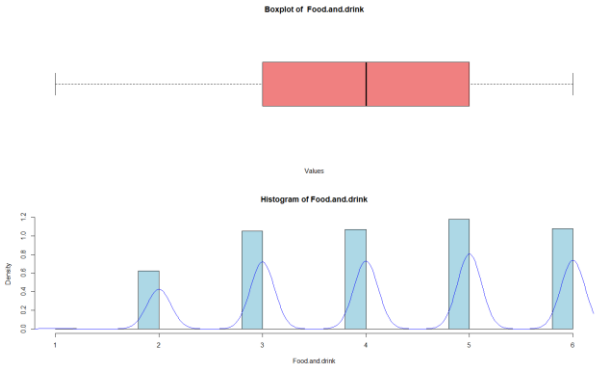

Figure 20 Boxplot and Histogram of Food and Drink

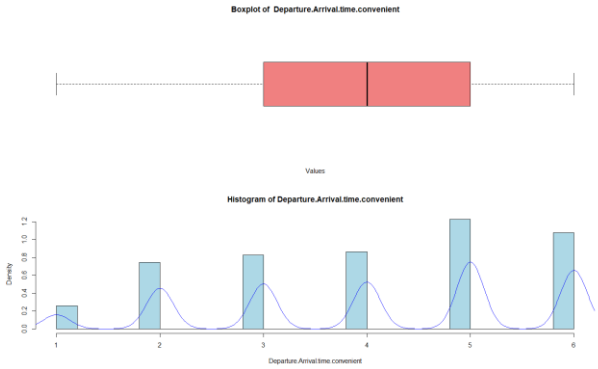

Figure 21 Boxplot and Histogram of Departure Arrival Time Convenient

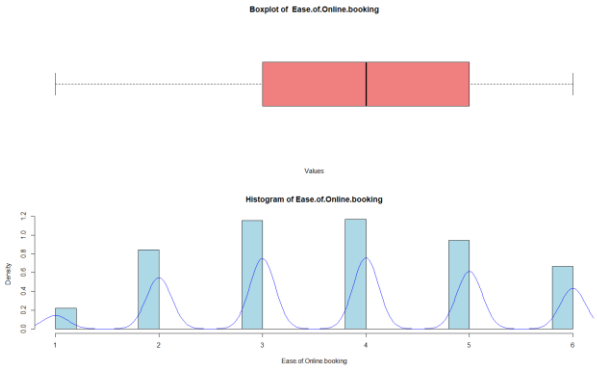

Figure 22 Boxplot and Histogram of Base of Online Booking

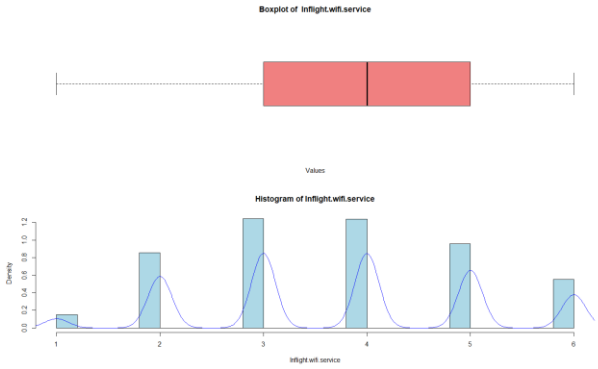

Figure 23 Boxplot and Histogram of Inflight Wifi Service

It can be seen other boxplot and histogram graphs above. There are no outlier in another features. The values before and after outlier handling are as in the output below:

```
Before Outlier Rate for  Flight.Distance  -->  0.0216658
After Outlier Rate for  Flight.Distance  -->  0
Before Outlier Rate for  Departure.Delay.in.Minutes  -->  0.1386976
After Outlier Rate for  Departure.Delay.in.Minutes  -->  0
Before Outlier Rate for  Arrival.Delay.in.Minutes  -->  0.134907
After Outlier Rate for  Arrival.Delay.in.Minutes  -->  0
Before Outlier Rate for  Checkin.service  -->  0.1236307
After Outlier Rate for  Checkin.service  -->  0
```

After that we use one-hot encoding for categorical features.

```
dummy <- dummyVars(" ~ .", data = dataset)
final_dataset <- data.frame(predict(dummy, newdata = dataset))
```

The correlation matrix in Figure 24 shows that there is a 100% negative correlation between the 2-class features in the dataset after one-hot encoding. These feature pairs are as follows:

GenderMale-GenderFemale

Customer.TypeLoyal.Customer-Customer.Typedisloyal.Customer

Type.of.TravelBusiness.travel-Type.of.TravelPersonal.Travel

satisfactionsatisfied-satisfactionneutral.or.dissatisfied

This shows that these attribute pairs are completely opposite to each other, with one attribute increasing in value while the other decreases. Such full negative correlations can cause problems in model training as they carry extra information in the dataset. Therefore, by selecting only one of these feature pairs and removing the other one, we aimed to train the model in a more effective and robust way.

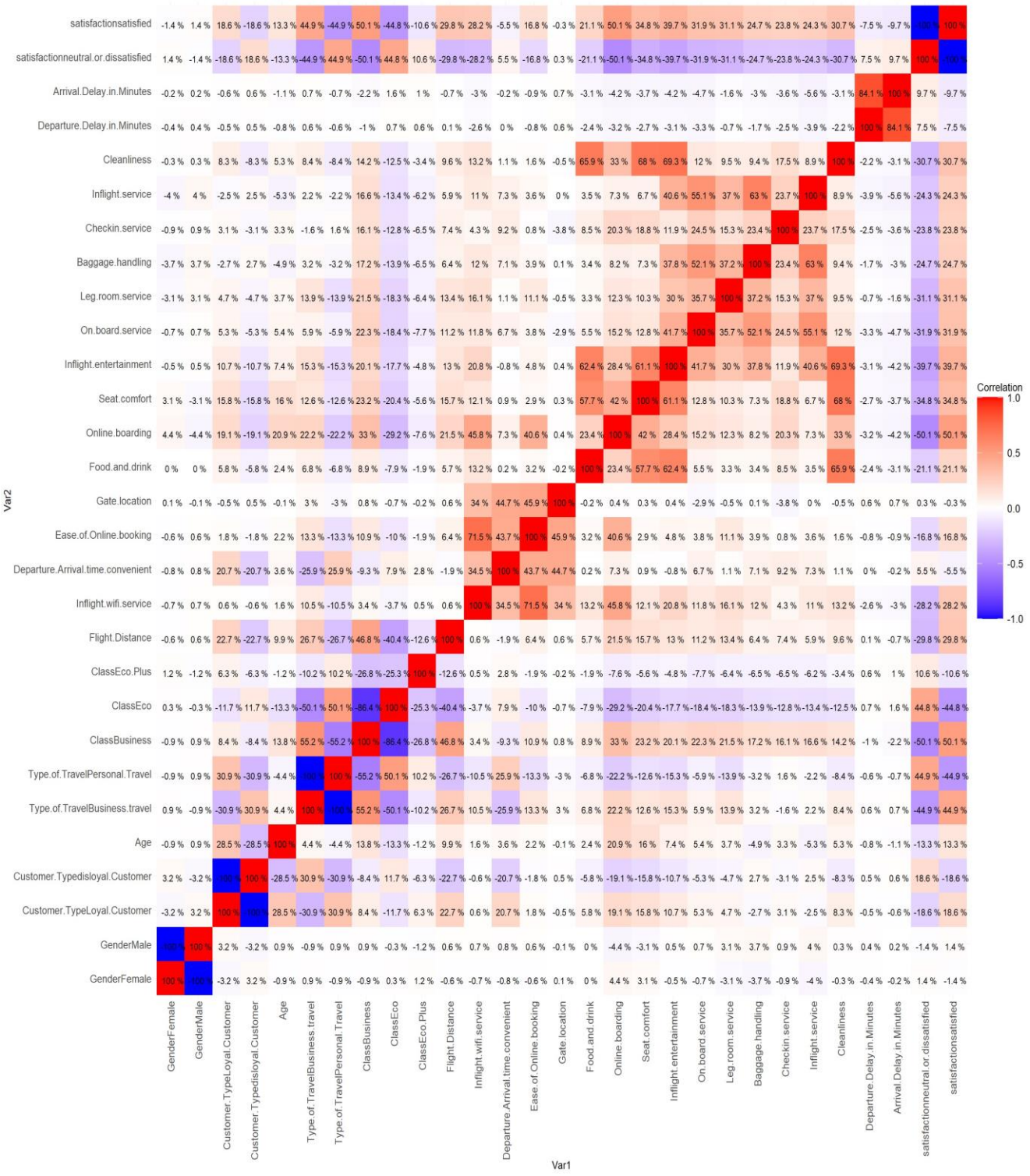

Figure 24 Correlation Matrix

The correlation matrix obtained after this process is as shown in Figure 25. One class of each feature pairs with over 90% correlation were removed.

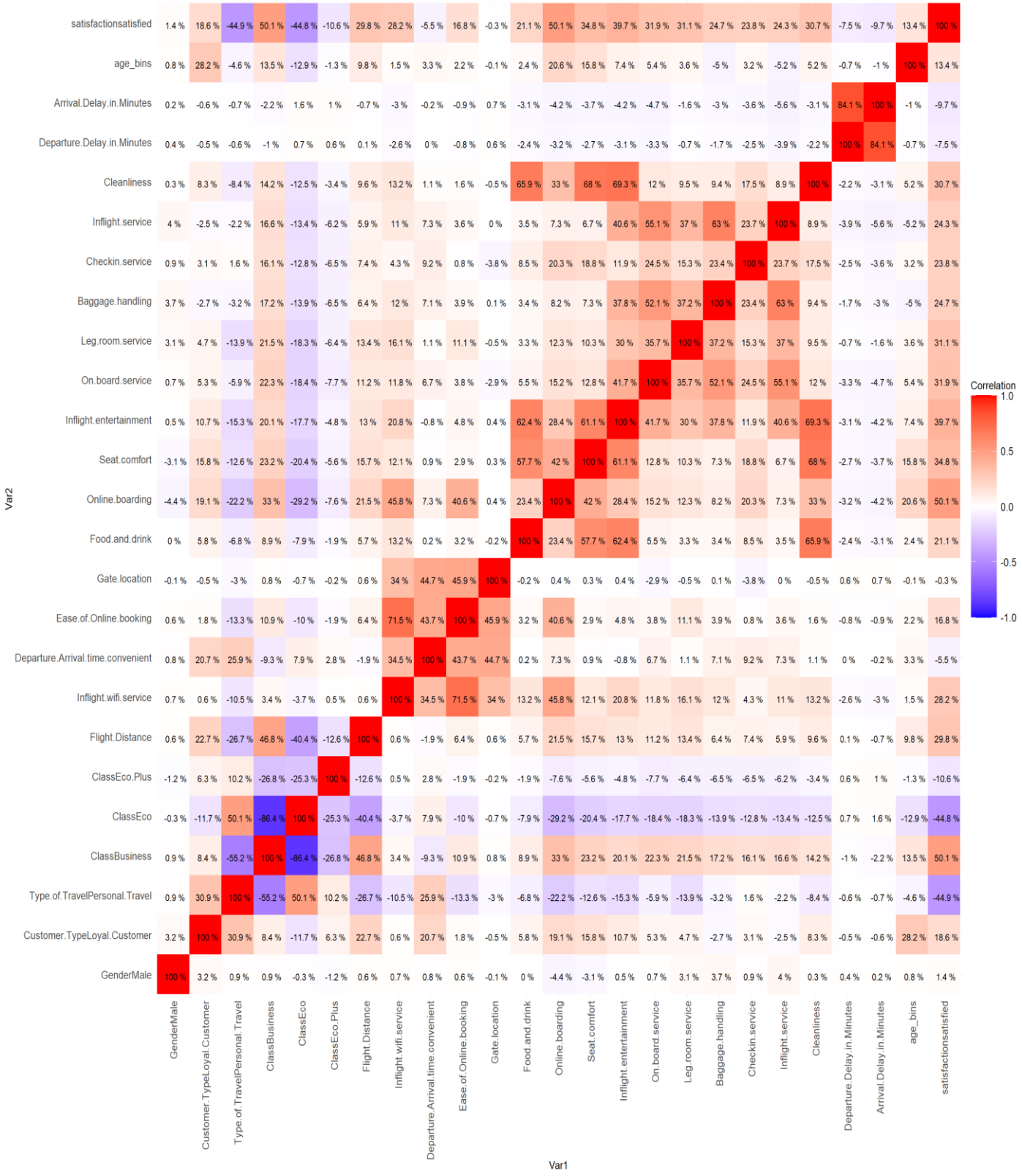

Figure 25 Final Correlation Matrix

UNDERSAMPLING

We created a new dataset by selecting 10,000 instances each of the "satisfaction" attribute belonging to the "satisfied" and "neutral or dissatisfied" classes.

```
sampled_data <- final_dataset %>%
  group_by(satisfactionsatisfied) %>%
  sample_n(size = 10000, replace = FALSE) %>%
  ungroup()
```

In this way, we made the size of our dataset more manageable and at the same time, we removed the imbalance of the target variable. That is, we equalized the number of samples for both classes. In the previous case, the imbalance of the target variable could have negatively affected the training of the model; however, with undersampling, we removed this imbalance and obtained a more reliable training set. It allowed the model to learn on both classes in a balanced way.

FEATURE SELECTION

Before proceeding with model training, we needed to make feature selection to get the best results. We used 2 methods for this. The first method used was PCA. PCA (Principal Component Analysis) stands for principal component analysis and is a data mining and dimensionality reduction technique used to reduce variability in multivariate data sets. PCA minimizes the amount of variability in the data set, creating new, fewer features. These new features try to reflect the underlying relationships between the features in the original dataset. The graph we extracted as a result of PCA is as shown in Figure 26.

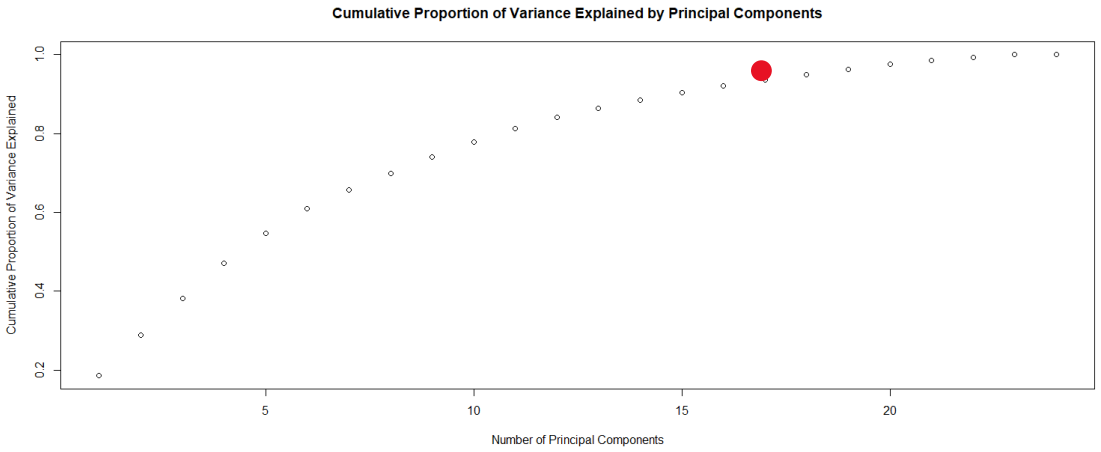

Figure 26 PCA Graph

As can be seen in the graph, a variance of over 90 percent can be obtained by using 17-18 features. The std dev, prop var, cum prop values obtained as a result of PCA are shown below.

Component	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Std Dev	2.1088	1.5724	1.50275	1.45443	1.34793	1.22327	1.0715	1.01046
Prop Var	0.1853	0.103	0.09409	0.08814	0.07571	0.06235	0.04784	0.04254
Cum Prop	0.1853	0.2883	0.3824	0.47054	0.54625	0.6086	0.65644	0.69898

Component	PC9	PC10	PC11	PC12	PC13	PC14	PC15	PC16
Std Dev	0.9883	0.97408	0.89422	0.81946	0.76085	0.68772	0.67609	0.65648
Prop Var	0.0407	0.03953	0.03332	0.02798	0.02412	0.01971	0.01905	0.01796
Cum Prop	0.7397	0.77922	0.81253	0.84051	0.86463	0.88434	0.90339	0.92134

Component	PC17	PC18	PC19	PC20	PC21	PC22	PC23	PC24
Std Dev	0.59538	0.5799	0.56412	0.54402	0.49053	0.42257	0.40452	1.28E-13
Prop Var	0.01477	0.01401	0.01326	0.01233	0.01003	0.00744	0.00682	0
Cum Prop	0.93611	0.95012	0.96338	0.97572	0.98574	0.99318	1	1

The other method we applied was RFE (Recursive Feature Elimination). In the context of machine learning and statistical modeling, RFE aims to identify the most important features by evaluating the importance of features (variables) in the dataset. RFE initially starts with the entire feature set and iteratively removes features to identify the ones that affect model performance the most. This process reduces the complexity of the model and at the same time minimizes the risk of overfitting.

Selected Features

Inflight.wifi.service, Online.boarding, Type.of.TravelPersonal.Travel, Checkin.service, Seat.comfort, Customer.TypeLoyal.Customer, Ease.of.Online.booking, Cleanliness, Baggage.handling, Inflight.entertainment, On.board.service, Inflight.service, Leg.room.service, age_bins, ClassBusiness, Departure.Arrival.time.convenient

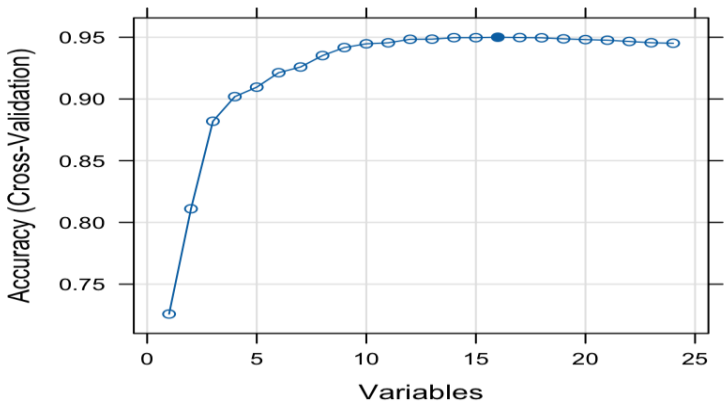

Figure 27 RFE Graph

MODELING

The dataset is divided into 80% training and 20% test data to train the machine learning model and evaluate its performance.

In order to obtain more reliable and generalizable results, this study uses the K-Fold Cross Validation method. This method is based on the principle of dividing the training dataset into k folds, using each fold as validation data in turn and using the remaining k-1 folds as training data. This allows the model to be trained and evaluated on different subsets of data.

Furthermore, the folds are fixed in order to ensure a fair comparison of the models tested during the study. That is, folds containing the same dataset were used for each model. This method allows us to evaluate the performance of different models more objectively. By ensuring this consistency, the reliability of the analysis is enhanced so that each model is given equal opportunities and the results can be reliably compared.

```
n <- 10

kfold_cv <- function(data, k) {
n <- nrow(data)
fold_size <- n %/% k
folds <- rep(1:k, each = fold_size)
return(folds)
}

folds <- kfold_cv(train_set, n)

for (i in 1:n) {
  train_data <- train_set[folds != i, ]
  validation_data <- train_set[folds == i, ]
   .
   . #modelling codes
   .
}
```

2.1 K- Nearest Neighbors Classifier

The K-NN (k-Nearest Neighbors) Classifier is a simple and effective machine learning algorithm used in a classification problem. The basic idea is to determine the class of a data point based on the class of its k-nearest neighbors around it.

The parameters used during the hyperparameter tuning process are as shown in Table 3.

Parameter	Values	Description
k	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}	Number of neighbours considered

Table 3 KNN Hyperparameters

The results are presented in the Table 4 below.

Fold	Train Accuracy	Train Kappa	Validation Accuracy	Validation Kappa
1	0.9091	0.8181	0.9263	0.8522
2	0.9112	0.8222	0.9156	0.8312
3	0.9108	0.8217	0.9125	0.8245
4	0.9110	0.8218	0.9044	0.8090
5	0.9119	0.8237	0.9206	0.8413
6	0.9098	0.8195	0.9288	0.8569
7	0.9096	0.8191	0.9213	0.8422
8	0.9105	0.8209	0.9213	0.8427
9	0.9104	0.8207	0.9175	0.8349
10	0.9110	0.8219	0.9169	0.8338
Average	0.9105	0.8210	0.9185	0.8369

Table 4 KNN Cross Validation Results

Test Accuracy	Test Kappa	Best Tune
0.9240	0.8481	K = 10

Table 5 KNN Test Results

		Actual	
		0	**1**
Prediction	**0**	1877	213
	1	91	1819

Table 6 Confusion Matrix of KNN Test Results

Metric	Value
Accuracy	0.924
95% CI	(0.9153, 0.932)
No Information Rate	0,508
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.8481
Mcnemar's Test P-Value	3.93E-12
Sensitivity	0.9538
Specificity	0,.8952
Pos Pred Value	0.8981
Neg Pred Value	0.9524
Prevalence	0.492
Detection Rate	0.4692
Detection Prevalence	0.5225
Balanced Accuracy	0.9245
'Positive' Class	0

Table 7 Evalution Metrics Results of KNN

2.2 Support Vector Machine Classifier (SVM)

SVM (Support Vector Machine) is a machine learning algorithm used especially in classification and regression problems. SVM uses a plane or hyperplane to classify data points into classes when building its learning model. Basically, it tries to find the widest margin (distance) between two classes and works to maximize this margin.

Parameter	Values
Cost	{0.1, 1, 10}
Gamma	{0.01, 0.1}
Kernel	{polynomial, radial}

Table 8 SVM Hyperparameters

Cost is a parameter that determines the level of error tolerance of the SVM. The Gamma parameter controls the width of the RBF kernel function. The Kernel parameter specifies the kernel function to be used. These functions transform the data space, making the non-linearly separable data set linearly separable.

Polynomial: Transforms the data according to polynomial degree and makes it separable.

RBF (Radial Basis Function): Provides non-linear separation using a radial basis kernel.

Fold	**Train Accuracy**	**Train Kappa**	**Validation Accuracy**	**Validation Kappa**
1	0.9835	0.9669	0.9463	0.8924
2	0.9833	0.9667	0.9425	0.8850
3	0.9840	0.9679	0.9375	0.8749
4	0.9847	0.9693	0.9294	0.8588
5	0.9840	0.9679	0.9356	0.8713
6	0.9822	0.9643	0.9531	0.9060
7	0.9833	0.9667	0.9513	0.9025
8	0.9843	0.9686	0.9419	0.8836
9	0.9830	0.9660	0.9506	0.9012
10	0.9835	0.9669	0.9375	0.8750
Average	0.9836	0.9671	0.9426	0.8851

Table 9 SVM Cross Validation Results

Test Accuracy	**Test Kappa**	**Best Tune**		
0.9483	0.8965	Cost = 10	Gamma = 0.1	Kernel = Radial

Table 10 SVM Test Results

		Actual	
Prediction		**0**	**1**
	0	1866	105
	1	102	1927

Table 11 Confusion Matrix of SVM Test Results

Metric	Value
Accuracy	0.9482
95% CI	(0.9409, 0.9549)
No Information Rate	0.508
P-Value [Acc > NIR]	<2e-16
Kappa	0.8965
Mcnemar's Test P-Value	0.8894
Sensitivity	0.9482
Specificity	0.9483
Pos Pred Value	0.9467
Neg Pred Value	0.9497
Prevalence	0.492
Detection Rate	0.4665
Detection Prevalence	0.4928
Balanced Accuracy	0.9482
'Positive' Class	0

Table 12 Evaluation Metrics Results of SVM

2.3 Decision Tree Classifier

This model is designed using a tree-like structure. The tree consists of a root node, branches, decision nodes and leaf nodes. The root node represents the most important feature and the branches lead to the branches of the tree based on the results of testing on that feature. Decision nodes contain the feature tests and leaf nodes represent classification or regression results. The tree building process continues by optimally partitioning the dataset and making decisions based on features.

Parameter	Values
cp	{0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10}

Table 13 Decision Tree Hyperparameters

The cp parameter stands for "complexity parameter". cp is used to control the size of the tree. This parameter limits the branching of a tree and thus prevents overfitting of the model. The larger the cp value, the simpler the decision tree generated. In other words, as the cp value increases, the branching of the tree decreases and a less complex model is obtained.

Fold	Train Accuracy	Train Kappa	Validation Accuracy	Validation Kappa
1	0.8927	0.7855	0.8969	0.7938
2	0.8923	0.7847	0.8838	0.7675
3	0.8897	0.7793	0.8856	0.7714
4	0.8914	0.7828	0.8881	0.7761
5	0.8923	0.7846	0.8719	0.7437
6	0.8903	0.7807	0.9044	0.8087
7	0.8898	0.7797	0.8831	0.7663
8	0.8897	0.7794	0.8900	0.7794
9	0.8897	0.7795	0.8913	0.7825
10	0.8899	0.7798	0.8875	0.7750
Average	0.8908	0.7816	0.8883	0.7764

Table 14 Decision Tree Cross Validation Results

Test Accuracy	Test Kappa	Best Tune
0.8885	0.7768	cp = 0.01

Table 15 Decision Tree Test Results

		Actual	
		0	**1**
Prediction	**0**	1659	142
	1	309	1890

Table 16 Confusion Matrix of Decision Tree

Metric	Value
Accuracy	0.8872
95% CI	(0.877, 0.8969)
No Information Rate	0.508
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.7741
Mcnemar's Test P-Value	5.43E-15
Sensitivity	0.843
Specificity	0.9301
Pos Pred Value	0.9212
Neg Pred Value	0.8595
Prevalence	0.492
Detection Rate	0.4148
Detection Prevalence	0.4502
Balanced Accuracy	0.8866
'Positive' Class	0

Table 17 Evaluation Metrics Resuls of Decision Tree

2.4 Naïve Bayes Classifier

The Naïve Bayes classifier is a supervised machine learning algorithm, which is used for classification tasks, like text classification. It is also part of a family of generative learning algorithms, meaning that it seeks to model the distribution of inputs of a given class or category. Unlike discriminative classifiers, like logistic regression, it does not learn which features are most important to differentiate between classes.

The parameters used during the hyperparameter tuning process are as shown in Table 4.

Parameter	Values	Description
usekernel	{TRUE, FALSE}	estimate the class conditional densities of metric predictors
laplace	{0, 0.5, 1}	value used for Laplace smoothing
adjust	{0.75, 1, 1.25, 1.5}	the bandwidth used is actually adjust*bw

Table 18 Naïve Bayes Hyperparameters

usekernel: Determines whether to use kernel density estimation (KDE) as the density estimation method. If TRUE, KDE is used. If FALSE, the normal distribution is used.

laplace: The value used for Laplace smoothing (additive smoothing). Laplace smoothing is used to avoid zero probabilities. If it is 0, no Laplace smoothing is performed. If it is a number greater than 0, it is added to each class-feature combination by this number.

adjust: This is the bandwidth value used for KDE. The bandwidth determines the amount of propagation of the KDE kernel function. If it is 0, the bandwidth is calculated automatically. If it is a number greater than 0, the bandwidth is set to this number.

Fold	Train Accuracy	Train Kappa	Validation Accuracy	Validation Kappa
1	0.8776	0.7552	0.8813	0.7628
2	0.8809	0.7619	0.8744	0.7488
3	0.8781	0.7562	0.8881	0.7764
4	0.8805	0.7611	0.8644	0.7283
5	0.8811	0.7623	0.8656	0.7311
6	0.8767	0.7535	0.8950	0.7902
7	0.8798	0.7595	0.8750	0.7505
8	0.8798	0.7598	0.8819	0.7624
9	0.8796	0.7592	0.8813	0.7626
10	0.8819	0.7638	0.8838	0.7675
Average	0.87959	0.75926	0.87906	0.75806

Table 19 Naïve Bayes Cross Validation Results

Test Accuracy	Test Kappa	Best Tune
0.88725	0.77414	usekernel = TRUE laplace = 0 adjust = 1.25

Table 20 Naïve Bayes Test Results

		Actual	
		0	**1**
Prediction	**0**	1659	142
	1	309	1890

Table 21 Confusion Matrix of Naïve Bayes Test Results

Metric	Value
Accuracy	0.8872
95% CI	(0.877, 0.8969)
No Information Rate	0.508
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.7741
Mcnemar's Test P-Value	5.425e-15
Sensitivity	0.8430
Specificity	0.9301
Pos Pred Value	0.9212
Neg Pred Value	0.8595
Prevalence	0.4920
Detection Rate	0.4148
Detection Prevalence	0.4502
Balanced Accuracy	0.8866
'Positive' Class	0

Table 22 Evaluation Metrics Resuls of Naïve Bayes

2.5 Neural Network

Neural networks are artificial neural networks inspired by the biological nervous system. These artificial neural networks are mathematical models used to perform complex information processing tasks. A neural network receives input data, processes it and learns to perform a specific task. Neural networks can consist of multilayer perceptrons (MLPs), with each layer containing a number of artificial neural cells. These neural cells process information by multiplying input data with weights and applying an activation function. Neural networks often use a back-propagation algorithm to adjust the weights throughout the learning process. With these features, neural networks have been successfully used in various tasks such as feature extraction, classification, regression and pattern recognition.

	Units	Activation	Optimizer	Loss
Layer Dense 1	64	Relu	RmsProp	Categorical Crossentropy
Layer Dense 2	32	Relu		
Layer Dense 3	2	Softmax		

Table 23 Neural Network Model

This model consists of a three-layer artificial neural network (Dense 1, Dense 2, Dense 3). The first layer, Dense 1, contains 64 neurons and uses the "Relu" (Rectified Linear Unit) activation function to process the input data. "Relu" is a function that outputs a positive value if the input value is positive and zero if the input value is negative.

The Dense 2 layer contains 32 neurons and also uses the "Relu" activation function. "Relu" helps stabilize the learning process by returning zero for negative inputs.

The Dense 3 layer contains 2 neurons and uses the "Softmax" activation function. The Softmax function converts the outputs into a probability distribution, producing outputs that are appropriately normalized for this problem. That is, it provides the estimated probability of each class.

Furthermore, RMSprop is a gradient-based optimization algorithm that adaptively adjusts the learning rate. It optimizes the training process by using a moving average of the previous gradient squares, adjusting the learning rates for each parameter separately.

Categorical Crossentropy is a loss function for multiclass classification problems. It encourages the model to make correct classifications by comparing real labels with the model's predictions. These two components combine to effectively train deep learning models.

The accuracy value, which we call Validation Split, is the accuracy value of the evaluation on the data separated as 20% of the training data. This validation set is used for early stopping. Early stopping is used to reduce the risk of model overfitting and increase generalization. During training, the performance of the model is evaluated on the validation set at the end of each epoch and if the performance no longer improves at a certain point (for example, if the validation set accuracy starts to decrease), the training is automatically stopped.

Using one of the callback functions, callback_early_stopping, the model's training process monitors the val_loss metric. The Patience parameter specifies the number of consecutive non-declining val_loss values during the training process and is used to determine the optimal performance of the model.

```
early_stopping <- callback_early_stopping(monitor = 'val_loss', patience = 7)
```

These parameters ensure that early stopping is triggered if val_loss does not decrease for 7 epochs during the training process. Thanks to this mechanism, the model can stop training before reaching its optimal performance, thus avoiding overfitting and increasing generalizability.

Fold	**Train Accuracy**	**Validation Split Accuracy**	**Cross Validation Accuracy**
1	0.9553	0.9403	0.9369
2	0.9605	0.9413	0.9413
3	0.9641	0.9448	0.9425
4	0.9592	0.9403	0.9262
5	0.9613	0.9441	0.9344
6	0.9611	0.9403	0.9506
7	0.9569	0.9378	0.9456
8	0.9623	0.9389	0.9419
9	0.9591	0.9431	0.9475
10	0.9596	0.9483	0.9294
Average	0.9599	0.9419	0.9396

Table 24 Neural Network Cross Validation Results

Best Loss	**Best Accuracy**	**Best Validation Loss**	**Best Validation Accuracy**	**Best Epoch**	**Best Step**
0.0941	0.9606	0.1185	0.9510	35	90

Table 25 Neural Network Test Results

As can be seen in Figure 28, while training the model on the training set, thanks to early stopping, the training was stopped when the value between the validation loss value and the training loss value started to open. Thus, overfit of the model is prevented.

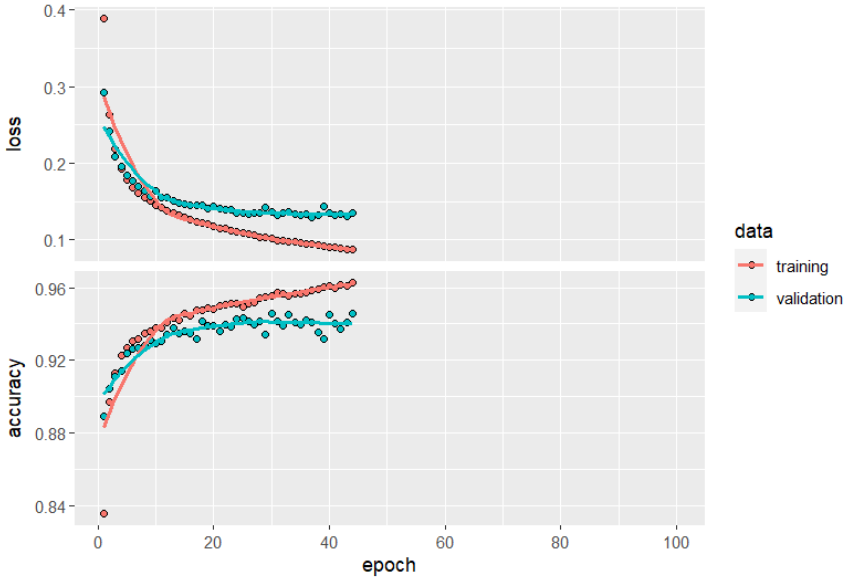

Figure 28 Accuracy and Loss per Epoch

		Actual	
		0	**1**
Prediction	**0**	1910	147
	1	58	1885

Table 26 Confusion Matrix of Neural Network

Metric	Value
Accuracy	0.9488
95% CI	(0.9415, 0.9554)
No Information Rate	0.508
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.8975
Mcnemar's Test P-Value	7.937e-10
Sensitivity	0.9705
Specificity	0.9277
Pos Pred Value	0.9285
Neg Pred Value	0.9701
Prevalence	0.492
Detection Rate	0.4775
Detection Prevalence	0.5142
Balanced Accuracy	0.9491
'Positive' Class	0

Table 27 Evaluation Metrics Results of Neural Network

2.6 Random Forest

Random Forest is a high-performance classification and regression algorithm widely used in machine learning. This algorithm is an ensemble learning method that combines multiple decision trees. Each decision tree is trained on a randomly selected sub-sample of data sets, providing diversity among the trees. Random Forest combines the predictions of these trees to obtain a more stable model with high generalization capabilities. It is also an effective method for feature selection because each tree can identify important features in the dataset by focusing on different features. Random Forest is robust to overfitting and can be successfully used in high-dimensional datasets and multi-class classification problems.

Parameter	Values
Mtry	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17}
Ntree	{25, 50, 100}

Table 28 Random Forest Hyperparameters

The Random Forest algorithm is an ensemble learning method based on two basic parameters, mtry and ntree. Mtry determines the number of features randomly selected in the construction of each decision tree, thus providing diversity by focusing each tree on different features. It is usually determined by rules such as square root (total number of features) or log2 (total number of features). Another important parameter, ntree, determines the number of decision trees in the ensemble. Larger ntree values usually provide better generalization ability, but can increase computation time. Choosing these parameters appropriately is important to optimize the performance of the Random Forest model and is often determined by techniques such as cross-validation.

Fold	Train Accuracy	Train Kappa	Validation Accuracy	Validation Kappa
1	0.94446	0.88891	0.95250	0.90495
2	0.94610	0.89219	0.94438	0.88874
3	0.94538	0.89075	0.94688	0.89366
4	094608	0.89213	0.93625	0.87251
5	0.94675	0.89350	0.94063	0.88126
6	0.94360	0.88719	0.96125	0.92231
7	0.94440	0.88879	0.95188	0.90371
8	0.94543	0.89083	0.95063	0.90120
9	0.94448	0.88894	0.95000	0.89999
10	0.94594	0.89186	0.94938	0.89875
Average	0.9453	0.8905	0.9484	0.8967

Table 29 Random Forest Cross Validation Results

Test Accuracy	Test Kappa	Best Tune
0.9585	0.9170	Mtry = 5, Ntree= 100

Table 30 Random Forest Test Results

		Actual	
		0	**1**
Prediction	**0**	1894	92
	1	74	1940

Table 31 Confusion Matrix of Random Forest

Metric	Value
Accuracy	0.9585
95% CI	(0.9519, 0.9645)
No Information Rate	0.508
P-Value [Acc > NIR]	<2e-16
Kappa	0.917
Mcnemar's Test P-Value	0.187
Sensitivity	0.9624
Specificity	0.9547
Pos Pred Value	0.9537
Neg Pred Value	0.9633
Prevalence	0.492
Detection Rate	0.4735
Detection Prevalence	0.4965
Balanced Accuracy	0.9586
'Positive' Class	0

Table 32 Evaluation Metrics Results of Random Forest

RESULTS AND CONCLUSION

	Cross Validated				Not Cross Validated	
	Train Accuracy	**Train Kappa**	**Validation Accuracy**	**Validation Kappa**	**Test Accuracy**	**Test Kappa**
KNN	0.9105	0.8210	0.9185	0.8369	0.9240	0.8481
SVM	0.9836	0.9671	0.9426	0.8851	0.9483	0.8965
Decision Tree	0.8908	0.7816	0.8883	0.7764	0.8885	0.7768
Naïve Bayes	0.8796	0.7593	0.8791	0.7581	0.8873	0.7741
Random Forest	0.9453	0.8905	0.9484	0.8967	0.9585	0.9170
Neural Network	0.9599		0.9396		0.9488	0.8975

Table 33 Overall Result

Random Forest shows the highest performance in all metrics except Train Accuracy; therefore, it can be considered as the most effective model. In the Cross Validated case, this model achieved high values of 0.9453 in Train Accuracy, 0.9484 in Validation Accuracy and 0.8964 in Validation Kappa. In the case of Not Cross Validated, it obtained high results such as 0.9585 in Test Accuracy and 0.9170 in Test Kappa. This model shows that it is able to learn the relationships between the variables in the dataset well.

The Neural Network also presents very impressive results and has an overall stable performance. In the Cross Validated case, this model achieved high values such as 0.9599 in Train Accuracy and 0.9396 in Validation Accuracy. In the Not Cross Validated case, it achieved impressive results such as 0.9488 in Test Accuracy and 0.8975 in Test Kappa. This shows that the model is able to capture the complexity and relationships in the dataset well.

The SVM model is also notable for its high accuracy rates. In the Cross Validated case, this model achieved very high values such as 0.9836 in Train Accuracy, 0.9426 in Validation Accuracy and 0.8851 in Validation Kappa. In the Not Cross Validated case, it obtained very high results such as 0.9483 in Test Accuracy and 0.8965 in Test Kappa.

The KNN and Decision Tree models show lower performance than the others, but still provide acceptable results. In the Cross Validated case, these models achieved values of 0.9105 and 0.8908 respectively in Train Accuracy, 0.9185 and 0.8883 respectively in Validation Accuracy, and finally 0.8369 and 0.7764 respectively in Validation Kappa. In the case of Not Cross Validated, they reached values of 0.9240 and 0.8885 in Test Accuracy and 0.8481 and 0.7768 in Test Kappa, respectively. These models show that they are moderately successful in separating the classes in the dataset.

The Naïve Bayes model has the lowest accuracy rates in this set. In the Cross Validated case, this model achieved low values such as 0.8796 in Train Accuracy, 0.8791 in Validation Accuracy and 0.7581 in Validation Kappa. In the case of Not Cross Validated, it obtained low results such as 0.8873 in Test Accuracy and 0.7741 in Test Kappa. This model shows that it is insufficient to separate the classes in the dataset.

In conclusion, according to this evaluation, Neural Network, Random Forest and SVM models are the most appropriate models for this dataset. KNN and Decision Tree models are considered as less suitable models. Naïve Bayes model is considered as an unsuitable model for this dataset.

REFERENCES

1. https://tensorflow.rstudio.com/tutorials/
2. https://keras.io/getting_started/
3. https://machinelearningmastery.com/
4. https://medium.com/
5. https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction
6. http://topepo.github.io/caret/train-models-by-tag.html