Overview

I. Define appropriate metrics
II. Visualization
III. Feature engineering
IV. Models
V. Conclusion and proposal

I. Define appropriate metrics

A. Precision

- Precision is a useful metric in cases where False Positive is a higher concern than False Negative.
- Precision (for class 1) is, out of all the predicted class values as 1 how many actually belong to the class
- **Precision** = **True Positive / (True Postive + False Positive)**

B. Recall

- Recall is a useful metric in cases where False Negative trumps False Positive.
- **Recall** tells us how many of the **actual positive** cases we were able to predict **correctly** with our model.
- **Precision** = **True Positive / (True Postive + False Positive)**

C. Accuracy

- The ratio of true predictions to total predictions
- **Accuracy = (TP + TN)/(TP + FP + TN + FN)**

D. F1 Score

- The harmonic mean of precision and recall
- **F1 Score = 2/(1/precison + 1/recall)**

The dataset is imbalanced with a ground truth of 5 values belonging to class 1 and 95 values belonging to class 0. And if our confusion matrix looks like this:

[[95, 0],

[5, 0]]

The accuracy is 95 % which looks great but the model is not working because it will predict all the values belonging to class 0.

That's the reason for precision and recall will be considered. To have a combined effect of precision and recall, F1 Score is the best choice

So F1-Score is the metric for this problem

II. Visualization

Please check "Visualization.ipynb" notebook to get more detail

III. Feature engineering

- Remove columns with 1 unique value: a constant value in that column across all observations this is not going to affect prediction (has nothing to teach the model about the target). Such features also can be removed. There are about 40 columns with the unique value, so they can be dropped from the data
- Missing value all in category features and number of missing data is not much. Fill NAN or missing values with 'Unknow' or with the highest frequency appearance in the column and remove the values with a frequency smaller threshold and assign them with other class
- One Hot Encoding will be used to deal with Category features (CatBoot model will not use one-hot encoding for category features)
- Drop out highly correlated features: Drop out highly correlated features: In many datasets, some of the features which are highly correlated means which are somewhat linearly dependent on other features. These features contribute very less to predicting the output but increase the computational cost.
- Oversampling with SMOTE: The SMOTE algorithm is one of the first and still the most popular algorithmic approach to generating new dataset samples, works by oversampling the underlying dataset with new synthetic points. The SMOTE algorithm is parameterized with `k_neighbors` (the number of nearest neighbors it will consider) and the number of new points you wish to create.
- As a final preprocessing step, The dataset will be imputed and scaled for numerical features. For Imputing, the median strategy will be used, i.e., missing values will be replaced by the median of their respective column. MinMax Scaling is also important to make sure that higher values are not necessarily given greater importance.
-

IV. Models

- All models will split 80% for training data, 20% for validation to choose hyperparameter, and save to CSV file using test data.

- K-fold strategy final results will be combined by k-models and the results label will count the most frequent result of the k-model.
- precision_recall_curve will be used to calculate F1-score and choose the best threshold
- The SMOTE was be tried but it was not good
- File submission will have two files, result model using 80% training data, 20% validation, and put the result on the test data, seconds file will use the K-fold strategy to combine the result of the k-model.

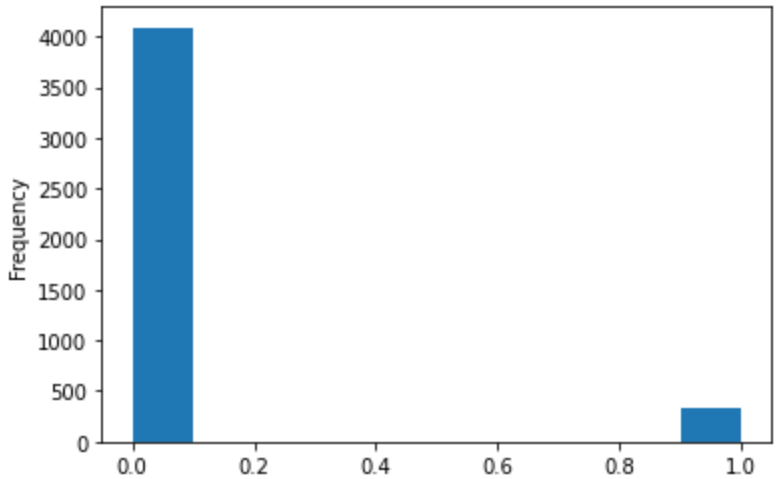
A. Logistic Regression

- Results split 80%, 20% training, and validation data. The result shown below is 20% validation data

```
threshold:  0.04865847923374146 Best_fscore: 0.26108374384236455
              precision    recall  f1-score   support

           0       0.96      0.92      0.94      2513
           1       0.20      0.36      0.26       146

    accuracy                           0.89      2659
   macro avg       0.58      0.64      0.60      2659
weighted avg       0.92      0.89      0.90      2659

Precision:       0.20384615384615384
Recall:          0.363013698630137
F1:              0.26108374384236455
Accuracy:        0.8871756299360662
Confusion matrix:
 [[2306  207]
 [  93   53]]
```

Test data submission chart

```
0.0    4089
1.0     333
Name: TARGET, dtype: int64
```



- Result 5-fold

```
FOLD: 1
threshold:  0.049673099899880786 Best_fscore: 0.23255813953488375
              precision    recall  f1-score   support

           0       0.97      0.86      0.91      2529
           1       0.15      0.50      0.23       130

    accuracy                           0.84      2659
   macro avg       0.56      0.68      0.57      2659
weighted avg       0.93      0.84      0.88      2659

Precision:
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s finished
<ipython-input-170-70176783c208>:4: RuntimeWarning: invalid value encountered in tru
  fscore = (2 * precision * recall) / (precision + recall)
```

```
 0.15151515151515152
Recall:          0.5
F1:              0.23255813953488375
Accuracy:        0.8386611508085746
Confusion matrix:
 [[2165  364]
 [  65   65]]
```

```
FOLD: 2
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s finished
```

```
threshold:  0.050251284275427986 Best_fscore: 0.24691358024691354
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      2528
           1       0.21      0.31      0.25       131

    accuracy                           0.91      2659
   macro avg       0.59      0.62      0.60      2659
weighted avg       0.93      0.91      0.92      2659

Precision:       0.20725388601036268
Recall:          0.3053435114503817
F1:              0.24691358024691354
Accuracy:        0.9082361790146671
Confusion matrix:
 [[2375  153]
 [  91   40]]
```

```
FOLD: 3
```

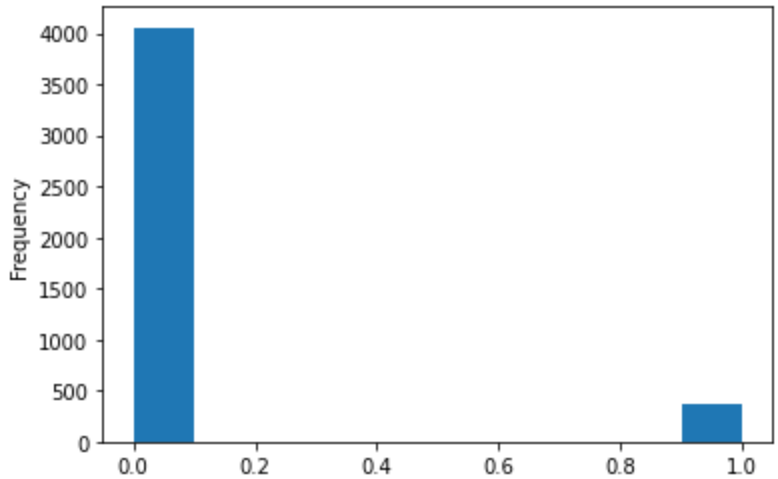
```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.2s finished
<ipython-input-170-70176783c208>:4: RuntimeWarning: invalid value encountered in true_d
  fscore = (2 * precision * recall) / (precision + recall)
```

```
threshold:  0.05010136607926262 Best_fscore: 0.22277227722772278
              precision    recall  f1-score   support

           0       0.96      0.91      0.94      2528
           1       0.16      0.34      0.22       131

    accuracy                           0.88      2659
   macro avg       0.56      0.63      0.58      2659
weighted avg       0.92      0.88      0.90      2659

Precision:       0.16483516483516483
Recall:          0.3435114503816794
F1:              0.22277227722772278
Accuracy:        0.881910492666416
Confusion matrix:
 [[2300  228]
 [  86   45]]
```

```
FOLD: 4
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s finished
<ipython-input-170-70176783c208>:4: RuntimeWarning: invalid value encountered in true_
  fscore = (2 * precision * recall) / (precision + recall)

threshold:  0.050299475781561714 Best_fscore: 0.20963172804532576
              precision    recall  f1-score   support

           0       0.96      0.93      0.94      2528
           1       0.17      0.28      0.21       130

    accuracy                           0.90      2658
   macro avg       0.56      0.61      0.58      2658
weighted avg       0.92      0.90      0.91      2658

Precision:       0.16591928251121077
Recall:          0.2846153846153846
F1:              0.20963172804532576
Accuracy:        0.8950338600451467
Confusion matrix:
 [[2342  186]
 [  93   37]]

FOLD: 5
threshold:  0.05001577775841227 Best_fscore: 0.23244552058111384
              precision    recall  f1-score   support

           0       0.97      0.91      0.94      2528
           1       0.17      0.37      0.23       130

    accuracy                           0.88      2658
   macro avg       0.57      0.64      0.58      2658
weighted avg       0.93      0.88      0.90      2658

Precision:       0.1696113074204947
Recall:          0.36923076923076925
F1:              0.23244552058111384
Accuracy:        0.8807373965387509
Confusion matrix:
 [[2293  235]
 [  82   48]]
```

Test data submission chart

```
0    4055
1     367
Name: TARGET, dtype: int64
```



B. Random Forest

- Results split 80%, 20% training, and validation data. The result shown below is 20% validation data

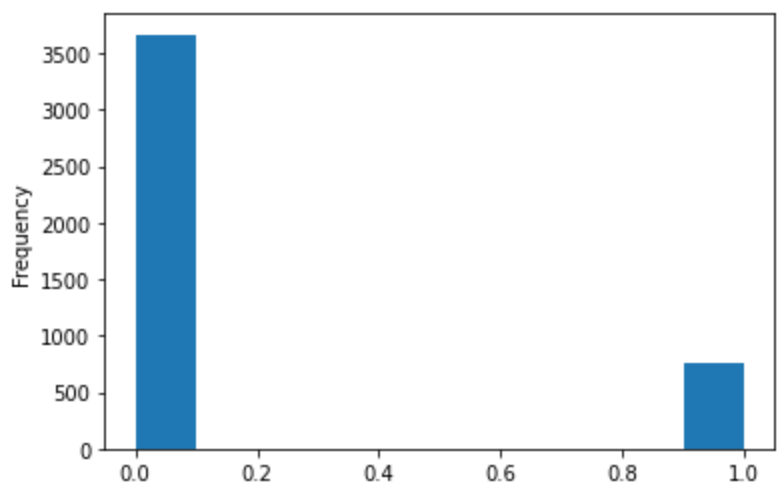
```
threshold:  0.11 Best_fscore: 0.2977941176470589
              precision    recall  f1-score   support

           0       0.97      0.87      0.92      2513
           1       0.20      0.55      0.30       146

    accuracy                           0.86      2659
   macro avg       0.59      0.71      0.61      2659
weighted avg       0.93      0.86      0.89      2659

Precision:       0.20351758793969849
Recall:          0.5547945205479452
F1:              0.2977941176470589
Accuracy:        0.8563369687852577
Confusion matrix:
 [[2196  317]
 [  65   81]]
```

Test data submission chart

```
threshold:  0.11 Best_fscore: 0.2977941176470589
              precision    recall  f1-score   support

           0       0.97      0.87      0.92      2513
           1       0.20      0.55      0.30       146

    accuracy                           0.86      2659
   macro avg       0.59      0.71      0.61      2659
weighted avg       0.93      0.86      0.89      2659

Precision:       0.20351758793969849
Recall:          0.5547945205479452
F1:              0.2977941176470589
Accuracy:        0.8563369687852577
Confusion matrix:
 [[2196  317]
 [  65   81]]
```

Test data submission chart

```
0.0    3668
1.0     754
Name: TARGET, dtype: int64
```



- Result 5-fold

```
FOLD: 1
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    0.5s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    1.3s finished
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent workers.
[Parallel(n_jobs=8)]: Done  34 tasks      | elapsed:    0.0s
[Parallel(n_jobs=8)]: Done 100 out of 100 | elapsed:    0.0s finished
<ipython-input-170-70176783c208>:4: RuntimeWarning: invalid value encountered in true_d
  fscore = (2 * precision * recall) / (precision + recall)
```

```
threshold:  0.07 Best_fscore: 0.21703296703296704
              precision    recall  f1-score   support

           0       0.98      0.79      0.88      2529
           1       0.13      0.61      0.22       130

    accuracy                           0.79      2659
   macro avg       0.55      0.70      0.55      2659
weighted avg       0.93      0.79      0.84      2659

Precision:       0.13210702341137123
Recall:          0.6076923076923076
F1:              0.21703296703296704
Accuracy:        0.7856336968785258
Confusion matrix:
 [[2010  519]
 [  51   79]]
```

```
FOLD: 2
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 8 concurrent worker
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    0.5s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    1.4s finished
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent workers
[Parallel(n_jobs=8)]: Done  34 tasks      | elapsed:    0.0s
[Parallel(n_jobs=8)]: Done 100 out of 100 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 8 concurrent worker
```

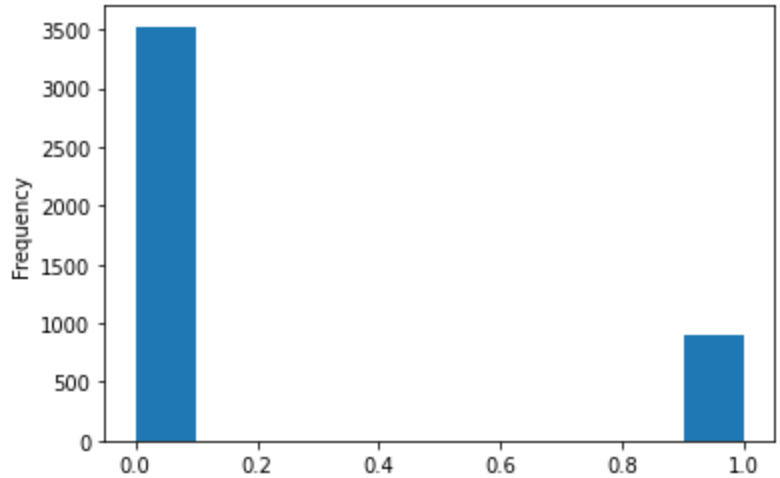
```
threshold:  0.12 Best_fscore: 0.2771855010660981
              precision    recall  f1-score   support

           0       0.97      0.89      0.93      2528
           1       0.19      0.50      0.28       131

    accuracy                           0.87      2659
   macro avg       0.58      0.69      0.60      2659
weighted avg       0.93      0.87      0.90      2659

Precision:       0.19230769230769232
Recall:          0.4961832061068702
F1:              0.2771855010660981
Accuracy:        0.8725084618277548
Confusion matrix:
 [[2255  273]
 [  66   65]]
```

```
FOLD: 3

[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    0.5s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    1.4s finished
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent workers.
[Parallel(n_jobs=8)]: Done  34 tasks      | elapsed:    0.0s
[Parallel(n_jobs=8)]: Done 100 out of 100 | elapsed:    0.0s finished
<ipython-input-170-70176783c208>:4: RuntimeWarning: invalid value encountered in true_
  fscore = (2 * precision * recall) / (precision + recall)

threshold:  0.09 Best_fscore: 0.2267343485617597
              precision    recall  f1-score   support

           0       0.97      0.84      0.90      2528
           1       0.15      0.51      0.23       131

    accuracy                           0.83      2659
   macro avg       0.56      0.68      0.57      2659
weighted avg       0.93      0.83      0.87      2659

Precision:       0.14565217391304347
Recall:          0.5114503816793893
F1:              0.2267343485617597
Accuracy:        0.8281308762692742
Confusion matrix:
 [[2135  393]
 [  64   67]]

FOLD: 4

[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    0.5s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    1.3s finished
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent workers.
[Parallel(n_jobs=8)]: Done  34 tasks      | elapsed:    0.0s
[Parallel(n_jobs=8)]: Done 100 out of 100 | elapsed:    0.0s finished
<ipython-input-170-70176783c208>:4: RuntimeWarning: invalid value encountered in true_
  fscore = (2 * precision * recall) / (precision + recall)

threshold:  0.1 Best_fscore: 0.23486238532110096
              precision    recall  f1-score   support

           0       0.97      0.86      0.91      2528
           1       0.15      0.49      0.23       130

    accuracy                           0.84      2658
   macro avg       0.56      0.68      0.57      2658
weighted avg       0.93      0.84      0.88      2658

Precision:       0.15421686746987953
Recall:          0.49230769230769234
F1:              0.23486238532110096
Accuracy:        0.8431151241534989
Confusion matrix:
 [[2177  351]
 [  66   64]]
```

```
FOLD: 5

[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 8 concurrent wo
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    0.5s

threshold:  0.13 Best_fscore: 0.24942263279445728
              precision    recall  f1-score   support

           0       0.97      0.90      0.93      2528
           1       0.18      0.42      0.25       130

    accuracy                           0.88      2658
   macro avg       0.57      0.66      0.59      2658
weighted avg       0.93      0.88      0.90      2658

Precision:       0.1782178217821782
Recall:          0.4153846153846154
F1:              0.24942263279445728
Accuracy:        0.8777276147479308
Confusion matrix:
 [[2279  249]
 [  76   54]]
```

Test data submission chart

```
0    3525
1     897
Name: TARGET, dtype: int64
```



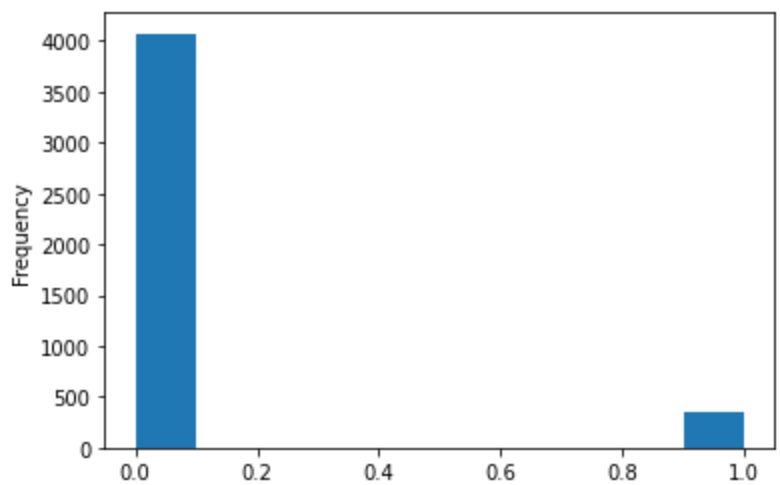
C. CatBoost

- Results split 80%, 20% training, and validation data. The result shown below is 20% validation data

```
Report result
threshold:  0.16755685920112623 Best_fscore: 0.3679525222551929
              precision    recall  f1-score   support

           0       0.97      0.95      0.96      2513
           1       0.32      0.42      0.37       146

    accuracy                           0.92      2659
   macro avg       0.65      0.69      0.66      2659
weighted avg       0.93      0.92      0.92      2659

Precision:       0.32460732984293195
Recall:          0.4246575342465753
F1:              0.3679525222551929
Accuracy:        0.919894697254607
Confusion matrix:
 [[2384  129]
 [  84   62]]
```

Test data submission chart

```
0.0    4073
1.0     349
Name: TARGET, dtype: int64
```



- Result 5-fold

```
FOLD: 1
Report result
threshold:  0.1763215101121 Best_fscore: 0.3037974683544304
              precision    recall  f1-score   support

           0       0.97      0.95      0.96      2529
           1       0.26      0.37      0.30       130

    accuracy                           0.92      2659
   macro avg       0.61      0.66      0.63      2659
weighted avg       0.93      0.92      0.92      2659

Precision:       0.25806451612903225
Recall:          0.36923076923076925
F1:              0.3037974683544304
Accuracy:        0.9172621286197818
Confusion matrix:
 [[2391  138]
 [  82   48]]
                        End report
```

```
FOLD: 2

<ipython-input-61-00f45e75f04f>:6: RuntimeWarning: invalid value encountered in true_
  fscore = (2 * precision * recall) / (precision + recall)

Report result
threshold:  0.08965553176528362 Best_fscore: 0.28515625
              precision    recall  f1-score   support

           0       0.97      0.88      0.92      2528
           1       0.19      0.56      0.29       131

    accuracy                           0.86      2659
   macro avg       0.58      0.72      0.60      2659
weighted avg       0.94      0.86      0.89      2659

Precision:       0.19160104986876642
Recall:          0.5572519083969466
F1:              0.28515625
Accuracy:        0.8623542685220007
Confusion matrix:
 [[2220  308]
 [  58   73]]
```

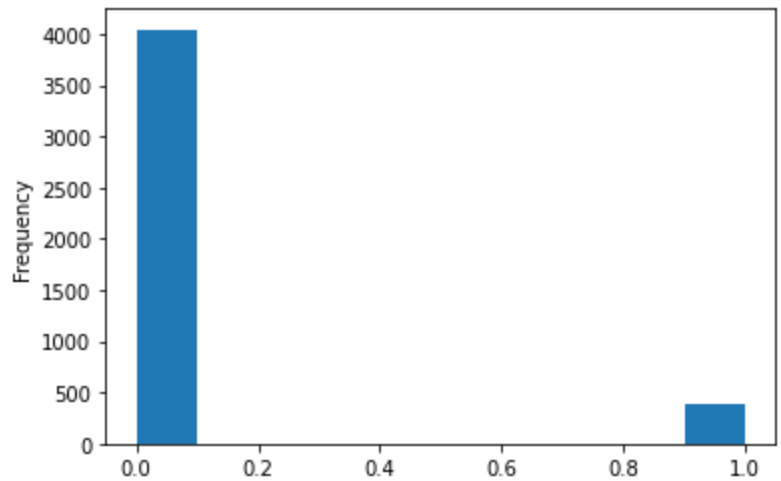
```
FOLD: 3

<ipython-input-61-00f45e75f04f>:6: RuntimeWarning: invalid value encountered in tru
  fscore = (2 * precision * recall) / (precision + recall)

Report result
threshold:  0.12864603338216116 Best_fscore: 0.289044289044289
              precision    recall  f1-score   support

           0       0.97      0.91      0.94      2528
           1       0.21      0.47      0.29       131

    accuracy                           0.89      2659
   macro avg       0.59      0.69      0.61      2659
weighted avg       0.93      0.89      0.91      2659

Precision:       0.2080536912751678
Recall:          0.4732824427480916
F1:              0.289044289044289
Accuracy:        0.8852952237683339
Confusion matrix:
 [[2292  236]
 [  69   62]]
```

```
FOLD: 4

<ipython-input-61-00f45e75f04f>:6: RuntimeWarning: invalid value encountered in true_
  fscore = (2 * precision * recall) / (precision + recall)

Report result
threshold:  0.18450717951415052 Best_fscore: 0.30597014925373134
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      2528
           1       0.30      0.32      0.31       130

    accuracy                           0.93      2658
   macro avg       0.63      0.64      0.63      2658
weighted avg       0.93      0.93      0.93      2658

Precision:       0.2971014492753623
Recall:          0.3153846153846154
F1:              0.30597014925373134
Accuracy:        0.9300225733634312
Confusion matrix:
 [[2431   97]
 [  89   41]]
```

```
FOLD: 5

<ipython-input-61-00f45e75f04f>:6: RuntimeWarning: invalid value encountered in true
  fscore = (2 * precision * recall) / (precision + recall)

Report result
threshold:  0.19238563762979952 Best_fscore: 0.3197026022304833
              precision    recall  f1-score   support

           0       0.97      0.96      0.96      2528
           1       0.31      0.33      0.32       130

    accuracy                           0.93      2658
   macro avg       0.64      0.65      0.64      2658
weighted avg       0.93      0.93      0.93      2658

Precision:       0.30935251798561153
Recall:          0.33076923076923076
F1:              0.3197026022304833
Accuracy:        0.9311512415349887
Confusion matrix:
 [[2432   96]
 [  87   43]]
```

Test data submission chart

```
0    4044
1     378
Name: TARGET, dtype: int64
```



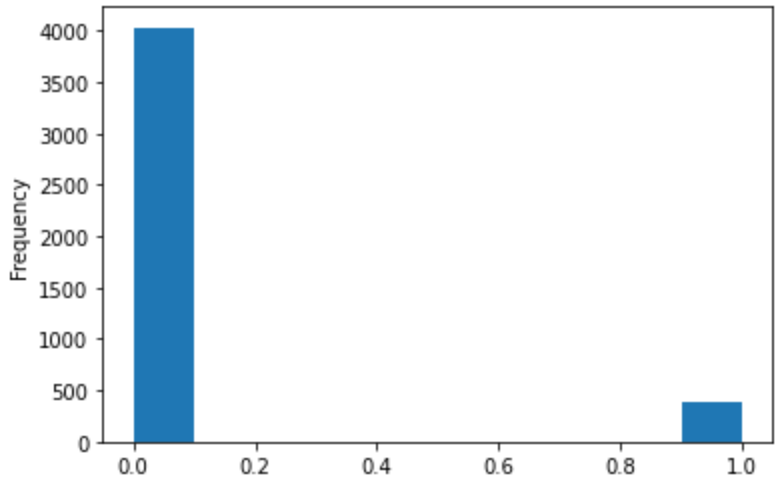
D. ANN Pytorch

- Results split 80%, 20% training, and validation data. The result shown below is 20% validation data

```
threshold:  0.099080645 Best_fscore: 0.29444444444444445
              precision    recall  f1-score   support

         0.0       0.96      0.94      0.95      2513
         1.0       0.25      0.36      0.29       146

    accuracy                           0.90      2659
   macro avg       0.60      0.65      0.62      2659
weighted avg       0.92      0.90      0.91      2659
```

-
- Test data submission chart

```
0.0    4032
1.0     390
Name: TARGET, dtype: int64
```



V. Conclusion and proposal

- Conclusion:

- The dataset is an imbalance between low risk and high-risk (products with low risk of about 95%, products high risk of about 5%) => Model in the training step can be easy bias so the more technical tunning and augmentation data with high risk have been needed
- There are about 40 columns with the unique value
- The category features are diverse and there are some valuable features with frequency only one time
- Missing value all in category features and number of missing data is not much
- The features with the same prefix such as: product_defect_rate_30, product_defect_rate_60, product_defect_rate_90, product_defect_rate_150, and so on are closely related to each other

- proposal:

- Collection of more data with high risk
- Find similarities between high risk and low-risk data and remove low-risk data with similarities with high risk to balance the data

- Needed domain knowledge to have a better understanding of the meaning behind the values of each column and reduce the number of observations by using under-sampling of the majority group so that it becomes equal to the number of observations of the minority group.