FRAUD DETECTION ON BANK PAYMENT BY USING MACHINE LEARNING REPORT

**INTRODUCTION:**

Fraud detection in bank payments using machine learning is a crucial application that can help financial institutions identify and prevent fraudulent activities in real-time. Machine learning algorithms can analyse large volumes of transaction data and identify patterns that are indicative of fraudulent behavior. Here's a general overview of how you might approach building a fraud detection system.

## BANKISM DATA:

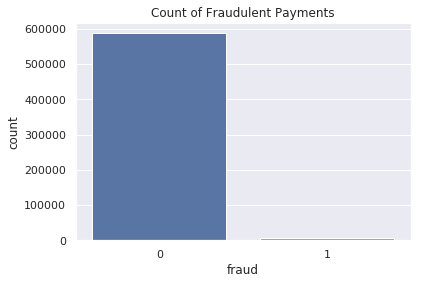
We detect the fraudulent transactions from the Banksim dataset. This synthetically generated dataset consists of payments from various customers made in different time periods and with different amounts. For more information on the dataset you can check the [Kaggle page](https://www.kaggle.com/ntnu-testimon/banksim1) for this dataset which also has the link to the original paper.

Here what we'll do in this kernel:

1. Exploratory Data Analysis(EDA)
2. Data Preprocessing
3. Oversampling with SMOTE
4. K-Neighbours Classifier
5. Random Forest Classifier
6. XGBoost Classifier
7. Conclusion

DATA: As we can see in the first rows below the dataset has 9 feature columns and a target column. The feature columms are :

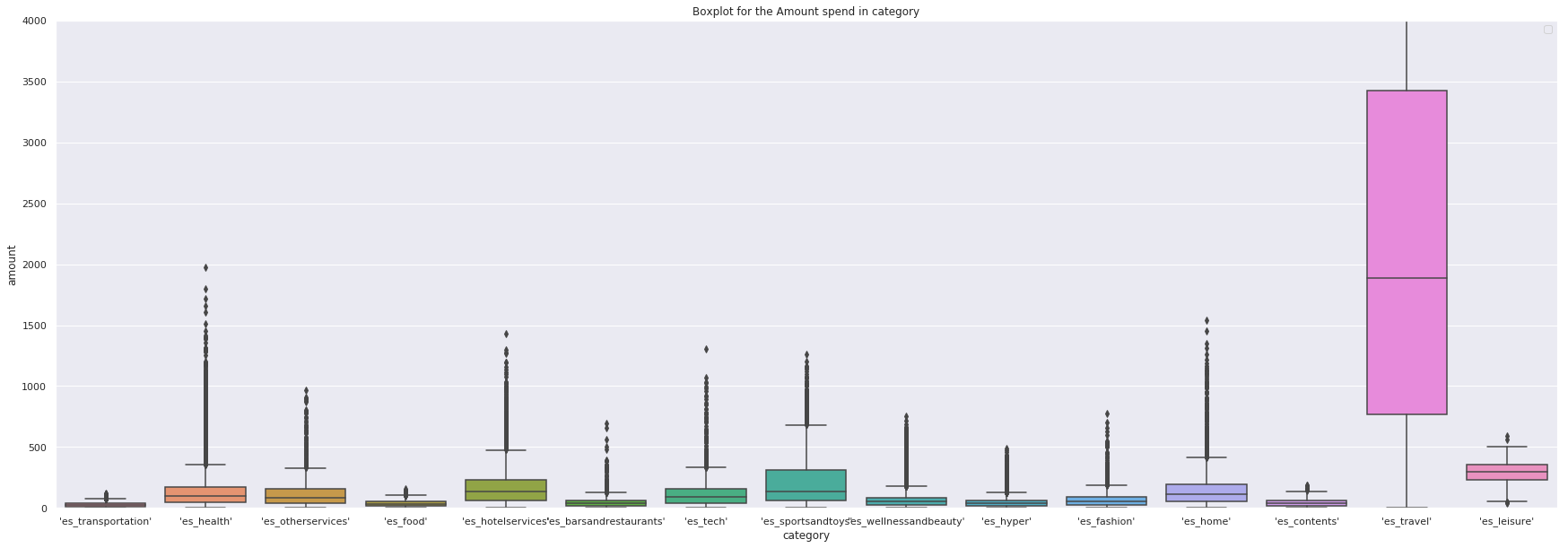
* **STEP**: This feature represents the day from the start of simulation. It has 180 steps so simulation ran for virtually 6 months.
* **Customer**: This feature represents the customer id
* **zipCodeOrigin**: The zip code of origin/source.
* **Merchant**: The merchant's id
* **zipMerchant**: The merchant's zip code
* **Age**: Categorized age
  + 0: <= 18,
  + 1: 19-25,
  + 2: 26-35,
  + 3: 36-45,
  + 4: 46:55,
  + 5: 56:65,
  + 6: > 65
  + U: Unknown
* **Gender**: Gender for customer
  + E : Enterprise,
  + F: Female,
  + M: Male,
  + U: Unknown
* **Category**: Category of the purchase. I won't write all categories here, we'll see them later in the analysis.
* **Amount**: Amount of the purchase
* **Fraud**: Target variable which shows if the transaction fraudulent (1) or benign (0)

# Create two data frames with fraud and non-fraud data

Number of normal examples: 587443

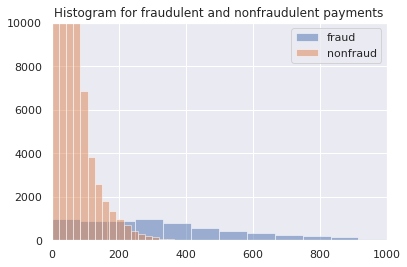
Number of fradulent examples: 7200

# Plot histograms of the amounts in fraud and non-fraud data



we can see in the histogram below the fradulent transactions are less in count but more in amount.

# Plot histograms of the amounts in fraud and non-fraud data



Looks like fraud occurs more in ages equal and below 18(0th category). Can it be because of fraudsters thinking it would be less consequences if they show their age younger, or maybe they really are young.

* Data Collection and Preprocessing**:** Collect historical transaction data, which includes features like transaction amount, time, location, transaction type, and any other relevant information. Preprocess the data by cleaning, transforming, and engineering features as needed. Also, balance the dataset if fraud cases are significantly lower in number compared to legitimate cases (imbalanced dataset).

.

## Oversampling with SMOTE

In a fraud detection project, it's common to encounter imbalanced datasets where the majority of transactions are non-fraudulent (normal) and a small fraction are fraudulent. This imbalance can lead to challenges when training machine learning models as they might struggle to effectively learn the patterns associated with the minority class (fraudulent transactions). One way to address this issue is by using oversampling techniques such as Synthetic Minority Over-sampling Technique (SMOTE)..

**Steps to Implement SMOTE:**

**Data Splitting:**  Divide the dataset into training, validation, and testing sets. The training set is used to train the model, the validation set helps tune hyperparameters, and the testing set evaluates the model's performance on unseen data.

**Model Selection**: Choose appropriate machine learning algorithms for fraud detection. Common choices include:

Logistic Regression

. Random Forest

. Gradient Boosting (e.g., XGBoost, LightGBM)

. Neural Networks

. Anomaly Detection Algorithms (e.g., Isolation Forest, One-Class SVM)

**Model Training**: Train the selected model using the training dataset. During training, pay attention to handling class imbalance using techniques such as oversampling, undersampling, or using algorithms designed to handle imbalanced data.

* **Model Evaluation** :Evaluate the model's performance using the testing dataset. Common metrics include precision, recall, F1-score, ROC-AUC, and accuracy. However, the choice of metrics depends on the ic business requirements and the cost associated with false positives and false negatives.
* **Threshold Selection:** Set an appropriate threshold for classifying transactions as fraudulent or legitimate. This depends on the desired balance between precision and recall and the business's risk tolerance.

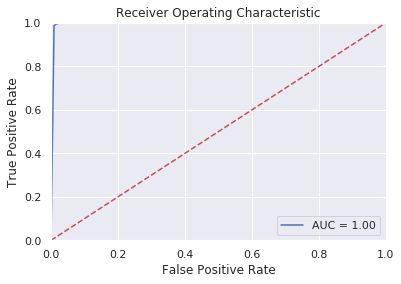
Deployment and Future Enhancements:

Successfully deploying the model into a real-time environment marked a significant achievement. To ensure its continued relevance, a robust monitoring and retraining strategy was put in place, enabling the model to adapt to emerging fraud patterns.

Business Recommendations:

Our findings suggest that integrating this fraud detection model into the bank's payment systems will yield substantial benefits, including minimizing financial losses, preserving customer trust, and reinforcing the bank's reputation for security and reliability.

K-Neighbours Classifier



## **Random Forest Classifier**

## XGBoost Classifier

## Conclusion

## In this kernel we have tried to do fraud detection on a bank payment data and we have achieved remarkable results with our classifiers. Since fraud datasets have an imbalance class problem we performed an oversampling technique called SMOTE and generated new minority class examples. I haven't put the classification results without SMOTE here but i added them in my github repo before so if you are interested to compare both results you can also check [my github repo](https://github.com/atavci/fraud-detection-on-banksim-data).