• PROJECT TOPIC
• (Predicting Credit Card Fraud)

Problem Statement

- For this project, I have chosen the banking sector, focus on fraudulent transactions as a case study. By using machine learning models, the models shall be trained using supervised learning techniques. The dataset contains thousands of financial transactions, respectively gotten from Kaggle (https://www.kaggle.com/code/turkayavci/fraud-detection-on-bank-payments/notebook)

Fraud

- Fraud is an intentional deception to secure unfair or unlawful gain, or to deprive a victim of a legal right. A fraudulent transaction is the unauthorized use of an individual's accounts or payment information.

Tools

- For this project, the tools we will be using are Python and Power BI. We decided to use Python because it is a high-level programming language, few lines of code for better output.
- (1)Python is a great tool, comprising machine learning algorithms. We shall load, explore, and visualize the fraud dataset in Jupiter notebook for better understanding.
- (2) Power BI to extract, load and transform the fraud dataset for insights by creating charts, and story telling.

Algorithms and Techniques

- Perform exploratory data analysis (EDA) on the dataset. This will help gain insights from the data structure (for example looking at outliers)
- The dataset is not a balanced one, so I will balance the classes (under sampling and oversampling) as learnt in class. Smote as best technique
- Create and train different models learned in class (Logistic regression, Decision Tree, K-Nearest Neighbor, SVM, Naïve Bayes, Confusion Matrix)
- When evaluating the models for example
- Precision: How correct the model is in predicting fraud
- Recall: if it predicts, how well it can still predict fraudulent or non-fraudulent transactions

Expectations

- To detect fraudulent transactions on the Banksim dataset
- Which algorithm predicts best with the dataset
- What story does the dataset tell us
- What is/are the categories/reasons behind fraudulent transactions

Metrics

- **Accuracy**

  When assessing a model performance, how accurate the dataset is, comes first in mind as a metric.
- **Relevant**

  The dataset would meet the requirements for the intended use to understand fraudulent transaction.
- **Completeness**

  This dataset does not have any missing values or missing data

A look at Our Dataset

- **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.
- **Amount**: Amount of the purchase
- **Fraud**: Target variable which shows if the transaction fraudulent(1) or benign(0)

Class observation before Smote

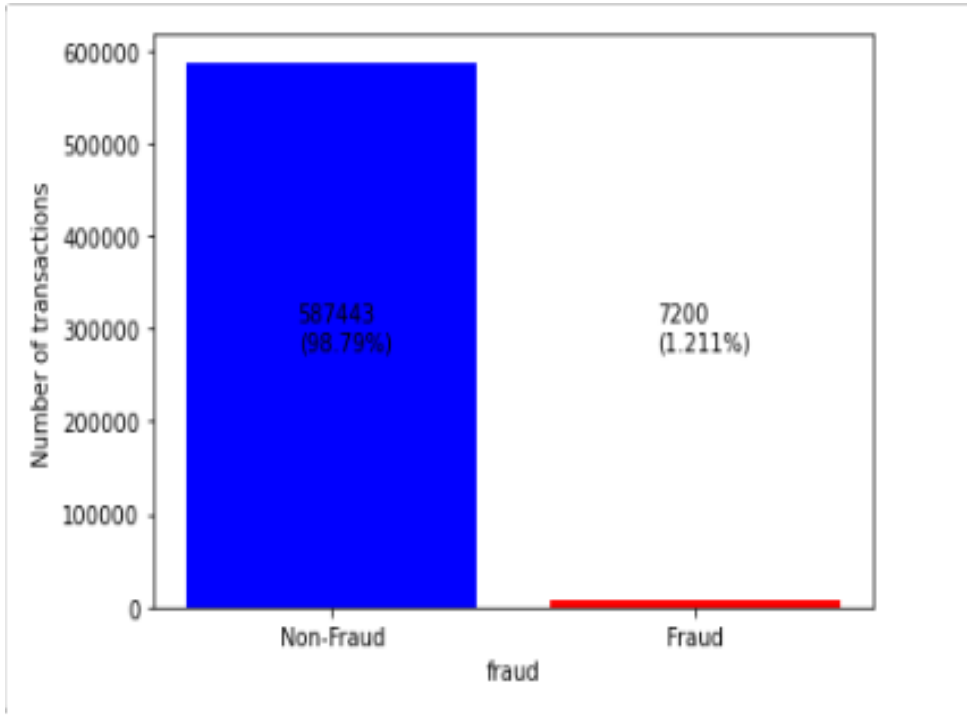


Fig 1: observation of both classes from the fraud dataset

As seen on figure 1 above, the dataset is highly imbalanced. Fraudulent activities account for just (1.211%) of the dataset, while non-fraudulent activities make up the larger part (98.79%).

TRANSACTIONS BY CATEGORY

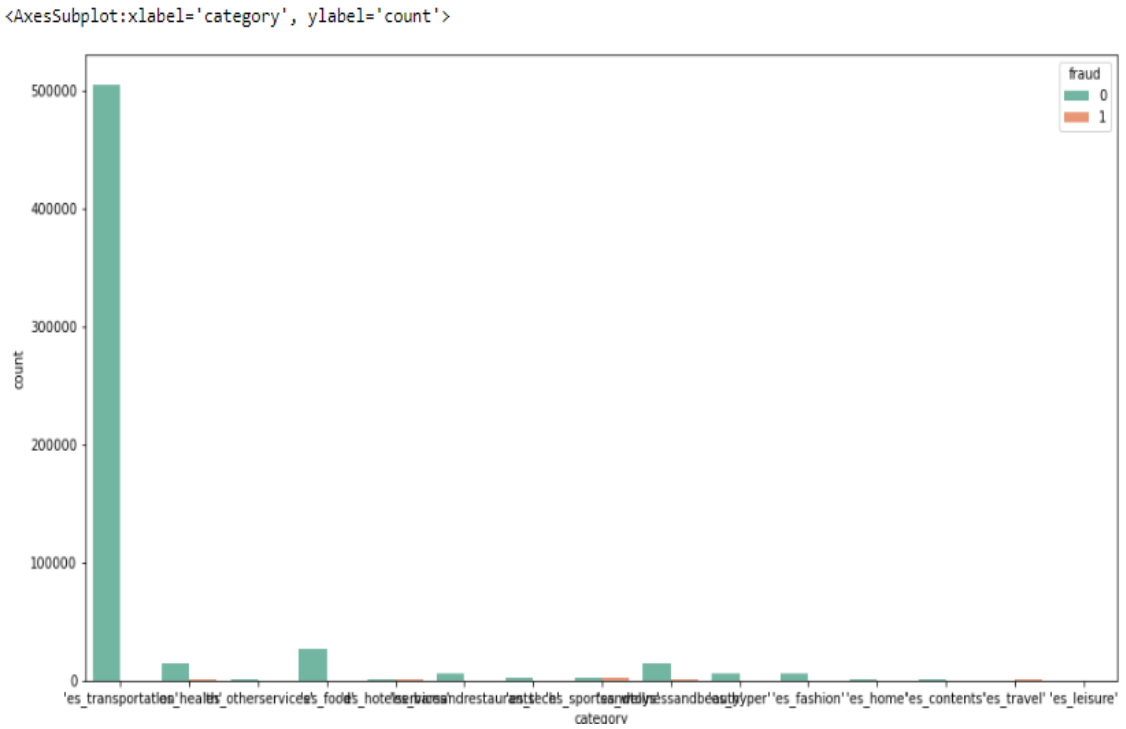
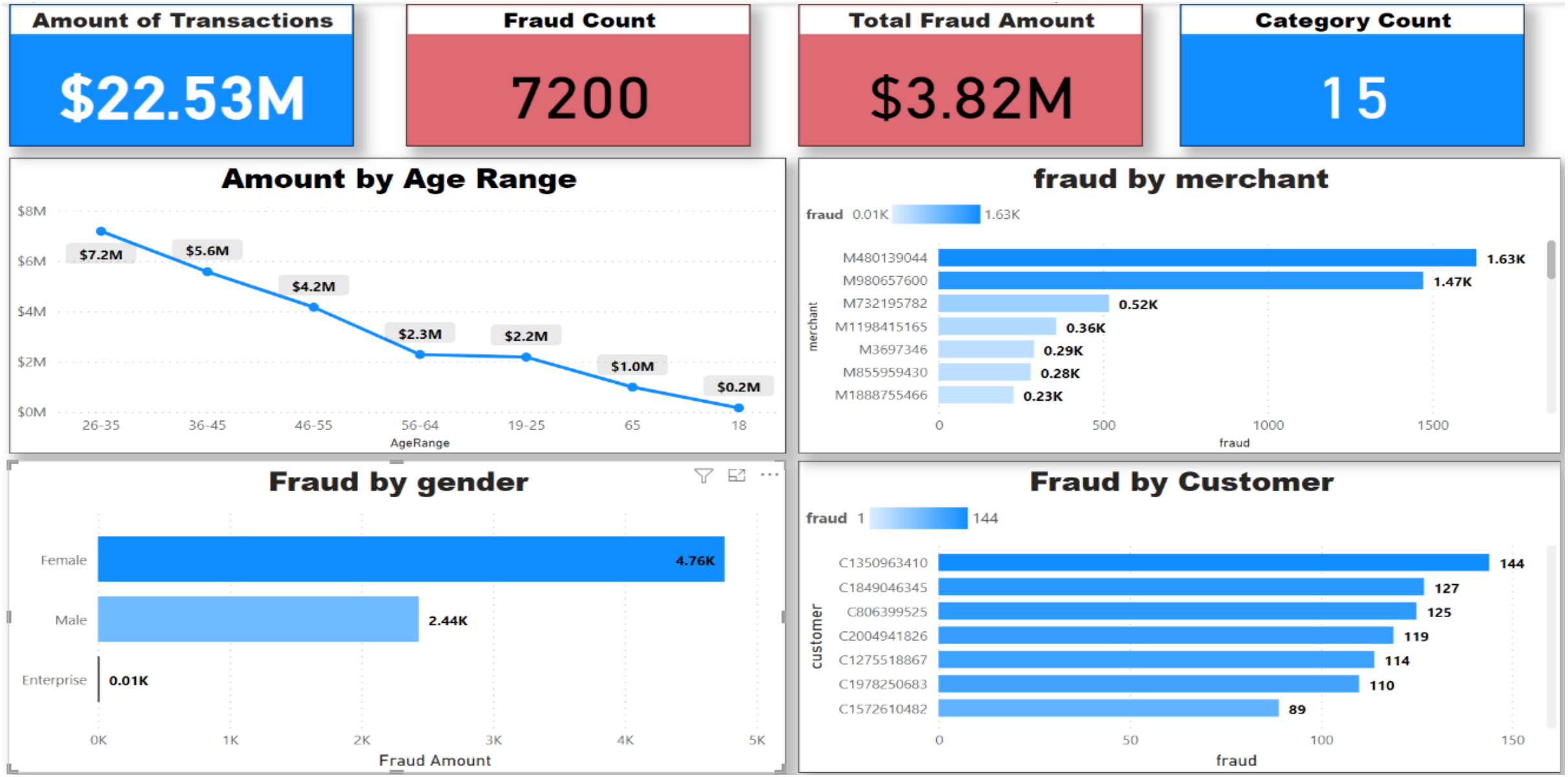


Fig 2: View of transactions by their category

We noticed the highest category in terms of spending and transactions is transportation. This can be likened to the constant movement of people from one location to another.



Insights on the Banksim fraudulent and non-fraudulent dataset

INSIGHTS

- The age range of 26-35years have the highest fraud record at $7.2Million, second by 36-45years and in bronze place 46-55years. This goes to say the youth are highly making things difficult for banks and people.
- Our second insight falls on gender. We visualized all genders and noticed two prominent were female and male, where the latter gender had a fraud count of 4758 followed by $2,503,555 fraudulent transactions. The former follows with a fraud count of 2442, fraudulent transactions at $1,315,801.
- Of the total fraudulent transactions, 43% happens with merchants M480139044 and M980657600. We later questioned what category customers transact with these merchants and the cost.

Top Fraud category: es_health, Total amount transacted: 664804.3899999999

Top Fraud category: es_sportsandtoys, Total amount transacted: 505311.62

In conclusion, Top fraudulent transactions happen in **es_health** and **es_sportandtoys** category with total amount transacted **664,804** and **505,311** respectively. But we should not forget **es_travel** is where most fraudulent transactions happen.

- Regarding customer fraud cases, customers (C1350963410, C1849046345, C806399525, C2004941826) were involved in transactions flagged as fraudulent.

SPENDING BY GENDER

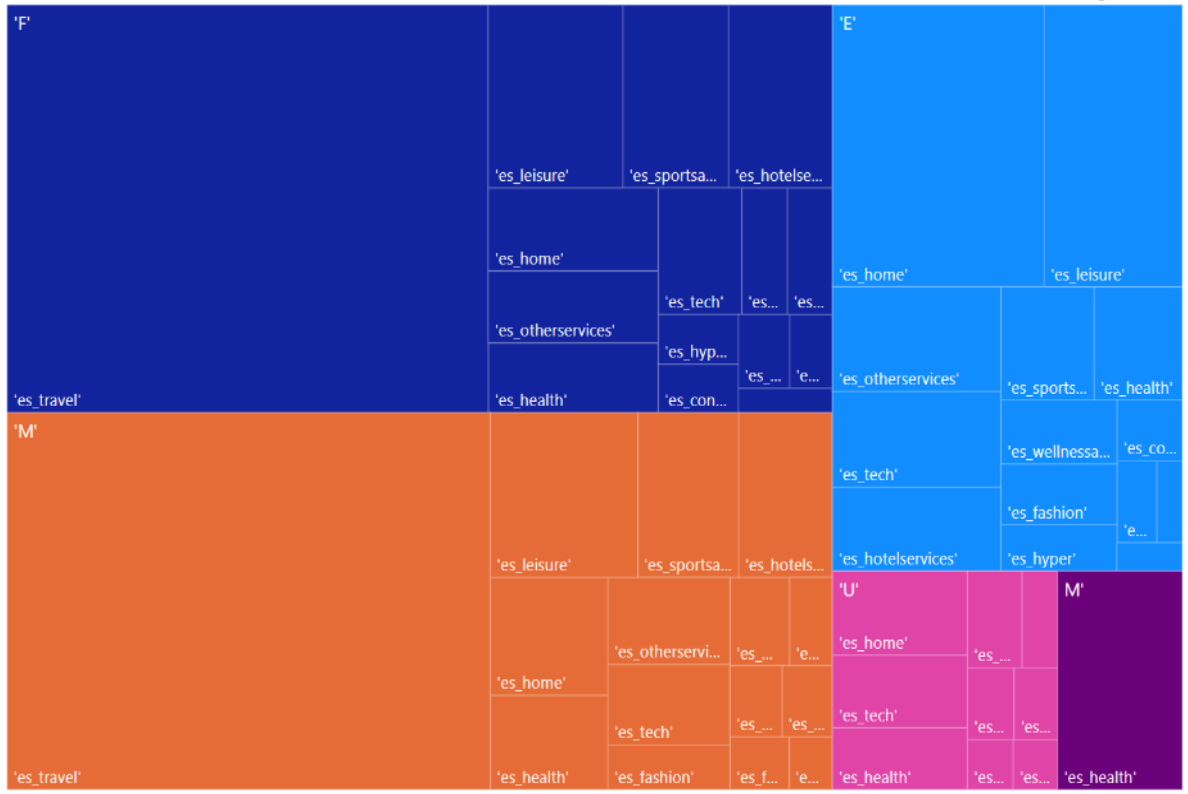


Fig 3: category with the highest spending rate by each gender

The male and female genders comprises the highest spenders, with travelling as the main spending habit. As if to say, they lavish the money from one location to another(**es_leisure, es_sportsandtoys, es_wellness and beauty**) Speaking on facts, all the transactions flagged as fraud had travelling as their basis.

FLAGGED AS FRAUD

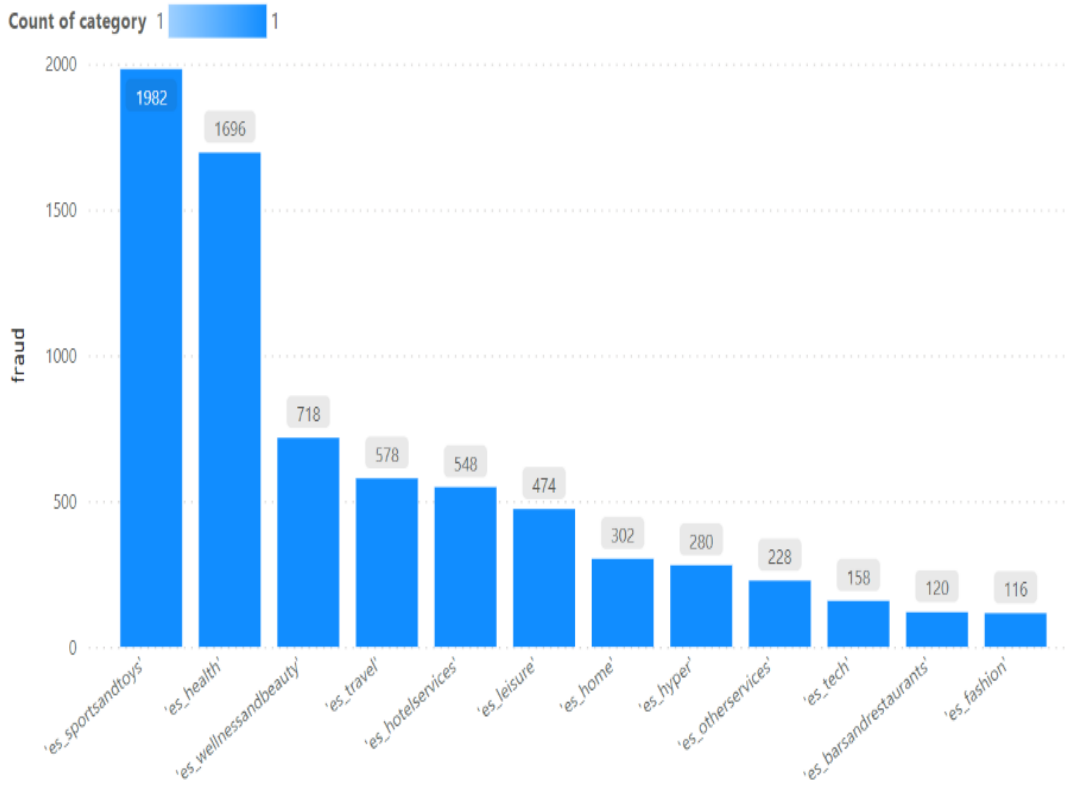


Fig 4: flagged fraud by category

Based on the chart, fraudulent transactions weighed the highest on (**sports, health, and beauty**).

```python
#SCRIPT FOR SVM
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix


for name,method in [('SVM', SVC(kernel='linear',random_state=100))]:
    method.fit(x_train2,y_train)
    predict = method.predict(x_test2)
    print(confusion_matrix(y_test,predict))
    print(classification_report(y_test,predict))
```

```
[[117429     60]
 [   719    721]]
              precision    recall  f1-score   support

           0       0.99      1.00      1.00    117489
           1       0.92      0.50      0.65      1440

    accuracy                           0.99    118929
   macro avg       0.96      0.75      0.82    118929
weighted avg       0.99      0.99      0.99    118929
```

Support Vector Machine (SVM) does well with the dataset in predicting both classes (fraud and non-fraudulent)
Looking at class 0, it rightly predicts at 99% while at 1, comes at 92% which is still pretty good.
An f1-score of 0.99 which shows assurance of the model.
But recall at 1.00, very good for a predictive model.
Weighted average of 0.99 which is good.

```python
#SCRIPT FOR LOGISTICAL REGRESSION
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

for name,method in [('Logistic Regression', LogisticRegression(solver='liblinear',random_state=100))]:
    method.fit(x_train2,y_train)
    predict = method.predict(x_test2)
    print(confusion_matrix(y_test,predict))
    print(classification_report(y_test,predict))
```

```
[[117354    135]
 [   605    835]]
              precision    recall  f1-score   support

           0       0.99      1.00      1.00    117489
           1       0.86      0.58      0.69      1440

    accuracy                           0.99    118929
   macro avg       0.93      0.79      0.84    118929
weighted avg       0.99      0.99      0.99    118929
```

For logistic regression (LR) in predicting the dataset.
A look at Precision of both classes (0 and 1). At 0 it confidently predicts non-fraudulent transactions at 0.99, while for fraudulent it predicts at 0.86. A good model for predicting both classes but better at finding non-fraudulent transactions.
Recall at 1.00, which is good for a predictive model.
Not as good like SVM with the dataset.

```python
#SCRIPT FOR DECISION TREE
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix

for name,method in [('DT', DecisionTreeClassifier(random_state=100))]:
    method.fit(x_train2,y_train)
    predict = method.predict(x_test2)
    print('\nEstimator: {}'.format(name))
    print(confusion_matrix(y_test,predict))
    print(classification_report(y_test,predict))
```

```
Estimator: DT
[[117158    331]
 [   625    815]]
              precision    recall  f1-score   support

           0       0.99      1.00      1.00    117489
           1       0.71      0.57      0.63      1440

    accuracy                           0.99    118929
   macro avg       0.85      0.78      0.81    118929
weighted avg       0.99      0.99      0.99    118929
```

Decision tree (DT) algorithm handles the class 0 very well and can predict at 0.99, but did an above average job with the class 1 at 0.71.
Recall is good at 1.00 same as f1-score
A weighted average of 0.99 is good for the model.

DT is better at predicting non-fraudulent transactions (0) than fraudulent transactions(1)

CONCLUSION

- From above plots its clear that across figures/features, the amount transacted is higher in Fraudulent cases
- The bank should take special interest in high value transactions and do a check before its approved (most banks do this now). They also have daily and weekly spendings to limit movement of suspicious funds/fraud.
- We have more fraudulent transactions in amounts than count
- Leveraging from the above charts we got merchants who are affected by these fraudulent transaction. Looking at the total amount transacted, we advise, the bank should flag these merchants and monitor transactions closely.
- Same recommendation should apply to our analysis for customer feature but flagging customers would create a tense banking environment, can lead to bad idea for PR, hence loosing customers. Another method is by observing customer buying power and locations.

ALGORITHMS

- SVM performed better than LR and DT with the Banksim dataset after balancing the lesser class with SMOTE.
- We would recommend using SVM for better quality and assurance in predicting fraudulent and non-fraudulent transactions.

FINAL OBSERVATION

- Situation where a transaction is flagged as fraud, but not a fraud, that is known as false positive. we will put a system in place to better analyze the situation. This will limit any friction between the bank and its customers.

REFERENCES

1. https://www.kaggle.com/code/turkayavci/fraud-detection-on-bank-payments/data
2. https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

3. **Professor Babalola, Data 2206-Capstone/Durham College**

4. **Professor Sergey R, Data 2205- Durham College**