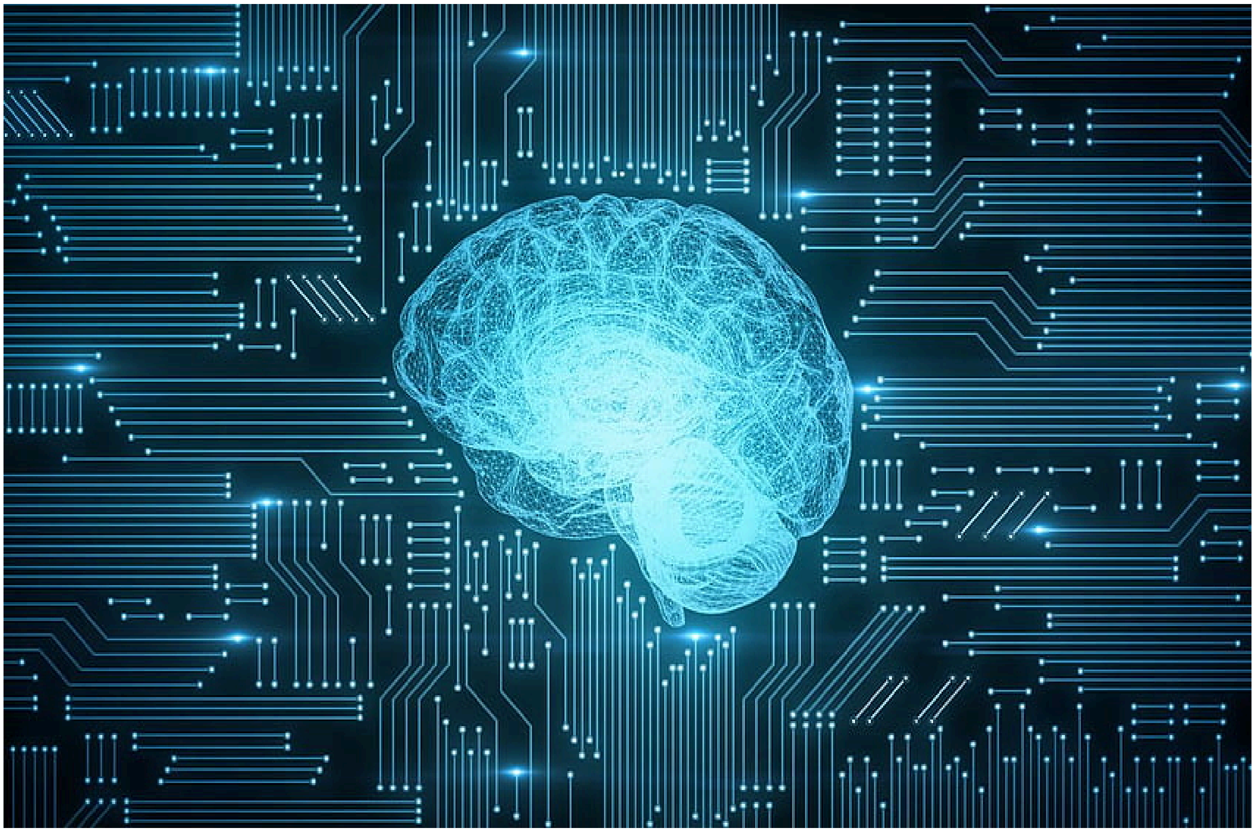
Logistic Regression



INTRODUCTION

The initial data looked like this :

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0
5	2.0	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	0.260314	-0.568671	...	-0.208254	-0.559825	-0.026398	-0.371427	-0.232794	0.105915	0.253844	0.081080	3.67	0
6	4.0	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	0.081213	0.464960	...	-0.167716	-0.270710	-0.154104	-0.780055	0.750137	-0.257237	0.034507	0.005168	4.99	0
7	7.0	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	-3.807864	0.615375	...	1.943465	-1.015455	0.057504	-0.649709	-0.415267	-0.051634	-1.206921	-1.085339	40.80	0
8	7.0	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.851084	-0.392048	...	-0.073425	-0.268092	-0.204233	1.011592	0.373205	-0.384157	0.011747	0.142404	93.20	0
9	9.0	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	0.069539	-0.736727	...	-0.246914	-0.633753	-0.120794	-0.385050	-0.069733	0.094199	0.246219	0.083076	3.68	0
10	10.0	1.449044	-1.176339	0.913860	-1.375667	-1.971383	-0.629152	-1.423236	0.048456	-1.720408	...	-0.009302	0.313894	0.027740	0.500512	0.251367	-0.129478	0.042850	0.016253	7.80	0
11	10.0	0.384978	0.616109	-0.874300	-0.094019	2.924584	3.317027	0.470455	0.538247	-0.558895	...	0.049924	0.238422	0.009130	0.996710	-0.767315	-0.492208	0.042472	-0.054337	9.99	0
12	10.0	1.249999	-1.221637	0.383930	-1.234899	-1.485419	-0.753230	-0.689405	-0.227487	-2.094011	...	-0.231809	-0.483285	0.084668	0.392831	0.161135	-0.354990	0.026416	0.042422	121.50	0
13	11.0	1.069374	0.287722	0.828613	2.712520	-0.178398	0.337544	-0.096717	0.115982	-0.221083	...	-0.036876	0.074412	-0.071407	0.104744	0.548265	0.104094	0.021491	0.021293	27.50	0
14	12.0	-2.791855	-0.327771	1.641750	1.767473	-0.136588	0.807596	-0.422911	-1.907107	0.755713	...	1.151663	0.222182	1.020586	0.028317	-0.232746	-0.235557	-0.164778	-0.030154	58.80	0
15	12.0	-0.752417	0.345485	2.057323	-1.468643	-1.158394	-0.077850	-0.608581	0.003603	-0.436167	...	0.499625	1.353650	-0.256573	-0.065084	-0.039124	-0.087086	-0.180998	0.129394	15.99	0
16	12.0	1.103215	-0.040296	1.267332	1.289091	-0.735997	0.288069	-0.586057	0.189380	0.782333	...	-0.024612	0.196002	0.013802	0.103758	0.364298	-0.382261	0.092809	0.037051	12.99	0
17	13.0	-0.436905	0.918966	0.924591	-0.727219	0.915679	-0.127867	0.707642	0.087962	-0.665271	...	-0.194796	-0.672638	-0.156858	-0.888386	-0.342413	-0.049027	0.079692	0.131024	0.89	0
18	14.0	-5.401258	-5.450148	1.186305	1.736239	3.049106	-1.763406	-1.559738	0.160842	1.233090	...	-0.503600	0.984460	2.458589	0.042119	-0.481631	-0.621272	0.392053	0.949594	46.80	0
19	15.0	1.492936	-1.029346	0.454795	-1.438026	-1.555434	-0.720961	-1.080664	-0.053127	-1.978682	...	-0.177650	-0.175074	0.040002	0.295814	0.332931	-0.220385	0.022298	0.007602	5.00	0

20 rows × 31 columns

The **TIME column** was just like the serial number and showed the number of seconds for the transaction since the first was made and added no value, hence it was removed.

CORRELATION MATRIX :

The correlation matrix on keen observation will show that no features are extremely related with each other and with the target variable.

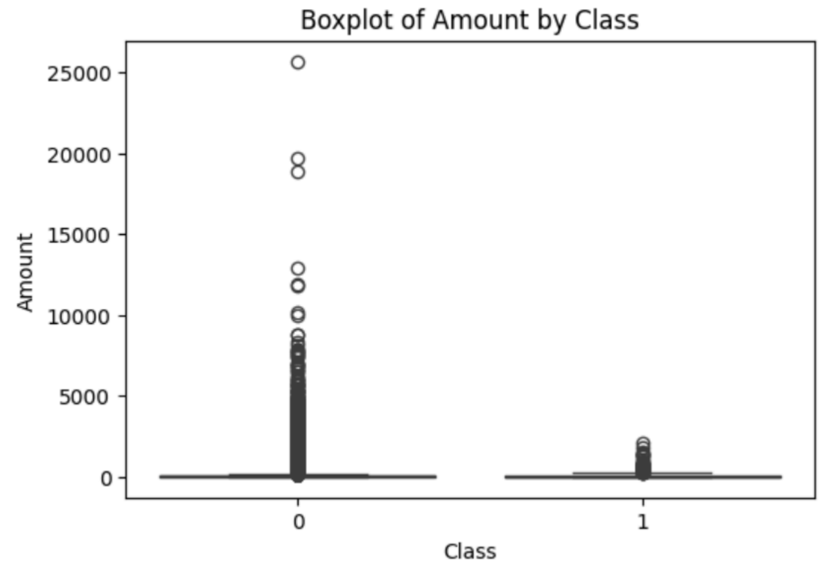
This also makes sense because on deep thinking one will realize that the fraudster would try the maximum to make a fraud transaction look like a normal transaction. Hence it is difficult to select or remove any column on that basis. Although AMOUNT table shared higher correlation compared to others and was kept in regard.

AMOUNT Column consisted of some outliers and that too in the non fraud transactions which did not expectedly play any role in formation of patterns for identifying fraud classes since the fraud cases were generally in range 0 to 3000 units .

Hence the amount above 8000 units was removed from the dataset as shown below.

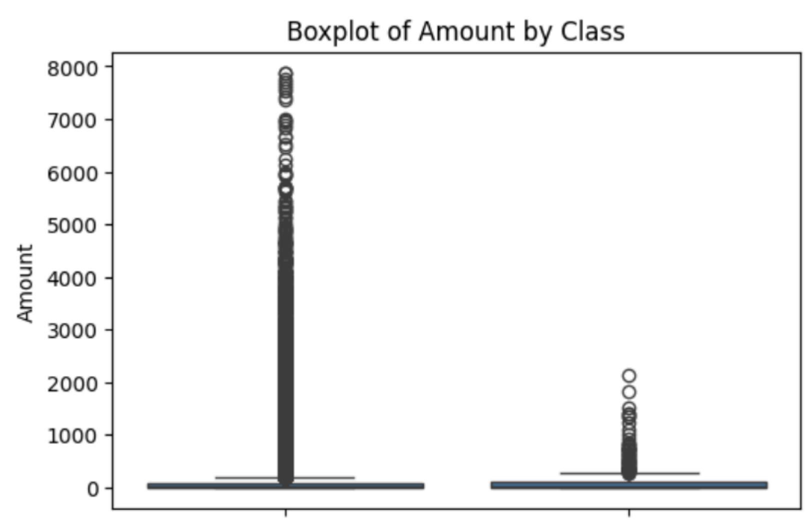
Class Imbalance:

Class	
0	284303
1	492




```
[65] data = data[data['Amount'] <= 8000]
```

```
plt.figure(figsize=(6, 4))
sns.boxplot(x='Class', y='Amount', data=data)
plt.title('Boxplot of Amount by Class')
plt.show()
```



MODEL IMPLEMENTATION

The data was split into test and train data as per usual custom. Model was fit with train data, and test data was tested on the algorithm built.

```
probs=mdl.predict_proba(xtest1)
threshold = 0.6
predictions = [1 if p[1] > threshold else 0 for p in probs]

from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
accuracy=accuracy_score(ytest1,predictions)
print(accuracy,"\n")
print(confusion_matrix(ytest1,predictions))
print(classification_report(ytest1,predictions))
```

```
0.9830295679189676

[[92256  1582]
 [   13   136]]
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     93838
           1       0.08      0.91      0.15       149

    accuracy                           0.98     93987
   macro avg       0.54      0.95      0.57     93987
weighted avg       1.00      0.98      0.99     93987
```

TRAINING DATA ACCURACY

```
training accuracy
0.9769196905854978

[[186080   4376]
 [    28    328]]
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    190456
           1       0.07      0.92      0.13       356

    accuracy                           0.98    190812
   macro avg       0.53      0.95      0.56    190812
weighted avg       1.00      0.98      0.99    190812
```

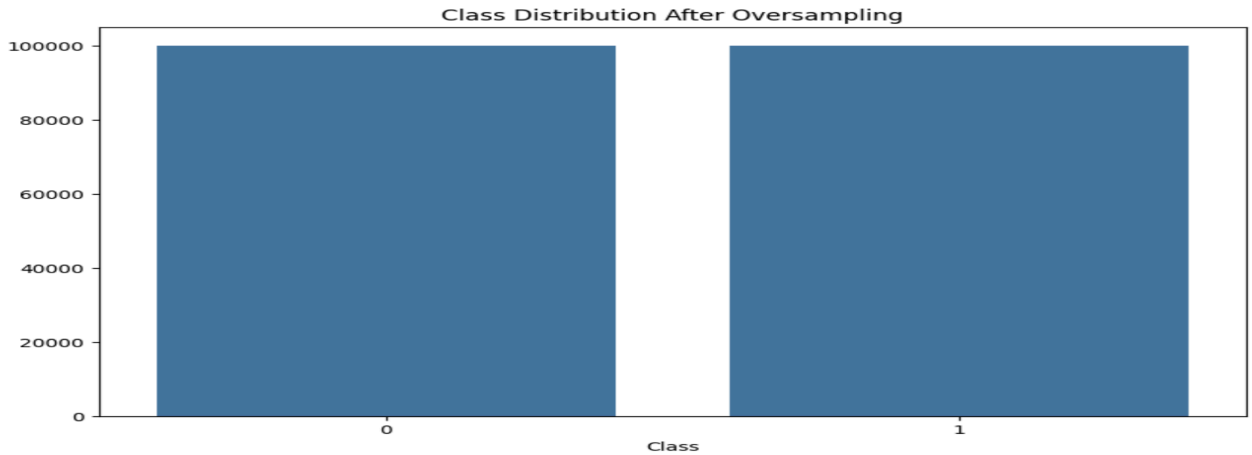
The similarity between train test data accuracy shows that the model does not overfit since both bias and variances are low.

The results are above shown, the threshold was set keeping in mind that the bigger aim is to reduce false negatives.

This is because it is still acceptable that non fraud are classified as fraud but if a fraud survives as non fraud the scenario will not be so good.

SMOTE

Smote was used to resample the Fraud class. As a result :



Classes got balanced.

```
[ ] newprobs=new_mdl.predict_proba(xtest1)
    threshold = 0.52
    predictions = [1 if p[1] > threshold else 0 for p in newprobs]

    accuracy=accuracy_score(ytest1,predictions)
    print(accuracy,"\n")
    print(confusion_matrix(ytest1,predictions))
```

```
0.9233298222094545

[[86647  7191]
 [   15   134]]
```

Hyperparameter tuning

Hyperparameter tuning was used to find best parameters, scoring was set as recall to maximize recall.

```
[12] param_grid = {'C': [0.1, 1, 10], 'solver': ['liblinear', 'saga']}
```

```
from sklearn.model_selection import RandomizedSearchCV
randomized_search = RandomizedSearchCV(LogisticRegression(max_iter=2000), param_grid, n_iter=5, cv=3, scoring
randomized_search.fit(x, y)

best_model = randomized_search.best_estimator_
best_model.fit(x, y)
```

```
LogisticRegression
LogisticRegression(C=10, max_iter=2000, solver='liblinear')
```

Results

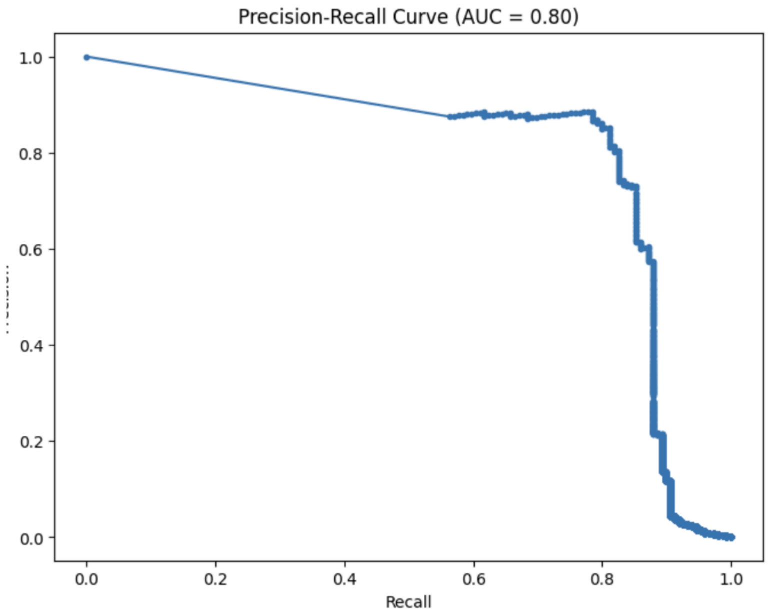
```
[[90524  3314]
 [   13   136]]
              precision    recall  f1-score   support

           0       1.00      0.96      0.98     93838
           1       0.04      0.91      0.08       149

    accuracy                           0.96     93987
   macro avg       0.52      0.94      0.53     93987
weighted avg       1.00      0.96      0.98     93987
```

threshold=0.35

```
0.9915201038441487

[[93057   781]
 [   16   133]]
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     93838
           1       0.15      0.89      0.25       149

    accuracy                           0.99     93987
   macro avg       0.57      0.94      0.62     93987
weighted avg       1.00      0.99      0.99     93987
```

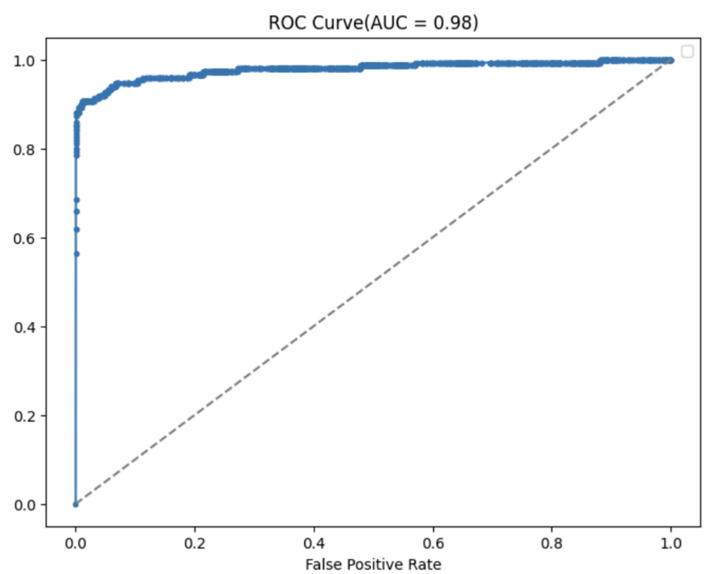
threshold=0.7

PR curve



The smoothness and area of the PR curve suffered a bit because the primary target was to control FN even at the cost of increased FP.

ROC curve



The smoothness and area of the ROC curve suffered a bit because the primary target was to control FN even at the cost of increased FP.

Conclusion

On lowering the thresholds along hyperparameter tunings and SMOTE resampling technique, FNs were reduced, as we could see for reducing 3 FNs , the cost was additional 2500 FPs.

On deep thinking one may realize that it is still bearable to have some FPs if as a reward FNs are getting reduced.

Since the class was highly imbalanced, no frauds detected will be way lower than non fraud cases.