PROJECT REPORT

ON

Fraud Detection Framework for Bank Account Applications

Carried Out at

UNDER THE SUPERVISION OF

Mr. ABHAY MANE

C-DAC-ACTS, Bangalore

Submitted By

Shaumit Kumar (220950125083)

Vishal Mishra (220950125112)

Saurabh Shinde (220950125077)

Sayali Mane (220950125078)

Shreya Saini (220950125089)

PG DIPLOMA IN BIG DATA ANALYTICS

C-DAC-ACTS, BANGALORE

CANDIDATE's DECLARATION

We hereby certify that the work being presented in the report entitled **Fraud Detection Framework for Bank Account Applications**, in partial fulfillment of the requirements for the award of PG Diploma Certificate and submitted in the department of PG-DBDA of the C-DAC Bangalore, is an authentic record of our work carried out during the period, 1st February 2023 to 14th March 2023 under the supervision of **Mr. Abhay Mane**, C-DAC Bangalore. The matter presented in the report has not been submitted by us for the award of any degree of this or any other Institute/University.

(Name and Signature of Candidates)

Shaumit Kumar (220950125083)

Vishal Mishra (220950125112)

Saurabh Shinde (220950125077)

Sayali Mane (220950125078)

Shreya Saini (220950125089)

Counter Signed by

ACKNOWLEDGMENT

We take this opportunity to express our gratitude to all those people who have been directly and indirectly with us during the competition of this project.

We pay thanks to Mr. Abhay Mane who has given us guidance and light to us during this major project. His versatile knowledge of "**Fraud Detection Framework for Bank Account Applications**" has eased us in critical times during the span of this Final Project.

We acknowledge here our debt to those who contributed significantly to one or more steps. We take full responsibility for any remaining sins of omission and commission.

From

Whole Team

CERTIFICATE

This is to certify that the work titled **Fraud Detection Framework for Bank Account Applications** is carried out by Shaumit Kumar (220950125083), Vishal Mishra (220950125112), Saurabh Shinde (220950125077), Sayali Mane (220950125078), Shreya Saini (220950125089) the bonafide students of Diploma in Big Data Analytics from Centre for Development of Advanced Computing, Electronic City, Bangalore from 1st February 2023 - 14 March 2023. The Course End Project work is carried out under my direct supervision and is 80% completed.

Mr. Abhay Mane

Data Scientist,

C-DAC, R & D,

Electronic City, Bangalore - 560100, India

ABSTRACT

The Fraud Detection Framework for Bank Account Applications project aims to develop a machine learning-based framework to detect fraudulent activities in bank account applications. The framework employs various data pre-processing techniques, feature selection methods, and classification algorithms to identify potentially fraudulent applications. The project utilizes a publicly available dataset of bank account applications and uses various metrics such as accuracy, precision, recall, and F1 score to evaluate the performance of the framework. The proposed framework can assist financial institutions in detecting fraudulent activities and preventing financial losses.

The proposed framework has the potential to assist financial institutions in detecting fraudulent activities and preventing financial losses. It can also provide valuable insights into the patterns and characteristics of fraudulent applications, which can inform the development of more effective fraud detection systems. Overall, the project has significant implications for improving the security and reliability of bank account applications and financial transactions.

We have developed an adaptive model based on the dataset published at NeurIPS2022 which is a Base dataset containing 1M instances based on a real-world dataset. We have 11029 fraud counts against 988971 non-fraudulent counts. Since the dataset is highly imbalanced, it possessed a great challenge for data preprocessing, data modeling, feature extraction, analysis, and analytics. We have used machine learning algorithms and deep learning algorithms such as Random Forest, Support Vector Machine, and ANN to build fraud detection models. When constructing a fraudulent bank account detection model, it is very important to extract the right features from transactional data.

We have used different undersampling methods in order to get the correct model for our problem statement. We are currently working on deployment and further modifications if required.

Table Of Contents

Table of Figures

Table of Tables

1. INTRODUCTION

Bank account fraud is a serious issue that can cause significant financial loss to both individuals and financial institutions. Fraudsters often create fake bank accounts using stolen or fabricated identities to carry out fraudulent activities such as money laundering, embezzlement, and identity theft. As a result, there is a growing need for robust fraud detection systems that can identify and prevent such activities in real time.

The project will explore various techniques and algorithms that can be used for fraud detection in the finance industry. We will analyze large volumes of data to identify patterns and trends that can be used to develop an effective fraud detection system. The project will demonstrate the importance of data analytics and machine learning in the finance industry and highlight the potential for technology to mitigate the risks associated with fraudulent bank accounts.

By building an efficient and accurate fraud detection system, banks can protect themselves from financial losses and maintain their reputation as trusted financial institutions. The project will also provide a powerful tool to protect individuals from financial losses caused by fraudulent bank accounts.

2. PROBLEM STATEMENT

The goal of this project is to develop a machine-learning model that can accurately detect fraudulent bank accounts that are created. In recent times, there is a wide variety of Bank fraudulent transactions, and there are fewer systems available to verify such transactions. So, in order to check whether an application from a customer is legitimate or false, we have developed a model to verify the applicant's details.

To address this challenge, we will develop a machine-learning model that can analyze various features such as personal information, transaction history, and behavior patterns to identify potentially fraudulent bank accounts. The model should be able to detect fraudulent bank accounts with high accuracy while minimizing false positives and false negatives.

The success of this project will demonstrate the potential for data analytics and machine learning to mitigate the risks associated with fraudulent bank accounts in the finance industry. It will also provide financial institutions with a powerful tool to protect their customers from financial losses and maintain their reputation as trusted financial institutions.

3. SOFTWARE REQUIREMENT

Jupyter-Lab: It is an open-source web-based tool that provides a flexible and powerful environment for working with data, code, and visualization.

Some of the features of JupyterLab are:

- Multi-Language Support: JupyterLab supports multiple programming languages, including Python, R, Julia, and others.
- Notebook Organization: JupyterLab provides a workspace where you can organize your notebooks, code files, and data files.
- Code Editor: JupyterLab comes with a powerful code editor that provides syntax highlighting, auto-completion, and other advanced features.
- File Browser: JupyterLab provides a file browser that allows you to navigate your file system and open files directly in the IDE.

Google-Colab: It is a cloud-based platform provided by Google that allows users to write, run, and share Python code using a Jupyter notebook interface. It provides free access to computing resources such as CPU, GPU, and TPU, and also supports collaborative work among users.

Due to limitations in our systems, we have utilized Google Colab where we have used both GPU and TPU as per our requirement in our code. It also allowed us easily share our work with each other and collaborate in real-time, which was beneficial for the team project.

.

4. LITERATURE SURVEY

Machine Learning Models / Classifier / Algorithms

i) Random Forest Classifier:

As the name suggests, a random forest consists of a large number of independent decision trees as a set. Each tree in the random forest will produce a category prediction, and the category with the most votes becomes the prediction of our model. The basic concept of random forest is simple but powerful: the wisdom of the crowd. From a data science perspective, the reason why the random forest model is so effective is that there are a large number of relatively unrelated models (trees). The random forest has an advantage over the decision tree as it corrects the habit of overfitting to its training set. A subset of the training set is sampled randomly to train each individual 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 datasets with many features and data instances training is extremely fast in the random forest.

ii) SVM:

Support Vector Machine is a supervised learning method, most widely used in statistical classification and regression analysis. It is also the focus of this project. Support Vector machines belong to a family of generalized linear classifiers which are characterized by their ability to both minimize empirical errors and maximize geometric edge regions. Hence support vector machines are also known as maximum edge region classifiers. The core principle of the support vector machine is: mapping the vectors into a higher dimensional space where a maximum spacing hyperplane is established. Two parallel hyperplanes are built on either side of the hyperplane that separates the data. Also, the separated hyperplanes maximize the distance between the two parallel hyperplanes. It is assumed that the greater the space or gap between the parallel hyperplanes, the smaller the total error of the classifier. In this project, SVM is the supervised learning algorithm used to solve multi-class classification.

iii) Gradient Boosting:

Gradient Boosting is a popular boosting algorithm. The main idea behind this algorithm is to build models sequentially. In gradient boosting each predictor corrects its predecessor’s error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of the predecessor as labels.

A gradient-boosting classifier is used when the target column is binary. There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

iv) XGB:

XGBoost (Extreme Gradient Boosting) is a popular gradient-boosting algorithm used for supervised learning problems. It is widely used for solving problems in structured or tabular data, including classification, regression, and ranking tasks. Gradient boosting is an ensemble learning technique that combines several weak learners (usually decision trees) to create a strong learner. The algorithm builds models in a stage-wise fashion, where each new model tries to correct the errors made by the previous models. XGBoost differs from traditional gradient boosting methods in that it regularizes the models and applies parallel processing, making it faster and more scalable.

v) Logistic Regression Classifier:

Logistic regression is the classical and the best bicategorical algorithm which is preferred when dealing with classification problems, especially bicategorical ones. The choice of algorithm is based on the principle of simplicity before complexity. A multinomial logistic regression algorithm can regenerate the model. It will be a better classification algorithm when the target field or data is a set field with two or more possible values. It is also more straightforward for any beginner to understand and directly see the weights of each feature. Then it is easier to update the model and incorporate new data for different problems.

Furthermore, it has a disadvantage. There is a limit to the data and the adaptability of the scene. Not as adaptable as the decision tree algorithm. But this is an issue that we can also determine in this project based on the actual situation whether the logistic regression has a better ability to adapt to an extensive data set of credit card transactions.

vi) Decision Tree:

The use of a decision tree is usually based on the known probability of various scenarios, and the decision tree is formed to find the possibility that the expected net present value is greater than or equal to zero to evaluate the risk of the training project. Also, it judges the feasibility of the decision analysis method. Then we know that because this decision branch is drawn as a graph much like the trunk of a tree, we name it a decision tree.

Decision trees are a primary classification and regression method, and learning typically involves three steps: feature selection, decision tree generation, and decision tree pruning.

In machine learning, a decision tree is a predictive model that represents a mapping between object properties and object values.A classification tree (decision tree) is a very commonly used classification method. Similar to the dataset classification problem mentioned in this paper, the decision tree is a technique that is often used to analyze data and can also be used to make predictions. That is why we chose it for the training of the fraud detection system.

As the prediction of the score is a much more important task according to our model therefore we are predicting the score on the basis of the given formula:

$$\textbf{Score} = 0.5 * TP + 0.5 * Deviation$$

Where TP is the True Positive value and Deviation is the deviation of outlier data from the standard data point. On the basis of this score, we made two classes 0 and 1. If the score is 1 it will move to class 1 and be termed as a legal transaction and if the score is 0 it will move to class 0 and be termed as a fraudulent transaction. At last, the accuracy is calculated on the basis of how many fraudulent transactions are there in our dataset and how many we predicted with the help of our model.

vii)Naive Bayes:

Naive Bayes is a probabilistic classification algorithm that is based on the Bayes theorem, which describes the probability of an event based on prior knowledge of conditions that might be related to that event. Naive Bayes assumes that the features used to classify the data are independent of each other, hence the name "naive".The algorithm works by calculating the probability of each class given the input features, using the Bayes theorem. It then selects the class with the highest probability as the predicted class. To calculate the probability of each class, Naive Bayes uses the frequency of each feature in the training set, and the conditional probability of each feature given the class.

Naive Bayes is often used in text classification tasks, such as spam filtering or sentiment analysis, where the input features are typically words or phrases. It is also commonly used in other classification tasks, such as image recognition or fraud detection.

vii)ANN: An artificial neural network (ANN) is a type of machine learning model that can be used for bank account fraud detection. ANN models are designed to simulate the behavior of the human brain by using layers of interconnected neurons to process data. In the case of bank account fraud detection, an ANN model can learn to identify patterns and anomalies in credit card transaction data to detect fraudulent activity. It can be effective for bank account fraud detection because they are able to learn complex patterns in large datasets, even when those patterns are not immediately obvious to human analysts. However, they can also be prone to overfitting or underfitting if not properly tuned, and may require significant computational resources to train & evaluate.

5. PROCESS FLOW DIAGRAM

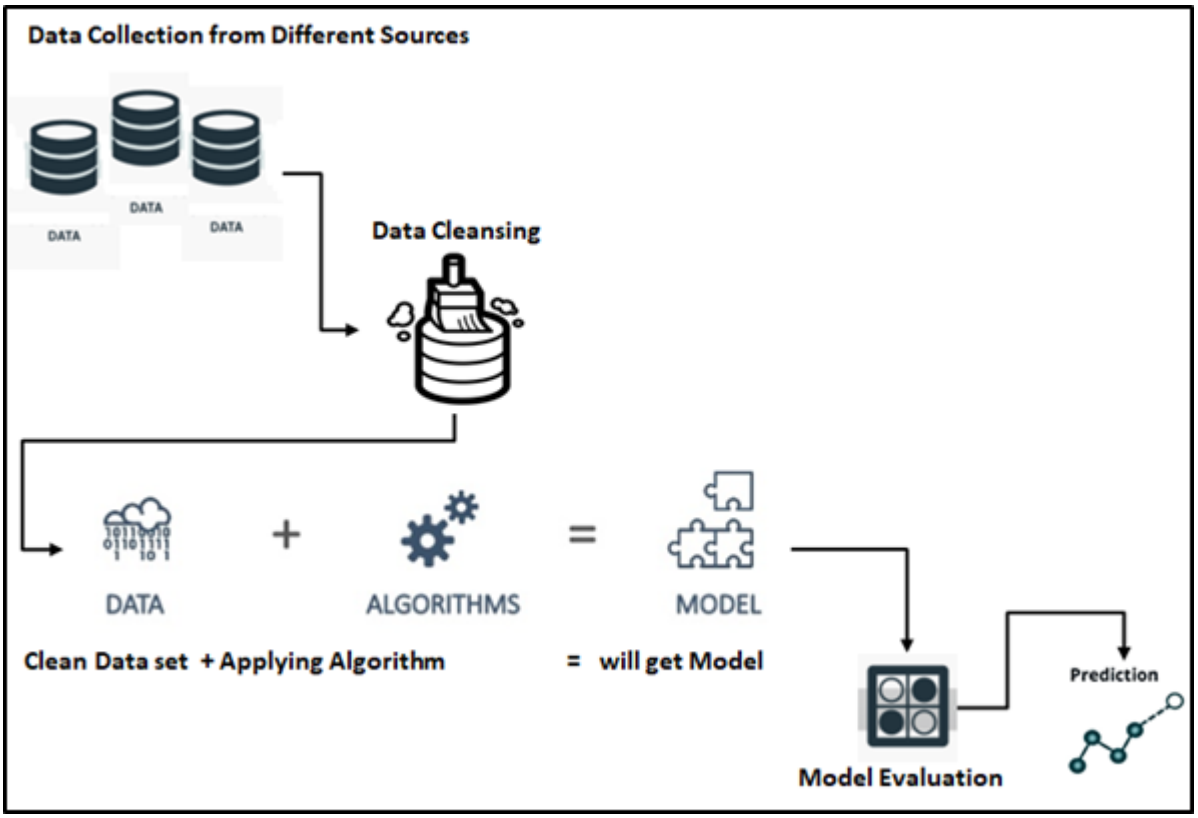


Fig. 1 Process Flow Diagram

1. **Data Collection:** Collect relevant data from various sources and store it in a suitable data storage system.
2. **Data Pre-processing:** Clean the data, eliminate irrelevant or missing values, and transform it into a suitable format for modeling.
3. **Model Selection:** Choose an appropriate machine learning algorithm that fits the problem, data, and requirements of the project.
4. **Training and Validation:** Train the selected machine learning algorithm using the prepared data. The validation set is then used to evaluate the performance of the trained model on unseen data.
5. **Evaluation:** Evaluate the performance of the model using appropriate metrics and testing data.
6. **Final Model:** The final model refers to the model that has been selected as the best candidate from a set of candidate models based on some evaluation metric or criteria.

6. DATA PRE-PROCESSING & PREPARATION

The dataset was created by Sérgio Jesus, José Pombal, Duarte Alves, André F. Cruz, Pedro Saleiro, and Pedro Bizarro on behalf of Feedzai. The Bank Account Fraud (BAF) suite of datasets has been published at NeurIPS 2022 and it comprises a total of 6 different synthetic bank account fraud tabular datasets.

BAF is a realistic, complete, and robust test bed to evaluate novel and existing methods in ML and fair ML, and the first of its kind. The label is contained in the fraud_bool field.

- Realistic data, based on a present-day real-world dataset for fraud detection.
- Biased data, each dataset has distinct controlled types of bias.
- Imbalanced data, this setting presents an extremely low prevalence of positive class.
- Dynamic data, with temporal data and observed distribution shifts.

Privacy-preserving, to protect the identity of potential applicants we have applied differential privacy techniques (noise addition), and feature encoding and trained a generative model.

The BAF suite comprises six datasets that were generated from a real-world online bank account opening fraud detection dataset. This is a relevant application for Fair ML, as model predictions result in either granting or denying financial services to individuals, which can exacerbate existing social inequities

Data Description: Each instance is a synthetic feature-engineered bank account application with the following fields:

1. income (numeric): Annual income of the applicant (in decile form). Ranges between [0.1, 0.9].
2. name_email_similarity (numeric): Metric of similarity between email and applicant's name. Higher values represent higher similarity. Ranges between [0, 1].
3. prev_address_months_count (numeric): Number of months in the previously registered address of the applicant, i.e. the applicant's previous residence, if applicable. Ranges between [−1, 380] months (-1 is a missing value).
4. current_address_months_count (numeric): Months in currently the registered address of the applicant. Ranges between [−1, 429] months (-1 is a missing value).

5. customer_age (numeric): Applicant's age in years, rounded to the decade. Ranges between [10, 90] years.
6. days_since_request (numeric): Number of days that passed since the application was done. Ranges between [0, 79] days.
7. intended_balcon_amount (numeric): Initial transferred amount for application. Ranges between [−16, 114].
8. payment_type (categorical): Credit payment plan type. 5 possible (anonymized) values.
9. zip_count_4w (numeric): Number of applications within the same zip code in the last 4 weeks. Ranges between [1, 6830].
10. velocity_6h (numeric): Velocity of total applications made in the last 6 hours i.e., the average number of applications per hour in the last 6 hours. Ranges between [−175, 16818].
11. velocity_24h (numeric): Velocity of total applications made in the last 24 hours i.e., the average
12. number of applications per hour in the last 24 hours. Ranges between [1297, 9586]
13. velocity_4w (numeric): Velocity of total applications made in the last 4 weeks, i.e., the average number of applications per hour in the last 4 weeks. Ranges between [2825, 7020].
14. bank_branch_count_8w (numeric): Number of total applications in the selected bank branch in the last 8 weeks. Ranges between [0, 2404].
15. date_of_birth_distinct_emails_4w (numeric): Number of emails for applicants with the same date of birth in the last 4 weeks. Ranges between [0, 39].
16. employment_status (categorical): Employment status of the applicant. 7 possible (anonymized) values.
17. credit_risk_score (numeric): Internal score of application risk. Ranges between [−191, 389].
18. email_is_free (binary): Domain of application email (either free or paid).

19. housing_status (categorical): Current residential status for the applicant. 7 possible (anonymized) values.
20. phone_home_valid (binary): Validity of provided home phone.
21. bank_months_count (numeric): How old is the previous account (if held) in month. .Ranges between [−1, 32] months (-1 is a missing value).
22. has_other_cards (binary): If the applicant has other cards from the same banking company.
23. proposed_credit_limit (numeric): Applicant's proposed credit limit. Ranges between [200, 2000].
24. source (categorical): Online source of the application. Either browser (INTERNET) or app (TELEAPP).
25. foreign_request (binary): If the origin country of request is different from the bank's country.
26. session_length_in_minutes (numeric): Length of a user session in a banking website in minutes. Ranges between [−1, 107] minutes.
27. device_os (categorical): Operative system of the device that made the request. Possible values are Windows, macOS, Linux, X11, or others.
28. keep_alive_session (binary): User option on session logout.
29. device_distinct_emails (numeric): Number of distinct emails in the banking website from the user device in the last 8 weeks. Ranges between [−1, 2].
30. device_fraud_count (numeric): Number of fraudulent applications with used devices. Ranges between [0, 1].
31. month (numeric): The month where the application was made. Ranges between [0, 7].
32. fraud_bool (binary): If the application is fraudulent or not.

Data preprocessing:

It is an important step in the data analysis process, as it involves preparing the data for analysis by cleaning, transforming, and reducing it. The following are some of the steps involved in data preprocessing:

A. Data collection: This involves gathering data from various sources, such as databases, APIs, or web scraping.
B. Data cleaning: This step involves identifying and correcting errors or inconsistencies in the data, such as missing values, duplicate entries, or incorrect data types.
C. Data transformation: This involves converting the data into a format that is suitable for analysis, such as scaling, normalization, or encoding categorical variables.
D. Data reduction: This involves reducing the dimensionality of the data by selecting only the relevant features or using techniques such as principal component analysis (PCA) to extract the most important components.
E. Handling outliers: Outliers are data points that are significantly different from the other data points in the dataset. They can be handled by removing them, transforming them, or treating them as a separate category.
F. Handling missing data: Missing data can be handled by imputation, which involves filling in the missing values using various techniques such as mean imputation.
G. Feature engineering: This involves creating new features from the existing features, such as combining features, creating dummy variables for categorical variables, or extracting text features.

The following steps were followed for data pre-processing:

a) We have imported the necessary packages and libraries.

b)We have checked for unique values in each column to find categorical columns.

c) Null Value treatment in the dataset. In the dataset null values have been filled with -1.

d) Replace null values with the mean of that column.

e) We have dropped columns:

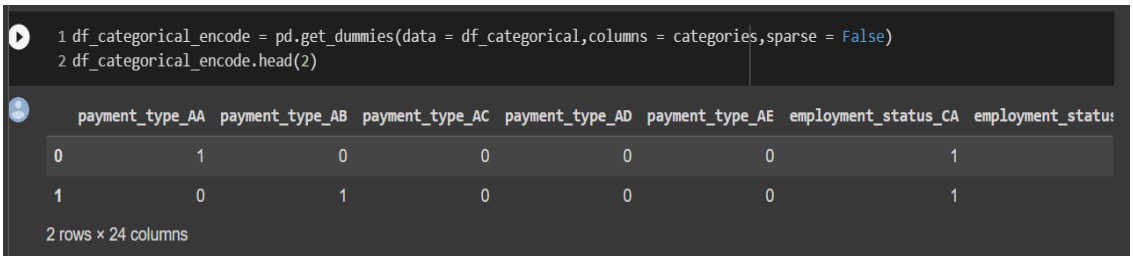
i)‘prev_address_month_count’ column as it is having more than 70% of null values.

ii)‘foreign_request’ column as it has imbalanced data which includes: 0 = 9,74,758 and 1 = 25,242.

iii)‘device_fraud_count’ column because of having only one quantity.

iv) ‘source’ column due to having imbalanced data: INTERNET = 99% and TELEAPP = 1%

f) Separating categorical columns from the dataset by using One Hot Encoding using pandas function ‘get_dummies’.



```
1 df_categorical_encode = pd.get_dummies(data = df_categorical,columns = categories,sparse = False)
2 df_categorical_encode.head(2)
```

	payment_type_AA	payment_type_AB	payment_type_AC	payment_type_AD	payment_type_AE	employment_status_CA	employment_statu
0	1	0	0	0	0	1	
1	0	1	0	0	0	1	

2 rows × 24 columns

Figure 2. One Hot Encoding using ‘get_dummies

g) Encoding categorical columns using One Hot Encoding: Each category is represented as a binary vector.

h) Scaling of numerical data using robust scaling as data in the dataset is not uniform and is having a number. of outliers.

j) Joining categorical and numerical columns to make a new dataset for training and testing of our model.

k)For feature selection, we used mutual_info_classif.

l) The reason for not dropping most of the columns for feature selection was the curve for the learning rate was stagnant.

Plotted correlation matrix:

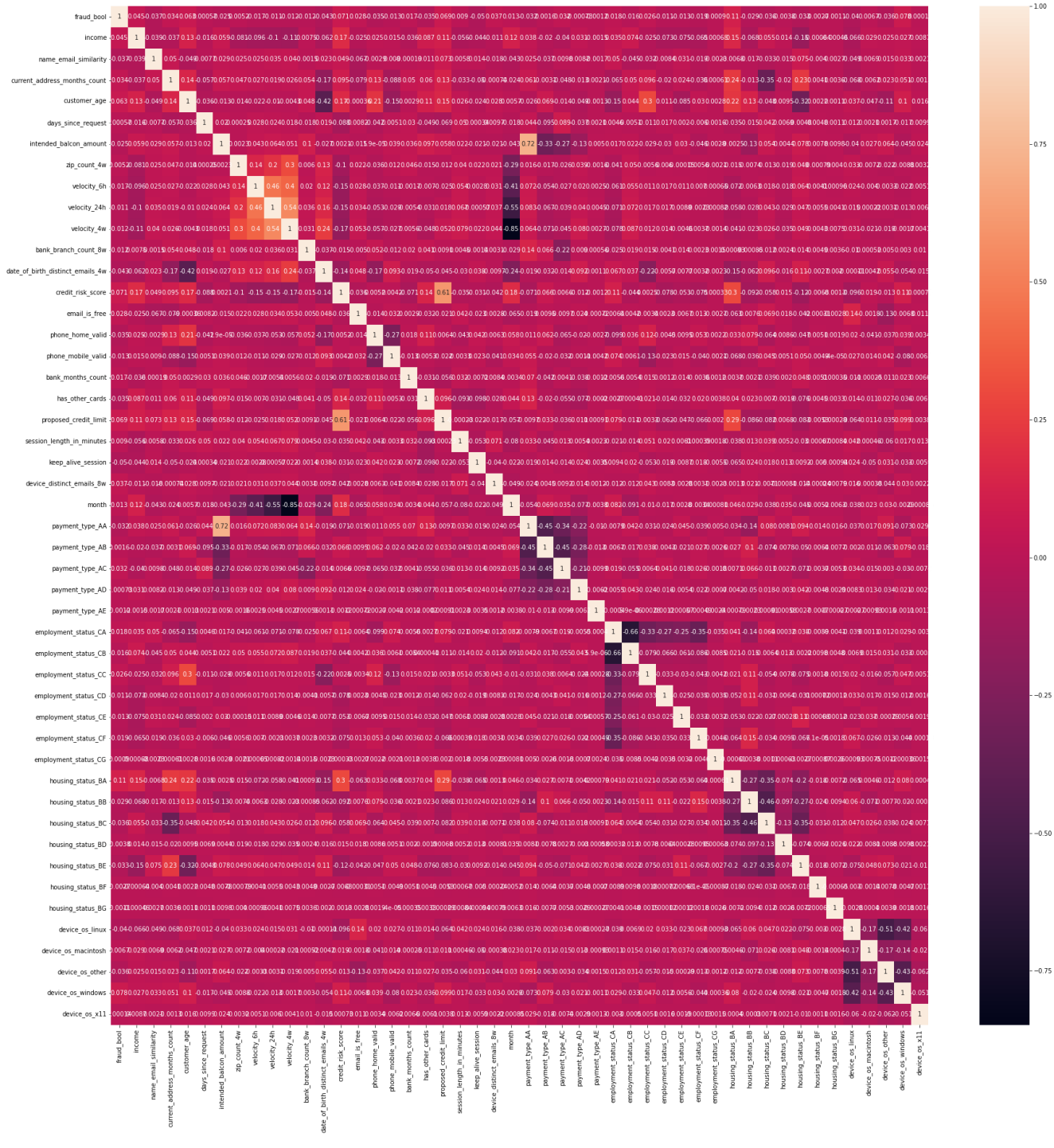


Figure 3. Correlation matrix for information gain

Required Packages:

We have utilized the following packages as per our requirement:

a) Data Manipulation: Pandas and Numpy
b) Model Training: Scikit Learn and Keras.

We have used various packages and libraries present inside Scikit Learn for our model preparation. Given below is the list of all the packages and libraries used:

i) For the Train-Test split, we have imported train_test_split from model_selection

ii) For different Models, we have imported different Models like

- from sklearn.naive_bayes import GaussianNB
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
- from sklearn.linear_model import LogisticRegression
- from xgboost import XGBClassifier
- from sklearn.naive_bayes import GaussianNB
- from sklearn import svm

iii) For Deep Learning: Keras from TensorFlow

We have used Keras for training our ANN model and the requirement are:

- from tensorflow import keras
- from tensorflow.keras.models import Sequential
- from tensorflow.keras.layers import Dense, SimpleRNN, Dropout

iv) For Visualization: Seaborn and MatplotLib

v) For Class Prediction: Scikit_learn

Again we have used different methods from Scikit_Learn.metrics for making a prediction on our model:

- ConfusionMatrixDisplay,classification_report
- roc_auc_score
- f1_score, confusion_matrix, accuracy_score,precision_score, recall_score

vi) For Saving the Model: Joblib for Machine Learning and Keras for ANN

7. PREDICTIVE MODELING

There are a few metrics that should be considered for predictive modeling.

(a) Accuracy: Accuracy is the most intuitive measure of performance. It is just the ratio of a correctly predicted observation to the total number of observations. We might think that our model is the best when the accuracy is high. Actually, high accuracy is good, but only if you have a symmetric data set, where false positives and false negatives are almost the same. Therefore, you need to look at other parameters to evaluate the performance of the model.

The formula for Accuracy :

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

(b) F1 Score: The F1 score is the weighted average of precision and recall. Therefore, this estimate takes into account both false positives and false negatives. Intuitively, it is not as easy to understand as precision, but F1 is usually more useful than precision, especially when your classes are unevenly distributed. When the cost of false positives and false negatives are the same, accuracy works best. When the cost of false positives and false negatives differ greatly, it is best to look for precision and recall at the same time.

The F1 Score formula:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})}$$

TP = number of true positives

FP = number of false positives

FN = number of false negatives

(c) Confusion Matrix: A confusion matrix (sometimes called a confusion table) is a two-row, two-column table that shows the number of false positives, false negatives, true positives, and true negatives, allowing for more detailed analysis. If the data set is not balanced, accuracy can produce false results; that is when the number of observations in different categories differs greatly. For example, if the data includes 95 cats and 5 dogs, a specific classifier can classify all observations as cats. The overall accuracy rate is 95 percent, but to be more precise, the detection rate (sensitivity) of the classifier for cats is 100 percent, but the detection rate for dogs is 0 percent.

In this case, the F1 value is even more unreliable. The confusion matrix is not limited to binary classification, but can also be used for multi-class classifiers.

The given figure shows the Confusion Matrix:

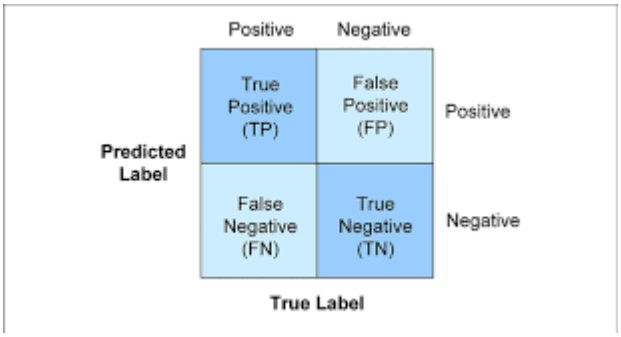


Figure 4: Confusion Matrix

(d) ROC / AUC CURVE: The receiver operating characteristic (ROC) curve is a graph showing the performance of the classification model under all classification thresholds. The curve shows two parameters:

a)True Positive Rate (TPR): TP/(TP+FN)

b)False Positive Rate (FPR): FP/(FP+TN)

ROC plots TPR against FPR: Area Under the ROC Curve (AUC) it measures the area under the ROC curve. Greater the area under the curve better the model. The AUC value ranges from 0 to 1. A model that predicts 100 percent incorrectly has an AUC of 0.0; a 100 percent accurate prediction has an AUC of 1.

(e) Matthews Correlation Coefficient Matrix: The Matthews Correlation Coefficient (MCC) is a metric used to evaluate the performance of binary classification models, and can be computed using a confusion matrix. The MCC ranges from -1 to 1, with 1 indicating a perfect classification, 0 indicating random classification, and -1 indicating total disagreement between predicted and actual classes. A value of 0 means that the algorithm performed no better than random, while a value of 1 means that it made perfect predictions.

The formula for MCC:

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

where,

MCC= Matthew's Correlation Co-efficient

TP = True Positive

TN= True Negative

FP=False Positive

FN= False Negative

Types of Sampling Methods Used:

After performing Exploratory Data Analysis, we first tried our model without sampling to check the unbalanced dataset which showed high accuracy and least F1 score against 1(fraud). So we performed Random under Sampling and Near Miss Under various algorithms and we received decent results. But to improve our results we went ahead with Near Miss Sampling which gave us good results on ANN.

Based on initial findings and testing on the dataset, we have used the following sampling methods:

(i) Random Undersampling: Undersampling is one of the techniques used for handling class imbalance. In this technique, we under-sample the majority class to match the minority class.

So in our example, we take a random sample of non-fraud classes to match a number of fraud samples.

This makes sure that the training data has an equal amount of fraud and non-fraud samples. Random undersampling or subsampling method contains examples randomly selected from the majority class in order to remove them from the training dataset. This results in fewer examples in the majority class in the converted version of the training data set. This process can be repeated until the desired class distribution is achieved, for example, each class has the same number of examples.

This approach may be more suitable for records with class imbalance. One of the limitations of subsampling is that it removes the most potentially useful, important, or potentially critical examples from the class to reach the limit of a robust solution. Since the examples are randomly deleted, there is no way to find or keep "good" or anything information-rich examples.

Models	Accuracy	Precision	Recall	F1-Score	False Positive
Random Forest	0.8	0.81	0.78	0.8	409
XGB	0.8	0.81	0.79	0.8	422
Gaussian NB	0.73	0.69	0.86	0.76	855
SVM	0.78	0.8	0.77	0.79	428
Gradient Boosting	0.81	0.83	0.81	0.82	386
Logistic Regression	0.76	0.8	0.77	0.79	440
Decision Tree	0.71	0.72	0.7	0.71	616
ANN	ROC AUC=0.89	0.1	0.01	0.01	0

Table No 1. : Model Training Result for Random-undersampling method

(ii) Near-Miss: The Near-Miss algorithm is a type of undersampling method used in machine learning to balance imbalanced datasets. The NearMiss algorithm works by selecting examples

from the majority class that is "nearest" to the examples in the minority class and discarding the rest. The "nearest" examples are determined based on a distance metric, such as Euclidean distance. We used version-1 of the near-miss algorithm instead of version-3 as accuracy was low in the latter one.

We have performed some hyper tuning of our Model ANN using Near Miss Under Sampling with respect to the following:

a) Number of dense layers : Had made no effect on accuracy.
b) Number of neurons change : Had not provided much effect
c) We used the sigmoid activation function in the last layer as softmax was not giving good results.(hint: The Softmax Layer is used when we need voting from the weighting.)
d) We used Adam optimizer, Max adam, and SGB optimizer but Adam was giving good results.
e) We have made modifications in the learning rate from 0.001 to 00.007 but there was no improvement in accuracy.

Models	Accuracy	Precision	Recall	F1-Score	False Positive	MCC
ANN	88	0.96	0.87	0.87	79	.77
SVM	88	0.94	.80	.86	106	.76
RANDOM FOREST	87	0.91	.83	.87	181	.75
XGBOOST	88	0.92	0.84	0.88	169	0.77
GRADIENT BOOSTING	87	.91	.82	0.87	170	0.75
LOGISTIC REGRESSION	84	0.88	0.79	0.83	227	0.69
NAIVE BAYES	80	0.97	0.63	0.76	48	0.65
DECISION TREE	69	0.72	0.62	0.66	544	0.38

Table No 2. : Model Training Results for Near Miss-undersampling method

8. CONCLUSION

We found out that by using Near Miss Undersampling Method through the ANN algorithm, our False Positives and False Negatives were reduced. To verify these claims we used the Classification Report. We also have used Matthew's Correlation Coefficient(MCC) as it considers both the positive and negative instances in the dataset, and hence provides a more balanced view of the classification performance. We got an F1 score of 88 as per our ANN Model. This project can be used to detect fraudulent bank account creation at the very initial stage.

9. REFERENCES

1.	Dynamic Stacked Ensemble with Entropy based Undersampling for the Detection of Fraudulent Transactions	https://ieeexplore.ieee.org/document/9417896/figures#figures
2.	Credit Card Fraud Detection - Machine Learning methods	https://ieeexplore.ieee.org/abstract/document/8717766
3.	Credit card fraud detection using machine learning techniques: A comparative analysis	https://ieeexplore.ieee.org/abstract/document/8123782
4.	Credit Card Fraud Detection using Machine Learning Algorithms	https://www.sciencedirect.com/science/article/pii/S187705092030065X
5.	Fraud detection using unsupervised machine learning techniques: A survey	R. Singh and P. Singh (2020)
6.	Combining machine learning techniques for bank fraud detection: A case study	D. A. Santos and E. F. R. Gomes (2021)