

***­­***

***Machine Learning Project***

***Credit Card Fraud Detection***

# Team 17

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Code | Sec | BN |
| Abdelrahman Hamdy Ahmed | 9202833 | 1 | 38 |
| Ziad Sherif | 9202586 | 1 | 26 |
| Zeyad Tarek | 9202588 | 1 | 27 |
| Abdelhameed Emad | 9202758 | 1 | 34 |

Date: 4/08/2024 Presented to: Eng. Mohamed Shawky

|  |  |  |  |
| --- | --- | --- | --- |
| **Workload Division** | | | |
| **Abdelrahman** | **Zeyad Tarek** | **Ziad Sherif** | **Abdelhameed** |
| * **Data Preprocessing** * Data Splitting * Scaling * Removing outliers * Data Distribution & Correlation * **Data Visualization** * **Feature Engineering** * Oversampling using SMOTE * Random Undersampling * Dimensionality Reduction * **Grid Search** | | * **Model Selection & Training** * Logistic Regression * KNN * SVM * Decision Tree * Random Forest * **Cross-validation** * **Performance Analysis** * ROC Curve * Confusion Matrix * Accuracy Measures | |

**Project Outline**

**1. Problem Description:**

Problem: Detecting credit card fraud is crucial for banks to protect customers from unauthorized charges. Using machine learning, banks analyze transaction data to flag suspicious activities based on patterns and anomalies, ensuring security against evolving fraud tactics. Regular updates and improvements in detection methods are essential to stay ahead of fraudsters.

Problem Definition and Motivation: The goal is to develop a **machine learning model** capable of accurately **detecting fraudulent credit card transactions** based on transaction data. This is motivated by the need to prevent financial losses for both customers and credit card companies, as well as to maintain trust in the financial system. Credit card fraud detection is crucial for protecting customers and minimizing financial losses. Machine learning models, including supervised algorithms like logistic regression and random forests, as well as unsupervised techniques like anomaly detection, are employed to identify fraudulent transactions. Evaluation metrics such as precision, recall, and AUC-ROC ensure the effectiveness of these models. Once trained, the selected model can be deployed to monitor real-time transactions, aiding financial institutions in promptly detecting and mitigating fraudulent activity.

**2. Evaluation Metrics:**

* **ROC (Receiver Operating Characteristic)**: ROC curves visualize the performance of a classification model across various threshold settings. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR). A higher area under the ROC curve indicates better model performance, with an AUC (Area under Curve) of 1 representing a perfect classifier.
* **Accuracy**: Accuracy is the proportion of correctly classified instances out of the total instances. It's a simple and intuitive metric, but it can be misleading in the presence of class imbalance. Accuracy doesn't account for the types of errors (false positives and false negatives) made by the classifier.
* **Recall (Sensitivity)**: Recall measures the ability of a classifier to find all relevant instances in the dataset. It is calculated as the ratio of True Positives to the sum of True Positives and False Negatives. High recall indicates that the classifier is good at identifying positive instances, but it may also classify some negative instances as positive.
* **Precision**: Precision measures the accuracy of positive predictions made by the classifier. It is calculated as the ratio of True Positives to the sum of True Positives and False Positives. High precision indicates that the classifier is conservative in labeling instances as positive, but it may miss some positive instances.

**3. Dataset and References:**

* **Dataset:** The dataset contains transactions made by credit cards in September 2013 by European cardholders. It includes 492 frauds out of 284,807 transactions, with highly unbalanced classes.
* **References:**
  + The dataset is sourced from Kaggle: Credit Card Fraud Detection
  + Additionally, a simulator for transaction data has been released as part of the practical handbook on Machine Learning for Credit Card Fraud Detection (Fraud Detection Handbook).

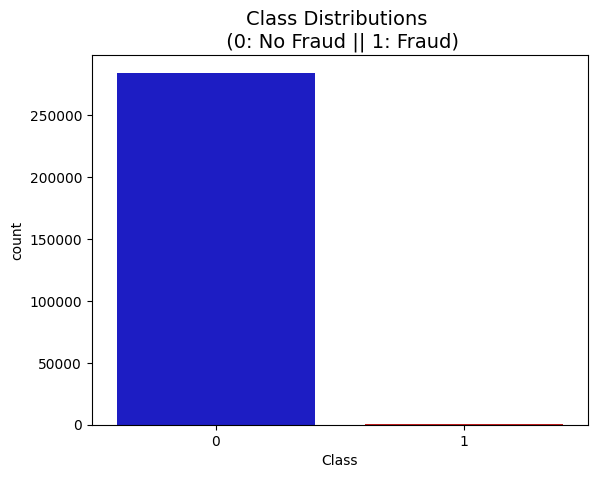
**Results & Conclusion**

First of all, it’s important to point out that the dataset was previously scaled and then went through PCA transformation which means the original features which were discrete had been represented in a continuous matter, removed the unnecessary columns and then transformed into a new set of orthogonal (uncorrelated) principle components (The V columns in the dataset basically) except for the time & amount columns.

This primarily helped in removing noise and redundant information from the dataset. Scaling ensured that all features contributed equally to the analysis. Therefore, we must also scale the time & amount columns. We’ll use a **Robust Scaler** to match the feature columns.

The dataset contains no null values and is heavily skewed (No Frauds = 99.83% & Frauds = 0.17%) but we’ll solve this issue.

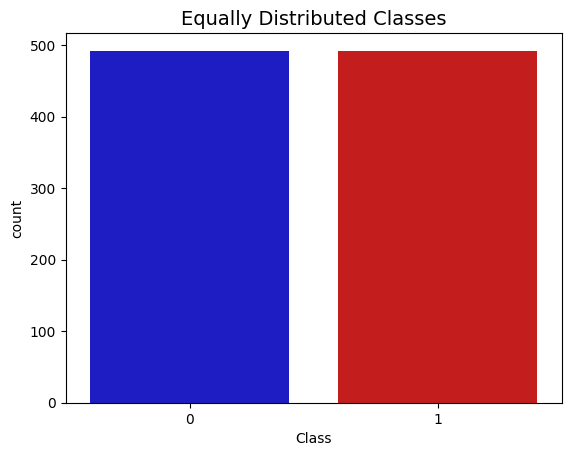
🡪 By observing the **data distribution**: Our dataset has way more normal transactions than fraudulent ones. If we use it directly for building models, they might make too many mistakes because they'll think almost every transaction is normal. But our goal is to teach the models to spot signs of fraud, not just assume everything is normal.



This led us to create a new sub-sample to achieve a 50/50 ratio of fraud and non-fraudulent transactions. This is to prevent overfitting and wrong correlations. There are **492** cases of fraud in our dataset so we can randomly get 492 cases of non-fraud to create our new sub dataframe.

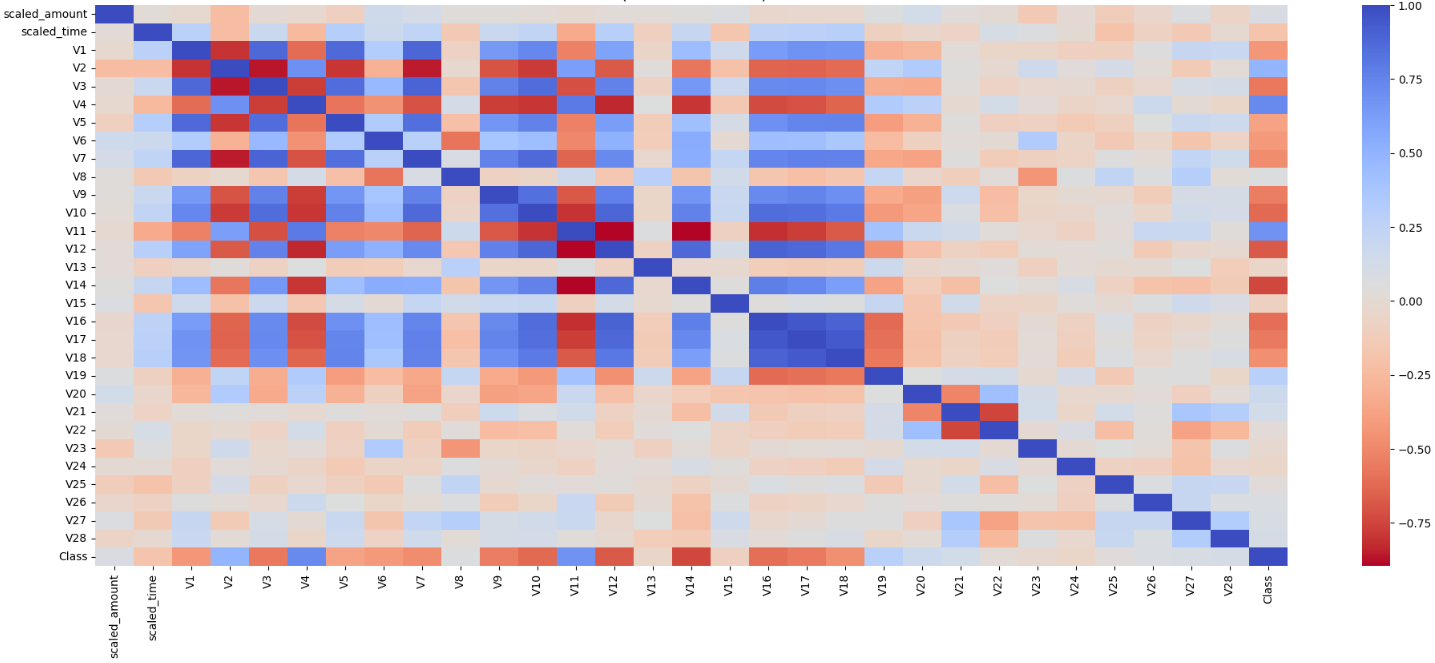
* Before proceeding with the Random Undersampling technique we have to separate the original dataframe. Why? For testing purposes, remember although we are splitting the data when implementing Random Undersampling or Oversampling techniques, we want to test our models on the original testing set not on the testing set created by either of these techniques. The main goal is to fit the model either with the dataframes that were undersample and oversample (in order for our models to detect the patterns), and test it on the original testing set.

After implementing random undersampling, we have a sub-sample of our dataframe with a 50/50 ratio with regards to our classes. We then shuffled the data to see if our models can maintain a certain accuracy every time we run this script.

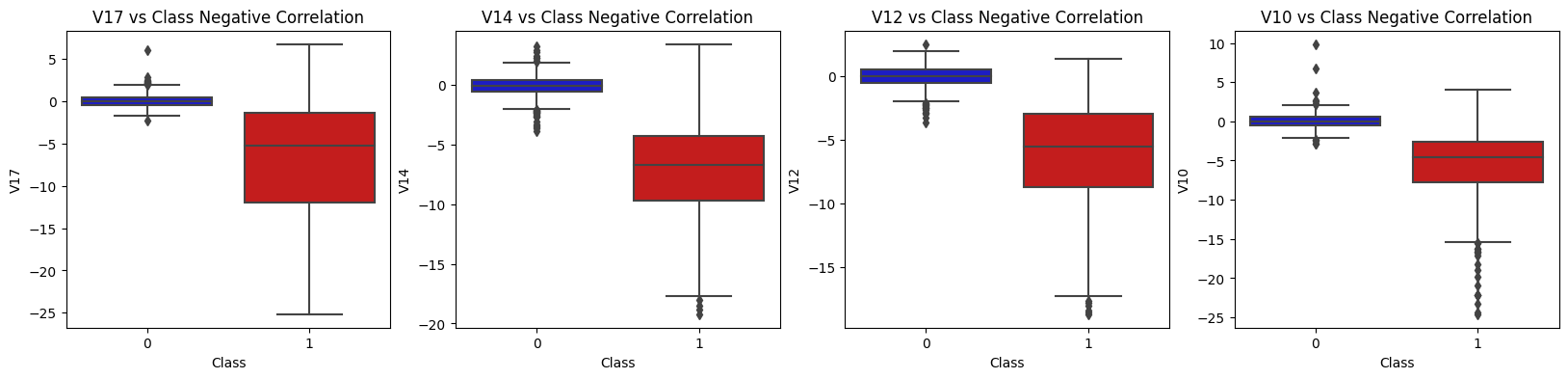


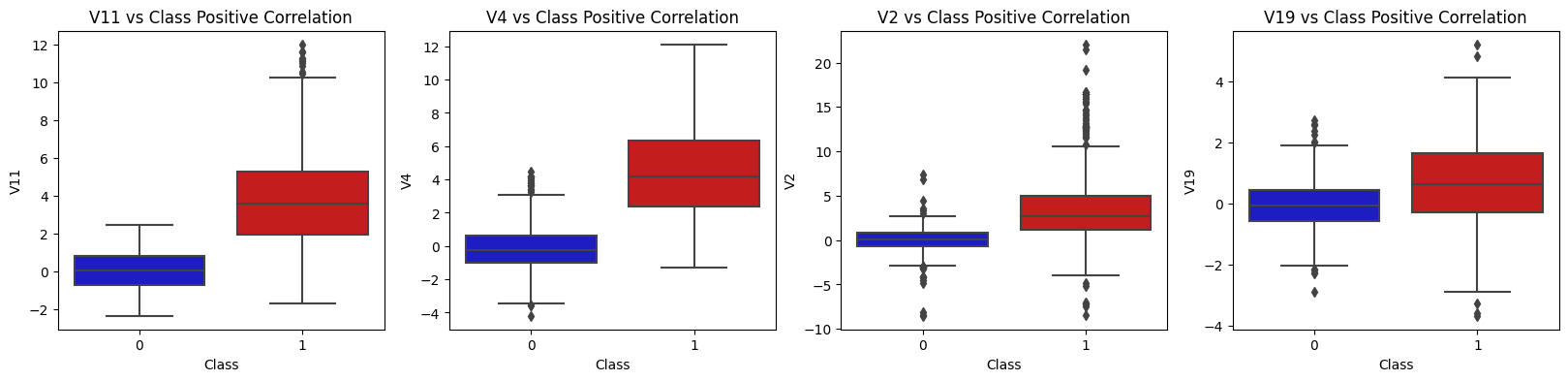
Next up, we observed the correlation between the features we have through the correlation matrix to know if there are features that influence heavily in whether a specific transaction is a fraud or not.

* Negative Correlations: V17, V14, V12 and V10 are negatively correlated.
* Positive Correlations: V2, V4, V11, and V19 are positively correlated.



We also used boxplots to have a better understanding of the distribution of these features in fraudulent and non-fraudulent transactions.





*Besides easily seeing the 25th and 75th percentiles (both end of the squares) it is also easy to see extreme outliers (points beyond the lower and higher extreme)*.

**Anomaly Detection**:

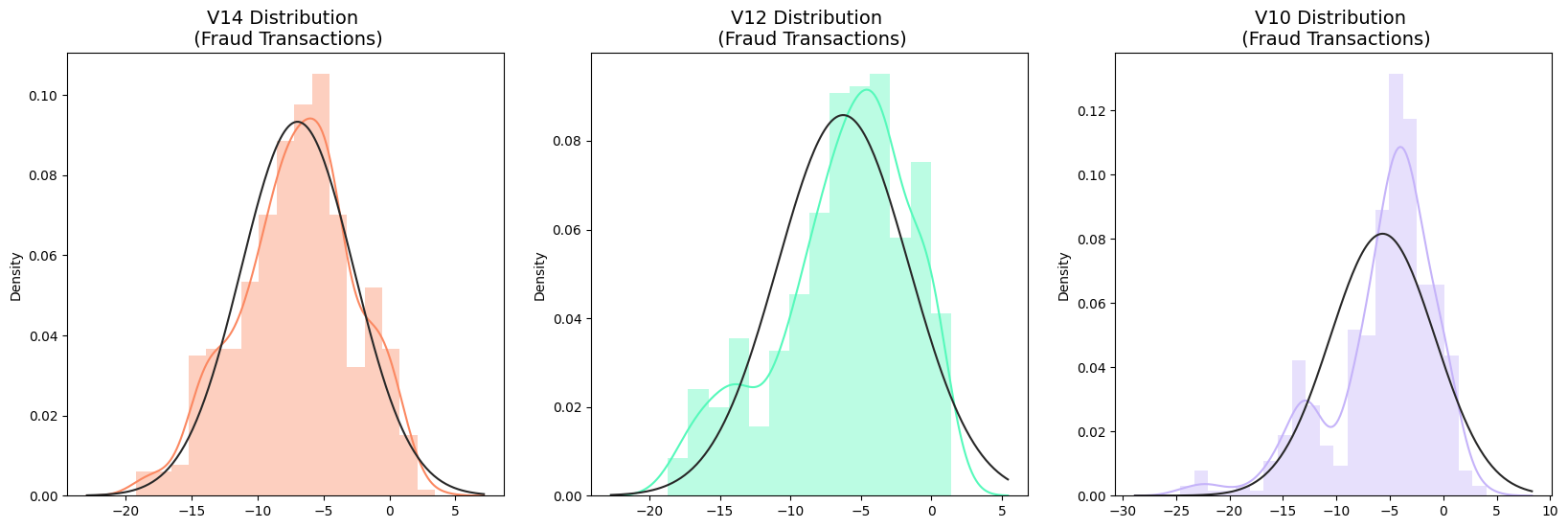
The aim here is to remove extreme **outliers** from features that have a high correlation with the classes. This will have a positive impact on the accuracy since they won’t shift the space towards them anymore.

For this, we will use the **Interquartile Range Method.** We calculate this by the difference between the 75th percentile and 25th percentile. Our aim is to create a threshold beyond the 75th and 25th percentile that in case some instance pass this threshold the instance will be deleted.

**Outlier Removal Tradeoff**:

We have to be careful as to how far do we want the threshold for removing outliers. We determine the threshold by multiplying a number (ex: 1.5) by the (Interquartile Range). The higher this threshold is, the less outliers will detect (multiplying by a higher number ex: 3), and the lower this threshold is the more outliers it will detect.

The Tradeoff: The lower the threshold the more outliers it will remove however, we want to focus more on "extreme outliers" rather than just outliers. Why? Because we might run the risk of information loss which will cause our models to have a lower accuracy. You can play with this threshold and see how it affects the accuracy of our classification models.



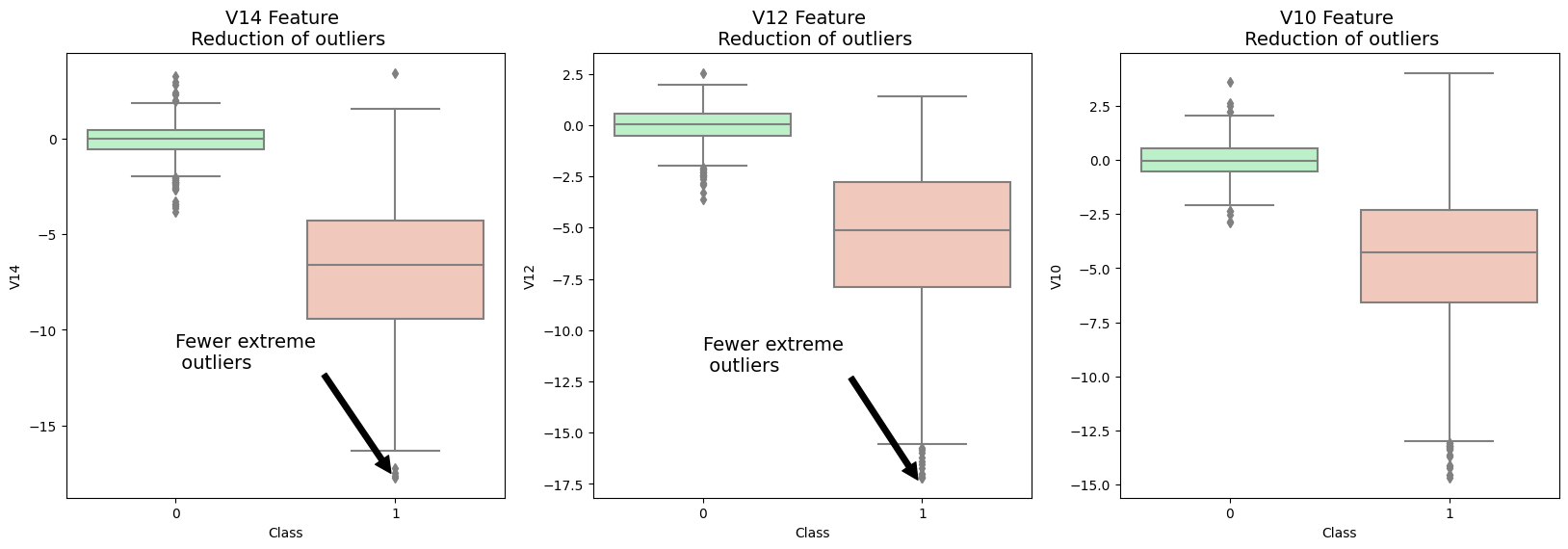
Sample Results:

V14 Lower: -17.807576138200666

V14 Upper: 3.8320323237414167

Feature V14 Outliers for Fraud Cases: 4

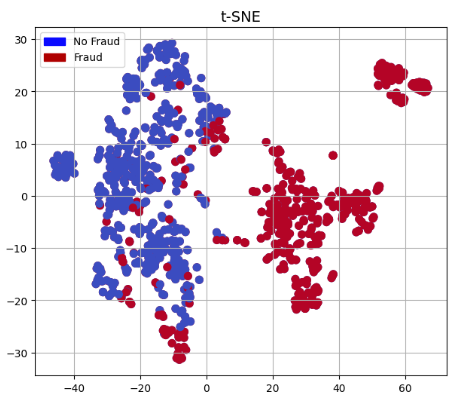
V14 outliers:[-18.8220, -19.21432, -18.49377, -18.04999]



**Dimensionality Reduction and Clustering**

We have also done an extra step before classification which was to apply Clustering using t-SNE which can pretty accurately cluster the cases that were fraud and non-fraud in the dataset, which gave us a good indication that further predictive models will perform well

T-SNE reduces high-dimensional data to a lower-dimensional space while preserving its structure. It computes pairwise similarities between data points in the high-dimensional space. Then, it constructs a similar probability distribution in the lower-dimensional space. Through iterative optimization, it adjusts the positions of data points to minimize differences between the two distributions using gradient descent. It employs a Student's t-distribution to measure similarity in the lower-dimensional space. Finally, t-SNE produces a lower-dimensional embedding of the data suitable for visualization and analysis.



**Model Training**

We trained 5 types of classifiers:

* Logistic Regression
* KNN
* SVM
* Decision Tree Classifier
* Random Forest

Before the training phase, we applied Grid Search to determine the parameters that gave the best predictive score for each estimator. The range of parameters was specified by us and it was selected purely based on experiments

* Logistic Regression yielded the highest training accuracy compared with the other 4 (SVM was a worthy competitor too)
* Logistic Regression has the best Receiving Operating Characteristic score (ROC), meaning that it pretty accurately separates fraud and non-fraud transactions.

Logistic Regression Has a training score of **95.0 %** accuracy score

KNeighborsClassifier Has a training score of **93.0 %** accuracy score

SVC Has a training score of **92.0 %** accuracy score

DecisionTreeClassifier Has a training score of **88.0 %** accuracy score

RandomForestClassifier Has a training score of **93.0 %** accuracy score

We applied **Cross-Validation**

Cross-validation is vital in machine learning for robustly estimating a model's performance. It assesses how well a model generalizes to unseen data by repeating the evaluation process on different data subsets. Cross-validation helps detect and mitigate overfitting by evaluating the model's performance across multiple folds of the data. It's instrumental in hyperparameter tuning, aiding in the selection of optimal model configurations. Additionally, cross-validation facilitates model selection by comparing the performance of different models on the same dataset.

Logistic Regression Cross Validation Score: 93.4%

Knearst Neighbors Cross Validation Score 92.87%

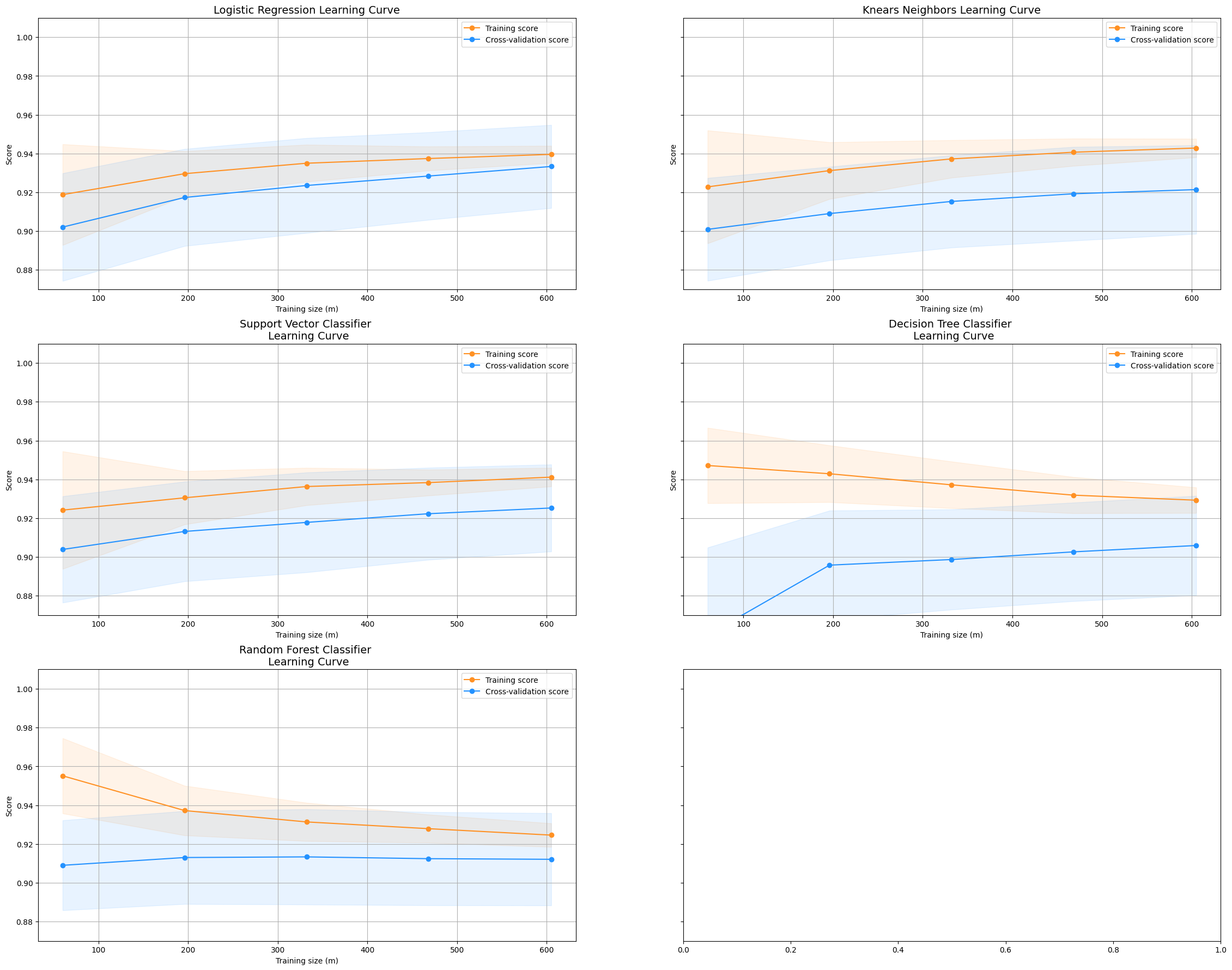
Support Vector Classifier Cross Validation Score 93.27%

DecisionTree Classifier Cross Validation Score 91.29%

Random Forest Classifier Cross Validation Score 91.42%

We also plotted the **Learning Curves** of each classifier:

* The wider the gap between the training score and the cross validation score, the more likely your model is overfitting (high variance).
* If the score is low in both training and cross-validation sets this is an indication that our model is underfitting (high bias)
* Logistic Regression Classifier shows the best score in both training and cross-validating sets.



Then, we calculated the ROC score which was our main metric to focus on:

Logistic Regression: 0.9753464501964734

KNearst Neighbors: 0.9259449595167177

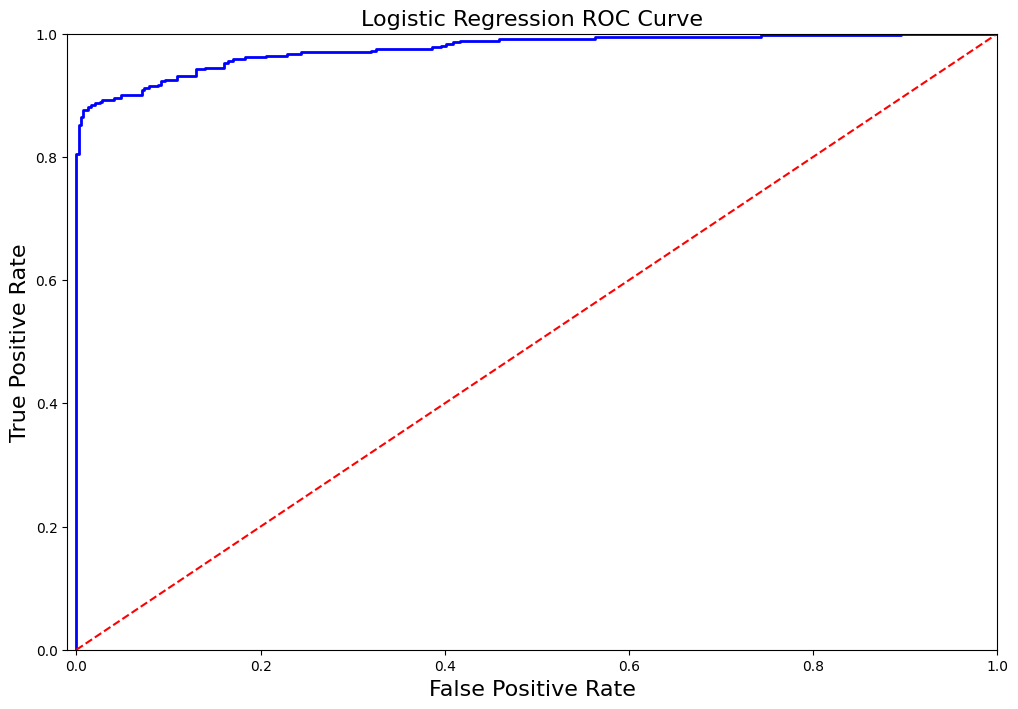
Support Vector Classifier: 0.9744584749199424

Decision Tree Classifier: 0.9106396218763546

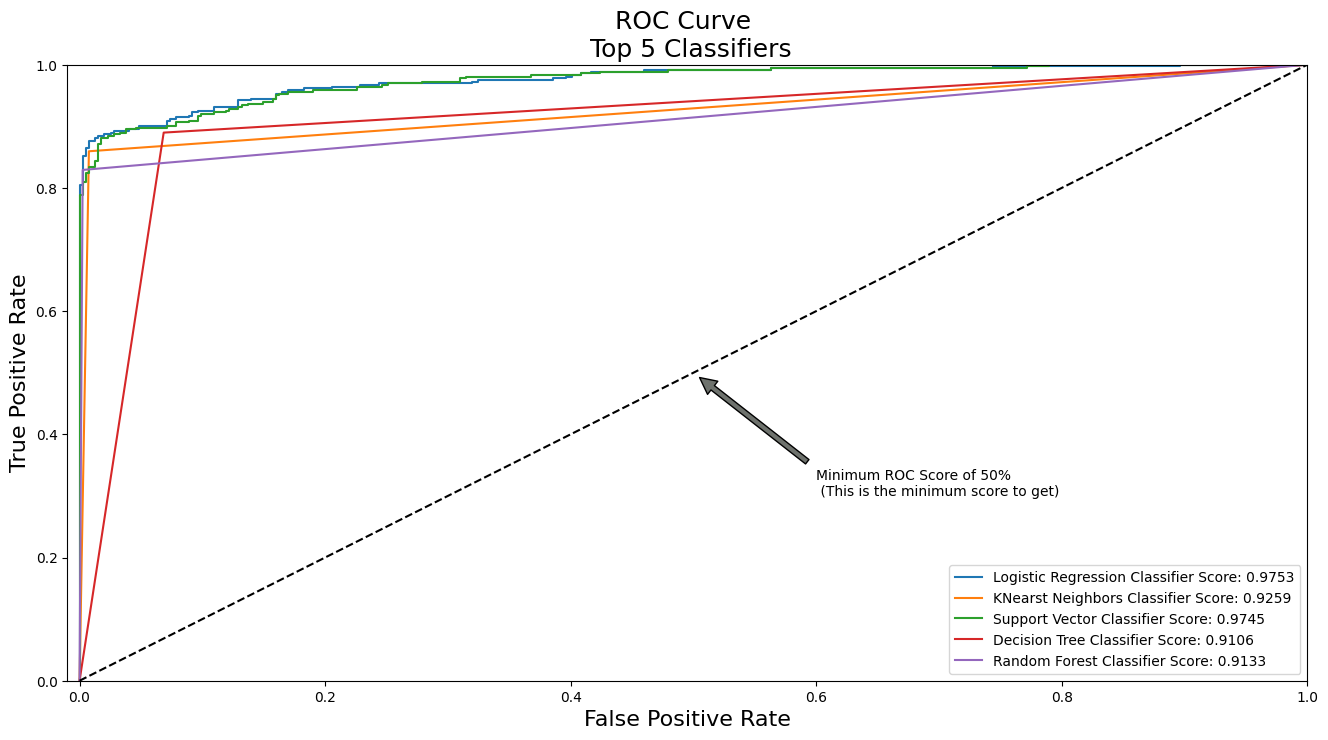
Random Forest Classifier: 0.9133315154311924

ROC (Receiver Operating Characteristic) curve is a graphical representation of a binary classifier's performance. It illustrates the trade-off between its true positive rate (sensitivity) and false positive rate (1-specificity) across various thresholds. The curve plots true positive rate against false positive rate, with the area under the curve (AUC) representing the classifier's overall performance. A higher AUC indicates better discrimination ability, with a value of 1 representing a perfect classifier. ROC analysis helps in assessing and comparing the performance of different classifiers in binary classification tasks

As we said, Logistic Regression scored the best here with SVC coming really close.



This was the ROC curve for all 5 classifiers



The breaking point in an ROC curve marks a threshold where the false positive rate (FPR) begins to increase rapidly. This occurs when the classifier becomes less selective, leading to more false positive classifications. While the true positive rate (TPR) may continue to increase, the rapid rise in FPR signifies a shift towards higher sensitivity but at the expense of reduced specificity. Identifying the breaking point is crucial for understanding the trade-off between sensitivity and specificity and determining an optimal operating point for the classifier. It helps in making informed decisions about the classifier's performance and suitability for the task at hand.

The curve of Logistic regression is the steepest meaning that the sensitivity of the model (TPR) is best at classifying true positives until a certain point. After that, the false positive rate starts to increase because that’s how far the model can get. The more left skewed that point is, the better the classifier.

*Another way different from Random Undersampling*

**Oversampling using SMOTE**

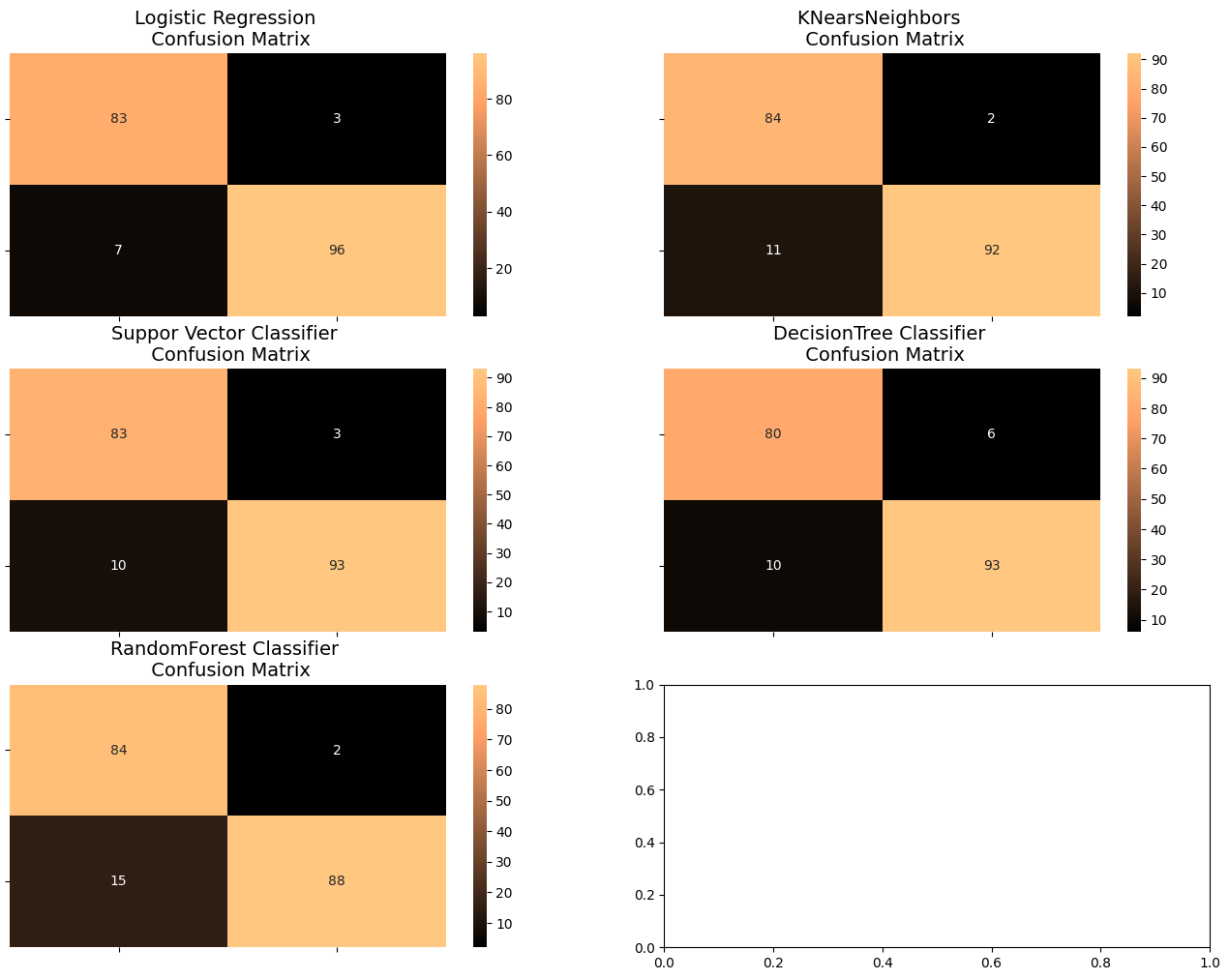
**SMOTE** stands for Synthetic Minority Over-sampling Technique. Unlike Random Undersampling, SMOTE creates new synthetic points in order to have an equal balance of the classes. This is another alternative for solving the "class imbalance problems".

Understanding SMOTE:

* Solving the Class Imbalance: SMOTE creates synthetic points from the minority class in order to reach an equal balance between the minority and majority class.
* Location of the synthetic points: SMOTE picks the distance between the closest neighbors of the minority class, in between these distances it creates synthetic points.
* Final Effect: More information is retained since we didn't have to delete any rows unlike in random undersampling.
* Accuracy & Time Tradeoff: Although it is likely that SMOTE will be more accurate than random under-sampling, it will take more time to train since no rows are eliminated as previously stated.

Therefore, we will test the Logistic Regression model on this technique.

***Now,* let’s test each classifier on the testing dataset and observe the confusion matrix of each.**

****

**FINAL RESULTS**

Logistic Regression:

precision recall f1-score support

0 0.92 0.97 0.94 86

1 0.97 0.93 0.95 103

accuracy 0.95 189

macro avg 0.95 0.95 0.95 189

weighted avg 0.95 0.95 0.95 189

-------------------------------------------------------------------------------------------------------

KNears Neighbors:

precision recall f1-score support

0 0.88 0.98 0.93 86

1 0.98 0.89 0.93 103

accuracy 0.93 189

macro avg 0.93 0.93 0.93 189

weighted avg 0.94 0.93 0.93 189

-------------------------------------------------------------------------------------------------------

Support Vector Classifier:

precision recall f1-score support

0 0.89 0.97 0.93 86

1 0.97 0.90 0.93 103

accuracy 0.93 189

macro avg 0.93 0.93 0.93 189

weighted avg 0.93 0.93 0.93 189

-------------------------------------------------------------------------------------------------------

Decision Tree Classifier:

precision recall f1-score support

0 0.89 0.93 0.91 86

1 0.94 0.90 0.92 103

accuracy 0.92 189

macro avg 0.91 0.92 0.91 189

weighted avg 0.92 0.92 0.92 189

-------------------------------------------------------------------------------------------------------

Random Forest Classifier:

precision recall f1-score support

0 0.85 0.98 0.91 86

1 0.98 0.85 0.91 103

accuracy 0.91 189

macro avg 0.91 0.92 0.91 189

weighted avg 0.92 0.91 0.91 189

Finally, we decided to make a quick comparison between the random undersampling and the oversampling using SMOTE technique for Logistic Regression and it turns out that oversampling is the way to go!

