Question 1

# It generates descriptive statistics of the variables in the data frame and provides a concise summary of the data.

describe(data)

# Due to the large data, only partial output is shown below.

# Firstly, load the data into a data frame.

data <- read.csv("./creditworthiness.csv")

Table

Description automatically generated with medium confidence

# A quicker way to check for missing values for the whole data.

From the output, there is no missing values in the data.

A picture containing text

Description automatically generated

To check for outliers

# Using R base (with the number of bins corresponding to the square root of the number of observations in order to have more bins than the default option):

Text

Description automatically generated

Chart, histogram

Description automatically generated

Text

Description automatically generated

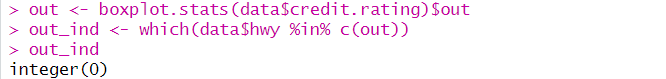
# A boxplot helps to visualize a quantitative variable by displaying five common location summary (minimum, median, first and third quartiles and maximum) and any observation that was classified as a suspected outlier using the interquartile range (IQR) criterion.

Chart, box and whisker chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated



# Print the values of the outliers directly on the boxplot with the mtext() function

# The which() function it is possible to extract the row number corresponding to these outliers



Chart, box and whisker chart

Description automatically generated

With the above methods, we can see that there are no outliers in the data.

A picture containing text

Description automatically generated

# After a quick analysis, it is noted there are rows of records that have not been classified. As such these rows of records cannot be used in the testing of the model for prediction. Hence I decide to excluded these data using a subset function

# The classifiedData has 1962 records that have a defined credit rating(column 46 is greater than 0). This data will be used for building and training a prediction model.

The unclassified data refers to the observations in the data set that do not have a defined credit rating (column 46 is equal to 0). This data will be used for making predictions based on the trained prediction model.





# To establish statistical property of various attributes against the credit.rating, I use correlation function to establish their relationship. In the code, the cor function is used to calculate the correlation between each attribute and the target attribute, "credit.rating". The abs function is used to obtain the absolute values of the correlation, since the correlation values are not important here, but rather the magnitude of their relationship.



Graphical user interface, text

Description automatically generated

# The order function is then used to sort the correlation values in descending order, so that the attributes with the highest correlation can be identified. The head function is used to show the top five attributes with the highest correlation.

A picture containing logo

Description automatically generated

No, it is not possible for a prediction model to obtain 100% prediction accuracy. The quality of the data used to train a model can have a significant impact on its accuracy. Noisy or incomplete data can cause the model to make incorrect predictions. When a model is too complex and fits the training data too closely but fails to generalize to new data. This can lead to poor prediction accuracy. The target variable is unbalanced which means one class has significantly more examples than the other, the model may be biased towards the more frequent class and not accurately predict the rare class.

It is obvious that functionary", "FI3O.credit.score", "re.balanced..paid.back..a.recently.overdrawn.current.acount", "credit.refused.in.past." and “gender” are the 5 interesting attributes because they have the highest correlation with the target attribute credit.ranking.

The correlation measures the strength and direction of the linear relationship between two variables. A high correlation between an attribute and the target attribute indicates that this attribute may be useful in predicting the target attribute.

The difficulty of pre-processing is quite easy because the data is very clear and clean without any missing values and outliers. This saves the trouble of finding and replacing the values. However, the data is not been normalised, there is a need to classified the data into two groups: one for building and training a prediction model and the other one is used for making predictions based on the trained prediction model.

--------------------------------------- SOM TRAINING ------------------------------------

Prepare a trained data set and do further analysis using visualization to confirm and conclude the findings. Visualization also helps for a more in-depth analysis and to gain insight on the relationships between the data.

# Select the columns 1,2,3,4,6 to train the SOM because they have the highest correlations

Self-Organizing Maps (SOM) is a type of unsupervised machine learning algorithm that is used for dimensionality reduction and visualization of complex data structures. SOMs can be used for various applications such as clustering, feature extraction, data visualization, and pattern recognition.

The training process of SOM involves organizing the input data into a two-dimensional grid of nodes, where each node represents a cluster of similar data points. During the training process, the weights of each node are adjusted such that similar input data points are grouped together. The training process continues until the weights of the nodes converge to stable values, and the nodes form a low-dimensional representation of the input data.

Once the SOM is trained, it can be used to classify new data points into one of the nodes in the grid, based on their similarity to the nodes' weights. This makes SOMs a useful tool for clustering and classification tasks.



# Create a new data frame which contains the first 45 columns of the classifiedData data frame.





# Converts the data\_train data frame into a matrix, and the scale() function is used to scale the values of each attribute in the matrix to have a mean of 0 and a standard deviation of 1. This helps in standardizing the values of the attributes, which can improve the performance of the SOM model.

# Assigns the names of the attributes in the data\_train data frame to the matrix data\_train\_matrix. This helps in keeping track of the names of the attributes for later use.



# Loading the kohonen library, and creating a hexagonal grid with 20 x 20 dimensions for a Self-Organizing Map (SOM). The SOM algorithm will use this grid to map the input data and help visualize the relationships between the input variables.

Text

Description automatically generated with low confidence

Text

Description automatically generated

# it trains the SOM using the "som()" function with some parameters:

* grid: The grid created using "somgrid" function.
* rlen: The number of training iterations to perform.
* alpha: The learning rate, which is set to decrease from 0.9 to 0.01 during the training process.
* n.hood: The type of neighborhood function to use, set to "circular".
* keep.data: Whether to store the data used in training for future use, set to TRUE.
* mode: The training mode, set to "online".
* normalizeDataLayers: Whether to normalize the data before training, set to false.

Text

Description automatically generated

Visualisation

# The source creates a color palette that ranges from cool blue to hot red, with n different shades and an optional argument alpha to control the opacity of the colors. The rainbow function is used to generate the color palette, and the end argument is set to 4/6 to restrict the range of colors to blue to red. The order of the palette is then reversed using [n:1] so that blue is at the bottom and red is at the top. The function returns the created color palette.



Text

Description automatically generated with medium confidence

# Plot the heatmap for the five interesting attributes / normalised values. Examples of plotting Functionary and "re.balanced..paid.back..a.recently.overdrawn.current.acount". The code is similar for the rest three attributes.

Text

Description automatically generated

Text

Description automatically generated

Chart

Description automatically generatedChart

Description automatically generatedChart, scatter chart

Description automatically generatedChart

Description automatically generated with medium confidence

Chart

Description automatically generated

The plots of these three attributes have two distinct clusters, with one cluster being much larger than the other, it suggests that there is a strong separation of the values of the attributes into two groups. The larger cluster indicates that the majority of the values of the attribute fall into this group.

This information could potentially be useful in making predictions, as the two clusters might represent two distinct populations with different characteristics. Further analysis is needed to understand the meaning of these clusters and how they relate to the target variable being predicted.

The plot of the functionary and gender attribute are very messy with many colors, it could mean that the attribute has a high variance and a large spread of values. This can indicate that the attributes has a high degree of variability, which could make it difficult to identify patterns or relationships and to build a precise prediction model.

Logo

Description automatically generated with low confidence

#plot a variable from the original data set (will be uncapped etc.) This function produces a menu for multiple heatmaps.

# Use other visualization tools to supplement the analysis to look for additional information

A picture containing graphical user interface

Description automatically generated

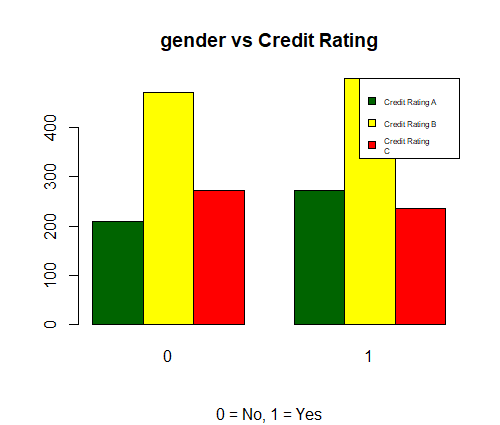
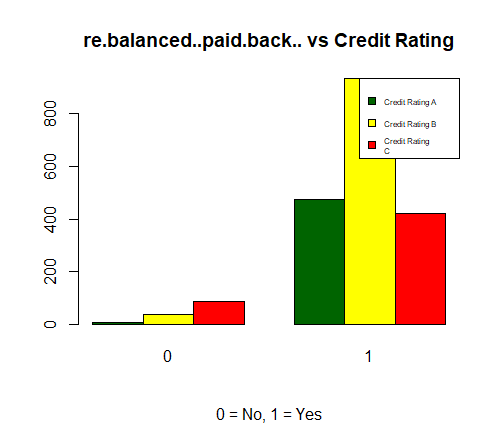
Text

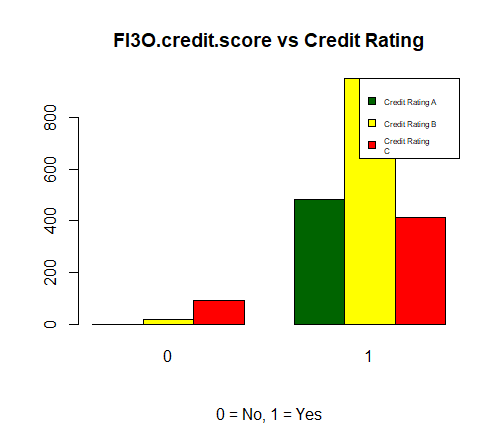
Description automatically generated

Chart, bar chart

Description automatically generated

190 individuals have credit rating of 1 and are not functionaries. 293 individuals have credit rating of 1 and are functionaries. Similarly, 785 individuals have credit rating of 2 and are not functionaries and 185 individuals have credit rating of 2 and are functionaries.

Chart

Description automatically generated

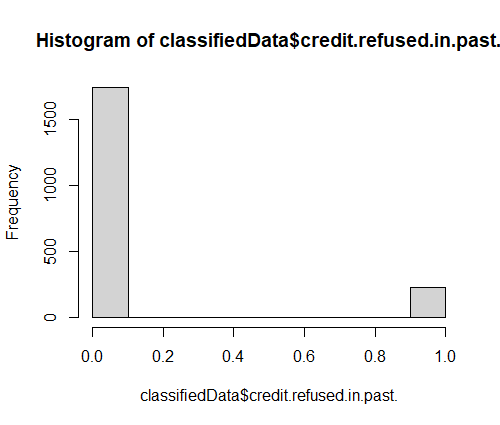
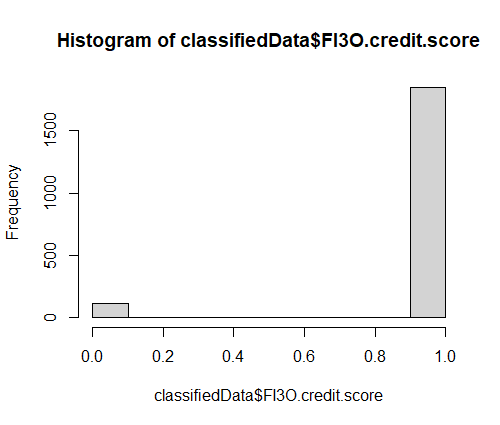
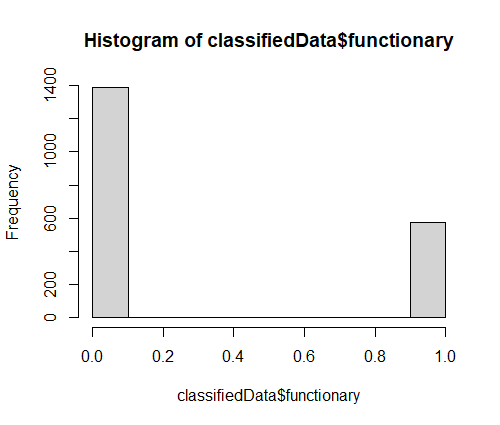
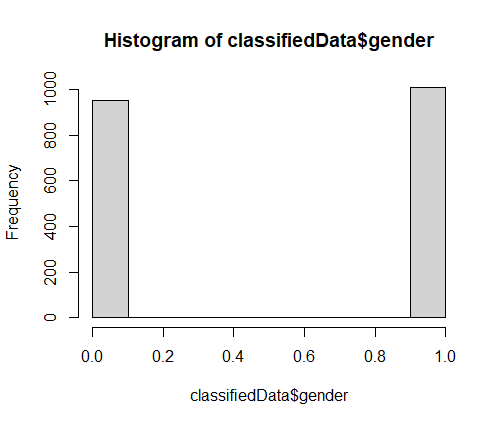
# We can have the same bar plot for the other attributes corresponding to credit.rating.

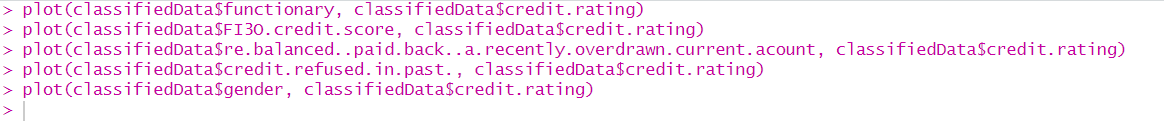
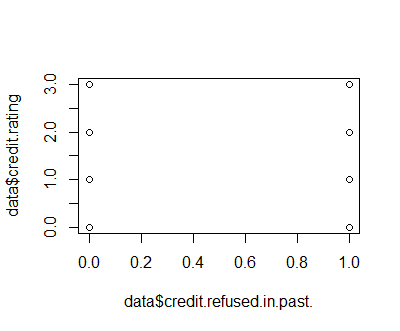
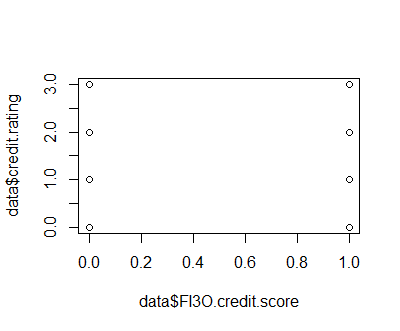
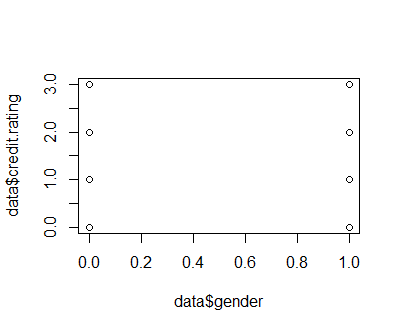
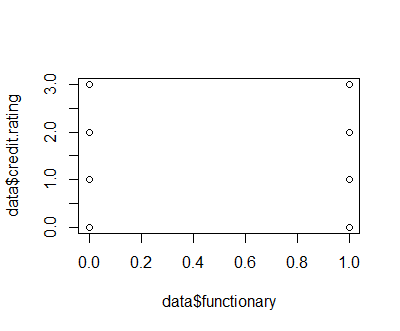
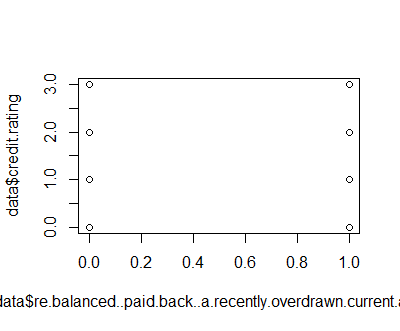
Text

Description automatically generated with medium confidence

# # Plot histogram for the five attributes

The height of each bar in the histogram represents the number of data points that fall within that bin's range. This histogram can help to visualize the distribution of the values in the data attributes, and identify any patterns, skewness, or outliers present in the data.





All the scatter plots are vertically parallel with the same value. This indicates a weak or no relationship between the two variables between the attribute and credit.rating. The dots are vertically parallel and spaced evenly, which means that there is no clear pattern between the two variables.

# Plot scatter for the five attributes

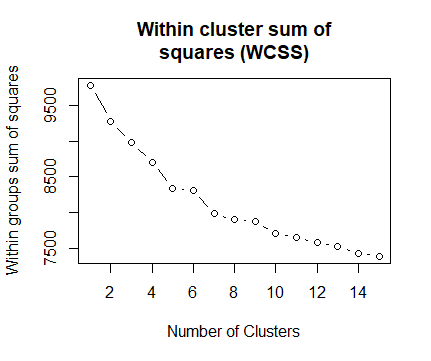
Question 2

--------------------------------- Clustering SOM results --------------------------------

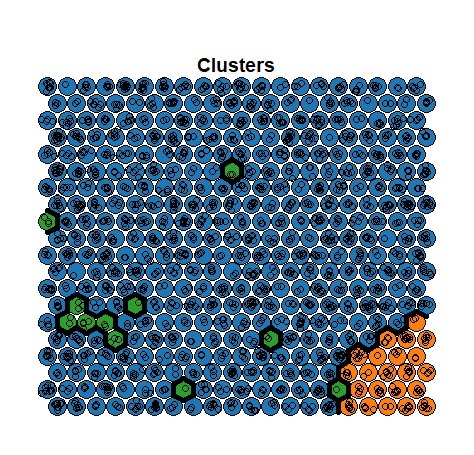
# show the WCSS metric for kmeans for different clustering sizes. Can be used as a "rough" indicator of the ideal number of clusters

Text

Description automatically generated



This graph displays the relationship between the number of clusters and the within-group sum of squares (WCSS) in the data. As the number of clusters increases, the WCSS decreases, indicating that the data is becoming more compact and homogeneous within each cluster. This graph can be used to help determine the optimal number of clusters in the data, as the "elbow" point in the graph represents the point at which the WCSS decreases at a slower rate, indicating that adding more clusters is unlikely to result in much additional improvement in terms of compactness and homogeneity within the clusters.

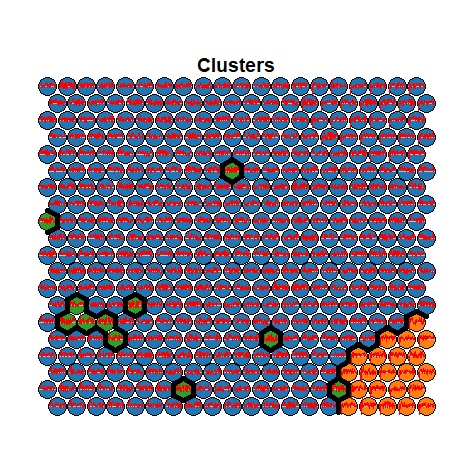
Text

Description automatically generated

The code performs hierarchical clustering on the codebook vectors of the Self-Organizing Map (SOM) model. The cutree function is used to cut the dendrogram generated by hclust into 3 clusters.

# Form clusters on grid use hierarchical clustering to cluster the codebook vectors

The plot of the SOM model shows two distinct clusters in different colors, with one cluster being much larger than the other. The larger cluster is shown in blue, while the smaller cluster is shown in orange. The add.cluster.boundaries function is used to highlight the boundaries between the two clusters. This indicates that there is a significant difference between the two groups of observations and that they can be separated into two separate clusters.



The plot is a representation of the SOM (Self-Organizing Map) model which is a type of unsupervised machine learning algorithm. The plot is showing the mapping of the data into two clusters represented by the blue and orange color. The blue cluster is much bigger than the orange cluster which means that the majority of the data belongs to the blue cluster. The boundaries of each cluster have been added using the add.cluster.boundaries function. The visualization of the codes, rather than just colors, shows the actual data points on the map and their association with the two clusters.

Text

Description automatically generated

This code is selecting 6 columns (indexes 1, 2, 3, 4, 6, and 9) from the classifiedData and unclassifiedData data frames to be used for modeling. The target data, stored in the 46th column, is then decoded into a binary matrix using the decodeClassLabels function. The data is then split into a training set and a test set using the splitForTrainingAndTest function with a ratio of 0.2 (20%) for testing data. The training and test set are then normalized using the normTrainingAndTestSet function.

# To train the MLP model to classified based on the following interested columns.

Text

Description automatically generated

The code above uses a pre-trained model to make predictions on both the training test set and the unknown set. The predict function takes the inputs and uses the model to predict the target values. The confusion matrix is then used to evaluate the accuracy of the predictions made by the model on the training test set and the training set. The code then outputs the first few entries of the actual targets, the decoded class labels and the predictions made on the unknown set.

# Predict the test. The predict() function in R is used to predict the values based on the input data.

Graphical user interface, text, application, email

Description automatically generated

The confusion matrices show the distribution of actual target values compared to the predicted values. The first confusion matrix shows the comparison between the actual targets of the training set and the predicted values from the model. The second confusion matrix shows the comparison between the actual targets of the testing set and the predicted values from the model. In both matrices, the rows represent the actual targets and the columns represent the predictions. The values in the cells of the matrix represent the number of observations that have the corresponding actual and predicted values. The values in the diagonal show the number of correct predictions.

Text

Description automatically generated

The accuracyTrain is calculated as the ratio of the number of correct predictions (sum of the diagonal of the confusion matrix) to the total number of predictions made (sum of all the elements of the confusion matrix).

The precisionTrain is calculated as the ratio of true positive predictions (elements of the diagonal of the confusion matrix) to the total number of positive predictions made (sum of elements in each row of the confusion matrix).

The recallTrain is calculated as the ratio of true positive predictions (elements of the diagonal of the confusion matrix) to the total number of actual positive cases (sum of elements in each column of the confusion matrix).

The F1 scoreTrain is the harmonic mean of precision and recall. It is a measure of the balance between precision and recall.

Text

Description automatically generated with medium confidence

The accuracy is 0.6132316 = 61.3%

This result can be considered as mediocre, but it depends on the context and the problem that the model is trying to solve. The nature of the data is very complex, and all the correlations to the credit.ranking attribute are very low. In this case, even collecting more data just barely increased the accuracy.

In conclusion, an accuracy of 0.61 could indicate that the model is not performing well, but in this case, the accuracy is considered good,

The code calculates the confusion matrix for the test set using the confusionMatrix() function and stores it in the confusionMatrixTest variable. The accuracy is then calculated by dividing the sum of the diagonal elements of the confusion matrix by the sum of all elements of the confusion matrix. The precision is calculated by dividing the diagonal elements of the confusion matrix by the row sums of the confusion matrix. The recall is calculated by dividing the diagonal elements of the confusion matrix by the column sums of the confusion matrix. Finally, the F1 score is calculated as the harmonic mean of precision and recall. The values of accuracy, precision, recall, and F1 score are printed for the test set.



# Increase the mlp size

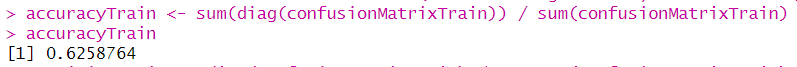
# To train the MLP model to classified based on the new interested columns.

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated





After selection more attributes and increase the training model with bigger size and matrix. The accuracy for accuracyTrain and accuracyTest has a slight increasement.

Text

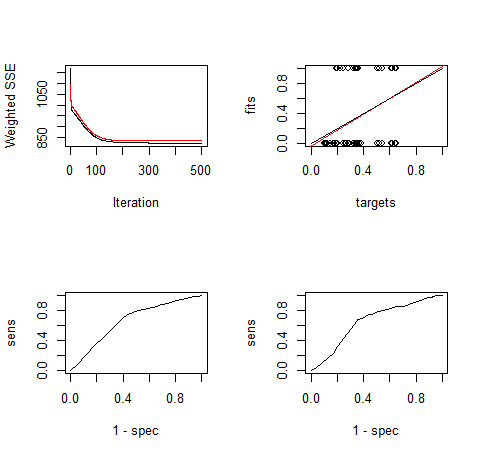
Description automatically generated

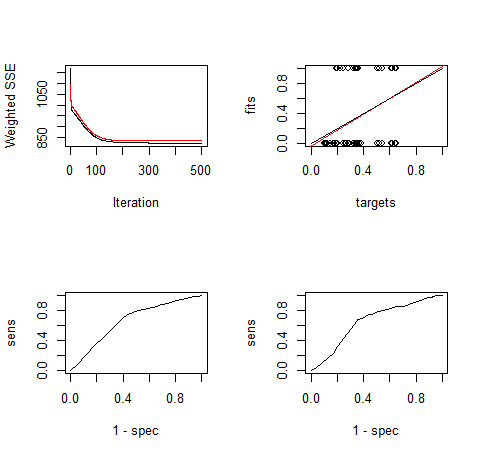
The head(trainTargets) output shows the first six rows of the trainTargets data, which contains the true class labels in binary form. Each row represents one sample and each column represents one class. The value is either 0 or 1 indicating whether a sample belongs to a certain class.

The head(classifiedData[, 46]) output shows the first six rows of the 46th column in the classifiedData data, which are the true class labels in numerical form. Each row represents one sample and each value represents the class label for that sample.

The head(predictUnknownSet) output shows the first six rows of the predicted class labels for the unknown data set (unknownValues) in numerical form. Each row represents one sample and each column represents the predicted probability of that sample belonging to a certain class. The values in each row sum up to 1 and indicate the confidence of the prediction.

Text

Description automatically generated



The plotIterativeError(model) shows the error rate of the model at each iteration during the training process. The error rate is a measure of the difference between the predicted values and the actual target values. The error rate decreases as the model is trained over more iterations, indicating that the model is learning from the training data and improving its predictions.

The plotRegressionError(predictTestSet[, 2], trainSet$targetsTest[, 2]) shows the regression error between the predicted values and the actual target values in the test set. The regression error is a measure of the difference between the two sets of values, and it is calculated as the sum of the squared differences between the predicted values and the actual target values. This plot shows that as the predicted values increase, the regression error also increases, indicating that the model is making less accurate predictions as the predicted values deviate further from the actual target values.

The plotROC(fitted.values(model)[, 2], trainSet$targetsTrain[, 2]) and the plotROC(predictTestSet[, 2], trainSet$targetsTest[, 2]) show the receiver operating characteristic (ROC) curve for the model. The ROC curve is a graphical representation of the trade-off between the true positive rate (TPR) and the false positive rate (FPR). The TPR is the proportion of actual positive cases that are correctly predicted as positive, while the FPR is the proportion of actual negative cases that are incorrectly predicted as positive. As the threshold for classifying cases as positive or negative is varied, the TPR and FPR change, resulting in a curve. The ROC curve for the model should be as close to the top-left corner as possible, indicating that the model is making few false positive predictions and a high proportion of actual positive cases are correctly predicted as positive. The plot shows that as the predicted values increase, the TPR also increases, but so does the FPR, indicating that the model is making more false positive predictions as it becomes more confident in its predictions.