DS Track Challenge 2

(Credit Card Dataset / Finance Challenge)

Indrani Singha Roy

Github Link

Introduction / Agenda

- Loading, Pre-processing
- Exploratory Data Analysis (EDA) + Feature Engineering
- Prepping the Dataset: Scaling & cleaning
- Metrics to Consider
- Fraud Detection Models (results + insights)
- Takeaways + Future Work

Loading & Pre-processing

- Used pd.read_parquet( ) native method to read in the 3 parquet files/ datasets. This allows us to take advantage of the more efficient file format to read the huge transaction dataset (6.8+M rows) into a data frame.

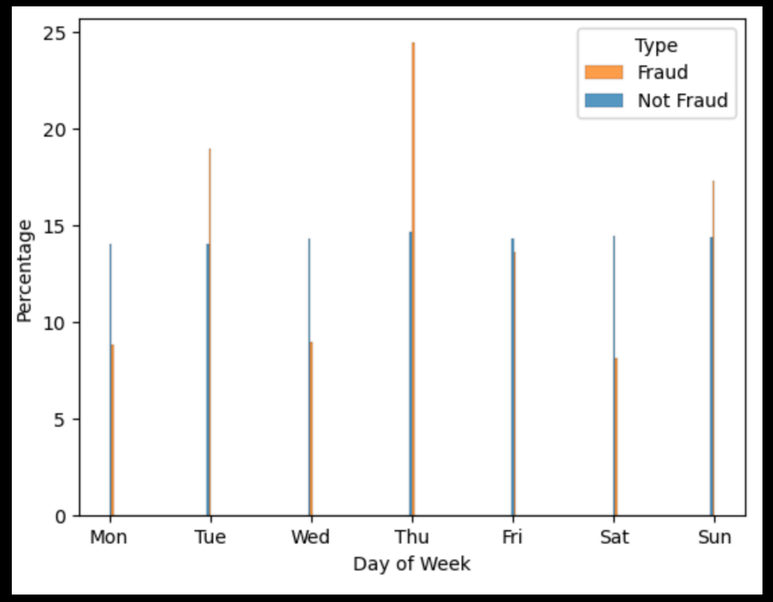
- The following are some of the basic characteristics of the 3 datasets:

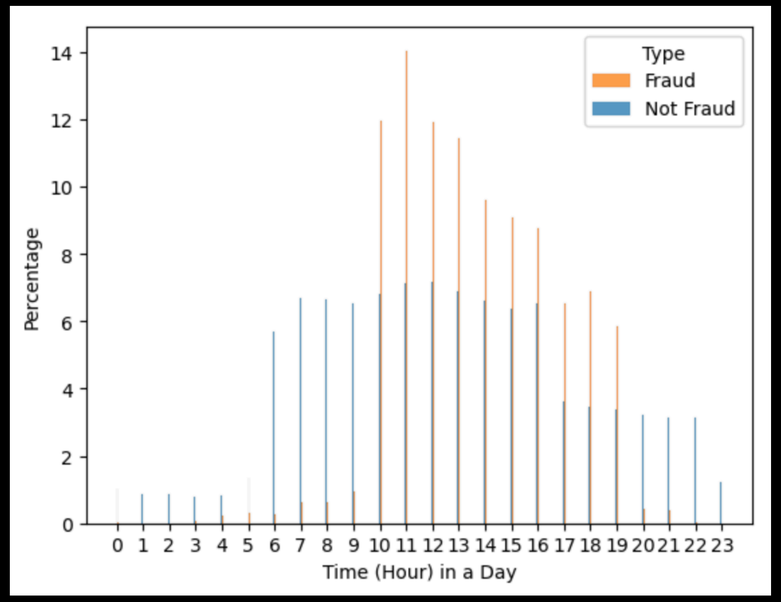
User Dataset	**Transaction Dataset**	**Credit Card Dataset**
2000 rows (index=User id)	6.86M Rows, 8412 fraud rows	6146 rows (user & card_index keys)
Missing : Apartment : 1472	Missing : Merchant State : 860764 rows ; Errors? : 6768768	Missing : NA

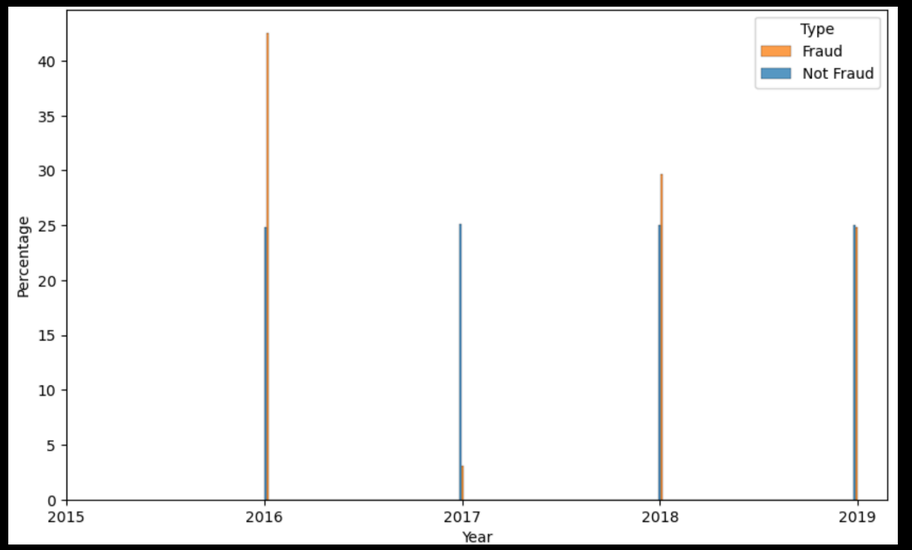
- Cleaning Operations : changing data types (stripping $ signs), Combining date parts columns

Feature Engineering + EDA

- Time related variables seemed to have a strong relation with when the frauds occur
- Variables in Users dataset like Current Age, Gender, Total Debt, Yearly Income - Person, Fico score don't appear to have much bearing on outcome



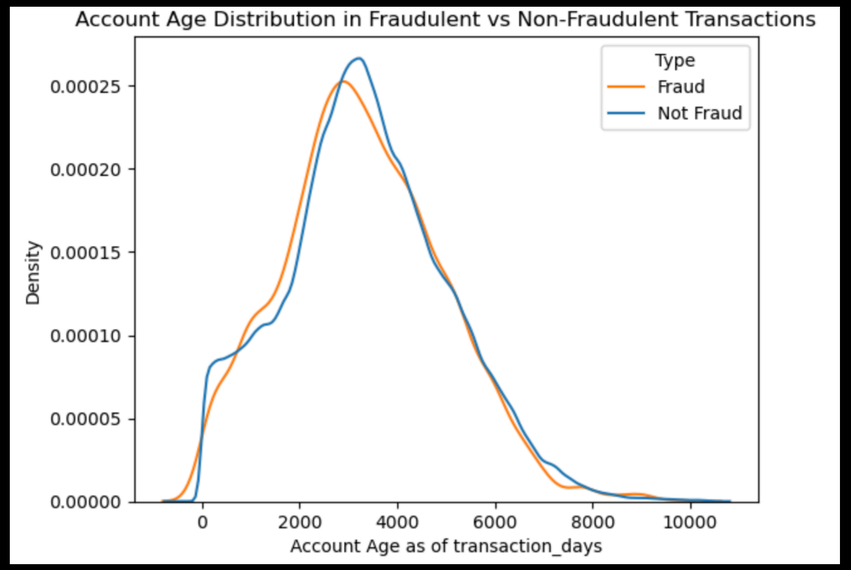


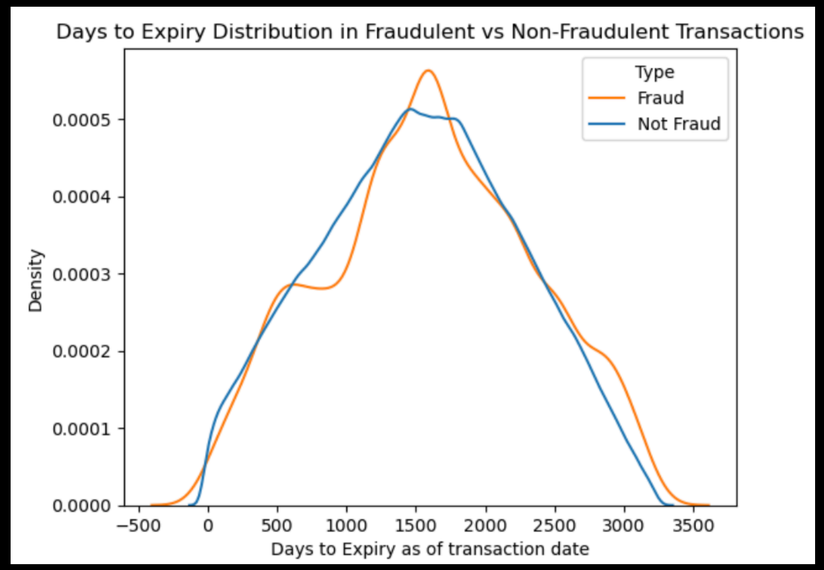


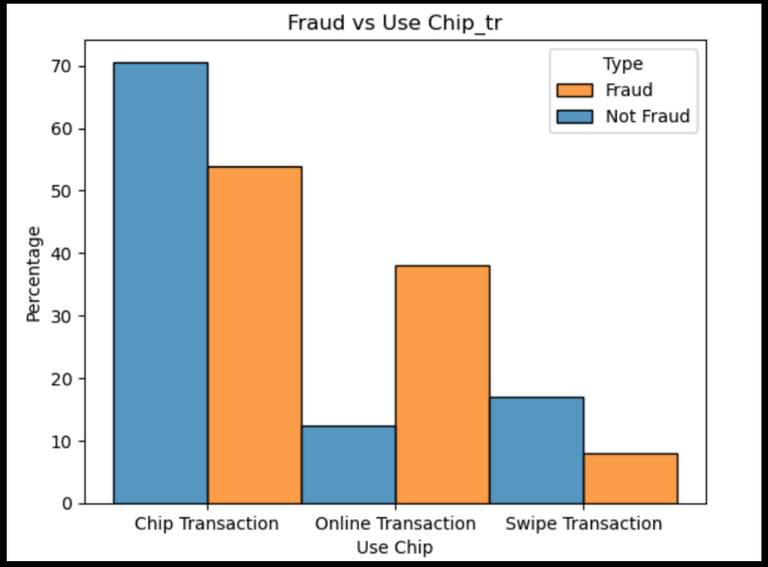
Feature Engineering + EDA (cont.)

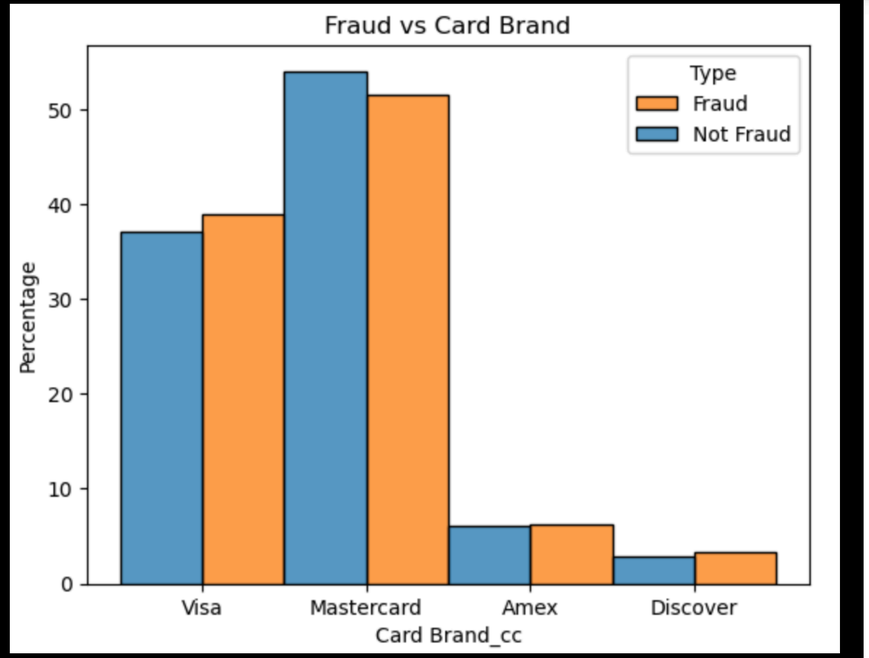
- Additional features / calculations explored:
  - Account age as of transaction date = Transaction Date - Accent created date
  - Days to expiry = Card expires date - Transaction date
  - Merchant City = User City
  - Merchant State = User State
  - Since a lot of blank Zip_tr (transaction zip codes) could be attributed to online transactions (860k out of 907k blank values), an additional feature was created 'Online transaction or not' (based on 'Use Chip_tr' field.
  - Calculating Zipcode distances (user & transaction zip codes using pgeocode library)
    - Created a Binary field 'Distance available'
    - Imputed missing Zip_tr with an outlier value so that it can be used as input to model

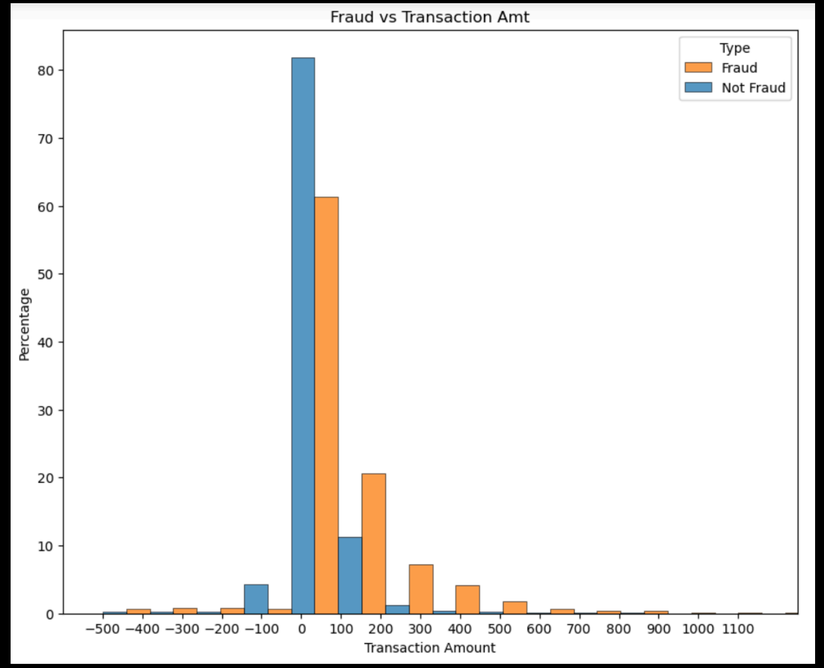
Feature Engineering + EDA (cont.)

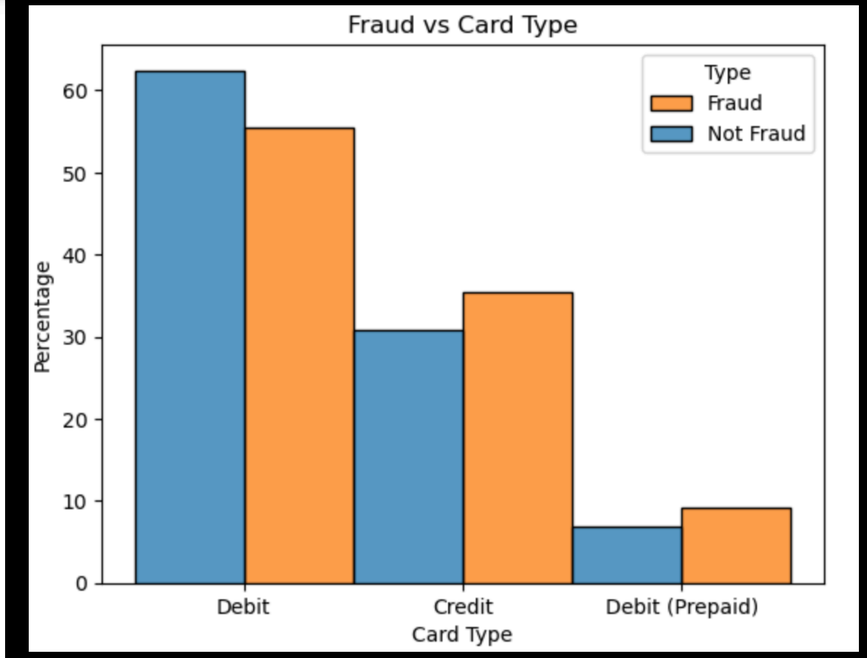




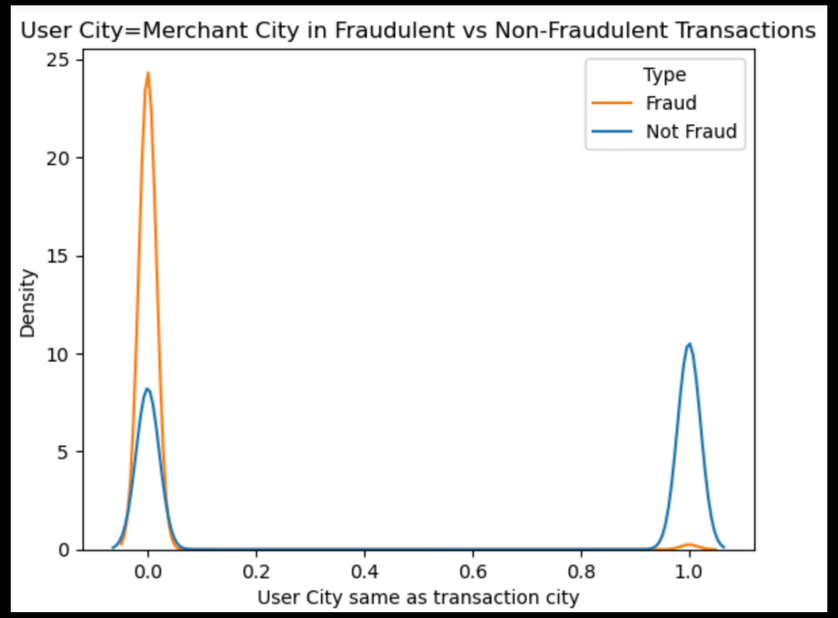


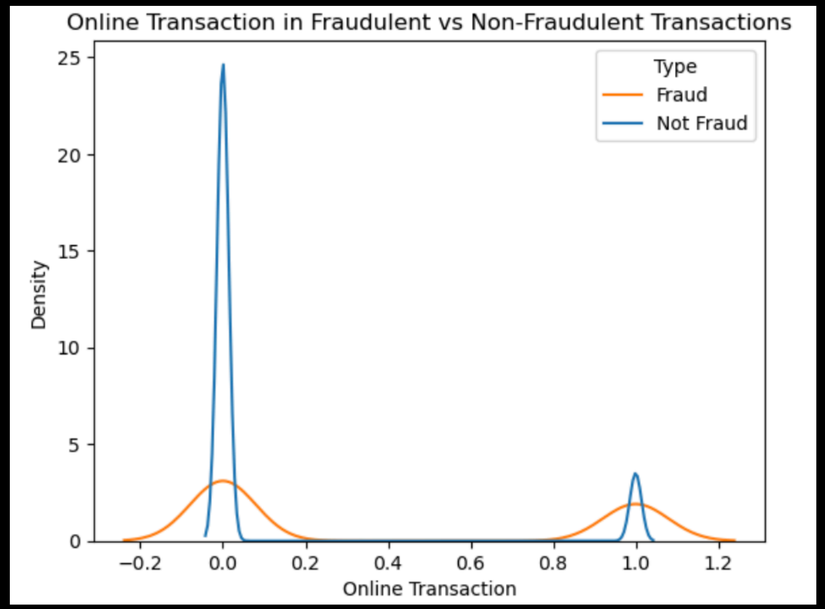


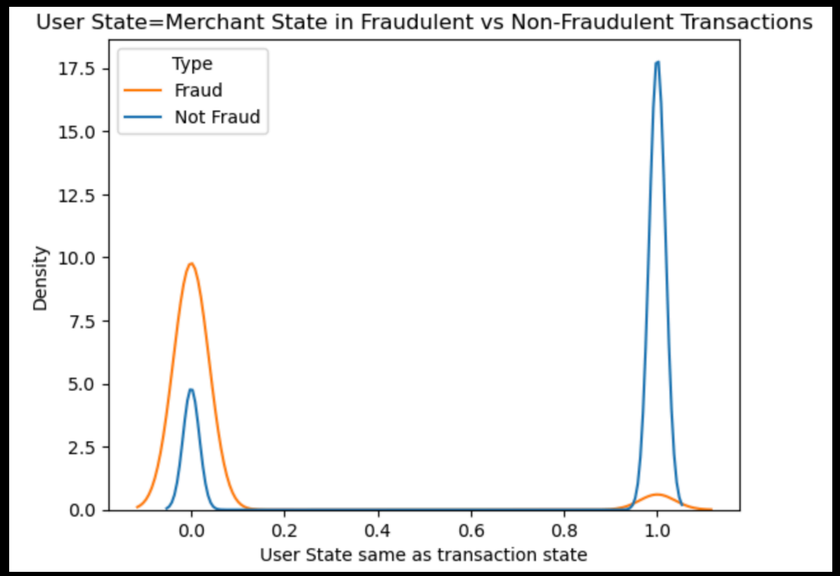


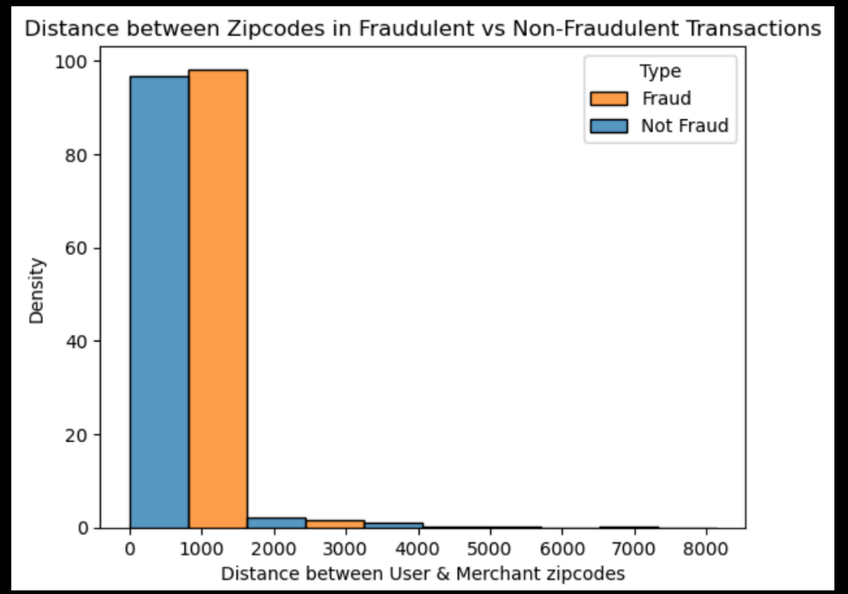


Feature Engineering + EDA (cont.)









Prepping the Dataset

Encoding Variables	• Low cardinality categorical variable with Binary Encoder (One hot) • High cardinality categorical variable with Label Encoder
Scaling & Resampling	• Using Standard Scalar for all numerical + encoded columns • Sub-sampling (5% data to train) • 343471 data points (240724 to train & rest for test/ validation) • Resampling using SMOTE
Train Test Split	• Using sklearn train_test_split method

Metrics

- Accuracy
- Precision
- Recall
- F1 Score
- ROC AUC Score

Confusion Matrix

	Predicted -ve	Predicted +ve
Actual -ve	TN	FP
Actual +ve	FN	TP

Fraud Detection

Baseline Models

Advanced Models

Baseline / Supervised Learning Models (Joined 3 datasets)

	Precision	Recall	F1-Score	Accuracy (Model)	ROC AUC Score (Model)
Logistic Regression	3%	73%	0.05	~95%	0.85
Random Forest	91%	56%	0.69	~100%	0.78
XGB Classifier	91%	67%	0.77	~100%	0.83
Light GBM Classifier	62%	67%	0.64	~100%	0.84
CatBoost Classifier	71%	76%	0.74	~100%	0.88
Decision Tree Classifier	54%	60%	0.57	~100%	0.8

Hyperparameter Tuning with Optuna

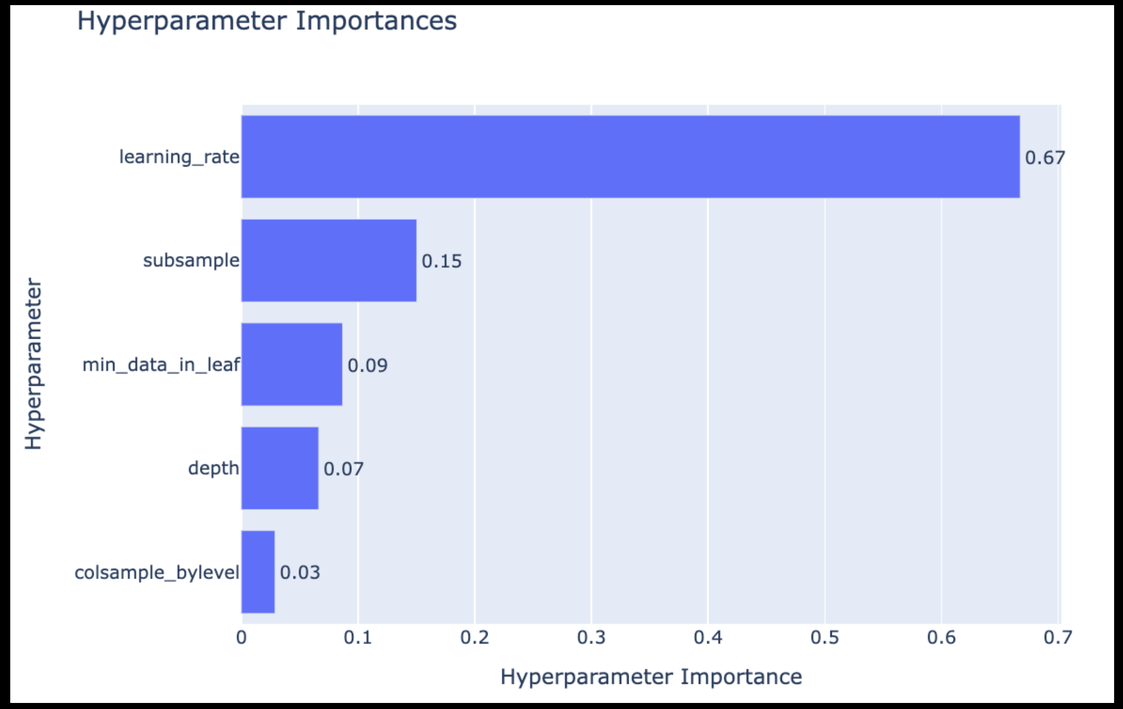
- Hyper Parameter Tuning on CatBoost

```
Catboost Results with Best Hyperparameters with Optuna:
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    103042
           1       0.77      0.72      0.75       126

    accuracy                           1.00    103168
   macro avg       0.89      0.86      0.87    103168
weighted avg       1.00      1.00      1.00    103168

Confuhttp://localhost:8888/notebooks/DSTrack_Fraud_Modeling-withallvar
rix:
 [[103015     27]
 [    35     91]]
Share of Non-Fraud in Test Data: 0.9988
F1 score of best XGB is 0.7459016393442622
Accuracy of best XGB is 0.9993990384615384
ROC-AUC Score:
0.8609800965733498
```

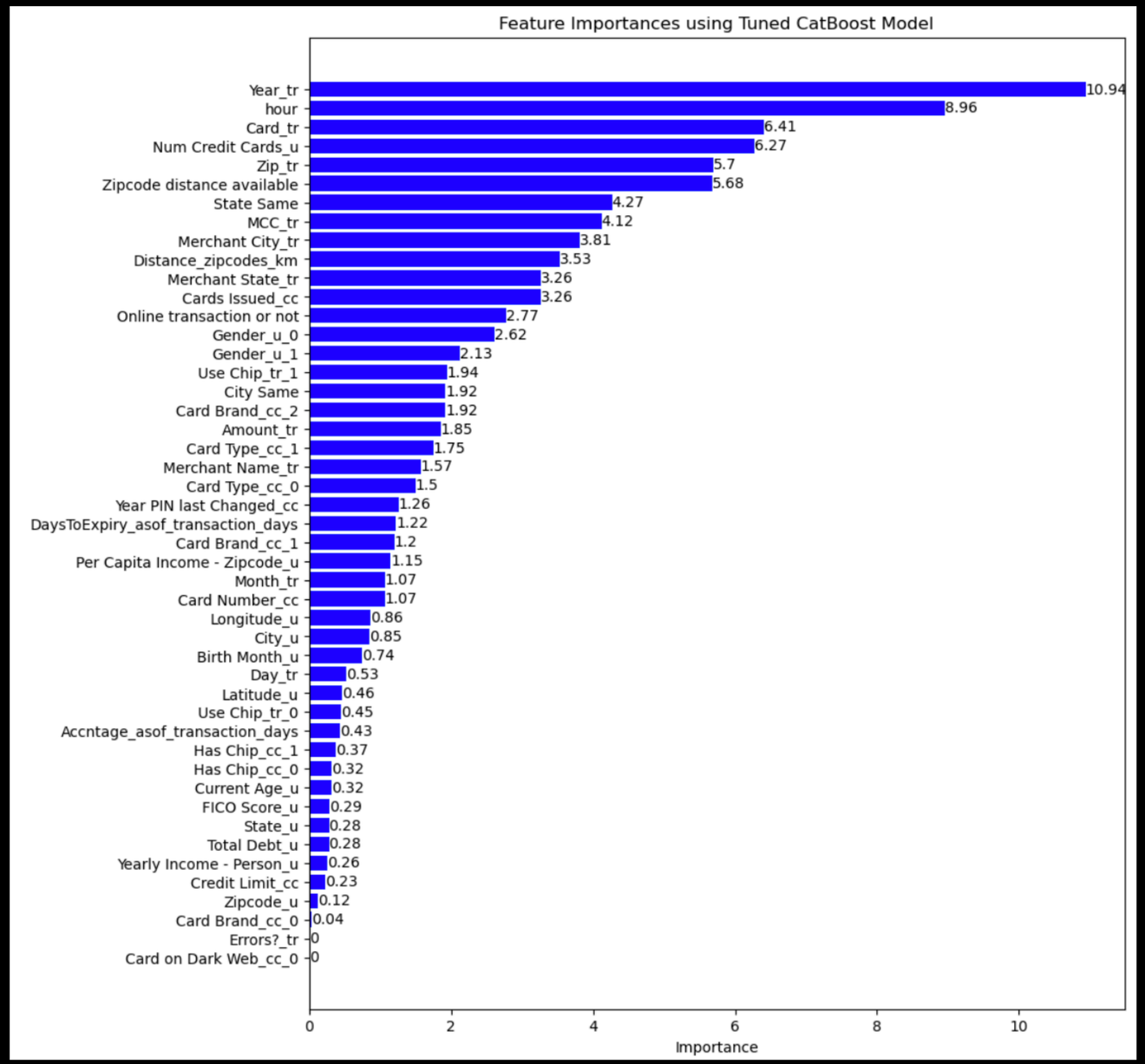
- Hyper Parameter Tuning on XGBoost

```
XGboost Results with Best Hyperparameters with Optuna:
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    103042
           1       0.82      0.71      0.76       126

    accuracy                           1.00    103168
   macro avg       0.91      0.85      0.88    103168
weighted avg       1.00      1.00      1.00    103168

Confuhttp://localhost:8888/notebooks/DSTrack_Fraud_Modeling-withallvariabl
 [[103022     20]
 [    37     89]]
Share of Non-Fraud in Test Data: 0.9988
F1 score of best XGB is 0.7574468085106384
Accuracy of best XGB is 0.999447503101737
ROC-AUC Score:
0.8530775553688541
```

Hyper parameter Importances



Feature Importances



Baseline / Supervised Learning Models (PCA dataset)

	Precision	Recall	F1-Score	Accuracy (Model)	ROC AUC Score (Model)
Logistic Regression	2%	82%	0.04	~95%	0.88
Random Forest	93%	59%	0.72	~100%	0.79
XGB Classifier	85%	63%	0.73	~100%	0.82
Light GBM Classifier	72%	71%	0.72	~100%	0.86
CatBoost Classifier	75%	73%	0.74	~100%	0.86
Decision Tree Classifier	63%	67%	0.65	~100%	0.84

Hyperparameter Tuning with Optuna (PCA model)

- Hyper Parameter Tuning on CatBoost

```
Catboost Results with Best Hyperparameters with Optuna:
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    103042
           1       0.78      0.75      0.76       126

    accuracy                           1.00    103168
   macro avg       0.89      0.87      0.88    103168
weighted avg       1.00      1.00      1.00    103168

Confuhttp://localhost:8888/notebooks/DSTrack_Fraud_Modeling-
rix:
 [[103016     26]
 [    32     94]]
Share of Non-Fraud in Test Data: 0.9988
F1 score of best XGB is 0.7642276422764228
Accuracy of best XGB is 0.9994378101736973
ROC-AUC Score:
0.8728897108683992
```

- Hyper Parameter Tuning on XGBoost

```
XGboost Results with Best Hyperparameters with Optuna:
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    103042
           1       0.77      0.78      0.77       126

    accuracy                           1.00    103168
   macro avg       0.88      0.89      0.89    103168
weighted avg       1.00      1.00      1.00    103168

Confuhttp://localhost:8888/notebooks/DSTrack_Fraud_Modeling-w
rix:
 [[103012     30]
 [    28     98]]
Share of Non-Fraud in Test Data: 0.9988
F1 score of best XGB is 0.7716535433070867
Accuracy of best XGB is 0.9994378101736973
ROC-AUC Score:
0.8887433171802652
```

Fraud Detection

Baseline Models

Advanced Models

Advanced Models

- Neural Networks (3 Dense layers , relu & sigmoid activation functions)

```
Confusion matrix:
[[103003     39]
 [    76     50]]
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    103042
           1       0.56      0.40      0.47       126

    accuracy                           1.00    103168
   macro avg       0.78      0.70      0.73    103168
weighted avg       1.00      1.00      1.00    103168

0.6982234551914877
```

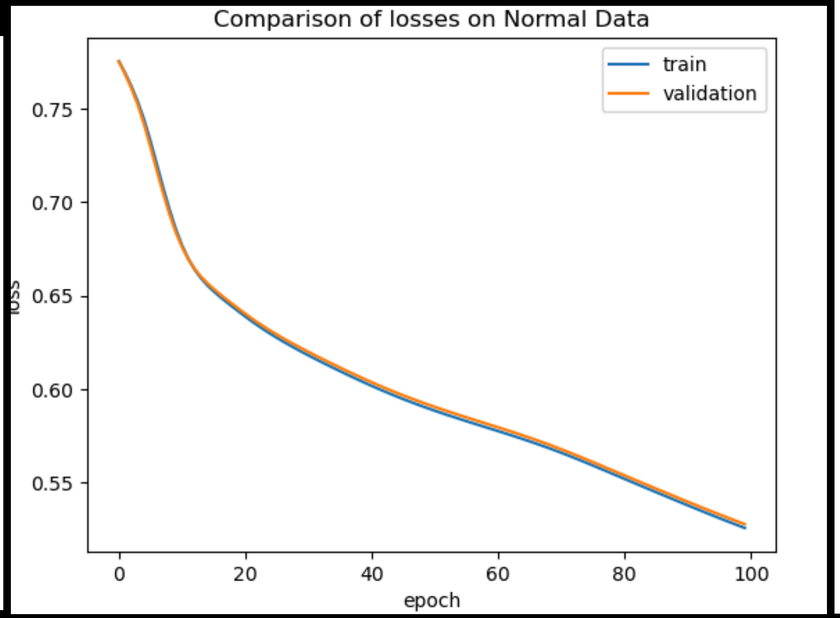
- Weighted Neural Networks

```
Confusion matrix:
[[98346  4696]
 [   10   116]]
Classification report:
              precision    recall  f1-score   support

   Not Fraud       1.00      0.95      0.98    103042
       Fraud       0.02      0.92      0.05       126

    accuracy                           0.95    103168
   macro avg       0.51      0.94      0.51    103168
weighted avg       1.00      0.95      0.98    103168

ROC AUC Score:
0.9375306355275688
```

Advanced Models (cont.)

- Auto Encoders
  - 343471 (421 fraud transactions only) subsample size divided into 240429 train & 103042 test size. Train has 295 fraud transactions & test has 126 fraud transactions.

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 32)]              0

 dense_11 (Dense)            (None, 20)                660

 dense_12 (Dense)            (None, 10)                210

 dense_13 (Dense)            (None, 10)                110

 dense_14 (Dense)            (None, 20)                220

 dense_15 (Dense)            (None, 32)                672

=================================================================
Total params: 1,872
Trainable params: 1,872
Non-trainable params: 0
_________________________________________________________________
```

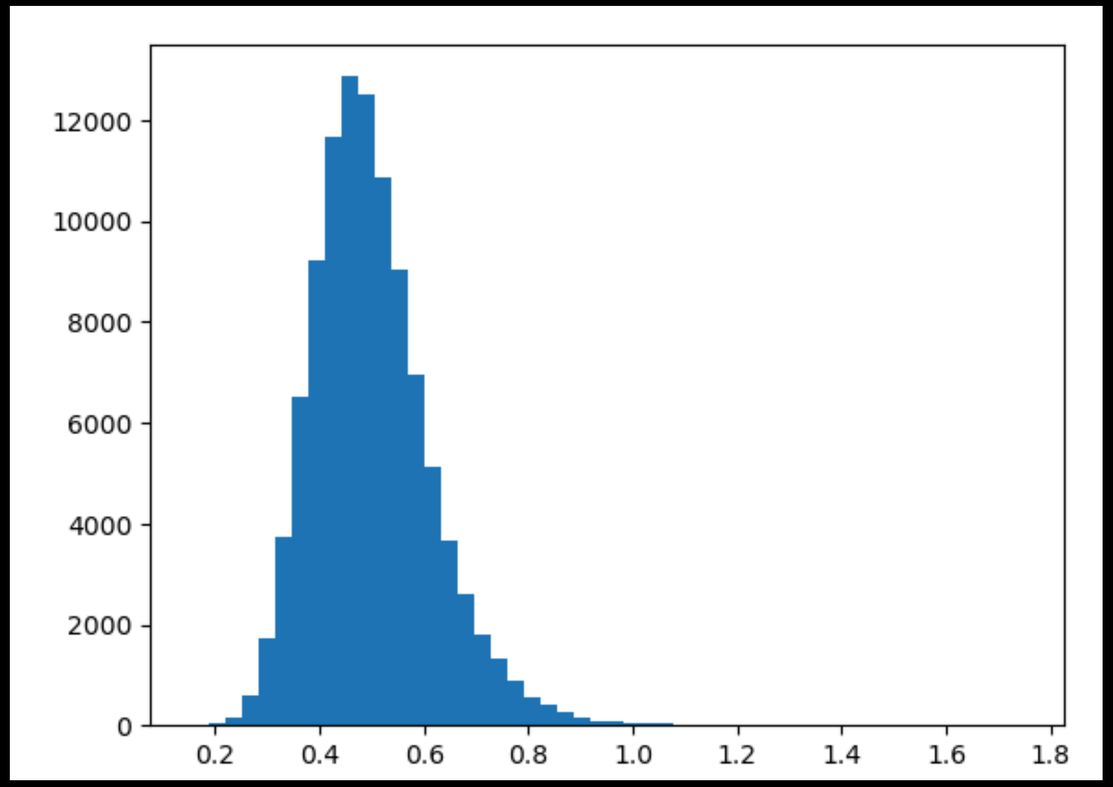



```
error_true=error[error['True_class']==0]
print("Normal Sample",error_true.describe())
error_false=error[error['True_class']==1]
print("Fraud Sample",error_false.describe())
```

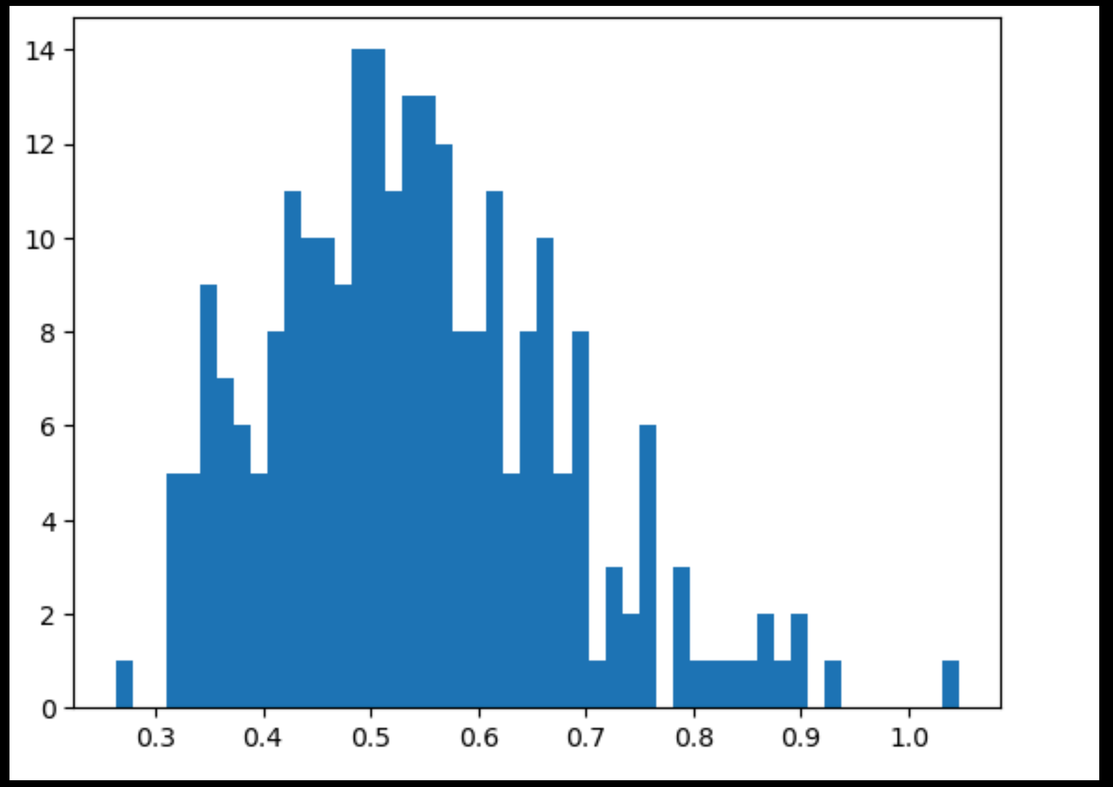
```
Normal Sample        reconstruction_error  True_class
count          103042.000000    103042.0
mean                0.523357         0.0
std                 0.411681         0.0
min                 0.045445         0.0
25%                 0.343578         0.0
50%                 0.450593         0.0
75%                 0.597401         0.0
max                31.781721         0.0
Fraud Sample         reconstruction_error  True_class
count             252.000000       252.0
mean                0.673267         1.0
std                 0.508295         0.0
min                 0.146181         1.0
25%                 0.377645         1.0
50%                 0.528578         1.0
75%                 0.802483         1.0
max                 5.086187         1.0
```

Advanced Models (Auto encoders cont.)

Losses on Test Class for Normal transactions



Losses on Test Class for Fraud transactions



This was further validated with t-SNE plots where the auto encoder arrangement did not do a good job of separating the two

Takeaways + Future Work

- Explainability Goes a Long Way…
- Feature Engineering not only makes the problem interesting or the results promising but helps one to understand a lot of intricate relationships & unpack patterns
- An interesting addition would be to consider all the transactions with Nan in transaction zip codes & populate them based on City + State values ; this might need a a different script accounting any API rate limits. This way we wouldn’t have to impute missing values. As seen previously, location difference plays a substantial role.
- With additional time, it would be interesting to use tSNE plots further to see if the feature engineering done or a better auto-encoder model could truly separate these transactions.