# *Fastag Fraud Detection*

## *Introduction*

The increase in the number of vehicles every year causes traffic at toll gates to increase significantly. Most toll gates are manually operated by an operator in each lane which often results in long queues on congested toll roads.

In response to this challenge, the National Highway Authority of India (NHAI) has introduced Fastag as an innovative solution. FASTag is a passive RFID tag used to make direct toll payments from customers linked to prepaid or savings/current cards. The tag is affixed to the windshield of the vehicle and allows the customer to drive through the toll road, without stopping to make the toll payment. The toll rate is directly deducted from the customer's linked account. The FASTag is also vehicle specific and once affixed to a vehicle, it cannot be transferred to another vehicle (Bhavar et al., 2023).

While Fastag brings the convenience of cashless toll payments and reduced waiting time at toll gates, it is also prone to abuse. Some examples of factors that lead to fraud in toll transactions include: stolen or misused Fastags, data manipulation by internal or external parties, identity theft of road users and some transactions that look suspicious or unusual. Such examples not only result in losses for toll road operators, but also disrupt the fair balance of tariffs.

In the face of these challenges, it is necessary to take appropriate measures to prevent Fastag abuse and ensure the integrity of the toll payment system. By developing a classification model to detect fraudulent transactions and identify suspicious activities associated with Fastag usage, the model aims to accurately classify transactions as genuine or fraudulent, thus enabling the identification and prevention of fraudulent activities such as stolen or misused Fastags, data manipulation, identity theft, and other suspicious transactions. Evaluation metrics will be determined based on business objectives, with a focus on maximizing detection accuracy while minimizing false positives (transactions incorrectly classified as fraudulent by the model) to ensure the integrity of the toll payment system and maintain fair toll rates for all users.

The input for this problem is fastag fraud using several classification algorithms such as Logistic Regression, decision tree, KNN, SVM and random forest to predict the probability of customer fraud.

## *Related Work*

In the context of related research, some works that are relevant to this project are:

1. Anis et al. (2015), made a study entitled A Comparative Study of Decision Tree Algorithms for Class Imbalanced Learning in Credit Card Fraud Detection. This study aims to find the best distribution among the classifiers, to get insights of credit data by Random Under Sampling (RUS) along with feature selection and conclude about useful models that can measure the credit card fraud risk more efficiently. The similarity between this paper and the project to be carried out is that both focus on fraud detection using classification techniques. The difference is that this paper focuses on credit card transactions using the decision tree algorithm while the project to be carried out is related to toll payment transactions using fastag at the toll gate by comparing several classification algorithms such as Logistic Regression, Decision Tree, KNN, SVM and Random Forest.
2. Sundarkumar et al. (2015), made a study entitled One-Class Support Vector Machine based undersampling: Application to Churn Prediction and Insurance Fraud Detection. The focus of this paper is to propose a methodology that uses One Class Support Vector Machine (OCSVM) based undersampling for fraud detection, and to demonstrate its effectiveness by applying it to two different datasets, namely the car insurance fraud dataset and the credit card customer disconnect dataset. This research utilizes several classification techniques, including Decision Tree (DT), Support Vector Machine (SVM), Logistic Regression (LR), Probabilistic Neural Network (PNN), and Group Method of Data Handling (GMDH), to compare the performance of the proposed methodology with existing classification methods. The similarity between this paper and the project to be carried out is that both focus on fraud detection using classification techniques. The difference is the type of data used. The paper uses a car insurance fraud dataset and a credit card customer relationship breakup dataset, while the project will use toll payment transactions using fastag.
3. Sahin et al. (2011), made a study entitled Detecting Credit Card Fraud by Decision Trees and Support Vector Machines. The focus of this paper is to develop and compare decision tree-based classification models and support vector machines (SVM) in credit card fraud detection. The goal is to evaluate the relative performance of SVM and decision tree methods in identifying credit card fraud using real datasets. The similarity between this paper and the project to be carried out is that both have a focus on fraud detection using classification techniques while the difference lies in the dataset used.

## *Dataset and Features*

The dataset used comes from Fastag Fraud Detection (kaggle.com). The dataset is 5000 data and there are attributes that capture the main details about toll transactions, including: Vehicle Information (Type, Dimension, Plate Number), Payment Details (FastagID, Transaction Amount, Amount Paid), Toll Location and

Time (TollBoothID, Timestamp), Lane Usage (Lane Type), Vehicle Speed and Fraud Indicator. Details of these attributes can be seen in the following table.

| Feature | Description | Data Type |
|---|---|---|
| Transaction_ID | A unique identifier assigned to each toll transaction, used for tracking and referencing purposes. | int |
| Timestamp | The exact date and time when the toll transaction occurred, crucial for chronological analysis. | object |
| Vehicle_Type | The type of vehicle involved in the transaction. | object |
| FastagID | A unique identifier linked to a prepaid toll payment account, enabling electronic toll collection without manual cash. | object |
| TollBoothID | A unique identifier assigned to each tollbooth, used for pinpointing locations and analyzing traffic patterns. | object |
| Lane_Type | The specific lane used for the transaction. | object |
| Vehicle_Dimensions | The length, width, and height of the vehicle, potentially influencing toll calculations and safety measures. | object |
| Transaction_Amount | The total cost. | int |
| Amount_paid | The actual amount paid during the transaction, reflecting any discounts, penalties, or outstanding balances. | int |
| Geographical_Location | The precise latitude and longitude of the tollbooth, enabling mapping and spatial analysis. | object |
| Vehicle_Speed | The speed of the vehicle as it passed through the tollbooth, potentially used for safety monitoring and traffic management. | int |
| Vehicle_Plate_Number | The speed of the vehicle as it passed through the tollbooth, potentially used for safety monitoring and traffic management. | object |
| Fraud_indicator | A flag indicating potential fraudulent activity associated with the transaction, | object |

The dataset above is divided into 3 data, namely training data, validation data and testing data with a ratio of 60-20-20. The models used are Logistic Regression, Decision Tree, KNN, SVM and Random Forest. The 5 models are compared in order to evaluate the performance of each model and determine which model is most suitable for the purpose of data analysis. This aims to select a model that provides the most accurate and reliable prediction results in detecting fraudulent transactions related to the use of Fastag in the toll payment system. By conducting a comparison between models, it is expected to identify the strengths and weaknesses of each

model in handling complex datasets, thus enabling better decision-making in the implementation of solutions to detect and prevent Fastag abuse.

Before entering the modeling stage, EDA (Exploratory Data Analysis) is first carried out on the training data. This is useful for understanding the structure and characteristics of existing data by examining the distribution of variables, descriptive statistics and data visualization in order to identify patterns, anomalies or trends that may exist in the data, besides that EDA helps identify relationships between variables, evaluate the presence of missing values or the presence of outliers in the data. Furthermore, data preprocessing is carried out such as categorizing the numerical variables "Vehicle_Speed", "Amount_paid", and "Transaction_Amount". This aims to facilitate the analysis and grouping of data based on the existing value range.

Then, the Weight of Evidence (WOE) and Information Value (IV) calculations were carried out on the existing variables.

- Weight of Evidence (WOE)
  WOE is used to measure the strength of each attribute in each category. WOE calculated as follows :

$$\text{WOE} = [\ln\left(\frac{\%Bad_i}{\%Good_i}\right)] \times 100$$

To better understand WOE, we would like to stress the following:

- %Bad is not equivalent %Bad Rate. %Bad is the percentage of bad accounts in a score band over all bad counts. Same understanding applies to %Good in the example.
- It does not matter whether %Good or %Bad should be chosen as the numerator. If the two measures exchange their positions in the division, the sign of WOE for each score band will reverse while the magnitude will remain unchanged. There is no impact on the subsequent calculation for IV.
- Multiplication by 100 is optional (Alec Zhixiao Lin, 2014).

The value of WOE can be seen in the following figure.

| Fraud_indicator | Fraud | Not Fraud | All | p_good | p_bad | WOE | contribution |
|---|---|---|---|---|---|---|---|
| **Vehicle_Speed_bin** | | | | | | | |
| (9.999, 55.0] | 148 | 667 | 815 | 0.276763 | 0.250847 | 9.831430e-02 | 0.002548 |
| (55.0, 67.0] | 142 | 574 | 716 | 0.238174 | 0.240678 | -1.046131e-02 | 0.000026 |
| (67.0, 82.0] | 154 | 626 | 780 | 0.259751 | 0.261017 | -4.865552e-03 | 0.000006 |
| (82.0, 118.0] | 146 | 543 | 689 | 0.225311 | 0.247458 | -9.376083e-02 | 0.002076 |
| All | 590 | 2410 | 3000 | 1.000000 | 1.000000 | -9.999995e-07 | -0.000000 |

| Fraud_indicator | Fraud | Not Fraud | All | p_good | p_bad | WOE | contribution |
|---|---|---|---|---|---|---|---|
| **Amount_paid_bin** | | | | | | | |
| (-0.001, 90.0] | 282 | 668 | 950 | 0.277178 | 0.477966 | -5.448805e-01 | 0.109405 |
| (90.0, 120.0] | 247 | 505 | 752 | 0.209544 | 0.418644 | -6.920918e-01 | 0.144717 |
| (120.0, 160.0] | 52 | 517 | 569 | 0.214523 | 0.088136 | 8.895283e-01 | 0.112425 |
| (160.0, 350.0] | 9 | 720 | 729 | 0.298755 | 0.015254 | 2.974702e+00 | 0.843331 |
| All | 590 | 2410 | 3000 | 1.000000 | 1.000000 | -9.999995e-07 | -0.000000 |

| Fraud_indicator | Fraud | Not Fraud | All | p_good | p_bad | WOE | contribution |
|---|---|---|---|---|---|---|---|
| Transaction_Amount_bin | | | | | | | |
| (-0.001, 100.0] | 72 | 787 | 859 | 0.326556 | 0.122034 | 9.842944e-01 | 0.201310 |
| (100.0, 130.0] | 171 | 577 | 748 | 0.239419 | 0.289831 | -1.910842e-01 | 0.009633 |
| (130.0, 290.0] | 179 | 515 | 694 | 0.213693 | 0.303390 | -3.504817e-01 | 0.031437 |
| (290.0, 350.0] | 168 | 531 | 699 | 0.220332 | 0.284746 | -2.564650e-01 | 0.016520 |
| All | 590 | 2410 | 3000 | 1.000000 | 1.000000 | -9.999995e-07 | -0.000000 |

| Fraud_indicator | Fraud | Not Fraud | All | p_good | p_bad | WOE | contribution |
|---|---|---|---|---|---|---|---|
| Vehicle_Type | | | | | | | |
| Bus | 102 | 325 | 427 | 0.134855 | 0.172881 | -2.484129e-01 | 0.009446 |
| Car | 80 | 351 | 431 | 0.145643 | 0.135593 | 7.149272e-02 | 0.000718 |
| Motorcycle | 0 | 440 | 440 | 0.182573 | 0.000000 | 1.211490e+01 | 2.211850 |
| SUV | 108 | 321 | 429 | 0.133195 | 0.183051 | -3.179551e-01 | 0.015852 |
| Sedan | 84 | 331 | 415 | 0.137344 | 0.142373 | -3.596494e-02 | 0.000181 |
| Truck | 103 | 341 | 444 | 0.141494 | 0.174576 | -2.101117e-01 | 0.006951 |
| Van | 113 | 301 | 414 | 0.124896 | 0.191525 | -4.275423e-01 | 0.028487 |
| All | 590 | 2410 | 3000 | 1.000000 | 1.000000 | -9.999995e-07 | -0.000000 |

| Fraud_indicator | Fraud | Not Fraud | All | p_good | p_bad | WOE | contribution |
|---|---|---|---|---|---|---|---|
| TollBoothID | | | | | | | |
| A-101 | 165 | 683 | 848 | 0.283402 | 0.279661 | 1.328632e-02 | 0.000050 |
| B-102 | 222 | 622 | 844 | 0.258091 | 0.376271 | -3.769994e-01 | 0.044554 |
| C-103 | 203 | 665 | 868 | 0.275934 | 0.344068 | -2.206813e-01 | 0.015036 |
| D-104 | 0 | 25 | 25 | 0.010373 | 0.000000 | 9.247004e+00 | 0.095923 |
| D-105 | 0 | 66 | 66 | 0.027386 | 0.000000 | 1.021778e+01 | 0.279823 |
| D-106 | 0 | 349 | 349 | 0.144813 | 0.000000 | 1.188320e+01 | 1.720845 |
| All | 590 | 2410 | 3000 | 1.000000 | 1.000000 | -9.999995e-07 | -0.000000 |

| Fraud_indicator | Fraud | Not Fraud | All | p_good | p_bad | WOE | contribution |
|---|---|---|---|---|---|---|---|
| Lane_Type | | | | | | | |
| Express | 289 | 988 | 1277 | 0.409959 | 0.489831 | -1.780055e-01 | 0.014218 |
| Regular | 301 | 1422 | 1723 | 0.590041 | 0.510169 | 1.454479e-01 | 0.011617 |
| All | 590 | 2410 | 3000 | 1.000000 | 1.000000 | -9.999995e-07 | -0.000000 |

| Fraud_indicator | Fraud | Not Fraud | All | p_good | p_bad | WOE | contribution |
|---|---|---|---|---|---|---|---|
| Vehicle_Dimensions | | | | | | | |
| Large | 313 | 987 | 1300 | 0.409544 | 0.530508 | -2.587945e-01 | 0.031305 |
| Medium | 197 | 632 | 829 | 0.262241 | 0.333898 | -2.415768e-01 | 0.017311 |
| Small | 80 | 791 | 871 | 0.328216 | 0.135593 | 8.840045e-01 | 0.170279 |
| All | 590 | 2410 | 3000 | 1.000000 | 1.000000 | -9.999995e-07 | -0.000000 |

| Fraud_indicator | Fraud | Not Fraud | All | p_good | p_bad | WOE | contribution |
|---|---|---|---|---|---|---|---|
| Geographical_Location | | | | | | | |
| 12.84197701525119, 77.67547528176169 | 135 | 444 | 579 | 0.184232 | 0.228814 | -2.167141e-01 | 0.009661 |
| 12.936687032945434, 77.53113977439017 | 109 | 490 | 599 | 0.203320 | 0.184746 | 9.579261e-02 | 0.001779 |
| 13.042660878688794, 77.47580097259879 | 110 | 498 | 608 | 0.206639 | 0.186441 | 1.028549e-01 | 0.002077 |
| 13.059816123454882, 77.77068662374292 | 153 | 457 | 610 | 0.189627 | 0.259322 | -3.130179e-01 | 0.021816 |
| 13.21331620748757, 77.55413526894684 | 83 | 521 | 604 | 0.216183 | 0.140678 | 4.296428e-01 | 0.032440 |
| All | 590 | 2410 | 3000 | 1.000000 | 1.000000 | -9.999995e-07 | -0.000000 |

- Information Value (IV)
  IV is used to measure the strength of each characteristic. The IV value is obtained from the sum of the contribution IV of each attribute. The following is how IV is calculated :

$$IV = \sum_{i=1}^{n} \left( (\%Bad_i - \%Good_i) \times \ln\left(\frac{\%Bad_i}{\%Good_i}\right) \right)$$

  In such cases, we consider following rules of thumb suggested by Siddiqi for evaluating IV still applies to the binary dependent variable for the alternative IV:

  < 0.02: unpredictive
  0.02 to 0.1: weak
  0.1 to 0.3: medium
  0.3 to 0.5: strong
  > 0.5: suspicious

  The IV and strength values for each characteristic can be seen in the following table.

| Characteristic | Information Value | Strength |
|---|---|---|
| Vehicle_Speed_bin | 0.004657 | Unpredictive |
| Lane_Type | 0.025835 | Weak |
| Geographical_Location | 0.067774 | Weak |
| Vehicle_Dimensions | 0.218895 | Medium |
| Transaction_Amount_bin | 0.258900 | Medium |
| Amount_paid_bin | 1.209878 | Strong |
| TollBoothID | 2.156231 | Strong |
| Vehicle_Type | 2.273485 | Strong |

The "Vehicle_Speed" column was removed due to its unpredictable power, leaving the variables "Vehicle_Type", "TollBoothID", "Geographical_Location", "Vehicle_Dimensions", "Lane_Type", "Amount_paid", and "Transaction_Amount". Furthermore, categorical data is processed using the One-Hot Encoding (OHE) method which is useful for converting categorical data into a form that can be processed by machine learning algorithms. OHE converts categorical variables into binary vectors that have a value of 0 or 1, where each binary number represents the presence or absence of that category. Next, it transforms the categories in the numeric values into sorted values. This process helps to simplify the input data and reduce the dimensionality of the feature space making it easier for the algorithm to process and learn from the data. After that, categorical and numerical data is merged and the data is ready for modeling.

## *Methods*

In this project, classification techniques are performed by comparing the performance of several algorithms such as Logistic Regression, Decision Tree, KNN, SVM and Random Forest.

1. Logistic Regression

   Logistic Regression is similar to OLS regression but is specifically designed to deal with a dichotomous dependent variable such as the one with which we are concerned (Aldrich, 1984; Liao, 1994; Menard, 1995; Mendenhall, 1996). In this project, the dependent variable is the fraud indicator which has a value of fraud and not fraud. Logistic Regression is a type of generalized linear model. Simple linear regression is not suitable when the variable to be predicted is binary. The vector *α = (α0, α1, α2, . . . , αn)* represents the coefficients, *X = (1, X1, X2, . . . , Xn)* the exploratory variables, and the model's error. The linear model is defined as follows*: Y = α0 + α1X1 + α2X2 + · · · + αnXn + = Xα + In* logistics regression, a logit link function *g* over [0, 1] in *R* is introduced, to force the linear combination of the variables to take values between 0 and 1: *g(p) = Xα*, where *p* is the probability of fraud risk that we are estimating (Makki, et al., 2016). The logit function is defined as:

$$g(p) = \ln \frac{p}{1-p} \quad \text{with} \quad p = \frac{e^{X\alpha}}{1+e^{X\alpha}}$$

2. Decision Tree

   A decision tree is a tree structure which attempts to separate the given records into mutually exclusive subgroups. To do this, starting from the root node, each node is split into child nodes in a binary or a multi split fashion related to the method used based on the value of the attribute (input variable) which separates the given records best. Records in a node is recursively separated into child nodes until there is no split that makes statistical difference on the distribution of the records in the node or the number of records in a node is too small. Each decision tree method uses its own splitting algorithms and splitting metrics. From the well-known Decision Tree algorithms, ID3, C5.0 and C&RT methods use impurity measures to choose the splitting attribute and the split values (Sahin, 2011).

3. KNN

   In this algorithm, test sample belongs to the class which has most votes in the k-nearest neighbor. One of the problems of this algorithm is that K should be set by user. For solving this problem K should be set using a formula to reach the highest possible accuracy. One of the features of these algorithms is that monitor database only once while other algorithms monitor database several times

(Khodabakhshi, M. 2016). KNN consists of the K nearest points to the one that we aim to predict. A specific norm is used to measure the distance between points. The new observation is assigned the class with the majority of the K nearest points. The norm usually used to measure the distance between two observations p and q (an observation is $\in \mathbb{R}^n$) is the euclidean distance given by (Makki et al., 2016):

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

4. SVM

SVM tries to find a hyperplane to separate the two classes while minimizing the classification error. SVM learns a separating hyperplane which maximizes the margin and produces good generalization ability. SVM's basic idea is to transform the attributes to a higher dimensional feature space and find the optimal hyperplane in that space that maximizes the margin between the classes. Briefly, SVM does this by using a polynomial, sigmoid, radial basis or a linear kernel function which satisfies the Mercer condition. The kernel function reflects the geometric relationship between the input record and the support vector, or the similarities of the features of the classes. SVM is a classification tool that aims at finding the hyperplane that separates data points in two classes optimally. Formally, a given training vector *xi* in *R n, i = 1, . . . , l, n* is the number of exploratory variables, and *l* is the number of observations in the train set. *y ∈ R l* taking the values of 1 and -1. The binary classification is done by solving the following optimization problem.

$$\text{minimize} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{l}\zeta_i$$

$$\text{subject to} \quad \begin{cases} y_i \times (w^T\Phi(x_i) + b) \geq 1 - \zeta_i \\ \zeta_i \geq 0, i = 1, \ldots, l \end{cases}$$

The hyperplane equation is defined as: $w^T\Phi(x_i)+b$, where w is a vector of weights, $\Phi(x_i)$A maps $x_i$ into a higher dimensional space. Slack variables $\zeta_i$are added, to allow for some errors or miscalculations, in case these points are not linearly separable. *C* is a cost parameter > 0 associated with these errors. The aim of minimizing $\frac{1}{2}||w||^2$is to maximize the distance between the two margins that is

equal to $\frac{2}{||w||^2}$ in order to find the hyperplane that best separates the two classes (Makki et al., 2016).

5. Random Forest

   Random Forest is an algorithm for classification and regression. It is a collection of decision tree classifiers. Random Forest adds additional randomness to the model. Rather than searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that results in a better model. Random forest has precedence over decision tree as it corrects the habit of over fitting to their training set. A subset of the training set is sampled randomly so that to train each single tree and then a decision tree is built, each node then splits on a feature selected from a random subset of the full feature set. Even for large data sets with many features and data instances training is extremely fast in random forest and because each tree is trained independently of the others. By using Random Forest algorithm the generalization error and over fitting were achieved (Niveditha, 2019).

## *Experiments/Results/Discussion*

In the experiment process, hyperparameters are used to control the training process and affect the performance of the resulting model. In this project, hyperparameters are performed on each model:

1. Baseline

   The baseline is used as a comparison or reference point to evaluate the performance of more complex models. The baseline function is commonly used in the context of predictive modeling to provide an initial understanding of how well the model being developed compares to a very simple model. Baselines often use very simple methods such as predicting the majority class in a dataset. The baseline hyperparameter used is 'strategy'. The available strategy is 'most_frequent' which will always predict the most frequent class in the training data.

2. Logistic Regression

   In Logistic Regression there are no hyperparameters mentioned. However, the Logistic Regression model has several hyperparameters such as 'penalty', 'C', 'solver' and others that can be set to control the complexity of the model.

3. SVM

   In SVM the best hyperparameter C has a value of 0.5. C is the penalty parameter for errors in the penalty function. While the kernel hyperparameter is linear. Kernel

is a kernel function used to transform the original feature space into a higher dimensional space.

4. Decision Tree
   In the decision tree the best hyperparameter max_depth is 10. Max_depth is the maximum depth of the decision tree.
5. Random Forest
   In random forest the best hyperparameter n_estimators is 300. N_estimators is the number of decision trees to be built in the ensemble. The larger the value of n_estimators the stronger the ensemble but also requires more time to train the model.
6. KNN
   In KNN the best hyperparameter n_neighbors is 10. N_neighbors is the number of nearest neighbors that will be used by the model to make predictions. The greater the value of n_neighbors the more complex the model.

In model performance evaluation, AUC (area under the Receiver Operating Characteristic Curve) is often used as a metric to measure how well the model can distinguish between positive and negative classes. The higher the AUC value, the better the model can distinguish between legitimate and fraudulent transactions. A high AUC value indicates that the model has a good ability to identify patterns that indicate fraud, thereby reducing the risk of loss due to fraud.

Fraud detection models that have a high AUC tend to generate fewer false positives and false negatives, meaning that fewer legitimate transactions are misclassified as fraudulent, and vice versa. This helps in optimizing the use of resources and reduces the need for manual investigation of misclassified transactions.

The performance of fraud detection models as measured by AUC can also affect a business's reputation. Businesses that are known to have safe and reliable payment systems tend to gain the trust of the public and stakeholders, while businesses with chronic fraud detection issues may experience a decline in reputation. The results of the 6 models evaluated using the AUC metric on the training and test data can be seen in the following table

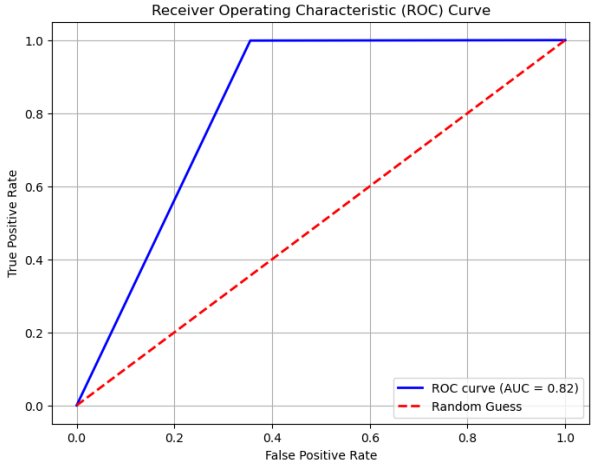
| model | AUC train | AUC test | Best param |
|---|---|---|---|
| baseline | 0.500000 | 0.500000 | {'strategy': 'most_frequent'} |
| logistic regression | 0.928686 | 0.921447 | {} |
| svm | 0.866842 | 0.880167 | {'C': 0.5, 'kernel': 'linear'} |
| decision tree | 0.924439 | 0.937875 | {'max_depth': 10} |
| random forest | 0.951643 | 0.936407 | {'n_estimators': 300} |
| knn | 0.879245 | 0.907910 | {'n_neighbors': 10} |

In the baseline model, the model performance is very low with an AUC of 0.5 on both training and test data, indicating that the model has no ability to distinguish between positive and negative classes. Furthermore, the Logistic Regression model achieved an AUC of around 0.93 on the training data and 0.92 on the test data without any additional hyperparameters. This model showed good and stable performance in both data sets. The SVM model has a value of C = 0.5 and has a linear kernel value, this model achieved an AUC of 0.86 on the training data and 0.88 on the test data. Then, the decision tree model has a maximum depth of 10 and achieves an AUC of around 0.92 in the training data and 0.94 in the test data. Furthermore, the random forest model with 300 estimators showed excellent performance with an AUC of around 0.95 in the training data and 0.94 in the test data. Finally, the k-Nearest Neighbors (KNN) model with 10 neighbors achieved an AUC of about 0.88 in the training data and 0.91 in the test data.

Furthermore, after obtaining the best hyperparameters using the training data. The next step is to apply it to the validation data to evaluate the performance of the model independently. By applying the best hyperparameters that have been selected by the model, we can retrain the model using all valid data. The results of AUC on valid data using the best hyperparameters can be seen in the following table.

| Model | AUC Score |
|---|---|
| Logistic Regression | 0.811 |
| SVM | 0.811 |
| KNN | 0.785 |
| Decision Tree | 0.831 |
| Random Forest | 0.828 |

Based on the performance evaluation with the AUC metric on the validation data, the best model is the decision tree model. This model achieves an AUC of around 0.831. Therefore, the decision tree model is the best choice in the context of this evaluation. Furthermore, the decision tree model is used on the test data where the AUC score obtained is 0.821 which shows how well the model can separate the positive and negative classes.



The Receiver Operating Characteristic (ROC) Curve graph above illustrates the performance of the model in distinguishing between positive (fraud) and negative (non-fraud) classes by comparing the True Positive Rate (TPR) with the False Positive Rate (FPR) at different threshold values.

The X-axis shows the False Positive Rate (FPR), which is the proportion of the non-fraud class that is incorrectly predicted as fraud by the model. Meanwhile, the Y-axis shows the True Positive Rate (TPR), which is the proportion of fraud classes that are correctly predicted by the model.

An ideal ROC curve would be close to the upper left corner (0.1) of the graph, indicating that the model has a high TPR and low FPR rate, thus having an excellent ability in separating positive and negative classes. In addition to the ROC curve (blue line), the red dashed line shows the reference for random prediction or "Random Guess", which gives an AUC area of 0.5.

From the graph above, the AUC score of 0.821 shows that the decision tree model has good performance in distinguishing between fraudulent and non-fraudulent transactions. For the confusion matrix value on the test data can be seen in the following figure:

| Fraud_indicator | Fraud | Not Fraud | All |
|---|---|---|---|
| row_0 | | | |
| Fraud | 127 | 1 | 128 |
| Not Fraud | 70 | 802 | 872 |
| All | 197 | 803 | 1000 |

Based on the crosstabulation results above, there are 197 transactions predicted as fraud by the model. Of these, 127 transactions are actually fraud, while 70 transactions are predicted as fraud but are not. On the other hand, there are 803 transactions predicted as non-fraud by the model. Of these, 802 transactions are actually non-fraud, while only 1 transaction is actually fraud but predicted as non-fraud.

From the entire test data (1000 transactions), the model successfully identified 128 fraud transactions and 872 non-fraud transactions correctly. However, there were 71 misclassified transactions, of which 70 were false positives (non-fraud transactions incorrectly predicted as fraud) and 1 false negative (fraud transactions incorrectly predicted as non-fraud).

## *Conclusion*

Based on the evaluation of model performance on training data, validation data, and test data, it can be concluded that the decision tree model shows the best performance in distinguishing between fraudulent and non-fraudulent transactions. By

achieving an AUC of 0.831 on validation data and 0.821 on test data, the decision tree model has a strong ability to separate positive and negative classes with good accuracy. Furthermore, to improve the performance of Fastag fraud detection, it is recommended to consider using algorithms such as gradient boosting or XGBoost. These algorithms are known for their ability to overcome class imbalance and model complex relationships between transaction features. By utilizing the power of these algorithms, we can improve the model's ability to detect more complicated and unexpected fraud patterns.

## *References*


Alec Zhixiao Lin, PayPal Credit, Timonium, MD. 2014. Expanding the Use of Weight of Evidence and Information Value to Continuous Dependent Variables for Variable Reduction and Scorecard Development. Dept. of Mathematics & Statistics, Univ of Maryland, Baltimore, MD.

G. Ganesh Sundarkumar. Vadlamani Ravi. V. Siddeshwar. 2015. One-Class Support Vector Machine based undersampling: Application to Churn prediction and Insurance Fraud detection. IEEE International Conference on Computational Intelligence and Computing Research.

G. Niveditha, K. Abarna, G. V. Akshaya. 2019. Credit Card Fraud Detection Using Random Forest Algorithm. International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT).

Maria Anis, Mohsin Ali, Amit Yadav. 2015. A Comparative Study Of Decision Tree Algorithms For Class Imbalanced Learning In Credit Card Fraud Detection. International Journal of Economics, Commerce & Management Vol. III, Issue 12, December 2015 Licensed.

Masoud Khodabakhshi. Mehdi Fartash. 2016. Fraud Detection in Banking Using KNN (K-Nearest Neighbor) Algorithm. 5th international conference on research in science and technology London-united kingdom.

Sara Makki, Zainab Assaghir, Yehia Taher, Rafiqul Haque, Mohand-Saïd Hacid, Hassan Zeineddine. 2016. An Experimental Study with Imbalanced Classification Approaches for Credit Card Fraud Detection. IEEE Access.

Vipul Bhavar, Vishakha Borkar, Tanvi Hirave, Harsh Vishwanathan, Ranjita Asati. 2023. Fastag Fraud Detection : A Literature Survey. International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 11 Issue IV Apr 2023.

Y. Sahin and E. Duman. 2011. Detecting Credit Card Fraud by Decision Trees and Support Vector Machines. Proceedings of the International MultiConference of Engineers and Computer Scientists 2011 Vol I, IMECS 2011, March 16 - 18, 2011, Hong Kong.