Credit Card Fraud Detection

Building a Machine learning model for detecting the fraud credit card transactions

Name : Siddhartha Pramanik

Objective

Credit card fraud is a form of identity theft in which criminals make purchases or obtain cash advances using a credit card account assigned to us. This can occur through one of your existing accounts, via theft of our physical credit card or our account numbers and PINs, or by means of new credit card accounts being opened in our name without our knowledge. Once they're in, thieves then run up charges and stick us and our credit card company with the bill.There were nearly 400,000 reports of credit card fraud reported to the Federal Trade Commission last year, a number that grew 44% from 2019 to 2020.

But observing the features , the fraud transactions can be distinguished . If we take some relevant data to train a ML model that can solve this problem up to a significant level . Here I have tried to do that

Data Collection

I have collected a relevant public dataset from kaggle which is a dataset of credit card transactions in the USA for the years 2019 and 2020. which has the following columns

```
data.columns
```

```
Index(['Unnamed: 0', 'cc_num', 'merchant', 'category', 'amt', 'first', 'last',
       'gender', 'street', 'city', 'state', 'zip', 'lat', 'long', 'city_pop',
       'job', 'trans_num', 'unix_time', 'merch_lat', 'merch_long', 'is_fraud',
       'year', 'hour', 'month', 'dayofweek', 'day', 'age_group',
       'latitude_difference', 'longitude_difference', 'dist'],
      dtype='object')
```

In this dataset , the target is the 'is_fraud' column , the other columns are the features .

Data Analysis and Visualization

This dataset contains imbalanced data .

```
data['is_fraud'].value_counts()
```

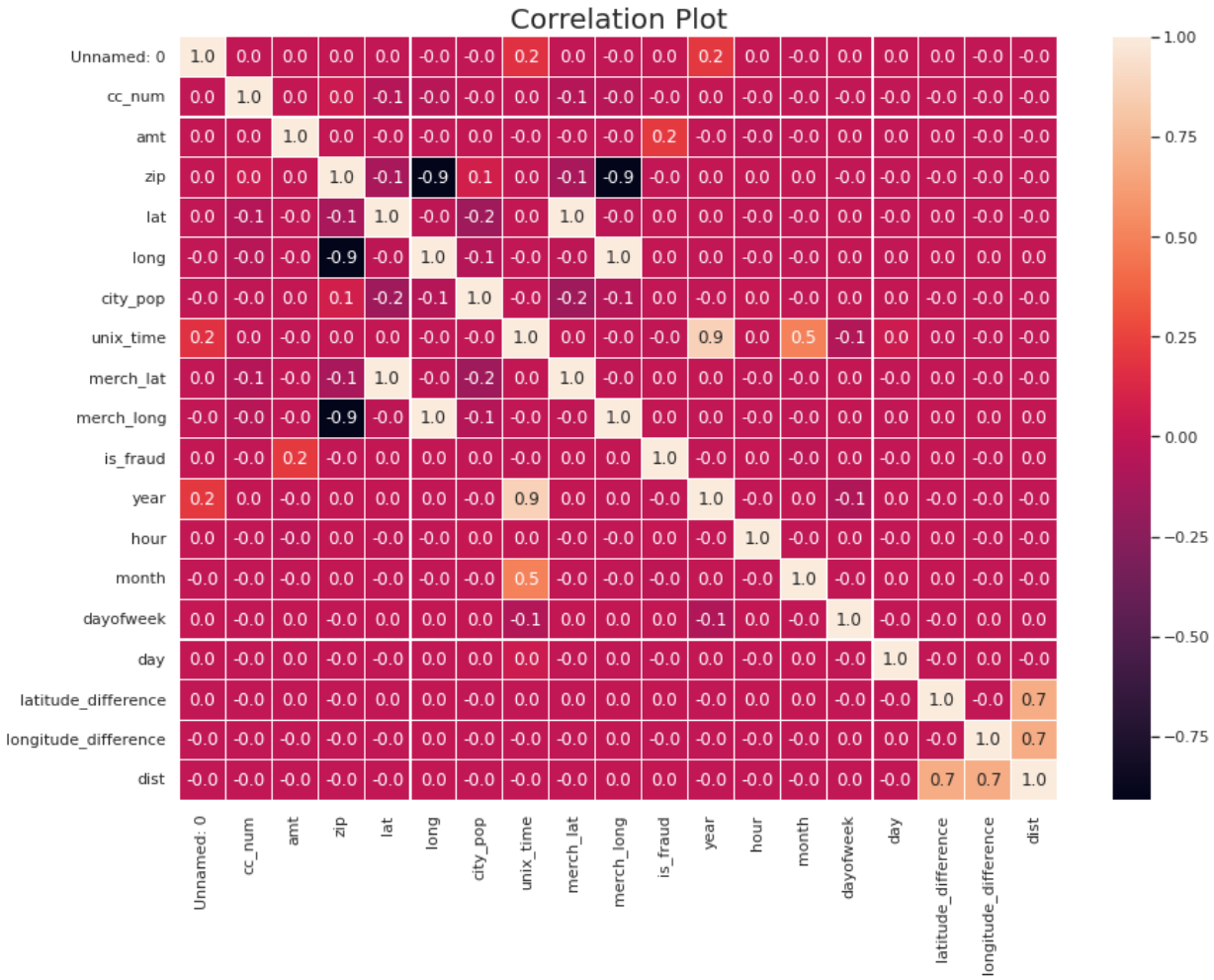
```
0    1842743
1       9651
Name: is_fraud, dtype: int64
```

We have to take care of this in the data preprocessing . But this does not have any null data

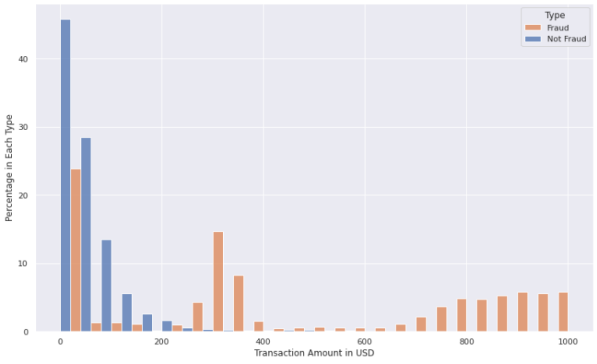
```
data.isnull().any().sum()
```

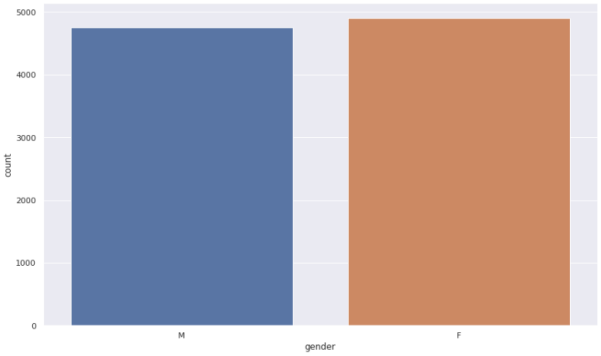
```
0
```

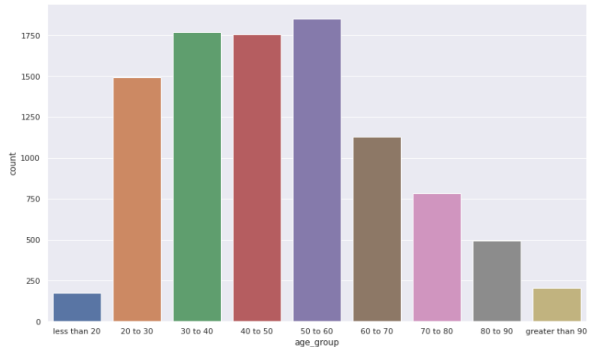
Using the visualizations we can see the relation between the columns and can understand which features are important .

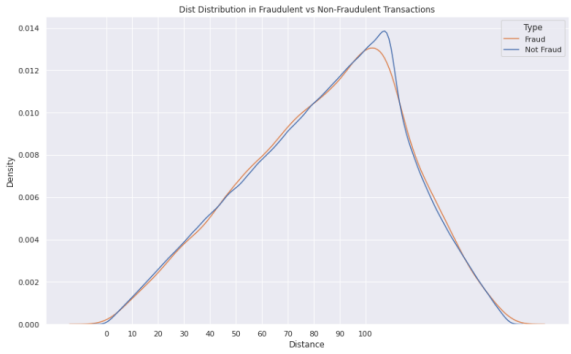


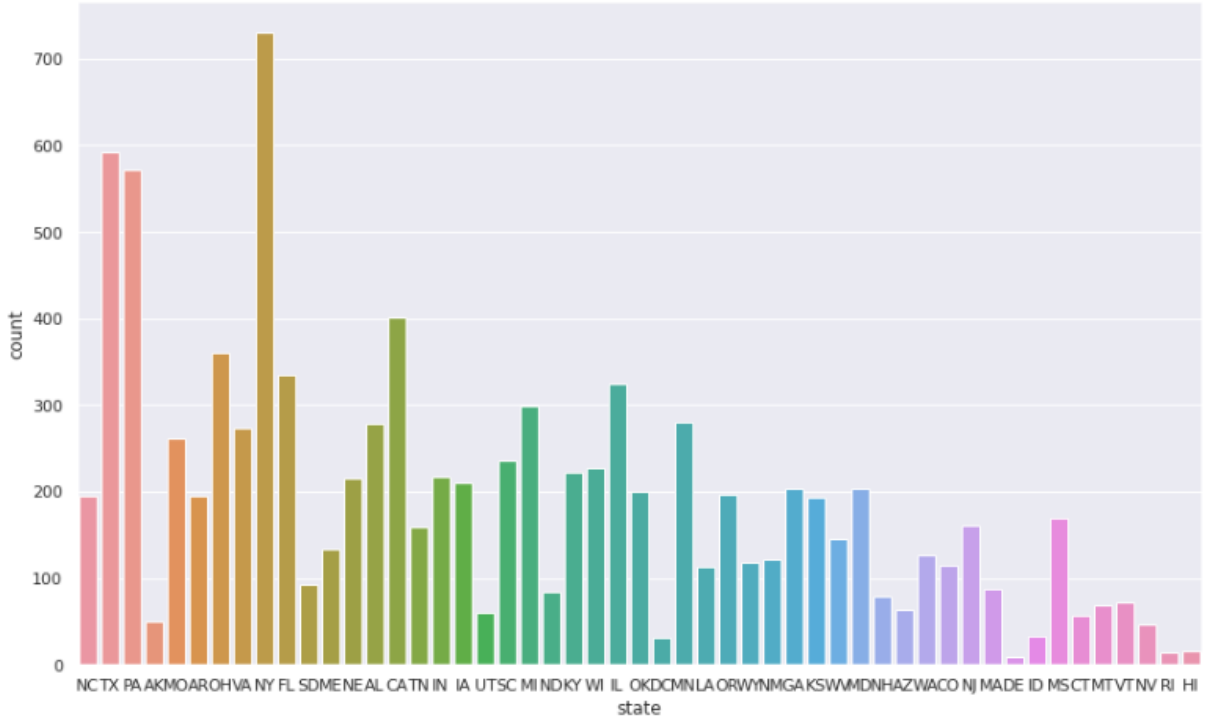
Also we can see the dependency on the amount , gender , age, distance and state in the graphs below











Data Pre-processing

There are so many columns which are not very relevant to the target so I have dropped them , After dropping them , the final columns are

```
data.columns
```

```
Index(['category', 'amt', 'gender', 'state', 'is_fraud', 'hour', 'month',
       'dayofweek', 'day', 'age_group', 'dist'],
      dtype='object')
```

As this data set is imbalanced, I have balanced it

```
non_fraud = data[data['is_fraud']==0]
fraud = data[data['is_fraud']==1]
```

```
non_fraud = non_fraud.sample(fraud.shape[0])
non_fraud.shape
```

```
(9651, 11)
```

```
[78] data = fraud.append(non_fraud, ignore_index=True)
```

```
[79] data['is_fraud'].value_counts()
```

```
1    9651
0    9651
Name: is_fraud, dtype: int64
```

As number of no fraud transactions is so much high , I have removed the extra non fraud transaction , This will prevent the model from being biased

I have applied one-hot encoding for some columns , to make them numerical data

```
print('One Hot Encoding is applied')
data = pd.concat([data,pd.get_dummies(data["gender"]),pd.get_dummies(data["age_group"]),pd.get_dummies(data["state"]),pd.get_dummies(data["category"])],axis=1)
data = data.drop(["gender","age_group","state","category"],1)
```

Also I have split the data into test and train using `train_test_split` function from 'sklearn'

```
y = data["is_fraud"]
x = data.drop(["is_fraud"], axis = 1)
# Standardization
scaler = StandardScaler()
x = scaler.fit_transform(x)
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 10)
```

Model Building

I have tried building the model using two libraries 'sklearn' and 'TensorFlow' .First I have used the `RandomForestClassifier` algorithm . I tried with different algorithms as well but this is giving me the best result .

For Tensorflow , I have created a sequential model with Conv1D and Dense layers

To stop overfitting , I have used `BatchNormalization and Dropout` Layers . Also I have used Adam Optimizer

```
epochs = 50
model = Sequential()
model.add(Conv1D(64, 2, activation='relu', input_shape = X_train[0].shape))
model.add(BatchNormalization())

model.add(Conv1D(32, 2, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(64, activation='relu'))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
```

I took 50 as my epoch number , But I used an EarlyStopping callback which is monitoring on Validation loss.

```
[97] callback = tf.keras.callbacks.EarlyStopping(
        monitor='val_loss',
        min_delta=0,
        patience=4,
        verbose=1,
        mode='auto',
        baseline=None,
        restore_best_weights=True
    )
    history = model.fit(X_train, y_train, epochs=epochs, validation_data=(X_test, y_test), verbose=1,callbacks = [callback])
```

Results and Conclusion

I evaluated the model using confusion matrix and classification report from 'sklearn' library

My 'sklearn' model is giving around 97% accuracy .

```
Confusion Matrix
 [[1904   37]
 [  65 1855]]
Classification Report
              precision    recall  f1-score   support

           0       0.97      0.98      0.97      1941
           1       0.98      0.97      0.97      1920

    accuracy                           0.97      3861
   macro avg       0.97      0.97      0.97      3861
weighted avg       0.97      0.97      0.97      3861
```

For the Tensorflow model , I have converted the prediction values to integers as the tensorflow predict function is giving float output , after that I applied the confusion matrix and classification report .

This model is giving around 96% accuracy.

```
[78] confusionMatrix
```

```
array([[1862,   79],
       [  93, 1827]])
```

```
print(classificationReport)
```

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.96      1941
           1       0.96      0.95      0.96      1920

    accuracy                           0.96      3861
   macro avg       0.96      0.96      0.96      3861
weighted avg       0.96      0.96      0.96      3861
```

Challenges

The main challenges that I faced are

Finding the proper data and finding the best algorithm for the model . Also I faced so many challenges that I learnt through this project

References

https://www.kaggle.com/

https://medium.com/