CREDIT CARD FRAUD DETECTION USING SUPPORT VECTOR MACHINE LEARNING MODEL

Jude Sedem Kojo Agboka

Student No: **B1680944**

University: Teesside University

Middlesbrough, UK

Email: Judeagboka@ymail.com

ABSTRACT— Today, using credit card and the fraud that goes along with it are both very common. Credit card fraud describes the real loss of a credit card, or the unauthorized disclosure of personal information associated with a credit card that results in financial loss. Retailers and credit card companies are unable to distinguish these fraudulent transactions from the millions of real ones. In earlier research, experiments to identify credit card fraud used Naive Bayes, Multilayer Perceptron, Random Forest, and Logistic Regression. The SVM model for classifying transactions as fraudulent or authentic is presented in this work. The study made use of a credit card fraud detection dataset from Kaggle. The SMOTE method was used to oversample since the dataset was significantly out of balance.

The dataset was divided into test and training halves, and a feature selection process was also performed. The results show how effective the SVM model is at identifying credit card fraud. It was shown that supervised machine learning algorithms are useful for detecting additional anomalies since they can control the skewness and produce the best classification outcomes.

Keywords—Credit Card fraud, SMOTE, Training & Testing, SVM – Support Vector Machine

I. OUTLINE

A. *Objectives*

- To find a good dataset for credit card fraud detection
- Prepare and pre-process an appropriate dataset for a Machine Learning Model.
- Develop a Machine learning Model (Support Vector Machine Model) to predict an outcome
- Testing and validating the model

B. *Aim*

- This paper's primary aim is to use supervised learning on real datasets to detect credit card fraud using (Support Vector Machine Model) to predict an outcome (Fraud or no fraud)

II. INTRODUCTION

A major issue that is destroying the economy, corporations, and governments is credit card fraud. Fraud is the term used to describe someone else using your account without your permission. Measures have been taken to stop these scam artists' behaviors.

Credit card fraud comes in two different forms. One includes stealing the card itself, while the other entails losing personal data like the card number, CVV code, card type, and other specifics. Before the cardholder becomes aware of it, fraudsters have already profited greatly. Various machine-learning techniques are used by businesses to identify this [1].

As a result, when fraud happens and is discovered, banks must invest resources in postmortem investigations to identify and prosecute offenders. This detection's processing time takes many days, and it was shown to be ineffective at discouraging fraud. The automated machine learning-based algorithm used by credit card firms to identify fraudulent transactions before they enter the database is called a fraud detection system (FDS). A perfect FDS should also reduce the number of false positives that obstruct legitimate transactions and hamper end users. These algorithms fall into a category termed supervised learning algorithms, which calls for the pre-labeling of sample data classes. To implement its algorithm, FDS gathers a large amount of historical data. Transaction records, however, frequently show an imbalance, with legitimate transactions vastly outweighing fraudulent ones. This article summarizes and assesses the Support Vector machine learning algorithm's performance in identifying fraudulent transactions. Employing SMOTE to balance out the data in unbalanced real-world datasets. The Fraud Detection System (FDS) is made to distinguish between fraudulent behavior and that of legitimate users to identify all types of fraud. Any type of credit card fraud can be found by a good FDS. It works on the premise that rather than focusing on fraud vectors, it can learn specific card users' usage patterns as well as fraudsters' spending patterns. Making an FDS turns into a binary classification issue. Here, regular transactions and fraudulent transactions are the two types of interest. These algorithms have some difficulties that prevent accurate classification. Class distortions, changes in fraudulent behavior to evade detection, seasonal changes in user activity, domain metrics, and missing truth tags are some of these difficulties. This paper's primary goal is to use supervised learning on real datasets. In our study, we employ SVM. When data is linearly categorized and processed in supervised learning, SVM is utilized [1], [2].

III. RELATED WORKS

Because fraud causes significant losses, researchers were driven to identify workable fraud prevention strategies. A few approaches have previously been developed and examined. Below is a brief description of a few of these. A prior research paper used GB, LR, RD, SVM, etc. Only after under-sampling the data and balancing the dataset were good precision and recall values attained. On LR, DT, and RF, comparisons were made between the underlying models. With 95.5% accuracy, RF outperformed the other two models, DT (94.3 D44) and LR (90% accuracy). Techniques for detecting outliers and K-nearest neighbors (KNN) are also useful in detecting fraud. It is proven to improve rates and fraud detection rates while reducing false alerts. Further, papers suggest the KNN method performed well. It is examined and evaluated against other conventional algorithms. In contrast to the work so far detailed, some classical algorithms and deep learning technology complete all evaluated procedures with about 80% accuracy. Deep neural network transactions for fraud detection are the subject of numerous papers. These models, nevertheless, need a lot of processing and work best with bigger datasets. Great outcomes may result from this strategy. The majority of the publications cited employed the under-sampling technique, which served as the inspiration for the employment of the oversampling technique. Consideration was given to a spectrum of traditional supervised learning algorithms. These include neural networks, tree-based algorithms, Bayesian methods, traditional Deep learning techniques, and hybrid algorithms. Using Random Forest (RF), a technique a classifier that incorporates many predictions. For fraud detection, neural networks (NN) and artificial neural networks are the most popular technique. NN determines values by identifying pertinent patterns based on learned patterns' associative memory. Artificial neurons coupled passively makeup ANNs. ANN is a backpropagation algorithm for training a feedforward neural network (NN). Data that has been linearly examined is categorized by Support Vector Machine (SVM). According to the Bayesian theory, which bases decisions on conditional probabilities. The best parameter for predicting binary probabilities is logistic regression (LR). Based on one or more attributes, an answer is given. The instance-based supervised learning classifier K nearest neighbors (KNN) uses a similarity-based classification technique [1], [2]

IV. METHODOLOGY

First, we got the data from Kaggle, a service that offers datasets for data analytics. 284,807 transactions were recorded in a public database in September 2013 for European Credit card users. This dataset comprises two features (deduction and amount) and 31 columns (attributes), 28 of which are marked with the labels "v1-

v28" to protect delicate information. The columns denote time, amount, and class. Time denotes the amount of time that has passed between the first and subsequent transactions. The amount is the sum that will be exchanged. Class 1 transactions are fraudulent, while Class 0 transactions are legitimate. To analyse and comprehend errors in your dataset graphically, use a variety of graphs to organize the data into categories.

• Data cleansing - One critical step in the data cleaning process is to fill in missing values. This issue can be resolved in a variety of ways. One of these is omitting the full tuples, which will most likely result in data skewness. Since there were no records with missing values in the source file that contained the actual transactions, entering this data was no longer a problem. The file has been cleaned up of meaningless value tuples because they do not help produce useful data and hence do not corrupt the data. The next set of improvements includes the removal of the time-related column and other superfluous columns.
• Data integration - Finding and resolving discrepancies between data values using European metrics like Euros instead of Dollars for the amount a client spent.
• Data transformation: In this step, all categorical data is compiled into a comprehensible numerical format. Transaction records include a variety of data types and ranges.
• Dimensionality reduction: It is the method utilized for data reduction. The likelihood of discovering incorrect data patterns should be reduced, and the chosen features should eliminate pointless elements. A typical transformation technique is Principal Component Analysis known as PCA. By using this approach, the feature selection issue is resolved from the perspective of numerical analysis. By identifying the right number of principle components, PCA completed the feature selection process [1], [3], [4].

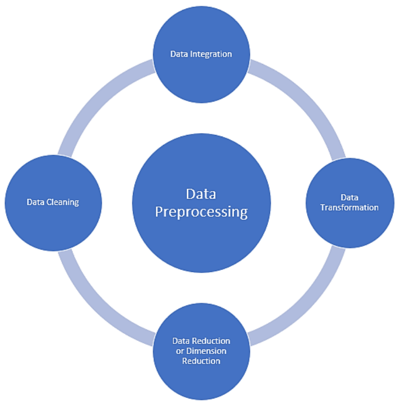

Figure 1. Shows the data pre-processing stages

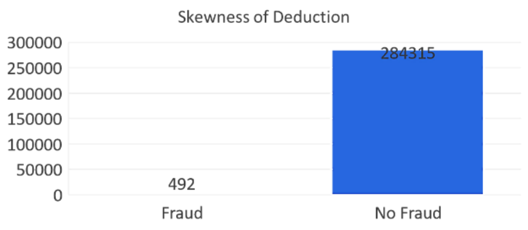

Figure 2. Demonstrates that there are significantly fewer illegal transactions than valid ones, which leads to skewness in the deductions.

Synthetic Minority Oversampling Technique was used to address this imbalance. The technique of oversampling is applied.

1. The algorithm chooses two or more instances of data that are similar (in this case, "deduction," which is 0 and 1 for no fraudulent and fraudulent).

2. After that, it randomly perturbs each instance by one feature at a time. It then chooses a value that is close to the nearby examples in terms of distance.

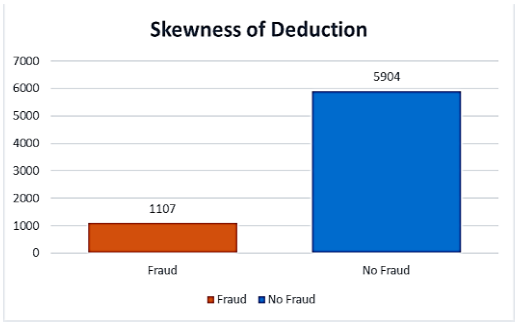

Figure 3. Shows a balanced deduction feature after using SMOTE

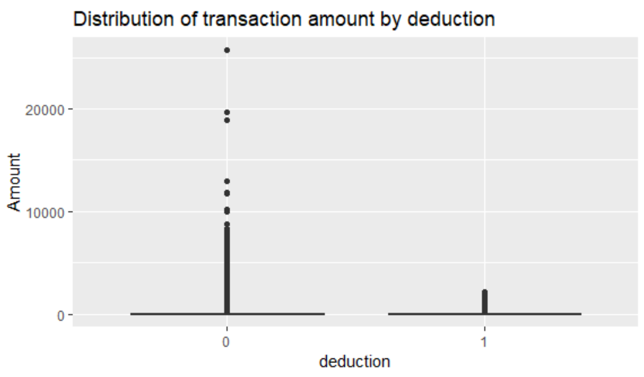

Figure 4. The money that was transacted and its deduction are shown in this graph. For non-fraudulent transactions, the transaction values are far more variable.

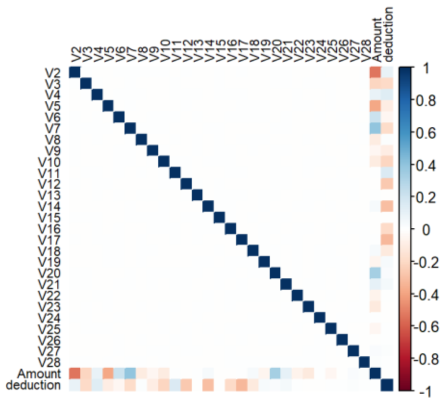

Figure 5. Shows the correlation and showcases if any variable correlate with each other. Most are not correlated. All the anonymous variables are independent of each other.

There are a couple of interesting correlations with the "Amount" and "deduction" feature.

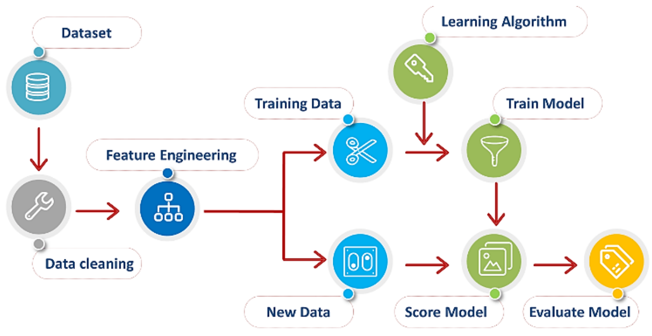

Figure 6. Shows the steps taken in the model evaluation

V. EXPERIMENT

The principles of statistics, optimization, and machine learning are the foundation of the Support Vector Machine (SVM).

It was first presented by Boser, Guyon, and Vapnik (1992) at the 5th Annual Computing Conference, and it has proven to be extremely helpful in many common categorization tasks.

SVMs employ empirical error and hypothesis space (a measure of how well the model fits the training data).

It employs a sparse approach and relies on the Support Vectors training subset for prediction.

It restricts the optimization problem that generates the hyperplane, requiring that it be positioned with a maximum margin or separation from the various sorts of classes [5]

25% of the data were used for the test set and 75% were used for training. Although being accurate is crucial to compare the categorization model's performance, other variables such as the F1 score, precision, TPR, FPR, recall, G-mean, and specificity are also examined. These rating scales are all valid, as shown [1]

True Positives (TP) are indications that a transaction is not fraudulent, such as when the expected class and actual class value agree.

True Negatives (TN) - For instance, if the transaction is fraudulent according to both the anticipated class and the actual class.

False Positives (FP) - For instance, if the actual class reveals that the transaction is not fraudulent, but the projected class says that it is.

False Negatives (FN) include situations when the transaction class value indicates that there was no fraud, but the forecasting class suggests that there was.

Precision is measured by the quantity of relevant positive categorized signals.

By way of the fraction of successfully predicted observations to all projected positive observations is known as accuracy for positive observations.

Precision equals TP / (TP + FP)

For this model, the precision value was low.

Specificity = TN/ (TN + FP)

That is how proficient the model is in preventing erroneous alarms.

- Sensitivity: Calculate the actual proportion of accurately detected positives.

Sensitivity is TP/ (TP + FN).

- Balanced Accuracy: the overall average rate of detection.

(Sensitivity + Specificity) / 2 equals balanced accuracy.

The easiest to understand is the accuracy performance statistic, which is simply the proportion of properly predicted observations to all observations. Our model must be the greatest if it is accurate, right? If the dataset's false positive and false negative rate numbers are roughly equivalent, accuracy is an excellent indicator. As a result, you must consider more factors while assessing the success of your model. Having a score of 0.959, our model is roughly 96% accurate.

Prevalence is calculated as (TP + FN)/ (TP + FP + FN + TN).

• Precision and Recall are weighted averaged to determine the F1 Score. As a result, while determining how to score, false positives and false negatives are taken into account. F1 is often more advantageous than accuracy, especially if you have an uneven class distribution, although it is not as intuitively simple to understand as accuracy. When false positive and false negative costs are about similar, accuracy performs best. It is preferable to include both Precision and Recall if the costs of false positives and false negatives differ significantly. In our case, it was, and as a result, the F1 score was 0.058, below the required level.

F1 Score is calculated as 2*(Recall * Precision) / (Recall + Precision).
The confusion matrix can be used to calculate all of the metrics given.
These measures were used to assess the performance of the model. Testing the model on both the original and the oversampled data revealed that sampling is crucial.

The SVM model achieved ;

- Precision: 0.030
- Recall: 0.877
- Sensitivity: 0.877
- Specificity: 0.959
- Accuracy: 0.959
- Auc: 0.956
- F1-score: 0.058

A. *Area Under Curve and ROC Curve (AUC)*

How effectively a classifier system works as the discrimination function is depicted in a two-dimensional diagram called a ROC curve.
The cut-off value fluctuates over the whole range of the predictor variable. The independent variable for the x-axis is the false positive rate of the predictive test. The dependent variable, or y-axis, for the predictive test is the genuine positive rate. Each point in the ROC space for a discrimination cut-off value of the prediction test represents a true positive/false positive data pair. If the probability distributions for the true positive and false positive are both known, the cumulative distribution function can be used to create a ROC curve. In practical applications, a data sample typically yields a single point for each option in the ROC space.
For each discriminating option cut-off, supply a single point in the ROC space. Point (0, 1), which would reflect 100% true positives and 0% false positives, would be the optimal result.
A gold standard technique for identifying genuine positive and genuine negative situations is required in order to create ratings for both true positives and false positives. When the truth is known, the effectiveness of a classifier or classification system is described by a confusion matrix, sometimes referred to as an error matrix.
Each row (or column) in a confusion matrix reports the numbers in a true class, such as the number of true fraudulent or true normal, whereas each column (or row) gives the numbers in a predicted class, such as the number of predicted fraudulent or predicted normal. A standard 22 contingency table reports four numbers:

		Predicted Class	
		Class = 1	Class = 0
Actual class	Class = 1	True Positive (Sensitivity)	False Negative
	Class = 0	False Positive	True Negative (Specificity)

Class = Deduction in our dataset
1 = Fraudulent
0 = Not fraudulent

1) Reliable positive or True Positives (TP; also known as sensitivity; a measure of the proportion of positives correctly predicted given that they are truly positive). The recall was above 0.5, which is excellent. The recall or sensitivity value is 88%, or 0.877 , 2) False Negative (FN; a measure of the percentage of predicted negatives given that they are positive), 3) False Positive (FP; a measure of the percentage of predicted positives given that they are negative), and 4) True Negative (TN; also known as specificity; a measure of the proportion of predicted negative, given that they are truly

negative). It is obvious that a classifier that is more accurate needs to have higher sensitivity and specificity.

The specificity is 1 – FP.

By setting the variable to be lower than the lowest value observed, the (0, 0) point in the ROC space is produced. When we increase the discriminating cut-off value to include more and more data points, we create a series of points in the ROC space that can be connected by a curve. A discriminating cut-off value bigger than the highest value observed results in the (1, 1) point. The diagonal line joining the coordinates (0, 0) and (1, 1) shows that test predictions are no more accurate than random guesses. As a point's position in the ROC space moves further from the diagonal line, the test's predictive power grows.

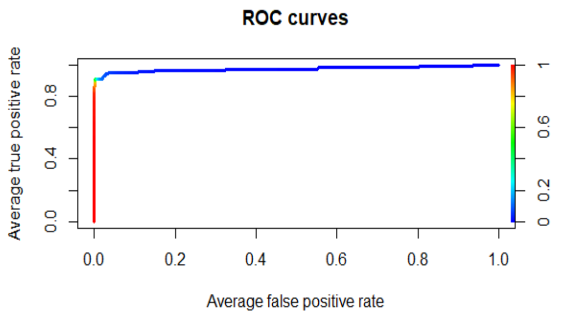

Figure 7. ROC Curve

AUC=0.5	No discrimination; for instance, toss a coin at random
0.6AUC>0.5	Ineffective discrimination
0.7≥AUC>0.6	Reasonable discrimination
0.8≥AUC>0.7	Good discrimination
AUC>0.9	Outstanding discrimination

Analysis of the obtained results shows that the accuracy is very high, but this does not mean that the results are perfect. Accuracy should be viewed "with care" - preferably interpreted in conjunction with other metrics[6]

VI. CONCLUSION

Even if we didn't achieve the system's ideal level of accuracy in fraud detection, we did succeed in building it.

We can get a lot closer to that objective with adequate time and information. There is still space for improvement, as there is with every undertaking of this nature.

It is possible to further enhance this model by adding new algorithms. The output of these algorithms must, however, follow the same structure as that of the other algorithms. When this prerequisite is satisfied, adding modules is as simple as adding code. This significantly increases the project's modularity and adaptability[3].

VII. ACKNOWLEDGMENT

I want to thank my colleagues and lecturer Al-Kafri for the immense effort to teach me the fundamentals of Machine Learning. I cannot thank them enough.

VIII. References

[1] Univerzitet u Istočnom Sarajevu. Faculty of Electrical Engineering, IEEE Industry Applications Society, Institute of Electrical and Electronics Engineers. Bosnia and Herzegovina Section, Institute of Electrical and Electronics Engineers. Serbia and Montenegro Section, and Institute of Electrical and Electronics Engineers, *2019 18th International Symposium INFOTEH-JAHORINA (INFOTECH) : proceedings : March 20-21, 2019, Jahorina, East Sarajevo, Republic of Srpska, Bosnia, and Herzegovina*.

[2] Amity University and Institute of Electrical and Electronics Engineers, *Proceedings of the 9th International Conference On Cloud Computing, Data Science and Engineering : Confluence 2019 : 10-11 January 2019, Uttar Pradesh, India.*

[3] S. P. Maniraj, A. Saini, S. Deep Sarkar, and S. Ahmed, "Credit Card Fraud Detection using Machine Learning and Data Science." [Online]. Available: www.ijert.org

[4] S. Misra, V. O. Matthews, A. Adewumi, O. S. Covenant University (Ota, IEEE Nigeria Section, and Institute of Electrical and Electronics Engineers, *Proceedings of the IEEE International Conference on Computing, Networking and Informatics (ICCNI 2017) : 29-31 October 2017, Covenant University, Canaanland, Ota, Ogun State, Nigeria*.

[5] M. Awad and R. Khanna, "Support Vector Machines for Classification," in *Efficient Learning Machines*, Apress, 2015, pp. 39–66. DOI: 10.1007/978-1-4302-5990-9_3.

[6] A.Used and D.T. Goal,"Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures".

IX. APPENDIX

```
library(DataExplorer)
library(ggplot2)
library(corrplot)
library(caTools)
library(DMwR)
library(e1071)
library(ROCR)
library(caret)
#The data is csv and its read
creditcard <- read.csv("C:/Users/ELITEBOOK/OneDrive/Desktop/DATA FOR CO2
R/Original_CO2/kk/creditcard.csv",sep = ',', header = TRUE)

#To check the correlation between the variables
correlation <- cor(creditcard[, -1], method = "pearson")
corrplot(correlation, number.cex = 1, method = "color", type = "full", tl.cex=0.7, tl.col="black")

#to check the correlation between amount and deduction
ggplot(creditcard, aes(x = factor(deduction), y = Amount)) + geom_boxplot() +
  labs(x = 'deduction', y = 'Amount') +
  ggtitle("Distribution of transaction amount by deduction")

#Creditcard is assigned the name 'nnew' & rows are removed that do not have target variable values
set.seed(7)
nnew <- creditcard[!(is.na(creditcard$deduction)),]
```

```
#to check if there is still any missing data
plot_missing(nnew)


#Change my feature deduction to a factor
nnew$deduction <- factor(nnew$deduction)

#Summary of the dataset
summary(nnew)

#Splitting the dataset

split <- sample.split(nnew$deduction, SplitRatio = 0.75)

training <- subset(nnew, split == TRUE)
testing <- subset(nnew, split == FALSE)


## Let's check the count of unique value in the target variable
#0 signifies no fraud & 1 signifies fraud
as.data.frame(table(training$deduction))



set.seed(7)

## Synthetic Minority Oversampling Technique To Handle deduction Imbalance In this Binary Classification
balanced.data <- SMOTE(deduction ~., training, perc.over = 200, k = 100, perc.under = 800)

#table showing a balanced deduction in the binary classification
as.data.frame(table(balanced.data$deduction))

#Make the train/test data compatible to SVM model
trainX <- subset(balanced.data, select = -c(deduction))
trainY <- subset(balanced.data, select = c(deduction))
testX <- subset(testing, select = -c(deduction))


#Create a Support vector machine (SVM Model) model
SVM_model_balanced_data_smote_final <- svm(x = trainX, y = trainY, type = "C-classification", kernel = "radial",
gamma = .01, cost = 25, probability=TRUE )

#Predict the performance using test data as input to the model
predicted_data<- predict(SVM_model_balanced_data_smote_final, newdata = testX, probability=TRUE)
predicted_probability <- attr(predicted_data, "probabilities")[,'1']

#Actual Performance from test data
actual_performance <- testing[,"deduction"]


prediction_test<- prediction(predicted_probability, actual_performance)

#to remove any plot setting interfering with current plot
```

```
dev.off()

#Precision/Recall graphs
performance_measures_test <- performance(prediction_test, "prec", "rec")
plot(performance_measures_test, avg= "threshold",colorize=TRUE,lwd= 3,main= " Precision/Recall graphs")
#sensitivity/Specificity graphs
performance_measures_test <- performance(prediction_test, "sens", "spec")
plot(performance_measures_test,avg= "threshold", colorize=TRUE, lwd= 3, main="Sensitivity/Specificity graphs")
#ROC CURVE
performance_measures_test<- performance(prediction_test, "tpr", "fpr")
plot(performance_measures_test, avg= "threshold",colorize=TRUE,lwd= 3, main = "ROC curves")

ks_table_test <- attr(performance_measures_test, "y.values")[[1]] - (attr(performance_measures_test,
"x.values")[[1]])
KS_prob_cutoff <- unlist(performance_measures_test@alpha.values)[which.max(ks_table_test)]
print(c(max(ks_table_test), KS_prob_cutoff))

predicted_performance <- factor(ifelse(predicted_probability >= KS_prob_cutoff, "1", "0"))

actual_performance <- factor(actual_performance)

#Confusion matrix

confusion_matrix  <- (confusionMatrix(predicted_performance, actual_performance, positive = "1"))
confusionMatrix(table(predicted_performance, actual_performance))

# Model Precision: tp/(tp+fp):
model_precision   <- confusion_matrix$table[2,2] / sum ( confusion_matrix$table[2,1:2])
# Model Recall: tp/(tp + fn):
model_recall      <- confusion_matrix$table[2,2] / sum (confusion_matrix$table[1:2,2])

# F-Score: 2 * precision * recall /(precision + recall)
model_fscore      <- 2 * (model_precision * model_recall) / (model_precision + model_recall)
model_accuracy    <- confusion_matrix$overall[1]
model_sensitivity <- confusion_matrix$byClass[1]
model_specificity <- confusion_matrix$byClass[2]

#Evaluation using AUC (Area under the curve)
auc <- performance(prediction_test, measure = "auc")
model_auc <- auc@y.values[[1]]
```