- **Data cleaning and Data Preparation**

1. First of all we had partitioned the dataset into numerical and categorical columns having the data-type as int/float and object respectively.
2. We converted the 'account opening date' column to an int data-type column by converting the cell elements to the number of days passed from an initial reference date . We kept the initial reference date as the previous day to the earliest date in the column. This also helps any learning model to capture sequential relations(if present) between the date and the account being a mule.
3. We converted the income column to two numerical columns containing the minimum income and the maximum income. NaN values were handled by first using a mask for the NaN value in obj data type (like using an integer not present in any min-max columns eg.'7') and then reconverting it to NaN value after splitting the columns.
4. We dropped 4 columns 'others_42','others_43','others_44' and 'others_45' because these columns had a large number of NaN values which were even more than the number of frauds so we thought that these columns carry little information. Moreover filling these values by imputation may also cause some false relations in the dataset due to the large number of fills.
5. As the dataset had many and randomly distributed NaN cells, we decided against dropping NaN values because some missing demographic information can sometimes be a cause of suspicion. So we replaced the categorical NaN values with the keyword 'Unknown'.
6. We label-encoded the categorical columns by **one hot encoding.**
7. Now that we have all the columns as numerical data type , we decided to adopt imputation method to fill the remaining empty cells in the numerical columns.For this we chose the model **RandomForestRegressor** to capture the relations in the cells having data and use this training to fill the Nan cells.
8. For deep learning we need to scale features to improve convergence,numerical stability and prevent bias , so we used **StandardScaler** to scale the features.
9. To handle binary imbalance we used **SMOTE resampling.**This helps remove the bias due to imbalance by synthetically generating samples for the minority class.
10. For the training of the Sequential model(discussed later) we had to reshape the data accordingly.

We have adopted the same methods (encoders and regressors) for both the files Dev-data and Validation data to make sure the same model fits well both.
Now that our data has been cleaned (all columns are numerical and there are no NaN cells), we proceed towards training .

- **Models and training**

Motivation and observations

1. There are 79 columns named as ‘txn_XX’ which refer to the transactions that the account holder made in these last 79 transactions. Now it is a common practice in banking to gain information about a customer from this **transaction history**. As these transactions are labeled as 1-79 ,we assumed that they might carry some information based on their sequenced values across the columns.
2. We had other columns like those containing demographic details and probably did not contain sequential data relation across the columns of the same row(sample).
3. Just as a trial run we had trained some **ensemble methods** of modeling which gave off good results.
4. The train dataset had a considerable binary imbalance having 2k out of 100k ones.So this needs to be taken care of during training and setting metrics otherwise the model may become biased towards the majority class during training.
5. Sometimes deep neural networks become too complex for the dataset and start capturing noise as data-relations so need **regularization**.
6. All the models in the ‘Details of the model’ section performed well individually on the dataset.

Details of the model

1. First of all we partitioned features of the dataset into two categories -'**Sequential features**' containing the 'txn_XX' columns and '**Non Sequential features'** containing the rest.
2. For the Sequential Features we trained an **LSTM model (long short term memory)**. LSTM s are a type of recurrent neural nets that are able to capture recent and distant relations with the past data. We are using the model to capture sequential relations to extract information out of a customer's transaction history. A snap of the model is given below.

```
n_features_seq=1
model_seq = tf.keras.Sequential([
    tf.keras.layers.LSTM(64, input_shape=(sequence_length, n_features_seq)),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(32, activation='relu'),

    tf.keras.layers.Dense(1, activation='sigmoid')
])
#model_seq is a LSTM model to capture the sequential relations and patterns in the transactions
model_seq.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy', tf.keras.metrics.Precision(),
                                                                         tf.keras.metrics.Recall()])
```

3. We are implementing Dropout to prevent the model from overfitting the dataset. Due to the binary imbalance in the data we are using our metrics as Precision and Recall because Accuracy does not give much information when the dataset is a large imbalanced dataset.We have tried activations functions such as LeakyReLu, tanh but **ReLu** showed better performance .
4. For the Non sequential dataset we are using **feed forward neural network** for training on the dataset to capture linear and nonlinear relations between the target and the features.A snap of the model is provided below.

```
model_non_seq = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_dim=X_non_seq_train_scaled.shape[1]),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
#model_non_seq to capture the patterns in non sequential data like demographic details, days of a
model_non_seq.compile(optimizer='adam', loss='binary_crossentropy',
                      metrics=['accuracy', tf.keras.metrics.Precision(), tf.keras.metrics.Recall()
                               ])
```

5. Here also we tried with different activation functions , and linear rate scheduler but best fitting combinations after fine tuning is the one shown above.Dropout layers to reduce overfitting.
6. Finally we trained the whole dataset(including sequential and non sequential columns) on an ensemble learning i.e. **XGBoost** .This model was also fine tuned for the dataset.

A snap of the model is shown below.

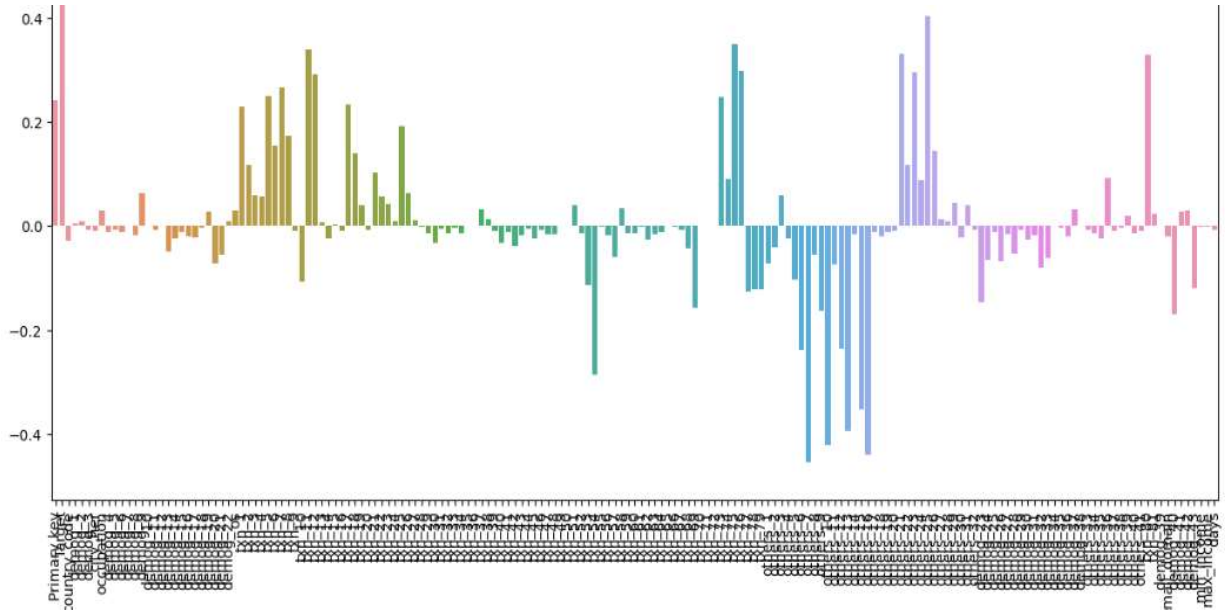
```
model = XGBClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=3,
    subsample=0.8,
    colsample_bytree=0.8,
    min_child_weight=1,
    gamma=0,
    random_state=42
)
```

7. So finally we took the probability scores from the three models and used their weighted average for calculating the final output probabilities. Now to obtain the optimum set of weights allotted to each probability score , we trained a multiple linear regression model against the features and the target . The final weights from the regression model are given below.

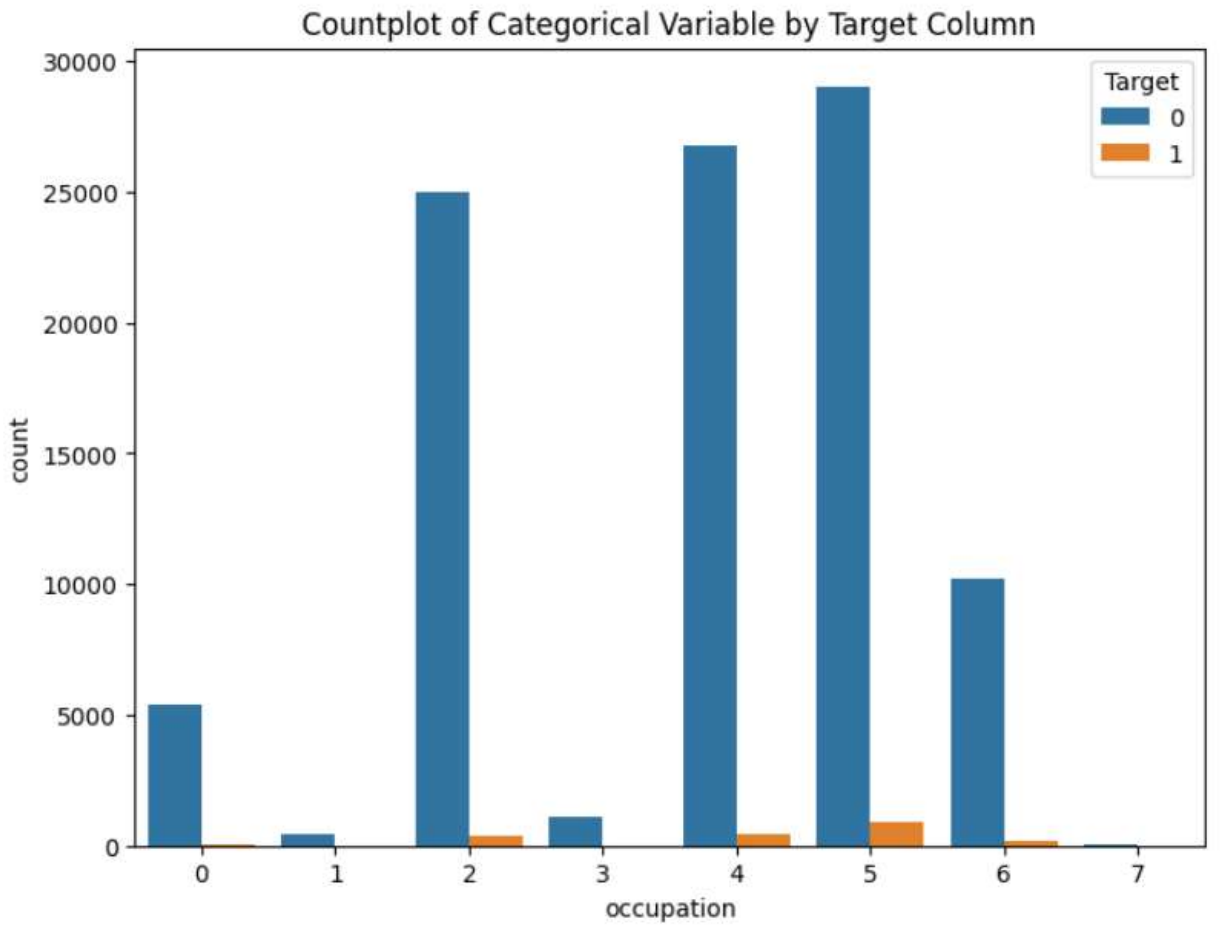
```
a1=0.1590 #sequential
a2=0.2727 #non sequential
a3=0.5682 #XGb
```

8. Rather than depending only on deep models for the classification we are also taking a weightage of the XGBoost classifier. This comes in handy in case the deep model overfits the training dataset and captures noise in the data even after taking proper methods to prevent overfitting .Basically if the deep models are capturing too complex features including noise it may create a problem however the ensemble method does not capture such intricate(possibly noisy) patterns, **thus overall decreasing the probability that comes just due to noise.**
9. Finally we replaced the probabilities of the columns with 0 , if both the minimum income and the maximum income is 0 . According to the P.S. for a mule account, it should have nonzero income into the account.

Insights:



The transaction part of the correlation plot focuses especially on the **analysis of start and final transactions as critical markers to differentiate between accounts that are fraudulent** and those that are not. Accounts that exhibit a regular and constant pattern of transactions, particularly in the beginning and end of the time series, are more likely to be connected to honest and non-fraudulent activity. On the other hand, a significant divergence, especially if it manifests itself as an abrupt increase in significant transactions at the conclusion of the time series, raises suspicions of possible theft. This emphasizes how important it is to evaluate the chronological progression of the first and last transactions in order to improve the accuracy of fraud detection algorithms.



0-homemaker
1-minor
2-other
3-retired
4-salaried
5-self employed
6-student

We can see a clear pattern here where 0 , 1 and 3 have minimal to no frauds whereas 2,4, 5 are the major occupations with frauds .the reasons being specifically 5 which is self employed is easier to fake while making an account .0 ,1 and 3 are less prone to fake verification details like you need a lot of verification to show retirement.