Fraud Detection Project Report

Youssef Mohamed Eladl

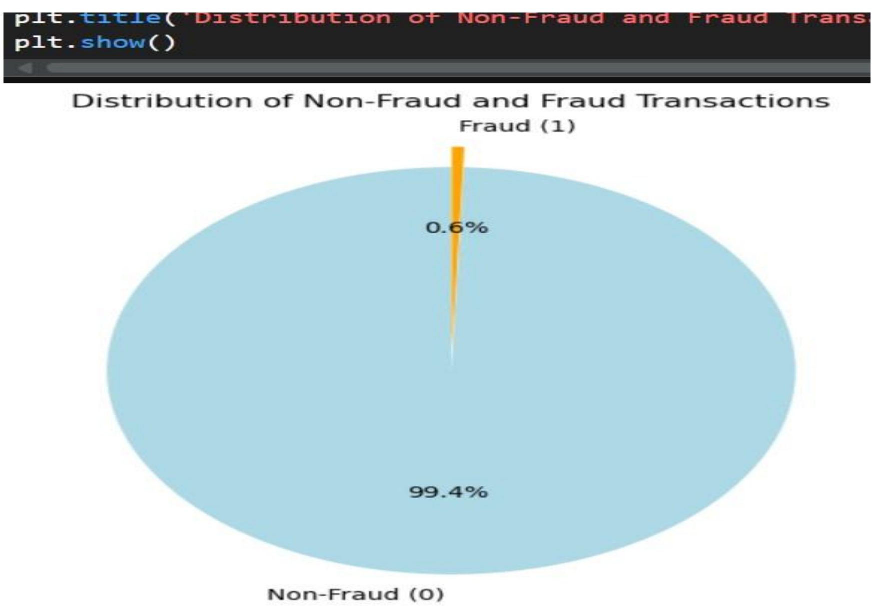
- https://github.com/youssefeladl
- http://www.linkedin.com/in/youssefeladl

1. INTRODUCTION

This project focuses on building a machine learning pipeline to detect fraudulent transactions using a structured dataset. It includes steps for data exploration, preprocessing, feature engineering, handling class imbalance, model selection, hyper parameter tuning, and evaluation.

2. DATA EXPLORATION

We began by exploring the dataset to understand its structure, distribution, and anomalies. Key steps included checking for null values, understanding data types, summarizing numerical features, and visualizing the distribution of fraudulent vs. non-fraudulent transactions.



3. DATA PREPROCESSING

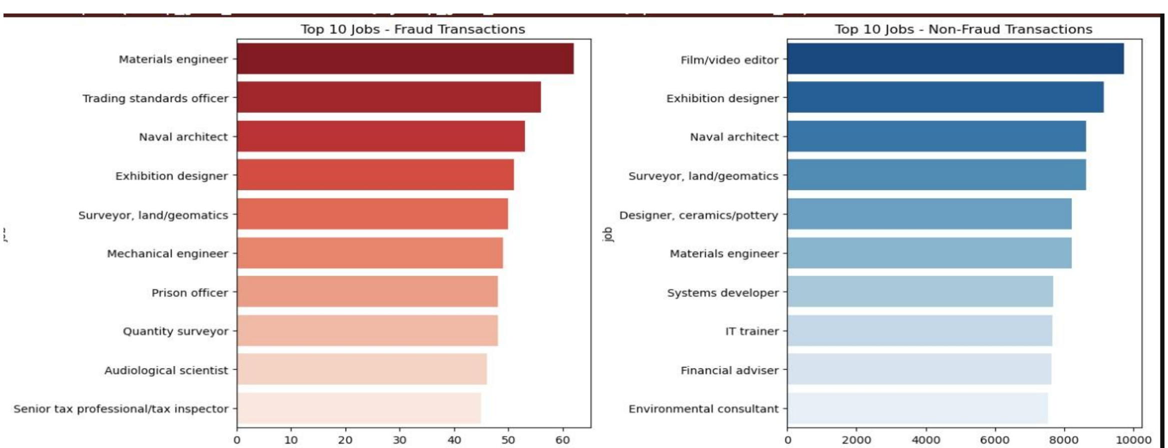
We converted datetime columns to proper formats, extracted features like hour, day, and month from timestamps, and created age groups. Irrelevant or redundant columns such as 'cc_num' were removed to focus on meaningful features.

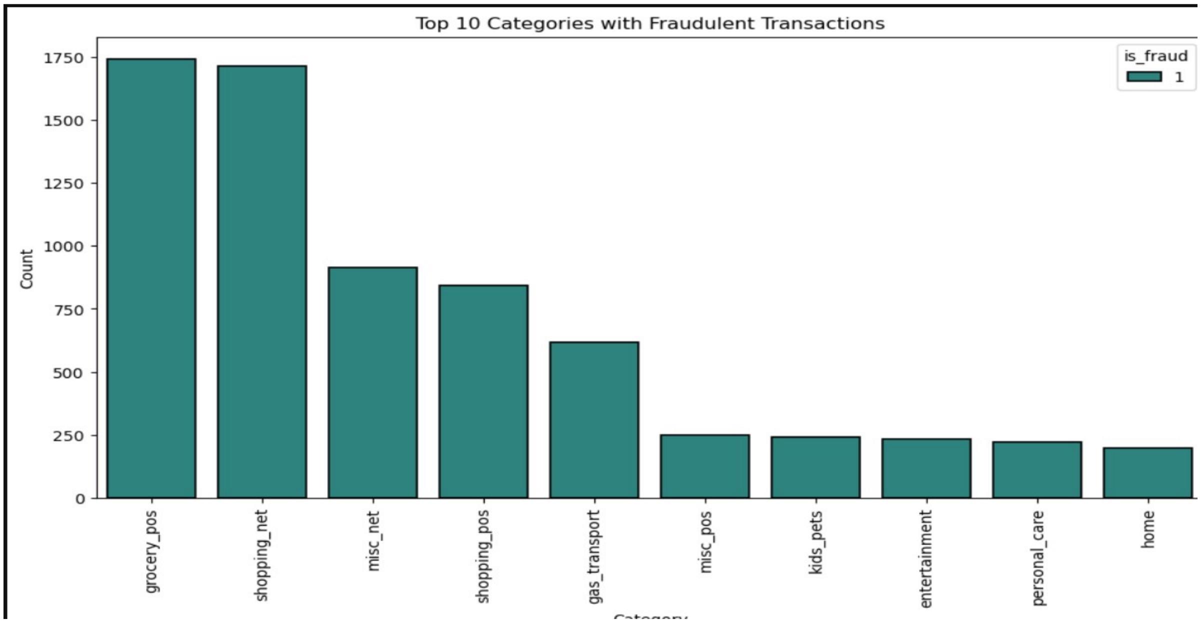
```
RangeIndex: 1296675 entries, 0 to 1296674
Data columns (total 23 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   Unnamed: 0             1296675 non-null  int64
 1   trans_date_trans_time  1296675 non-null  object
 2   cc_num                 1296675 non-null  int64
 3   merchant               1296675 non-null  object
 4   category               1296675 non-null  object
 5   amt                    1296675 non-null  float64
 6   first                  1296675 non-null  object
 7   last                   1296675 non-null  object
 8   gender                 1296675 non-null  object
 9   street                 1296675 non-null  object
 10  city                   1296675 non-null  object
 11  state                  1296675 non-null  object
 12  zip                    1296675 non-null  int64
 13  lat                    1296675 non-null  float64
 14  long                   1296675 non-null  float64
 15  city_pop               1296675 non-null  int64
 16  job                    1296675 non-null  object
 17  dob                    1296675 non-null  object
 18  trans_num              1296675 non-null  object
 19  unix_time              1296675 non-null  int64
 20  merch_lat              1296675 non-null  float64
 21  merch_long             1296675 non-null  float64
 22  is_fraud               1296675 non-null  int64
dtypes: float64(5), int64(6), object(12)
memory usage: 227.5+ MB
```

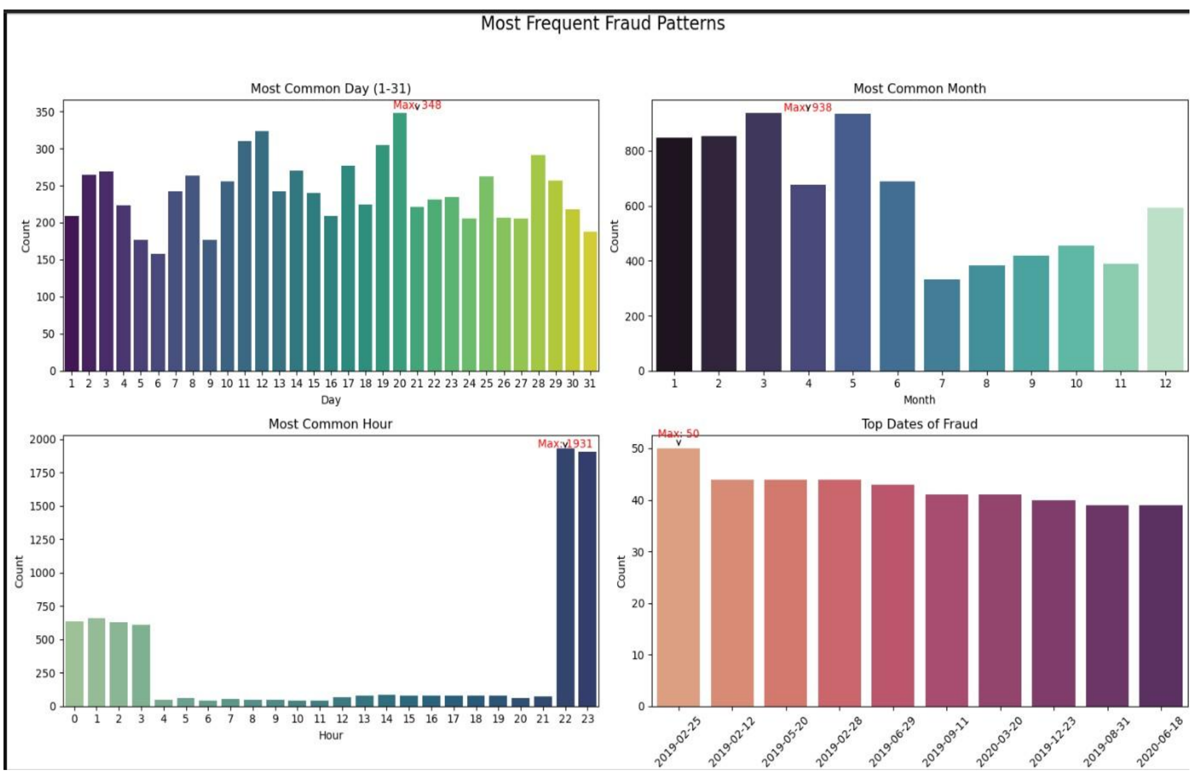
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1296675 entries, 0 to 1296674
Data columns (total 13 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   trans_date_trans_time  1296675 non-null  datetime64[ns]
 1   cc_num                 1296675 non-null  int64
 2   merchant               1296675 non-null  object
 3   category               1296675 non-null  object
 4   amt                    1296675 non-null  float64
 5   gender                 1296675 non-null  object
 6   city                   1296675 non-null  object
 7   state                  1296675 non-null  object
 8   city_pop               1296675 non-null  int64
 9   job                    1296675 non-null  object
 10  age                    1296675 non-null  int32
 11  dis_to_merch           1296675 non-null  float64
 12  is_fraud               1296675 non-null  int64
dtypes: datetime64[ns](1), float64(2), int32(1), int64(3), objec
```

4 EDA

Some of insights.

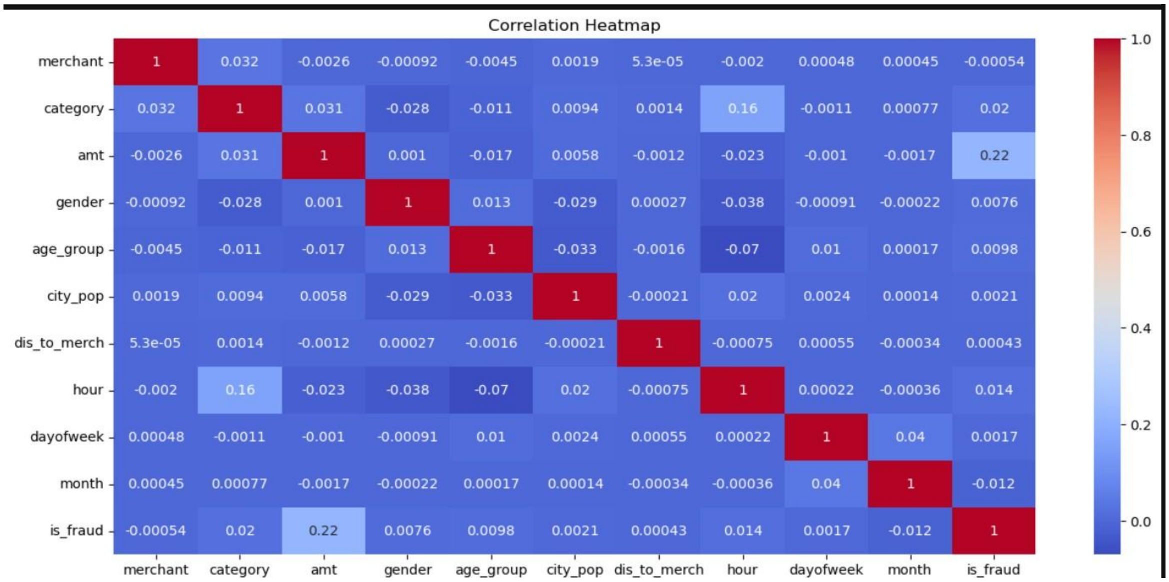


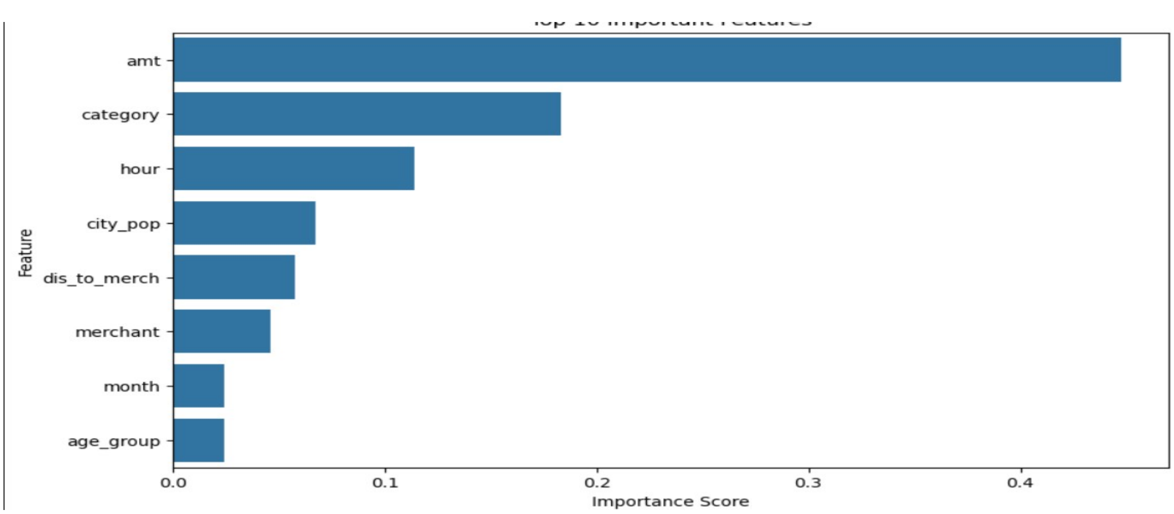




5. FEATURE ENGINEERING

We used SelectKBest to choose the most informative features. Features like 'merchant', 'category', 'amt', 'age_group', 'city_pop', 'dis_to_merch', 'hour', and 'month' were selected.





6. HANDLING CLASS IMBALANCE

The dataset was heavily imbalanced. We used SMOTE (Synthetic Minority Oversampling Technique) on the training set to synthetically generate samples for the minority class (fraudulent transactions). This helped the model to generalize better.

7 SPLITTING DATA AND VALIDATION STRATEGY

The dataset was split into training, validation, and testing sets. Stratified sampling ensured equal class representation. Cross-validation was used for model selection and hyperparameter tuning.

```
[142… X=new_df.drop(columns=['is_fraud','merchant','gender','dis_to_merch','dayofweek','month'])
      y=new_df['is_fraud']

[143… from sklearn.model_selection import train_test_split
      X_train_val,X_test,y_train_val,y_test=train_test_split(X,y,test_size=0.2,random_state=42, stratify=y)
      X_train,X_val,y_train,y_val=train_test_split(X_train_val,y_train_val, test_size=0.2, random_state=42, stratify=y_train_val)

[144… from imblearn.over_sampling import SMOTE
      sm = SMOTE(random_state=42)
      X_train_smote, y_train_smote = sm.fit_resample(X_train, y_train)

[ ]:  from sklearn.ensemble import RandomForestClassifier
      rf = RandomForestClassifier(random_state=42)
      rf_params = {
          'n_estimators': [100, 200],
          'max_depth': [None, 10, 20],
          'min_samples_split': [2, 5],
          'min_samples_leaf': [1, 2]
      }
      rf_grid = GridSearchCV(rf, rf_params, cv=5, scoring='f1', verbose=1, n_jobs=-1)
      rf_grid.fit(X_train_smote, y_train_smote)
      rint("Best Parameters:", rf_grid.best_params_)
      print("Random Forest Classification Report:")
      print(classification_report(y_test, rf_grid.predict(X_test)))

      Fitting 5 folds for each of 24 candidates, totalling 120 fits
```

8. MODEL BUILDING AND SELECTION

Several models were trained including Random Forest and LightGBM. Hyperparameters were tuned using GridSearchCV. LightGBM and Random Forest gave the best results.

9. ENSEMBLE LEARNING

A VotingClassifier ensemble combining LightGBM and Random Forest was implemented. This model achieved the highest performance with improved precision and recall.

10. EVALUATION METRICS

Models were evaluated using Precision, Recall, F1-Score, and Accuracy. Ensemble models showed balanced tradeoffs between minimizing false positives and maximizing fraud detection.