Report on

Machine Learning Project:

It’s a Fraud

Detecting if a credit card transaction is fraudulent

Submitted to:

Debmalya Sen

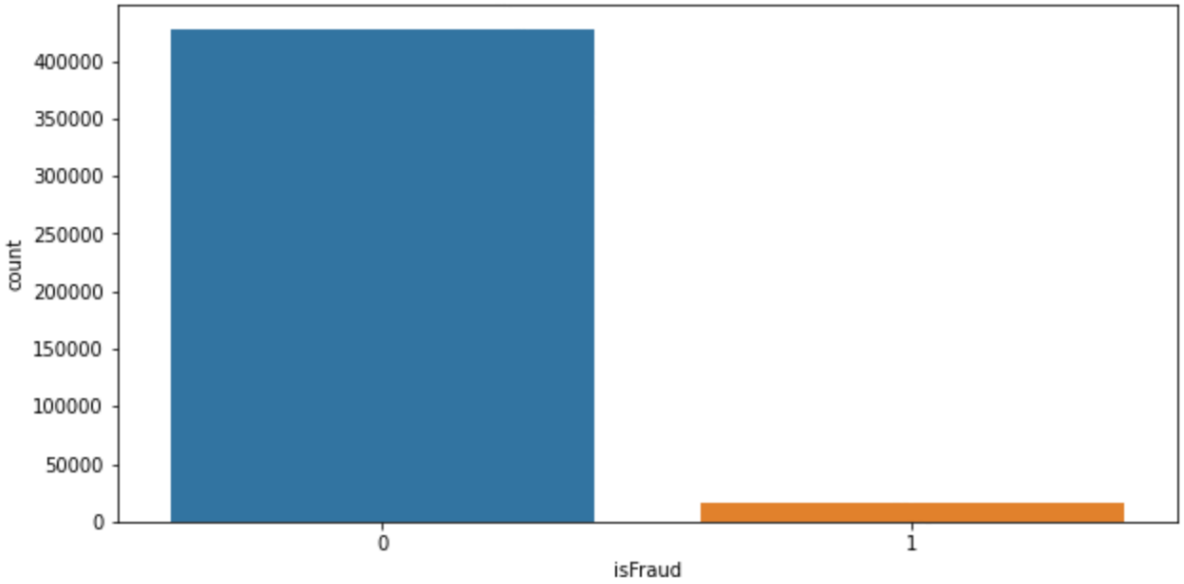
Submitted by:

Srishtiraj Gupta MT2022117

Yaman Goyal MT2022134

EDA

- Target feature: ‘isFraud’

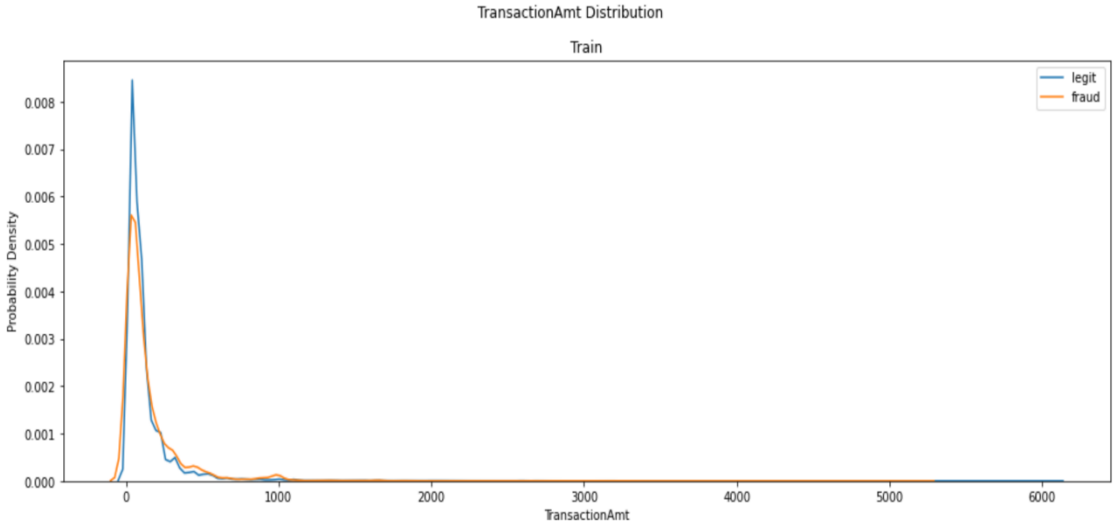


  - the dataset is extremely imbalanced. Resampling is required to obtain unbiased results.

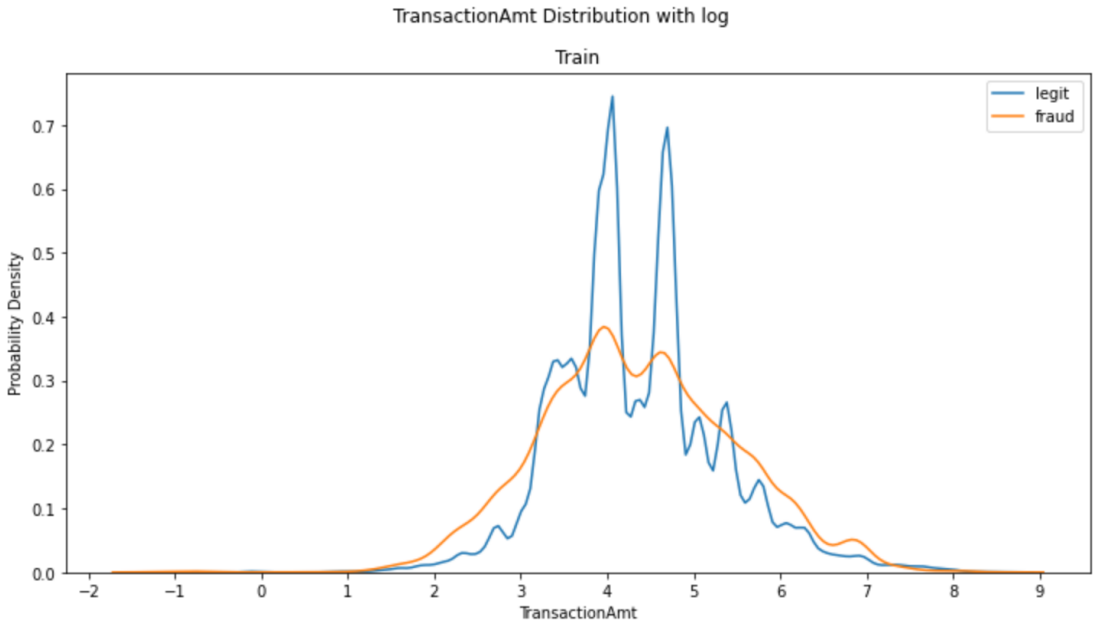
- ‘ProductCD’:

  - There is higher chance that the transaction is a fraud if it's product is C.

- ‘TransactionAmt’:

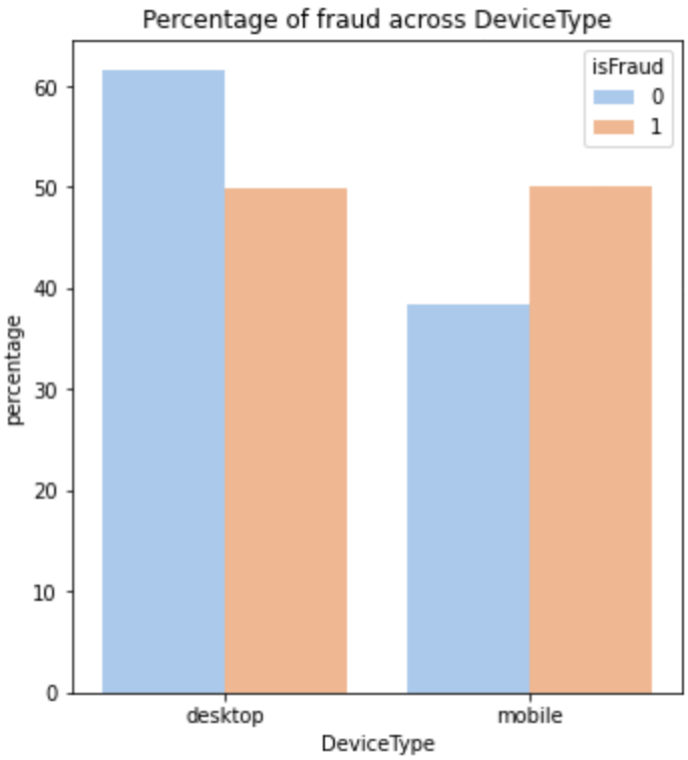


  - highly skewed, so we took log of this feature to remove skewness.



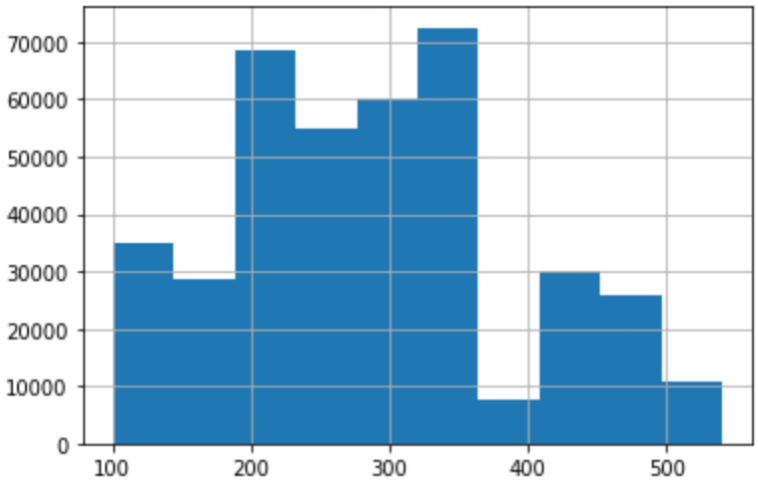
  - The transactions with LogTransactionAmt larger than 5.5 (244 dollars) and smaller than 3.3 (27 dollars) have higher frequency and probability density being fraudulent. On the other hand, LogTransactionAmt from 3.3 to 5.5 have higher chance being legit.

‘DeviceType’:

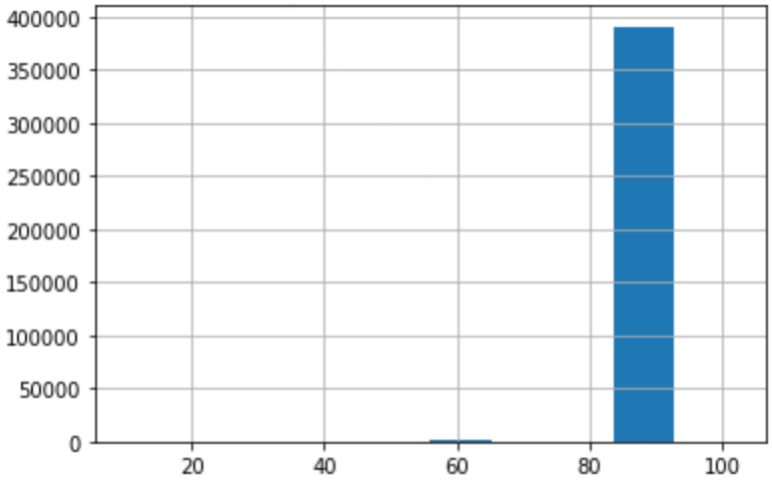


- Both Desktop and mobile have the same percentage of fraud, so this feature doesn’t provide much information.

- ‘addr1’ and ‘addr2’:

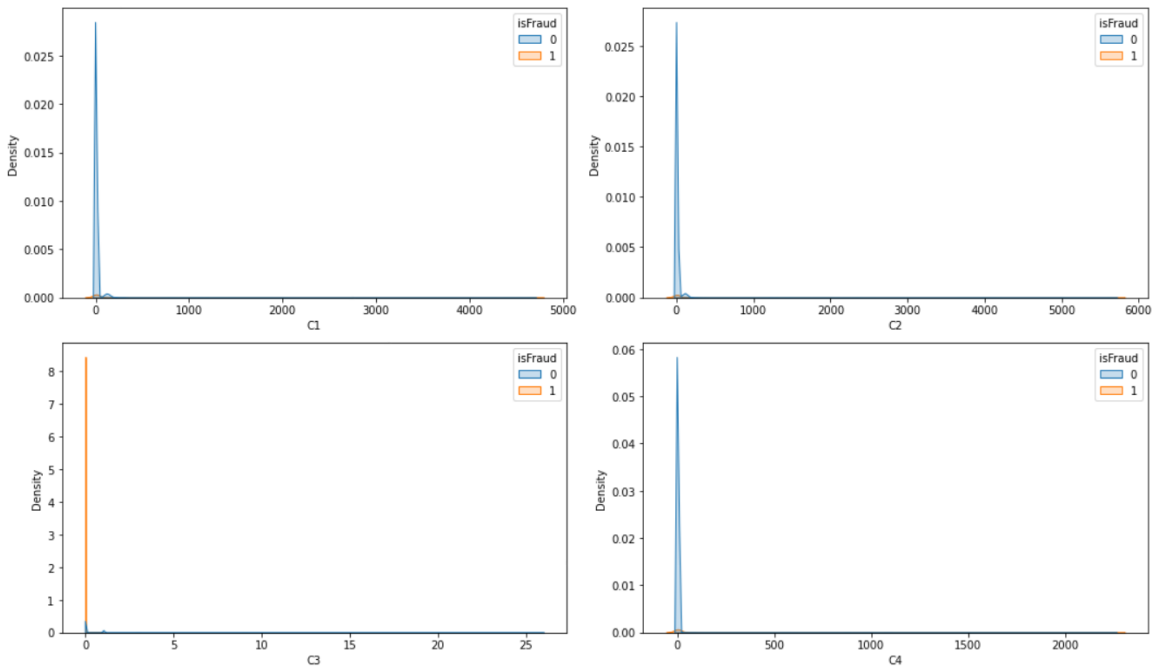


- addr1: this feature probably tells about the region within a country, where transaction originated from.



- addr2: this feature tells the country of transaction. As its variance is very low (88% values are same), this feature wont contribute much in training of model. Hence we decided to drop it.

- ‘C’ values:



- Plotting *kdeplot* against them, its found that all ‘C’ features are highly right-skewed. The mean and 75% values differ a lot.

Data Preprocessing and Feature Engineering:

- Null value handling: it was found that 45% of data was null. Following steps were taken to remove them:
    - Dropping rows: those columns which had less than 1% null values, those corresponding rows were dropped. Such features are: {card2, card3, card4, card5, card6, D1, V95, V279, V281}
    - Dropping columns: if any column has more than 70% null values, those features are removed. They are: {V138 - V278, DIST2, R_emaildomain, D12, D13, D14, DeviceType, DeviceInfo, D6 – D9, V322 – V339, id_01 – id_38}
    - Filling the null with values:
        - Mean: those features whose mean and 75% are similar. Ex: 'D' and 'V' values.
        - Median: if the values of a feature are too scattered, i.ee mean is too far off from 25%, 50% and 75% values, then median is filled. Ex: dist1
        - Mode: none.
        - Hybrid: filling the nulls in categorical columns (randomly) with same ratio of categories as originally given. We went with this approach rather than filling mean, median or mode, as all of them were introducing biasness in data.
        - 'NA': we have replaced the null values in categorical columns with 'NA'. Ex: P_emaildomain
- Checking duplicate entries: none found.

- Removing columns with less than 1% variance: these features have almost all entries with same value. They are dropped.
- Correlation matrix and removal of corelated columns: features which have correlation greater than 90% with any other feature are dropped.

- Handling Categorical features:
    - 'M' features are categorical containing binary or ternary classes. They are manually replaced to numerical values.
    - Following categorical features remained after preprocessing: {ProductCD, card4, card6, P_emaildomain}
    - Label encoding is done to convert these categorical features to numerical features. We wanted number of features to be as least as possible hence we avoided one-hot encoding.

- Normalizing dataset: MinMaxScaler is applied to normalize dataset. Feature scaling is done to transform the values of different numerical features to fall within a similar range. This avoids the training model to be biased towards certain values.
- Removing unimportant features: features like ‘TransactionID’ which serve as key for transactions wont help in training desired model. Such features are dropped.

- After all the preprocessing, the dataset is brought down to 113 features compared to 434 features originally.

- Re-sampling the dataset: As seen from EDA, given dataset is highly imbalanced. We went with RandomUnderSampler to handle this imbalance.

```
Original dataset shape Counter({0: 418575, 1: 15007})
Resampled dataset shape Counter({0: 15007, 1: 15007})
```

Model Evaluation and Hyperparameter-tuning:

- Non-tree based models:

S.no	Model	Optimal Hyperparameters	Kaggle score
1.	Logistic Regression	{'C': 0.1, 'max_iter': 5000, 'penalty': 'l2', 'solver': 'liblinear'}	0.79
2.	K nearest neighbours	{'n_neighbors': 6, 'weights': 'distance'}	0.794
3.	SVM	{'C': 3, 'gamma': 0.02, 'kernel': 'rbf'}	0.792
4.	Naïve Bayes	-	0.675

- Tree-based models:

S.no	Model	Optimal Hyperparameters	Kaggle score
1.	XGBoost	{'colsample_bytree': 0.4, 'learning_rate': 0.1, 'max_depth': 20, 'min_child_weight': 1}	**0.881**
2.	Random Forest	{'max_features': 'sqrt', 'max_samples': 2, 'min_samples_split': 2, 'n_estimators': 50}	0.819
3.	Decision Tree	{'criterion': 'entropy', 'max_depth': 20, 'min_samples_leaf': 50}	0.790
4.	Neural Network	-	0.798

XGBoost performs best among all classification model employed to solve the given problem. This was expected as XGBoost handles large datasets and imbalance in data very well. It minimises L1 and L2 regularised objective function. They contain a convex loss function and model complexity penalty rank. The model is trained iteratively with gradient boosting: new trees are inserted which predict the errors of previous trees that are then combined with previous trees, and the loss minimisation takes place through gradient descent algorithm.