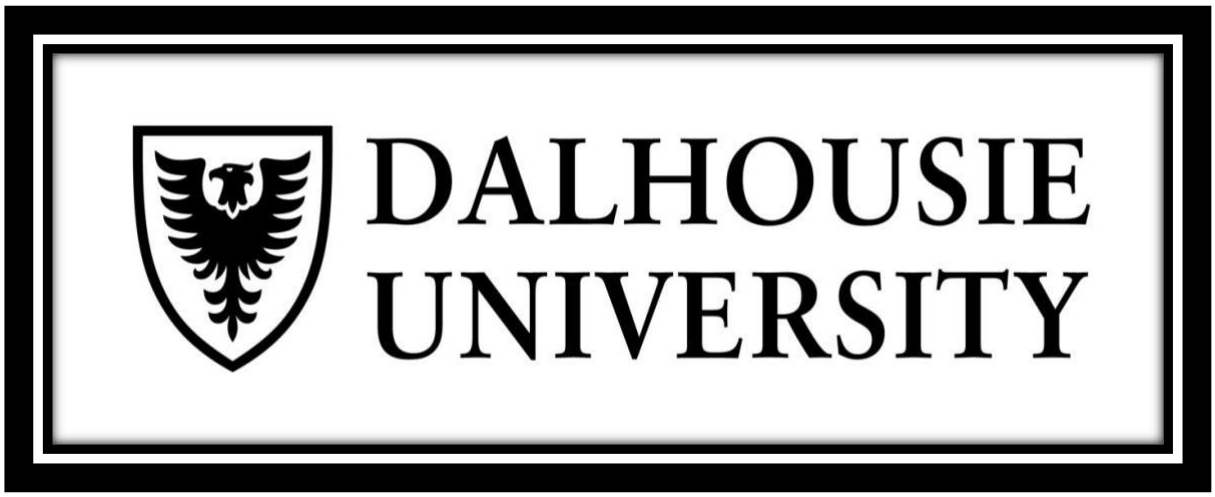


CSCI-6409

PROCESS OF DATA SCIENCE

FRAUD DETECTION ON BANKING PAYMENTS

Course Instructor: Dr. Axel Soto

GROUP MEMBERS:

Name	**B00 ###**
SARTHAK PANDIT	B00900388
PALLAVI CHERUKUPALLI	B00887062
GURYASH SINGH DHALL	B00910690
AMANDEEP SINGH MATTA	B00886925
SAGAR DEVESH	B00905507

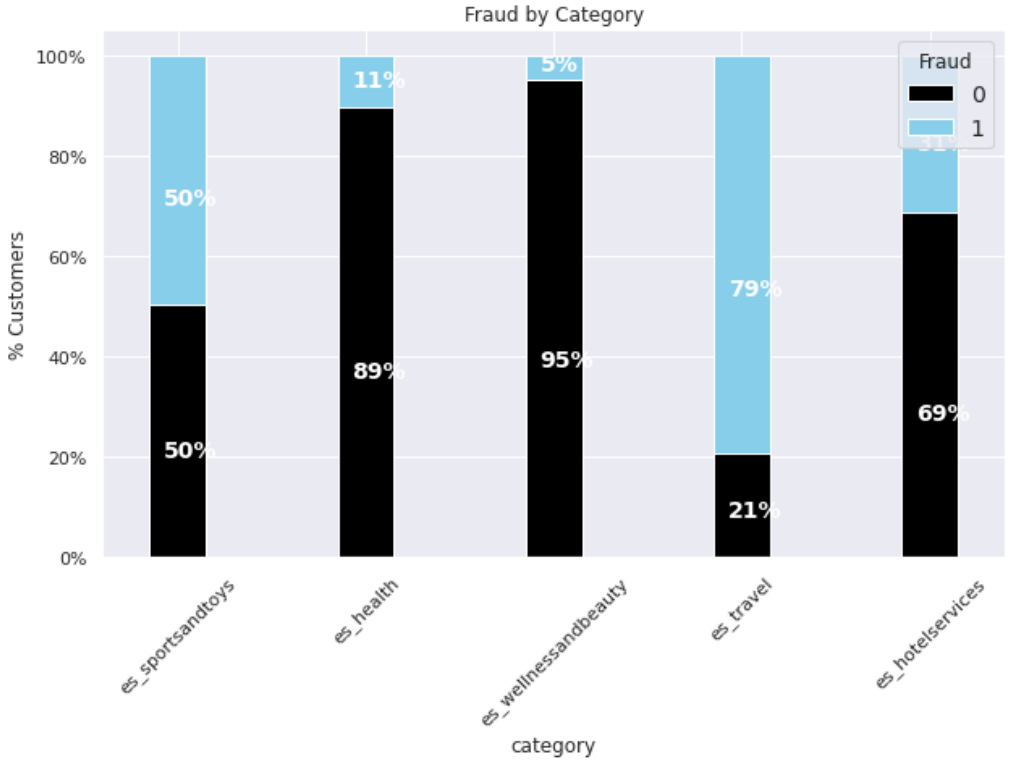
Link to GitHub Repository: https://git.cs.dal.ca/spandit/csci6409-fraud-detection-bank-payments

1. Data understanding and pre-processing –

We have used the Banksim dataset for our project to detect fraudulent transactions. Banksim is an abbreviation for Bank Payment Simulation. We have the 'fraud' feature as our target variable which will provide us with the information of whether a particular transaction is fraudulent or non-fraudulent.

Data Understanding:

We skimmed through the original data and found an interesting pattern regarding the number the fraudulent transactions per each category. After analysing, we found that 'travel', 'sports and toys', 'hotel services', 'health' and 'wellness and beaty' are the top 5 categories based on the number of fraudulent transactions, with 'travel' being at the top of the list. Below is the visualisation of the pattern that we observed:



Data Cleansing:

For data cleansing, we checked for missing values, however the dataset does not have any missing values. The dataset also has several columns in which the values are within quotation marks. These quotation marks will not play any role in influencing the performance of the model. Hence, we removed all the quotation marks from whichever features that had them.

Oversampling:

Another transformation that we applied to the dataset was SMOTE (Synthetic Minority Oversampling Technique) oversampling. The target feature 'fraud' is skewed in which 98.79% of the transactions are normal while only 1.21% are fraudulent transactions. This imbalance introduces bias into the dataset which would have affected the model's predictions. Hence, we performed oversampling in which we duplicated the instances of the minority class (fraudulent transactions) to achieve some balance in the target feature.

The selected Banksim synthetic dataset serves the purpose of our project, as it has all the features that could potentially lead a model into predicting whether a transaction is fraudulent or not. More importantly, it has a binary class target feature 'fraud', which eventually gives us the information about the class of prediction.

2. Literature Review

Source 1:
Credit Card Fraud Detection Algorithm using Logistic Regression [3]

This source of literature aims to compare the performance of four machine learning models - Logistic Regression, K-Nearest Neighbor, Decision Tree Classification and Random Forest Algorithm in detecting fraudulent transactions in bank payments. We have chosen logistic regression implementation from this paper and have worked on how the implementation could be improved in our approach. The dataset used for training the models is the Banksim dataset.

The target feature in the dataset reveals whether a transaction is fraudulent or non-fraudulent, however the number of fraudulent transactions naturally is far too small as compared to non-fraudulent transactions. The class instances (1 and 0) in the target feature are imbalanced. This introduces some bias in the dataset. To address this bias, the authors of the paper have used the under-sampling technique for balancing the dataset. Using under sampling, however, may lead to loss of potentially important data as events are removed without any consideration for what they are and how useful they might be for the analysis. In our approach, we have used oversampling instead of under sampling to get rid of the bias, as using this method ensures that no information from the original training set is lost as we keep all the members of the majority or minority class.

Authors of the paper have mentioned about the strength of different features in the dataset, however they have not highlighted feature selection or feature elimination. Feature selection is important as it helps in reducing the number of input variables, thereby reducing the overall computational cost of the model. Keeping this in mind, we have implemented Recursive feature elimination (RFE) for selecting the most prominent features to be used as input variables. RFE removes the weakest features until a specified number of features is reached. By default, it eliminates half the

features based on their importance. In our case, RFE eliminated 15 features out of 30 which reduced the computational complexity of the model significantly.

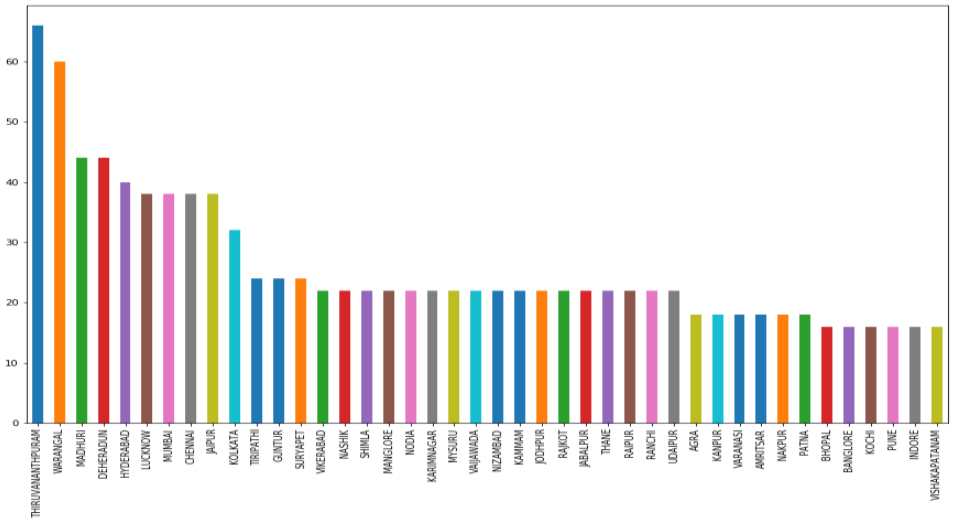
The paper shows the implementation of the logistic regression algorithm with a good accuracy, but it does not talk about hyperparameter tuning. Hyperparameter tuning helps in choosing the best set of parameters to minimize the error and draw out the best performance from a model. We have implemented hyperparameter tuning using the RandomizedSearchCV technique in which random combinations of the hyperparameters are used to find the best solution. Implementing this gives us the best set of parameters to be used in the logistic regression model.

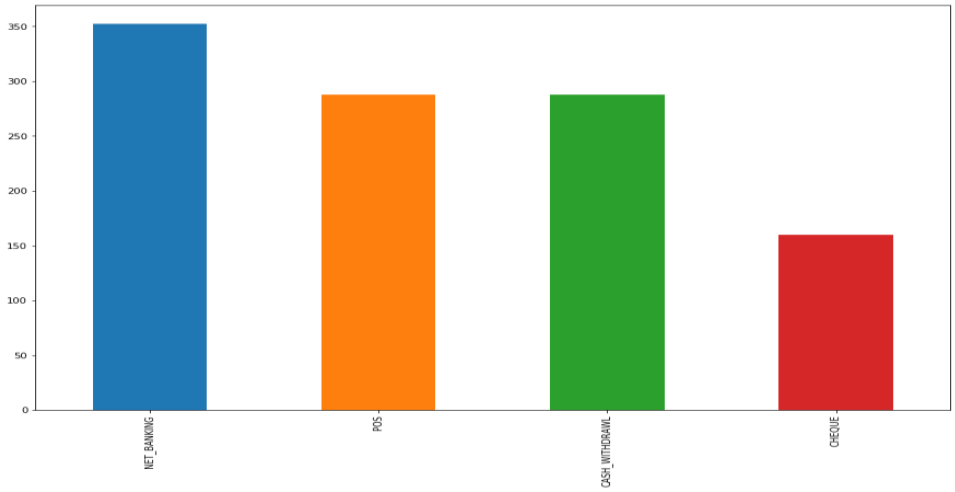
Source 2 -
Credit Card Fraud Detection Algorithm using Decision Trees based Random Forest Classifier [2]

In this source of literature research, the focus is on the detection of fraudulent transactions using the Random Forest classifier algorithm. The problem statement here is trying to solve a supervised classification problem, with the help of decision trees to classify the dataset. Once the classification is done, the results are evaluated using a confusion matrix. This paper aims at improving the accuracy of fraud detection by using real-time transactional data from a European banking institution.

The study systematically follows the steps to implement the system, starting with extraction, cleaning, pre-processing, classifying the data into training and testing sets, applying a random forest algorithm to the dataset and finally deriving the results and evaluating them.

The strength of the study is the diversity of the dataset considered and the varied analysis performed on the dataset. The paper brings together insights on how fraudulent credit card transactions are performed in different locations as well as different banking environments.





However, the paper has very little information on the evaluation process. It doesn't provide details on how the accuracy and confusion matrix is calculated and performed. Also, it lacks the insights that are derived from the same.

Also, the paper doesn't talk about the hyperparameter tuning of the random forest algorithm. The paper doesn't address the issue of the original model failing to give good accuracy and how it can be tuned to give better results.

With the help of this research paper source, we will be using the same approach in implementing the random forest algorithm and using similar techniques to clean, pre-process and classify the dataset. This paper helps enrich our solution and helps us better understand the significance of the random forest algorithm and how it can be used to evaluate the predictive analysis of fraudulent transactions.

Source 3:

Detecting Credit Card Fraud by Decision Trees and Support Vector Machines [1]

This piece of literature is a comparative study on the decision tree and the Support Vector Machines to detect credit card fraud and improve financial security. In this work, fraud detecting models were constructed using seven classification approaches. The paper highlights the benefits of using data mining techniques, such as decision trees and SVMs, to the problem of detecting credit card fraud with the aim of lowering the risk to the bank.

The study highlights that the decision trees stand out by providing better results than the SVM model. However, it also states that with larger datasets, the performance of the SVM model can match the decision trees in terms of accuracy.

The decision tree methods C5.0, C&RT, and CHAID as well as SVMs with kernels of polynomial, sigmoid, linear, and radial basis functions were used to construct the classifier models. The C&RT uses the Gini coefficient and the maximum number of surrogates as parameters to build the model. Surrogates are one way to handle missing values. Gini can help in selecting the best features for itself to build the model by taking the impurity measure into the eye. This is one of the perks of using a decision tree.

Apparently, the performance for the SVM models was nearly 99% on the train data and 83% on the test data. On the other hand, the decision trees had training accuracy of about 90% and a test average of about 87%. After analyzing the results, the SVMs performed very well in the testing data while failing to yield similar results in the training set. This clearly indicates that the SVM model has completely overfit the data.

The fraudulent datasets are usually biased as it includes a greater number of non-fraud transactions than fraudulent ones. To make it unbiased, the author used stratified sampling by segregating data based on strata. However, for this project, we changed the approach to oversampling through smote to get biased data. We used the classification and Regression Tree model as the accuracy remained consistent through all the three sets used in the model for training by adhering to the standards of overfitting. We will also be using the accuracy score as the measure of evaluation metric to determine the performance.

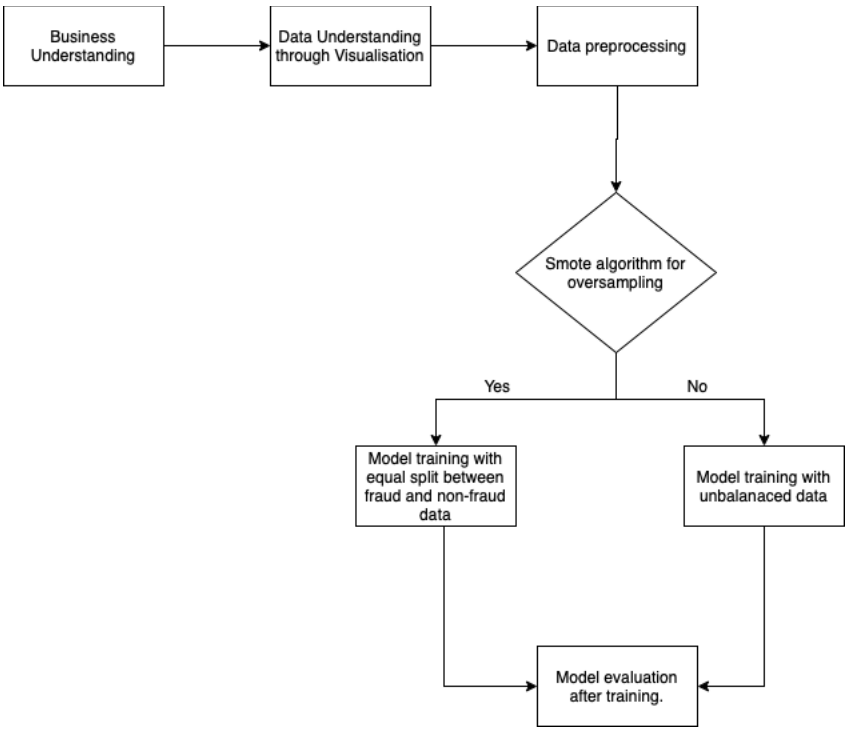
The classification and the regression trees use the Gini coefficient, which is the impurity measure used to determine how to split the tree. Gini helps in determining the chances of picking up the wrong instance from a dataset. Also, decision trees make use of the number of surrogates for dealing with the missing data which can lessen the burden on the preprocessing of the data. Also, Gini can help in reducing the workload on the feature selection method. It can calculate the Information Gain (IG) on its own thereby building the model with greater accuracy without much manual intervention.

As a part of the research, we built six decision tree models. Three before smote and three later. The first one is a generic decision tree based on KBest features, following which is the Gini and entropy model. The current paper has only been implemented using Gini. In our project, we are also implementing the model using entropy. We have

also plotted the learning curve based on the misclassification error. Finally, we plotted a confusion matrix for the Gini and entropy model after smote.

This paper helped us in bringing a better insight into the decision tree in terms of practical development. We could use that similar technique to build and evaluate our model.

3. Description of the solution



Project description

1. **Business Understanding:**

   In this section of the project, we understood the requirement of the project, the problems faced by the user in the area of bank fraud detection. We discovered a dataset that was artificially created and had payments from multiple clients made across a range of durations and quantities. The target variable in the dataset was fraud or non-fraud through which we concluded that we must solve a classification problem.

2. **Data understanding:**

   The team made sure to gather the relevant data during this project phase. We looked at the data's format, the quantity of records, and any null values. Then, we looked more closely at the data. It can be queried, visualised, and the relationships between the data are found. Using the data quality plan for

categorical and continuous features, we also pinpoint the problems with the data's quality. In this step we also noticed that the fraud data and non-fraud data was quite imbalanced.

3. **Data pre-processing:**

   We then performed some pre-processing steps. We first saw that some of the columns ('step', 'amount', 'fraud') had quotes in their data, so we first removed the quotes. We then converted all the columns into their relevant data type. The team also performed one hot encoding on categorical columns ('age', 'gender', 'category').

4. **Oversampling using Smote:**

   As there was imbalance between the fraud and non-fraud data, we used Smote algorithm to make the data balanced. We approached the problem using 2 ways, first we built the model using smote algorithm and second way in which we didn't use smote for the data and built the data on that model.

5. **Model building:**

   We then built models in 2 phases (with and without SMOTE). The model selected are Logistic Regression, Decision Tree Classifier and Random Forest Classifier.

6. **Evaluation:**

   We then evaluated the model accuracy and plotted a confusion matrix to analyse the results.

	step	customer	age	gender	zipcodeOri	merchant	zipMerchant	category	amount	fraud
0	0	C1093826151	4	M	28007	M348934600	28007	es_transportation	4.55	0
1	0	C352968107	2	M	28007	M348934600	28007	es_transportation	39.68	0
2	0	C2054744914	4	F	28007	M1823072687	28007	es_transportation	26.89	0
3	0	C1760612790	3	M	28007	M348934600	28007	es_transportation	17.25	0
4	0	C757503768	5	M	28007	M348934600	28007	es_transportation	35.72	0
...	...	...	...	...	...	...	...	...	...	...
594638	179	C1753498738	3	F	28007	M1823072687	28007	es_transportation	20.53	0
594639	179	C650108285	4	F	28007	M1823072687	28007	es_transportation	50.73	0
594640	179	C123623130	2	F	28007	M349281107	28007	es_fashion	22.44	0
594641	179	C1499363341	5	M	28007	M1823072687	28007	es_transportation	14.46	0
594642	179	C616528518	4	F	28007	M1823072687	28007	es_transportation	26.93	0

Sample data

The above sample data presents customer transactions. The dataset has information regarding the customer age, gender, zipcodeOri, merchant, zipMerchant, category, amount and fraud. With the use of visualization and scatter plot matrix, we selected features which effect whether a fraud will occur or not. The features that we selected

were age, merchant, category and amount. Our target feature is fraud or not fraud which is a categorical feature. After understanding the type of the problem, which is classification, we selected 3 algorithms which are mentioned in above section.

4. Data Analysis, Results and Evaluation

The Banksim dataset consists of 10 features representing financial transactions, namely step, customer, age, gender, zipcodeOri, merchant, zipMerchant, category, amount and fraud. In the following sections, we will briefly run through all the features in the dataset, explaining each of them and analysing if there are any relevant patterns within them.

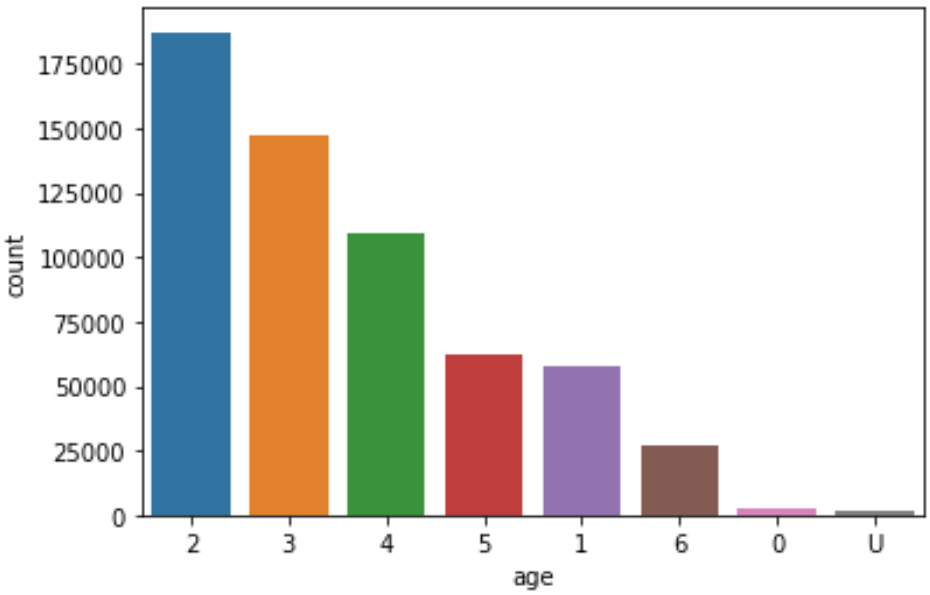
Step: Dataset stretches over six months, with the feature 'step' representing the number of days between each transaction and the first transaction of the dataset.

Customer: This feature represents the customer ID of each customer, and it does not really play a significant role in model predictions.

Age: The feature 'Age' is categorized into the following 8 sections:-

- 0: <= 18,
- 1: 19-25,
- 2: 26-35,
- 3: 36-45,
- 4: 46:55,
- 5: 56:65,
- 6: > 65
- U: Unknown

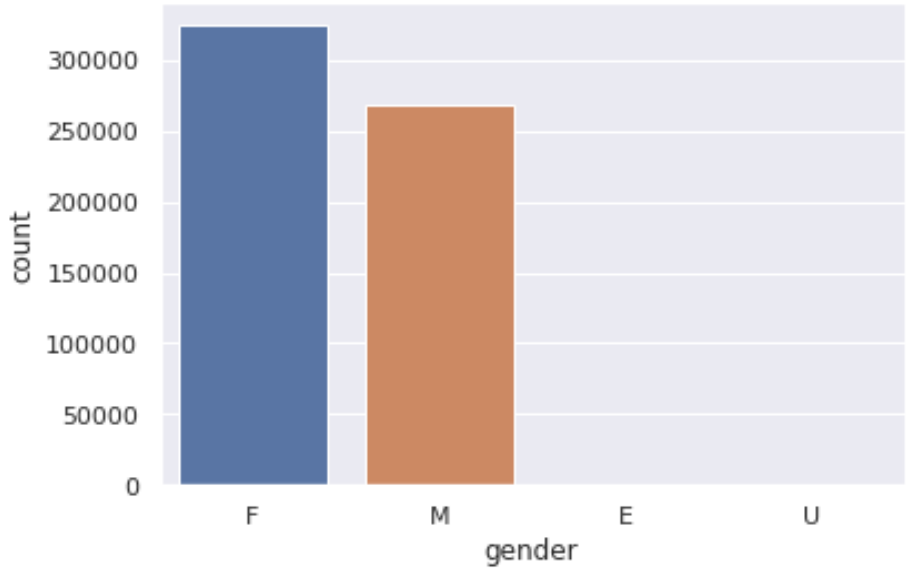
This feature represents the age of all the customers that were involved in the transactions as per the Banksim dataset. The below bar plot shows the number of customers of each age group in the dataset. The age group '2' with customers between the age of 26 and 35 years is the largest group, whereas customers below 18 years of age are the least in number.



Gender: This feature represents the gender of the customer. Below are the 4 classes:

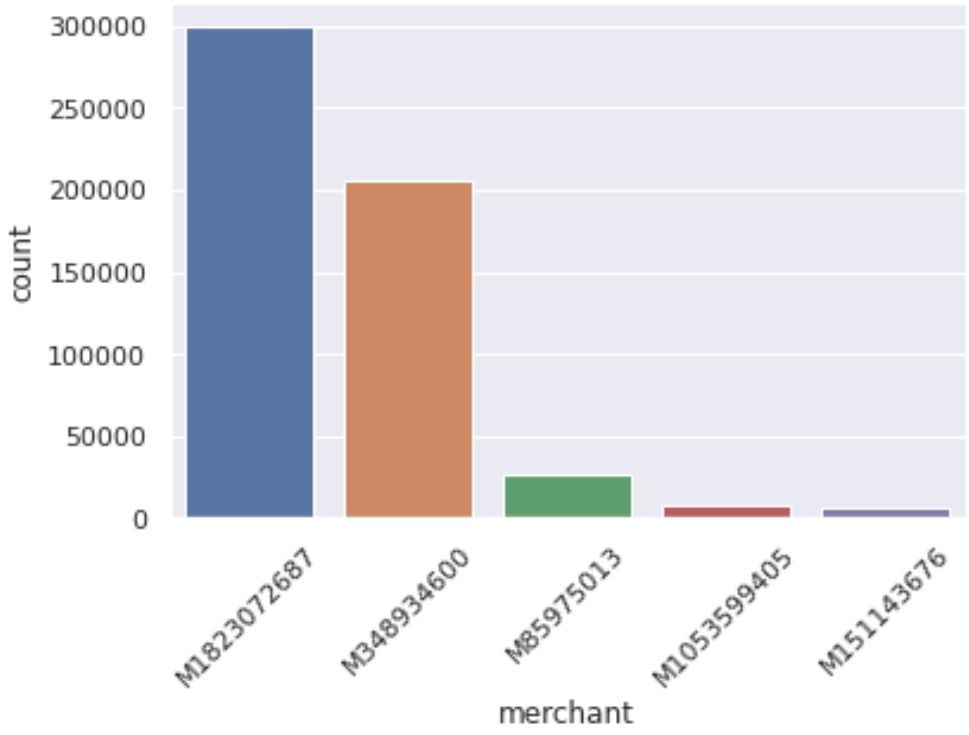
- F: Female,
- M: Male,
- E: Enterprise,
- U: Unknown

The below bar plot shows the number of customers of each gender class in the dataset. It comprises of more female customers as compared to male customers, whereas the ‘Enterprise’ and ‘Unknown’ classes have negligible numbers.



zipCodeOrigin: This feature represents the zip code of origin/source.

Merchant: This feature represents the merchants' ID. The below bar plot represents the count of each merchant ID.

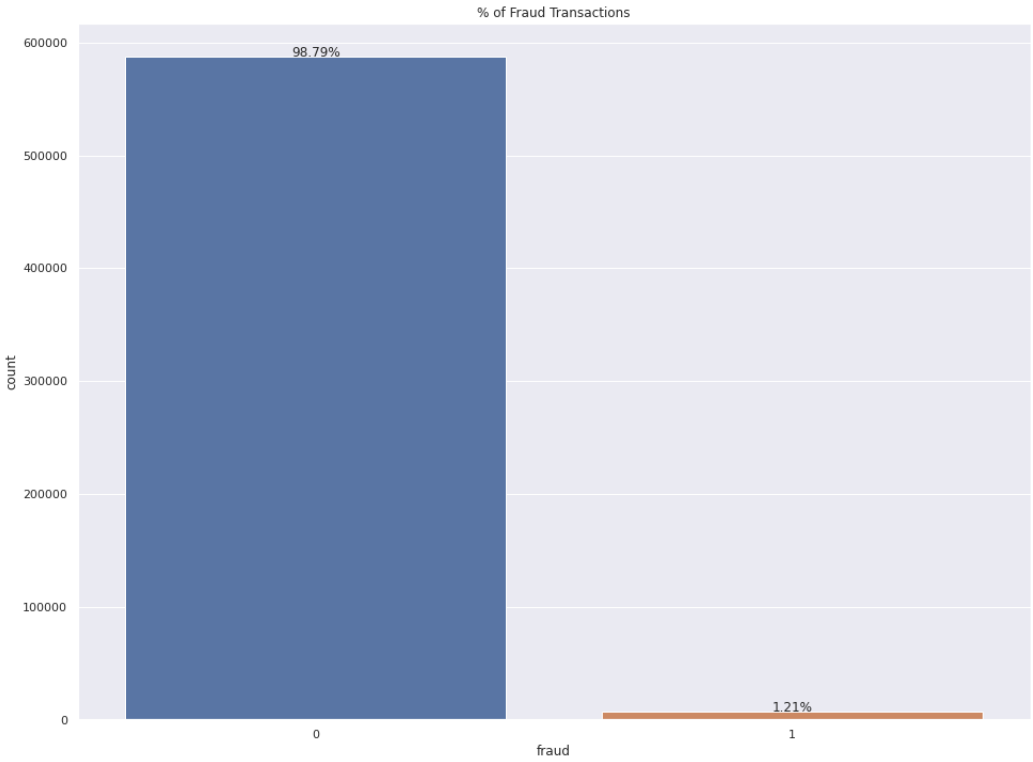


Category: This feature represents the category of the purchase. Below are the different categories: -

- es_transportation
- es_food
- es_hyper
- es_barsandrestaurants
- es_contents
- es_wellnessandbeauty
- es_fashion
- es_leisure'
- es_otherservices
- es_sportsandtoys
- es_tech
- es_health
- es_hotelservices
- es_home
- es_travel

Amount: This feature represents the amount of the purchase.

Fraud: 'Fraud' is the target feature, in which 0 represents 'normal' transactions, whereas 1 represents 'fraudulent' transactions This feature however is unbalanced, as the number of fraudulent transactions is exceedingly small as compared to the number of non-fraudulent transactions. The below bar plot shows that 98.79% of the transactions are non-fraudulent transactions, whereas only 1.21% are fraudulent transactions.



Hence, for balancing the dataset, we used the SMOTE oversampling technique by increasing the number of minority class ‘fraud’ by generating instances from the same class. Oversampling in this case is the preferred choice as it does not lead to any loss of critical information. Under sampling on the other hand would mean dropping observations of the majority class, which could lead to losing out on some valuable information.

In our approach, we have used three different models to predict whether a transaction is fraudulent or non-fraudulent. The three models we have used are Logistic regression, Decision tree and Random Forest algorithm. For each model, we have fit the data twice, once before oversampling and once after oversampling to compare the results. We will go through the implementation, evaluation and results for each model in the following sections :-

As part of the model implementation, there were a total of 3 different machine learning algorithms, these models are -

i) **Logistic Regression** - Logistic regression is a statistical model that models the probability of one event occurring over another. It is used to predict a binary outcome – in our case the outcomes are fraudulent or non-fraudulent transactions.
ii) **Decision Tree** - Decision trees are supervised machine learning operations that model decisions, outcomes, and predictions using a flowchart-like tree structure.
iii) **Random Forest** - A random forest is a meta estimator that fits several decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

- **Feature selection**

Before fitting the data to the model, we performed feature selection using Recursive Feature Elimination (RFE) method and k-best algorithms. RFE is a feature selection technique that selects the features recursively considering smaller and smaller sets of features.
In our case, RFE helped us eliminate half the number of features that were unimportant, thereby reducing the number of input variables from 30 to 15. Feature selection helps to reduce the model complexity significantly.
K-best algorithm selects features with the highest k-scores. With the help of 'k' parameter in the method, we can select the threshold on the number of features. For our models, Decision tree and Random Forest we have used 6 as the feature selection threshold.

- **Implementing the model (Before SMOTE oversampling)**

The next step was to fit the data to the model before oversampling. However, before implementing the model,
For Logistic regression model, we split the data into train set and test set in the ratio of 80:20.
For Decision model, we split our data into training and testing in the ratio of 70:30.
For Random Forest model, we split our data into training, testing and validation in the ratio of 40:30:30.

The train set was used to train the model and the test set was used to evaluate the model's performance.

- **Evaluation**

i) **Logistic Regression**:

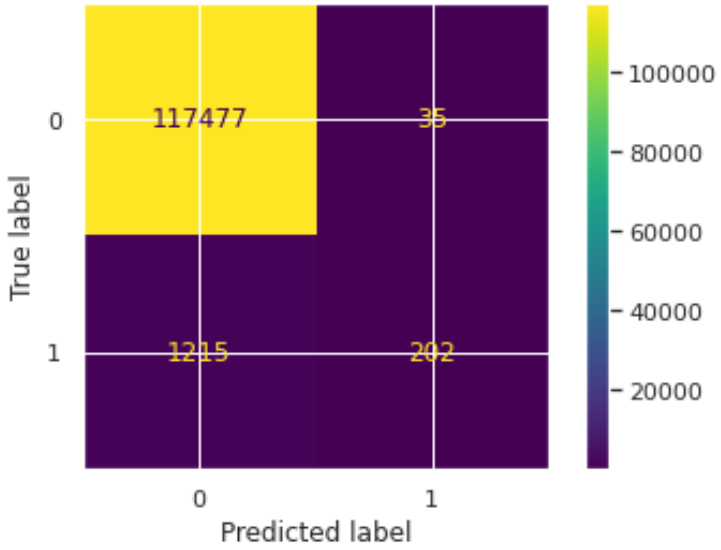
For evaluating the logistic regression model, we used accuracy score, confusion matrix, classification report and Receiver Operating Characteristic (ROC) Curve. The accuracy of the model on data before oversampling 99%. The below classification report that we obtained provides insights into statistics like precision, recall, f1-score and accuracy.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	117512
1	0.85	0.14	0.24	1417
accuracy			0.99	118929
macro avg	0.92	0.57	0.62	118929
weighted avg	0.99	0.99	0.99	118929

Classification report for logistic regression

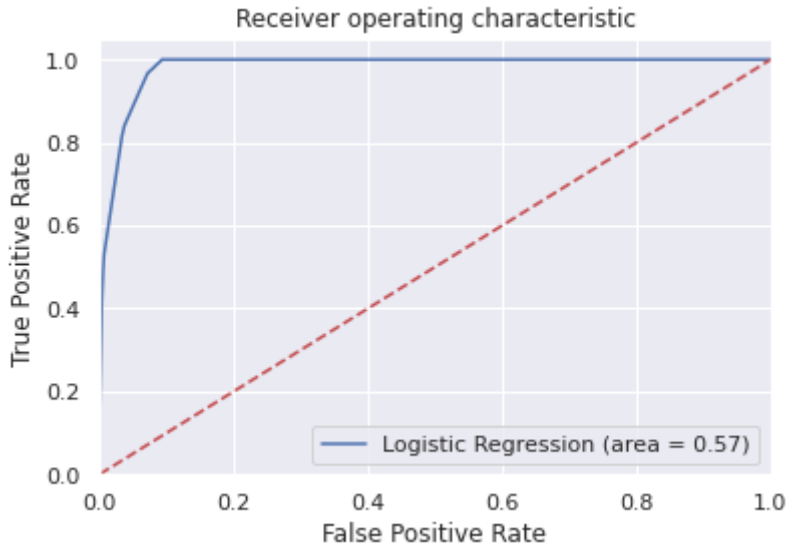
The confusion matrix below gives us an idea of how well the model has performed. As per the matrix, we can see that the model correctly predicted 117477 instances of the

positive class (True positives) and 202 instances of negative class (True negatives). Whereas there are 35 instances of false positives and 1215 instances of false negatives.



Confusion matrix for logistic regression

Below is an image of the ROC curve of the model. It is a plot of the false positive rate (x-axis) versus the true positive rate(y-axis). The area under the curve represents the degree of measure of separability. Higher the AUC, the better the model is at predicting the correct classes. The ROC AUC score of this model is 0.57.



ROC Curve

ii) **Decision Tree** – For evaluating decision tree algorithm, we used accuracy score as an evaluation metric. The model's accuracy score before SMOTE oversampling comes out to be 99% on the testing data.

```
#Measuring accuray for the model
print (metrics.accuracy_score(y_test.astype('int'), predicted_dt_entropy))
```

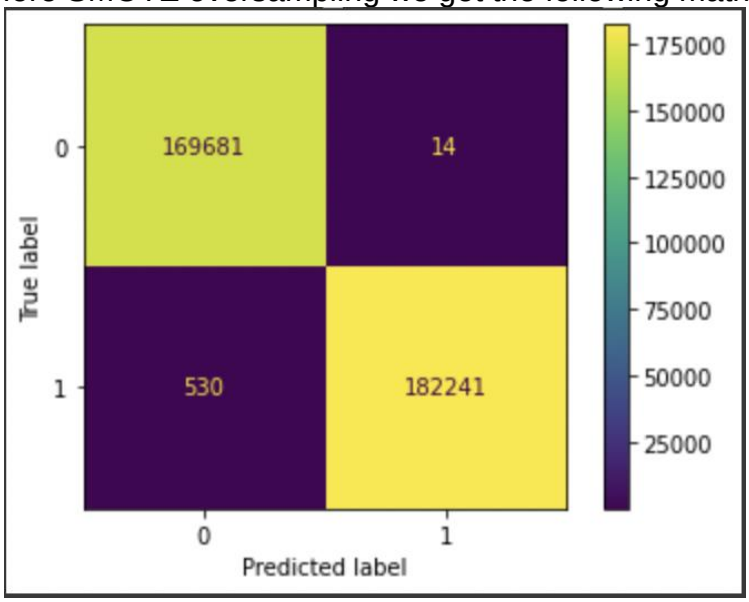
```
0.9947307349503623
```

iii) **Random Forest** – For evaluating Random Forest model, we used accuracy score and confusion matrix. Before SMOTE oversampling, this model yields an accuracy score of 93% on the training data with a minor variance in accuracy scores for training and validation datasets.

```
#Printing the accuracy scores of training, testing and validation data
print ('Training data :', train_score)
print('Testing data :', test_score)
print('Validating data :', valid_score)
```

```
Training data : 0.9355213488979772
Testing data : 0.9346632015570296
Validating data : 0.9350859373670084
```

We also plotted a confusion matrix as part of the evaluation for random forest, before SMOTE oversampling we get the following matrix -



The above confusion matrix was plotted on the testing data, once the model had evaluated the target variable. Based on the above results, we can see that there are **530** transactions that were misclassified as False Positives and a total of **14** transactions that were misclassified as False Negative.

Majority of the transactions were correctly classified as True Positives and True Negatives.

Implementing the model (After SMOTE oversampling)

As mentioned earlier, we performed SMOTE oversampling to get a balanced dataset. After oversampling, we retrained the logistic regression model to observe the changes that oversampling brings to the table.

- **Evaluation**

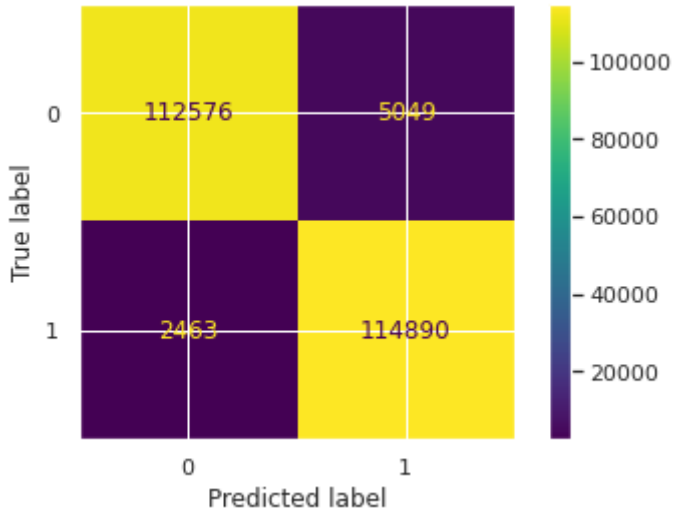
i) Logistic Regression -

We followed the same procedure for evaluating the model after SMOTE oversampling. The accuracy of the new model turned out to be 97% as compared to 99% when the model was trained before oversampling. The below classification report provides further details

	precision	recall	f1-score	support
0	0.98	0.96	0.97	117625
1	0.96	0.98	0.97	117353
accuracy			0.97	234978
macro avg	0.97	0.97	0.97	234978
weighted avg	0.97	0.97	0.97	234978

Classification report for logistic regression

The confusion matrix below shows that the new model got 112576 true positives and 114890 true negatives. We can see that the number of true negatives has drastically increased due to oversampling.



Confusion Matrix

The new model has got a better ROC curve with ROC AUC score of 0.97. This suggests that the second model has made more correct predictions as compared to the first model, since the first model had an ROC AUC score of 0.57.

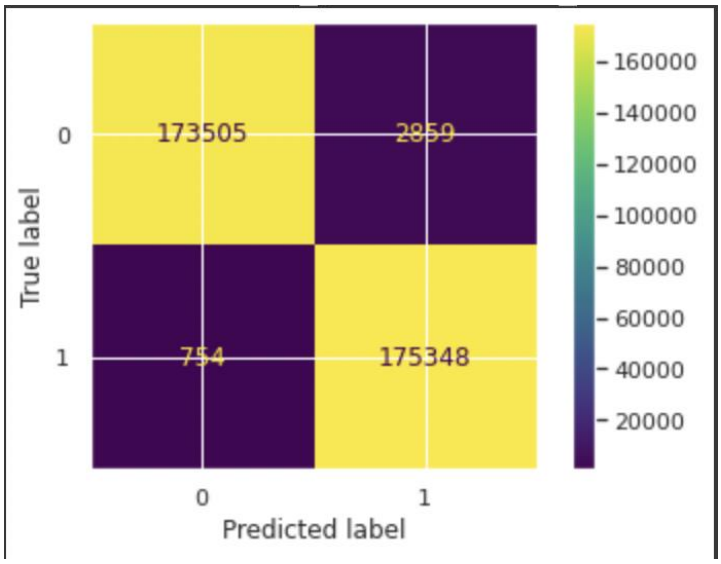
ROC curve

ii) Decision Tree –

After implementing SMOTE oversampling, we can see the accuracy for decision trees algorithm drops to 96%. This drop in accuracy score is expected after oversampling as the model tries to generalise the redundant dataset samples.

```
#Measuring the accuracy
print (metrics.accuracy_score(t_test_smote.astype('int'), predicted_value_after_smote))
```

```
0.9689643823801445
```



From the above confusion matrix, plotted after oversampling we can see that, around **754** transactions were classified as False Positives and around **2859** transactions were classified as False Negative.

Majority of the transactions have been correctly classified as True Positives and True Negatives.

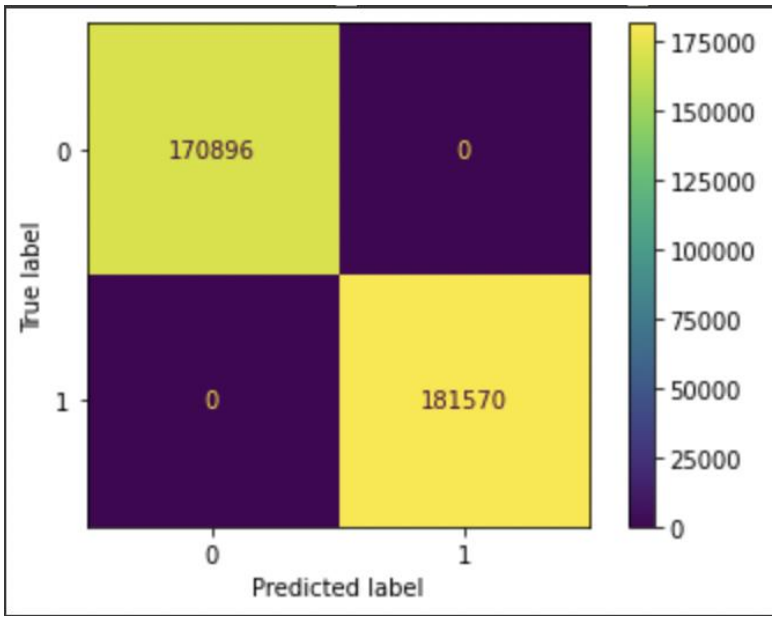
iii) Random Forest –

After implementing SMOTE oversampling, the accuracy of the model dropped to 93%. This dip in accuracy is expected due to the oversampling of dataset.

```
#Printing the accuracy scores of training, testing and validation data
print ('Training data :', train_score_hyptun)
print('Testing data :', test_score_hyptun)
print('Validating data :', valid_score_hyptun)
```

```
Training data : 0.935525604633645
Testing data : 0.934646178638507
Validating data : 0.9349071967225208
```

Accuracy for training, testing and Validation data



Confusion Matrix for Random Forest Algorithm

From the above confusion matrix, we can see that all the transactions have been correctly classified. There are no misclassified transactions.

- **Hyperparameter tuning**

For the first logistic regression models, we used RandomizedSearchCV to find the most optimal parameters. With this technique, random combinations of the hyperparameters are used to find the best solution for the built model. For the first logistic regression model, the ideal parameters are ('solver': 'lbfgs', 'penalty': 'l2', 'C': 10), using which we get the accuracy of 99.47%. For the second logistic regression model, we used HalvingRandomSearchCV of sklearn. In HalvingRandomSearchCV, a new class of successive Halving is taken, where training is performed on the subsets of data, rather than on all the data. Given the size of the dataset after oversampling,

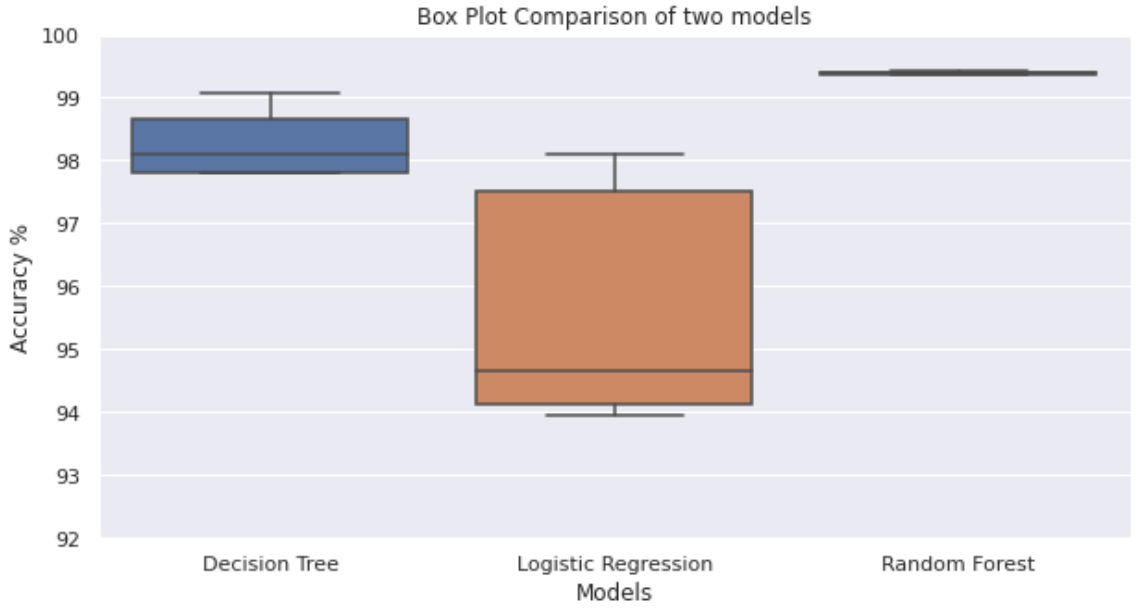
we found HalvingRandomSearchCV to be more appropriate than RandomizedSearchCV.

For the Decision Tree, hyperparameter tuning was done using entropy as criterion, random state of 100, maximum depth of 12, and minimum sample of leaves as 5. After using the parameters, we get an accuracy score of 98.8%.

For Random Forest, hyperparameter tuning was done using maximum depth of 1000 decision trees and n_estimators as 30. This yielded an accuracy score of 95.7%

Comparison of the three models

We plotted a box plot to draw out a comparison between the three models. As per the box plot below, the Random Forest model gave the highest accuracy in predictions, whereas the logistic regression model provided the lowest accuracy. Hence, random forest algorithm proved to be the best classification algorithm to segregate fraudulent transactions from the non-fraudulent ones in our project.



Box Plot comparison

5. Conclusion

Some of the algorithms that we implemented worked well for skewed data in which the non-fraud transactions are greater than the fraud transactions. Logistic regression and Decision Tree gave higher accuracy before Smote oversampling technique. This means that in a real- world scenario where fraud transactions are less in number, we can use Logistic regression and Decision Tree. Random Forest Algorithm gave good accuracy for equal split between fraud and non-fraud data rather than skewed data. Random forest algorithm can be used when the fraud and non-fraud transactions are almost equal.

One of the limitations to our approach is that we did not perform under sampling in which the non-fraud transactions might be made equal to the fraud transactions.

Considering the size of our data size, it would have led to less resource usage. However, under sampling may also lead to loss of potentially important data. Therefore, keeping this in mind, we opted for oversampling over under sampling to eliminate the bias in the dataset.

However, oversampling also has its own limitations. Although we got rid of the biasness in the dataset with oversampling, it comes at a cost of lower accuracy of the model. It is important for the models to correctly classify the False negative instances. False negatives here signify how many fraudulent transactions are classified as non-fraudulent, and higher false negative classifications can turn out to be detrimental for any business organisation. For all the three models that we have used – Logistic regression, Decision tree and Random Forest, we observe that after oversampling the accuracies of the model have reduced, which may also lead to slightly higher False negatives. Hence, it is a trade-off between achieving balance in the dataset and accuracy of the resulting model.

Another limitation of our approach is that we have used synthetic data which has less issues than a real-world data. The real-world data could have more noise than our data, so we might have to apply more data pre-processing steps on a real-world dataset. With synthetic data, we get more confidence in data quality and balance, whereas in a real-world dataset, we might have to perform a lot of pre-processing to get a dataset that can be fit into a model.

Improvements/ Future Work:

1. The number of hyperparameters can be increased, that gives us a variance and broader picture of the results. It will also give us the perspective of which parameters are bringing our accuracy score down and the parameters that might lead to an increase in the overall performance of our model.
2. We can implement this process on a more realistic dataset, that does not limit to fraud detection but also helps in detecting other features such as geographical locations, high-risk customers, or any politically-exposed-person. This model can benefit a lot to the banking sector.
3. We could increase our scope of data in such a way that we get more false-positives and false-negatives in our confusion matrices. This will help us train the model better and understand the nuances in each model with respect to classification. We could have dealt with more realistic data that are used by banks. These datasets rarely have a fixed pattern and are difficult to decode using the machine learning algorithms.

References

[1]Iaeng.org
, 2022. [Online]. Available:
http://www.iaeng.org/publication/IMECS2011/IMECS2011_pp442-447.pdf. [Accessed: 03- Aug- 2022].

[2]Citeseerx.ist.psu.edu
, 2022. [Online]. Available:
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.675.408&rep=rep1&type=pdf. [Accessed: 03- Aug- 2022].

[3]"Fraud Detection on Bank Payments Using Machine Learning", Ieeexplore.ieee.org

, 2022. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9726104. [Accessed: 03- Aug- 2022].

[4]"Fraud Detection on Bank Payments", Kaggle.com, 2022. [Online]. Available: https://www.kaggle.com/code/turkayavci/fraud-detection-on-bank-payments. [Accessed: 03- Aug- 2022].