QUESTION 2

1- Data Exploration

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	SpecialDay	Month	OperatingSystems	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
0	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.000000	0.0	Feb	1	1	1	1	Returning_Visitor	False	False
1	0	0.0	0	0.0	2	64.000000	0.000000	0.100000	0.000000	0.0	Feb	2	2	1	2	Returning_Visitor	False	False
2	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.000000	0.0	Feb	4	1	9	3	Returning_Visitor	False	False
3	0	0.0	0	0.0	2	2.666667	0.050000	0.140000	0.000000	0.0	Feb	3	2	2	4	Returning_Visitor	False	False
4	0	0.0	0	0.0	10	627.500000	0.020000	0.050000	0.000000	0.0	Feb	3	3	1	4	Returning_Visitor	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12325	3	145.0	0	0.0	53	1783.791667	0.007143	0.029031	12.241717	0.0	Dec	4	6	1	1	Returning_Visitor	True	False
12326	0	0.0	0	0.0	5	465.750000	0.000000	0.021333	0.000000	0.0	Nov	3	2	1	8	Returning_Visitor	True	False
12327	0	0.0	0	0.0	6	184.250000	0.083333	0.086667	0.000000	0.0	Nov	3	2	1	13	Returning_Visitor	True	False
12328	4	75.0	0	0.0	15	346.000000	0.000000	0.021053	0.000000	0.0	Nov	2	2	3	11	Returning_Visitor	False	False
12329	0	0.0	0	0.0	3	21.250000	0.000000	0.066667	0.000000	0.0	Nov	3	2	1	2	New_Visitor	True	False

12330 rows × 18 columns

This dataset is imbalanced.

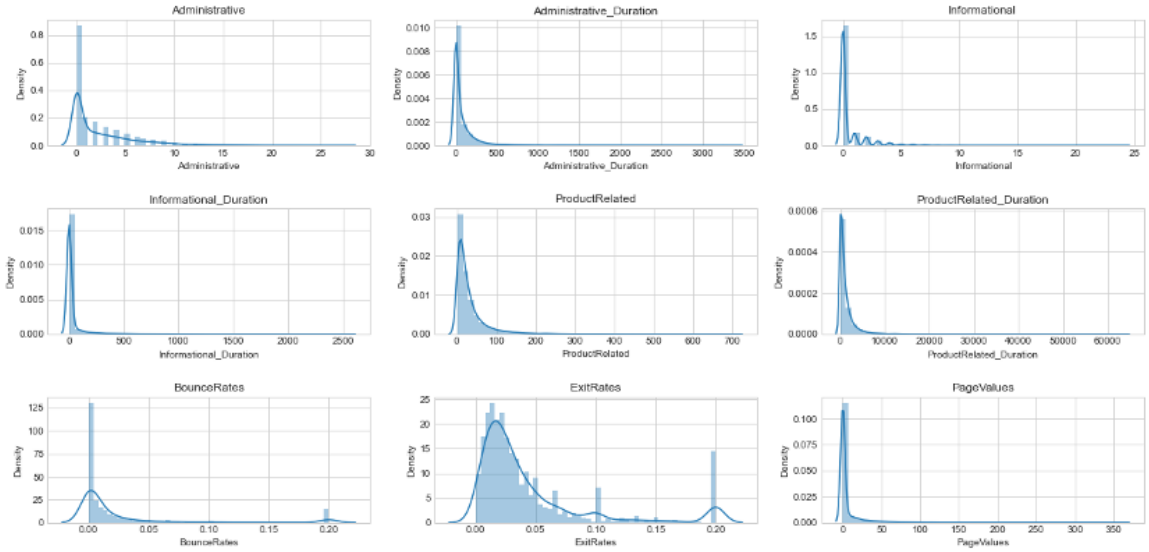
```
False    10422
True      1908
Name: Revenue, dtype: int64
```

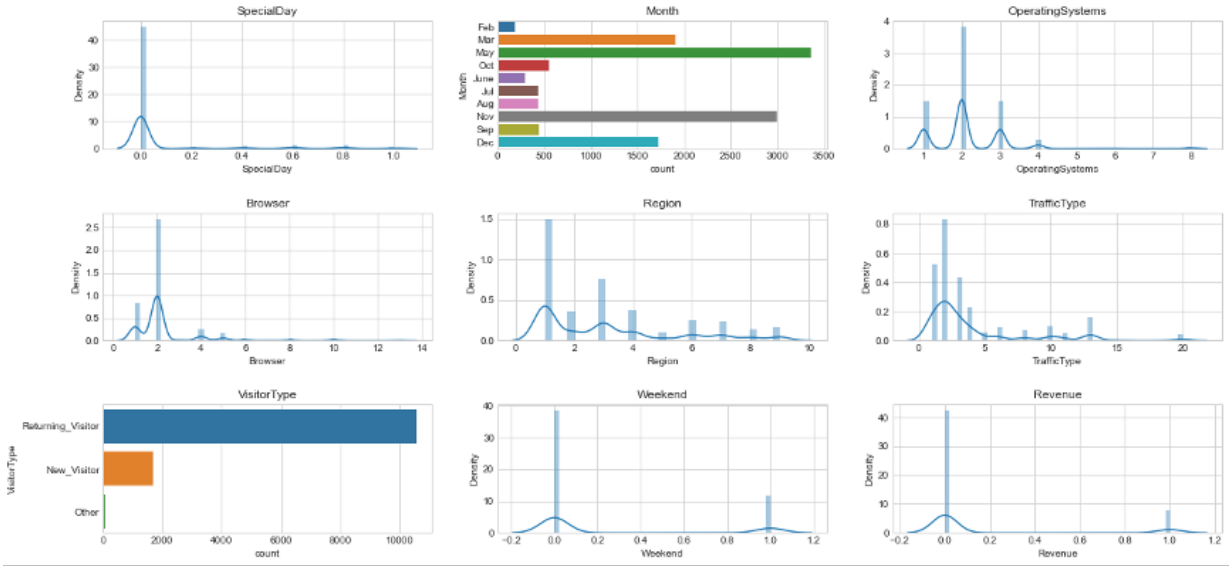
Univariate Analysis

Explore the Features

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	SpecialDay
count	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000
mean	2.315166	80.818611	0.503569	34.472398	31.731468	1194.746220	0.022191	0.043073	5.889258	0.061427
std	3.321784	176.779107	1.270156	140.749294	44.475503	1913.669288	0.048488	0.048597	18.568437	0.198917
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	7.000000	184.137500	0.000000	0.014286	0.000000	0.000000
50%	1.000000	7.500000	0.000000	0.000000	18.000000	598.936905	0.003112	0.025156	0.000000	0.000000
75%	4.000000	93.256250	0.000000	0.000000	38.000000	1464.157213	0.016813	0.050000	0.000000	0.000000
max	27.000000	3398.750000	24.000000	2549.375000	705.000000	63973.522230	0.200000	0.200000	361.763742	1.000000

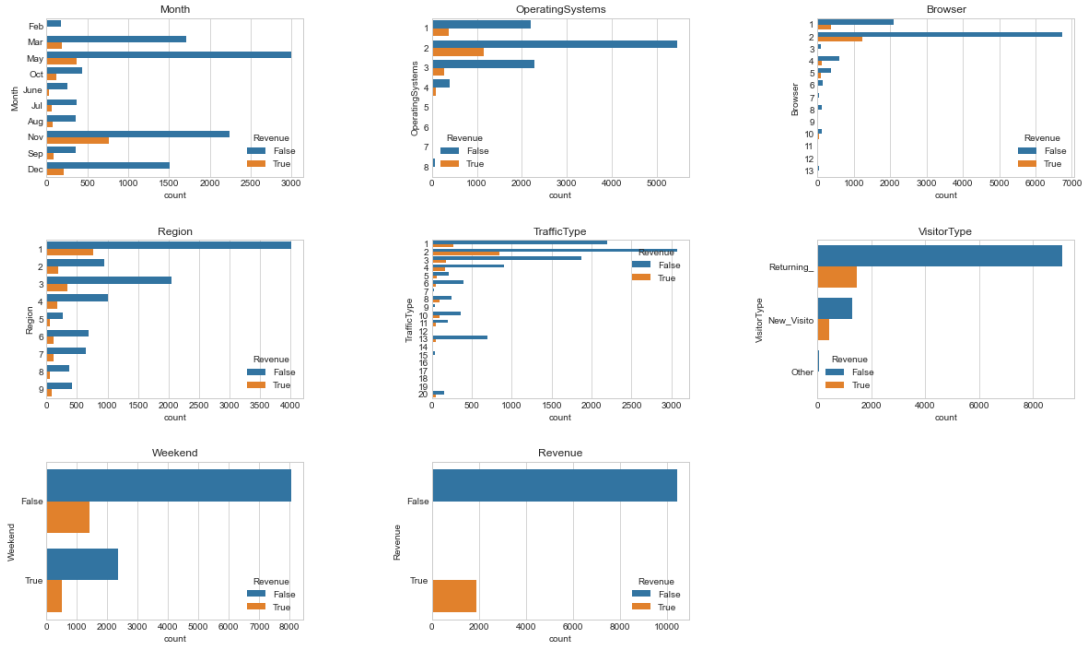
To learn more about the columns, visualize the data distribution of columns is important.





Bivariate Analysis

In this part, we will try to separate categorical features by Revenue.



Outliers

To detect outliers, z score has been used and if the z score is higher than 3, it is outlier.

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	SpecialDay	Month	OperatingSystems	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
0	0.0	0.0	0.0	0.0	2.0	64.000000	0.000000	0.100000	0.000000	0.0	Feb	2	2	1	2	Returning_Visitor	False	False
1	0.0	0.0	0.0	0.0	2.0	2.666667	0.050000	0.140000	0.000000	0.0	Feb	3	2	2	4	Returning_Visitor	False	False
2	0.0	0.0	0.0	0.0	10.0	627.500000	0.020000	0.050000	0.000000	0.0	Feb	3	3	1	4	Returning_Visitor	True	False
3	0.0	0.0	0.0	0.0	19.0	154.216667	0.015789	0.024561	0.000000	0.0	Feb	2	2	1	3	Returning_Visitor	False	False
4	0.0	0.0	0.0	0.0	3.0	738.000000	0.000000	0.022222	0.000000	0.4	Feb	2	4	1	2	Returning_Visitor	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10024	3.0	145.0	0.0	0.0	53.0	1783.791667	0.007143	0.029031	12.241717	0.0	Dec	4	6	1	1	Returning_Visitor	True	False
10025	0.0	0.0	0.0	0.0	5.0	465.750000	0.000000	0.021333	0.000000	0.0	Nov	3	2	1	8	Returning_Visitor	True	False
10026	0.0	0.0	0.0	0.0	6.0	184.250000	0.083333	0.086667	0.000000	0.0	Nov	3	2	1	13	Returning_Visitor	True	False
10027	4.0	75.0	0.0	0.0	15.0	346.000000	0.000000	0.021053	0.000000	0.0	Nov	2	2	3	11	Returning_Visitor	False	False
10028	0.0	0.0	0.0	0.0	3.0	21.250000	0.000000	0.066667	0.000000	0.0	Nov	3	2	1	2	New_Visitor	True	False

10029 rows × 18 columns

2301 observations are marked as outlier.

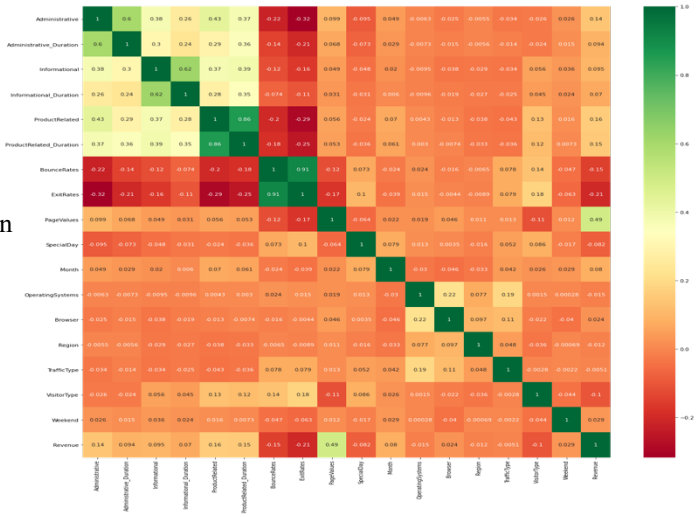
Feature Selection

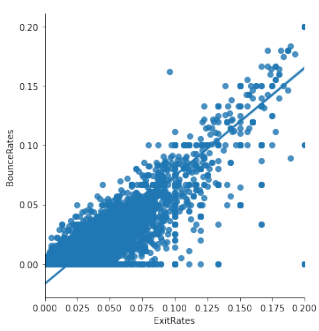
There are several ways to do feature selection.

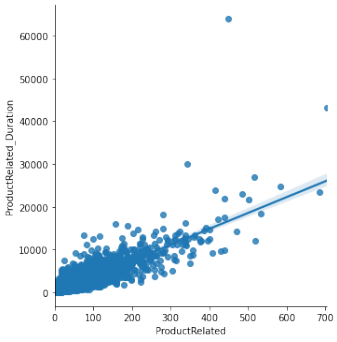
1- Correlation Table

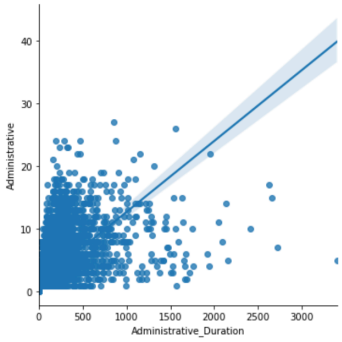
In the heatmap, it is clear that, there are correlated columns.

- ProductRelated - ProductedRelated_Duration
- ExitRates – BounceRates
- Informational - Informational _Duration
- Administrative - Administrative_Duration









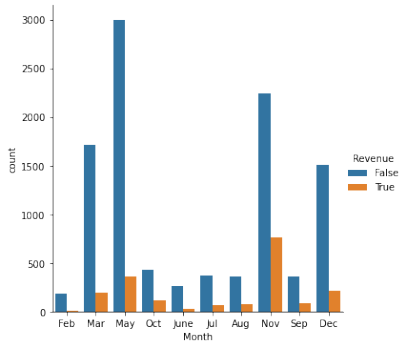
2- Chi2 statistic and Mutual Information

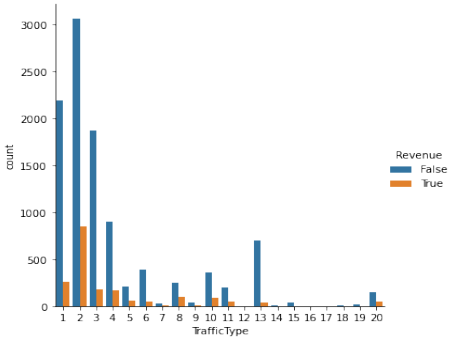
These two techniques give better result with categorical features.

	Feature	Chi2_Score
0	Month	86.163696
5	VisitorType	37.547523
2	Browser	8.873291
6	Weekend	8.120464
3	Region	3.037565
4	TrafficType	1.283194
1	OperatingSystems	1.037132

	Feature	Mutual_Information
4	TrafficType	0.017223
0	Month	0.012462
1	OperatingSystems	0.007813
6	Weekend	0.002254
5	VisitorType	0.001839
3	Region	0.000347
2	Browser	0.000000

Lets plot the most importance features of them.





3- Anova Test

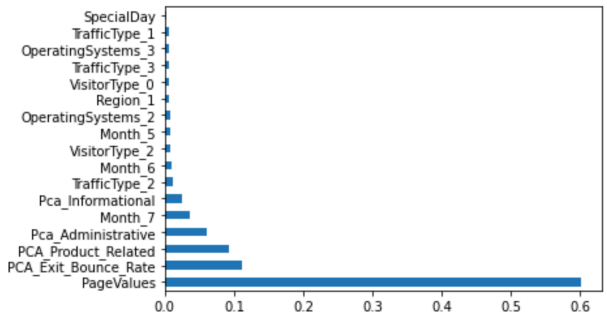
I will use the ANOVA stat test to find the best numerical features.

In the table, it is clear that the “PageValues” feature has a great effect on Revenue. It is also fitted with our findings in the correlation table.

	Feature	ANOVA
8	PageValues	3949.262960
7	ExitRates	552.286502
4	ProductRelated	317.844350
5	ProductRelated_Duration	293.027603
6	BounceRates	286.375674
0	Administrative	242.586667
2	Informational	112.751843
1	Administrative_Duration	108.928515
9	SpecialDay	84.080012
3	Informational_Duration	61.306613

4- RandomForest Feature Importance/PCA

At first, I used onehotencoder to encode categorical data and train the RandomForest model and use feature_importance. This gives the most important features. But before doing that, I used the PCA to reduce the dimensionality of correlated columns. The most important features can be seen in the table below.



PRINCIPAL COMPONENT ANALYSIS

"ExitRates", "BounceRates" → “PCA_Exit_Bounce_Rate”

"ProductRelated", "ProductRelated_Duration" → “PCA_Product_Related”

"Informational", "Informational_Duration" → “Pca_Informational”

"Administrative", "Administrative_Duration" → “Pca_Administrative”

2- PREPROCESSING

a) Nan Values

There is no nan values in the dataset.

b) Encoding

Machine learning models work with numerical values better than others. To convert features as categorical to numeric, OneHotencoder has been used.

One Hot Encoder

	Aug	Dec	Feb	Jul	June	Mar	May	Nov	Oct	Sep
0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
12325	0	1	0	0	0	0	0	0	0	0
12326	0	0	0	0	0	0	0	1	0	0
12327	0	0	0	0	0	0	0	1	0	0
12328	0	0	0	0	0	0	0	1	0	0
12329	0	0	0	0	0	0	0	1	0	0

12330 rows × 10 columns

LabeEncoder

	Month
0	2
1	2
2	2
3	2
4	2
...	...
12325	1
12326	7
12327	7
12328	7
12329	7

12330 rows × 1 columns

c) Scaling

Difference distributions of data is a problem for model training because it will give more importance to data with higher values. To solve it, MinMaxScaler has been used.

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	SpecialDay
count	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000
mean	0.085747	0.023779	0.020982	0.013522	0.045009	0.018676	0.110957	0.215364	0.016279	0.061427
std	0.123029	0.052013	0.052923	0.055209	0.063086	0.029913	0.242442	0.242983	0.051328	0.198917
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.009929	0.002878	0.000000	0.071429	0.000000	0.000000
50%	0.037037	0.002207	0.000000	0.000000	0.025532	0.009362	0.015562	0.125782	0.000000	0.000000
75%	0.148148	0.027438	0.000000	0.000000	0.053901	0.022887	0.084063	0.250000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

d) Power Transform

Many machine learning algorithms work better with Gaussian distributed features, and many of them don't have this necessity. I will use Yeo-Johnson Transform because Cox-Box Transform doesn't support negative variables and 0.

e) Sampling

Sampling methods have a crucial role to make balanced of imbalanced datasets. There are several sampling methods as RandomOverSampling, RandomUnderSampling, SMOTE, and ADASYS.
Simply, RandomUnderSampling decreases the number of majority labels and others increase minority labels to make the dataset balanced. I discovered, there is no best sampling method for all models.

3- MODEL IMPLEMENTATION & EVALUATION

1- KNeighborsClassifier

Simply, the logic of KNN is based on similar labels/points are located at close range. This is why it is suitable for classification. One downside of KNN is that it is very sensitive to meaningless features and works best with lower dimension datasets.

2- Support Vector Machine

SVM can separate data both linearly and non-linearly. It uses kernel-trick to transform data and make it more separately. This is why I choose SVM because data can be non-linear and not separable with basic linear functions. To get better results, I will use SVM with Adaboost.

3- Random Forest

Random Forest is an ensemble method of the DecisionTree. Simply, it is a decision tree with bagging but it also adds randomness to choosing features to find the best features and best results. It is also one of the best to avoid overfitting. This is why I used it instead of DecisionTree.

4- XGBoost and LigthGBM

XGBoost and LigthGBM are also tree-based model. They are an improved version of boosting and they can handle overfitting with regularization. Also, to handle an imbalanced dataset, weight parameter can be used to give more weight to minority label. Compared to Random forest which is a bagging-algorithm, XGBoost and LigthGBM are boosting-algorithm and help to reduce both variance and bias.

5- Voting Classifier

KNN, RandomForest, and XGBoost&LigthGBM have been used to get better results in a voting-classifier.

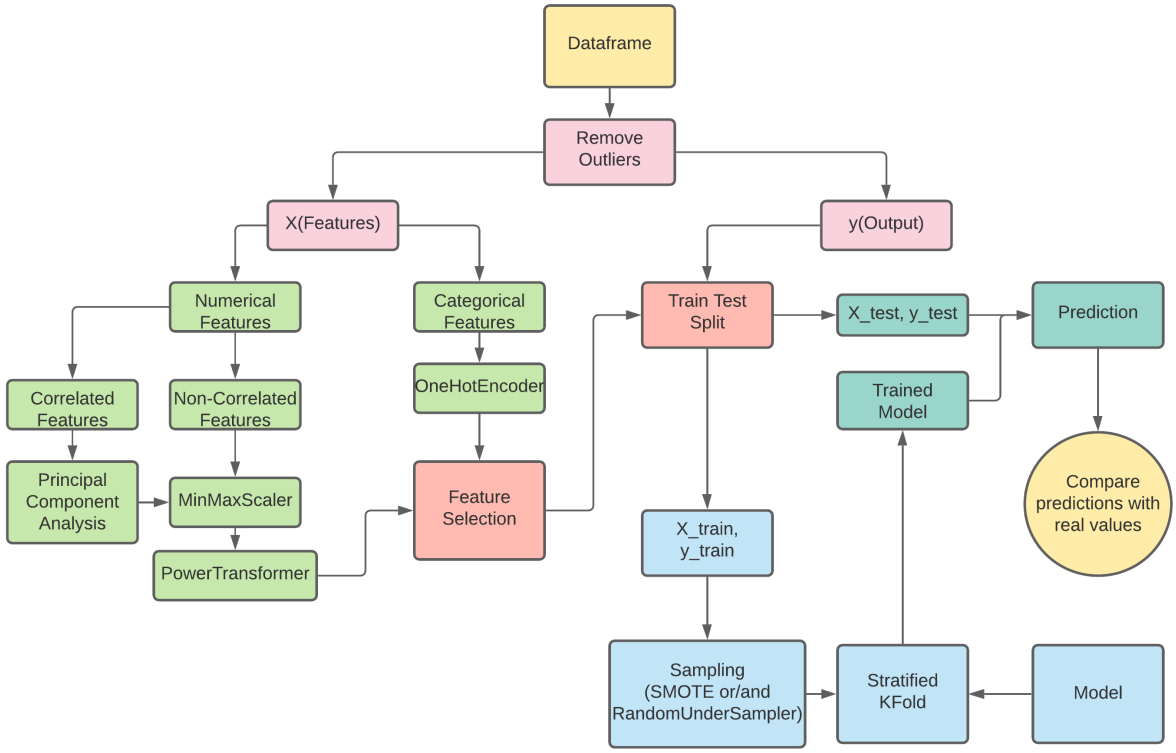
- Pipeline has been used in all of these models. In some of them, preprocessing and sampling parts have been done in the pipeline.

- Before applying sampling techniques, the dataset was divided with train_test_split and sampling techniques were only applied on the training dataset. To get correct results, the test dataset must be original.

- In all models, StratfiedKFold has been used to train models.

- To find the best parameters, GridSearchCV has been used in all models.

SUMMARY OF ML MODEL



Model Results

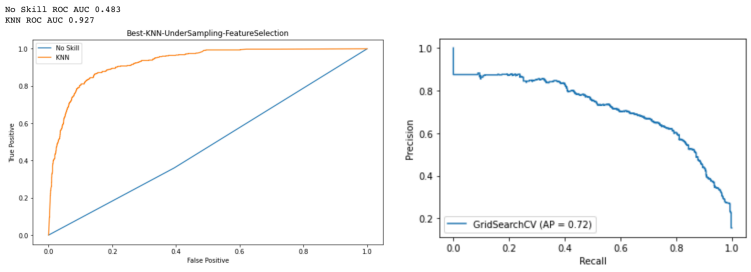
1- KNN

- KNN-UnderSampling-FeatureSelection

```
Validation Recall: 0.7714889277389279
Recall Test: 0.7987421383647799
Precision Test: 0.6018957345971564
F2 Test: 0.7497048406139315
F1 Test: 0.6864864864864866
[[2354  252]
 [  96  381]]
              precision    recall  f1-score   support

       False       0.96      0.90      0.93      2606
        True       0.60      0.80      0.69       477

    accuracy                           0.89      3083
   macro avg       0.78      0.85      0.81      3083
weighted avg       0.91      0.89      0.89      3083
```

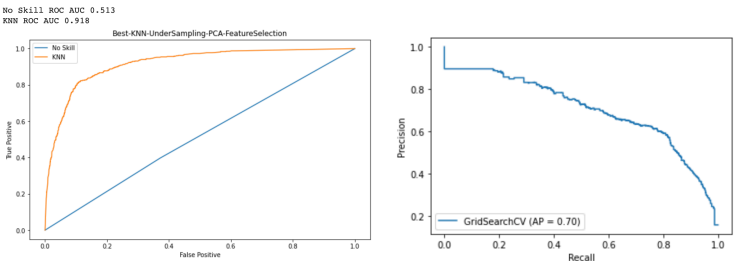


- KNN-UnderSampling-PCA-FeatureSelection

```
Validation Recall: 0.7629912341179946
Recall Test: 0.8070539419087137
Precision Test: 0.590288315629742
F2 Test: 0.7518361035948976
F1 Test: 0.6818580192813322
[[2331  270]
 [  93  389]]
              precision    recall  f1-score   support

       False       0.96      0.90      0.93      2601
        True       0.59      0.81      0.68       482

    accuracy                           0.88      3083
   macro avg       0.78      0.85      0.80      3083
weighted avg       0.90      0.88      0.89      3083
```

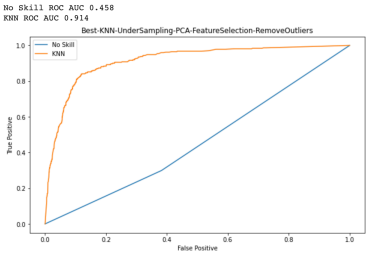


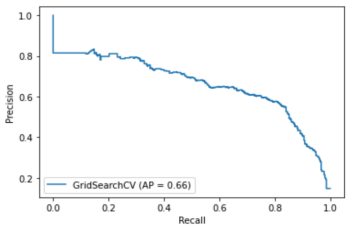
- KNN-UnderSampling-PCA-FeatureSelection-RemoveOutliers

```
Validation Recall: 0.7555382485289027
Recall Test: 0.8032345013477089
Precision Test: 0.5764023210831721
F2 Test: 0.7446276861569217
F1 Test: 0.6711711711711711
[[1918  219]
 [  73  298]]
              precision    recall  f1-score   support

       False       0.96      0.90      0.93      2137
        True       0.58      0.80      0.67       371

    accuracy                           0.88      2508
   macro avg       0.77      0.85      0.80      2508
weighted avg       0.91      0.88      0.89      2508
```





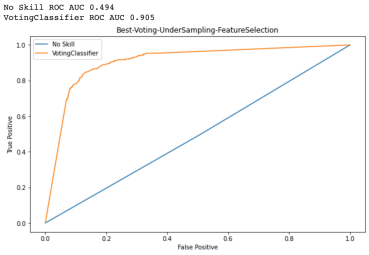
2- SUPPORT VECTOR MACHINE

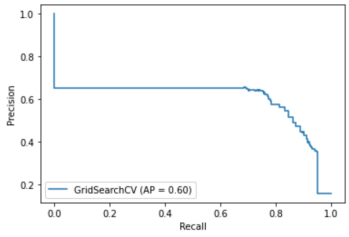
- SVC-UnderSampling-FeatureSelection

```
Validation Recall: 0.8389971031864949
Recall Test: 0.8475609756097561
Precision Test: 0.5422626788036411
F2 Test: 0.7617829740591887
F1 Test: 0.6613798572561459
[[2239  352]
 [  75  417]]
              precision    recall  f1-score   support

       False       0.97      0.86      0.91      2591
        True       0.54      0.85      0.66       492

    accuracy                           0.86      3083
   macro avg       0.75      0.86      0.79      3083
weighted avg       0.90      0.86      0.87      3083
```



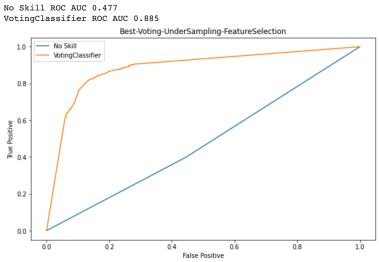


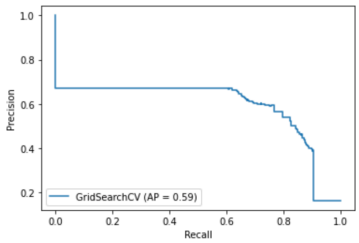
- SVC-UnderSampling-PCA-FeatureSelection

```
Validation Recall: 0.796727456940223
Recall Test: 0.7984189723320159
Precision Test: 0.5634588563458857
F2 Test: 0.7369573148485953
F1 Test: 0.6606704824202779
[[2264  313]
 [ 102  404]]
              precision    recall  f1-score   support

       False       0.96      0.88      0.92      2577
        True       0.56      0.80      0.66       506

    accuracy                           0.87      3083
   macro avg       0.76      0.84      0.79      3083
weighted avg       0.89      0.87      0.87      3083
```



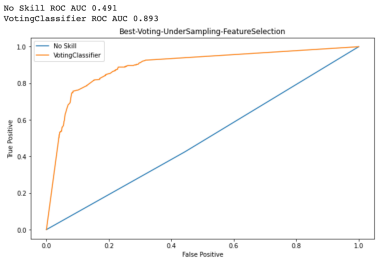


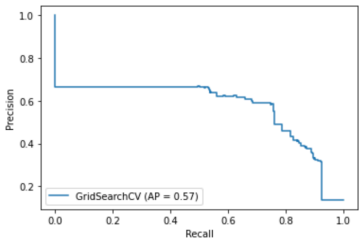
- SVC-UnderSampling-PCA-FeatureSelection-RemoveOutliers

```
Validation Recall: 0.7757903357903357
Recall Test: 0.7624633431085044
Precision Test: 0.5485232067510548
F2 Test: 0.707290533188248
F1 Test: 0.6380368098159509
[[1953  214]
 [  81  260]]
              precision    recall  f1-score   support

       False       0.96      0.90      0.93      2167
        True       0.55      0.76      0.64       341

    accuracy                           0.88      2508
   macro avg       0.75      0.83      0.78      2508
weighted avg       0.90      0.88      0.89      2508
```





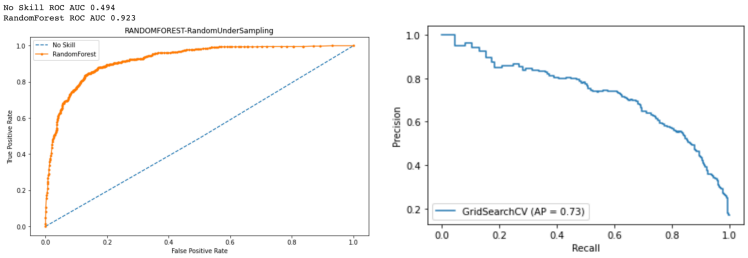
3- RANDOM FOREST

- RandomForest-UnderSampling-FeatureSelection

```
Validation Recall: 0.8558485665767656
Recall Test: 0.8580121703853956
Precision Test: 0.5065868263473053
F2 Test: 0.75347345920912
F1 Test: 0.6370481927710844
[[2178  412]
 [  70  423]]
              precision    recall  f1-score   support

       False       0.97      0.84      0.90      2590
        True       0.51      0.86      0.64       493

    accuracy                           0.84      3083
   macro avg       0.74      0.85      0.77      3083
weighted avg       0.89      0.84      0.86      3083
```

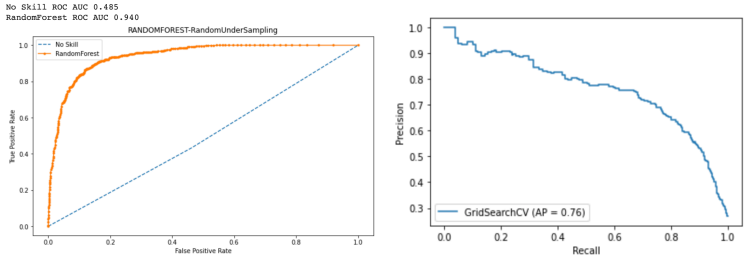


- RandomForest-UnderSampling-PCA-FeatureSelection

```
Validation Recall: 0.8451798561151079
Recall Test: 0.8830409356725146
Precision Test: 0.5551470588235294
F2 Test: 0.7897489539748954
F1 Test: 0.6817155756207676
[[2207  363]
 [  60  453]]
              precision    recall  f1-score   support

       False       0.97      0.86      0.91      2570
        True       0.56      0.88      0.68       513

    accuracy                           0.86      3083
   macro avg       0.76      0.87      0.80      3083
weighted avg       0.90      0.86      0.87      3083
```

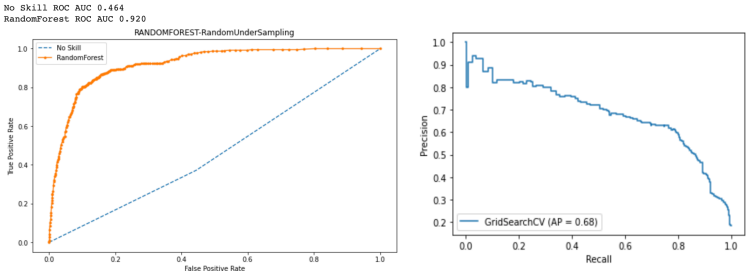


- RandomForest-UnderSampling-PCA-FeatureSelection-RemoveOutliers

```
Validation Recall: 0.8339622641509434
Recall Test: 0.8423772609819121
Precision Test: 0.5258064516129032
F2 Test: 0.7518450184501845
F1 Test: 0.6474677259185699
[[1827  294]
 [  61  326]]
              precision    recall  f1-score   support

       False       0.97      0.86      0.91      2121
        True       0.53      0.84      0.65       387

    accuracy                           0.86      2508
   macro avg       0.75      0.85      0.78      2508
weighted avg       0.90      0.86      0.87      2508
```



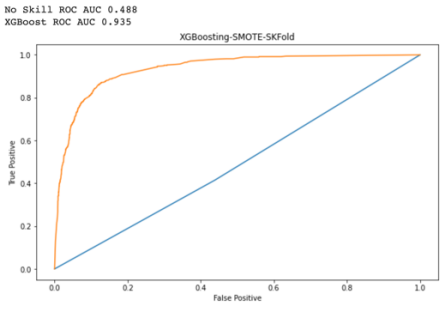
4- XGBoost & LigthGBM

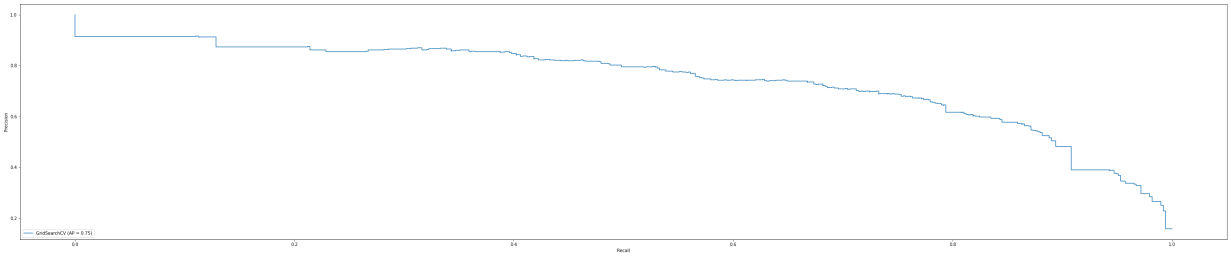
- XGBoosting-SMOTE-FeatureSelection

```
Validation Recall: 0.8631591686157014
Recall Test: 0.8591836734693877
Precision Test: 0.5759233926128591
F2 Test: 0.7822370865849128
F1 Test: 0.6895986895986896
[[2283  310]
 [  69  421]]
              precision    recall  f1-score   support

       False       0.97      0.88      0.92      2593
        True       0.58      0.86      0.69       490

    accuracy                           0.88      3083
   macro avg       0.77      0.87      0.81      3083
weighted avg       0.91      0.88      0.89      3083
```



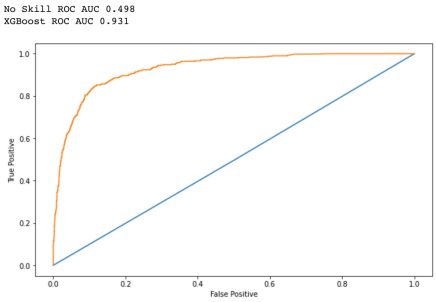


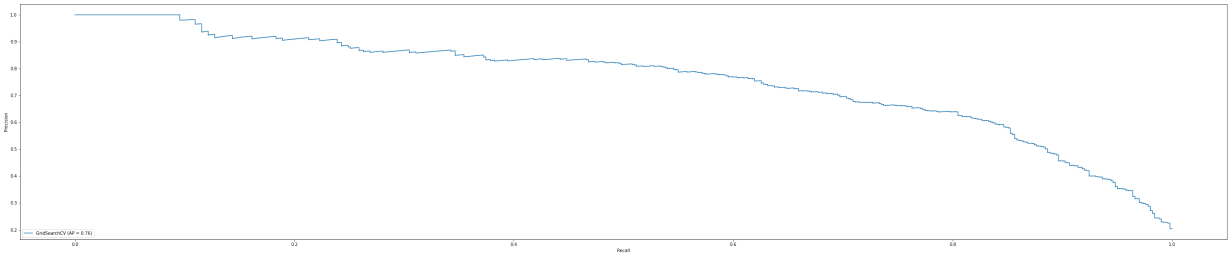
- XGBoosting-SMOTE-RandomUnderSampler-FeatureSelection

```
Validation Recall: 0.8854591836734693
Recall Test: 0.8386454183266933
Precision Test: 0.594632768361582
F2 Test: 0.7750368188512519
F1 Test: 0.6958677685950414
[[2294  287]
 [  81  421]]
              precision    recall  f1-score   support

       False       0.97      0.89      0.93      2581
        True       0.59      0.84      0.70       502

    accuracy                           0.88      3083
   macro avg       0.78      0.86      0.81      3083
weighted avg       0.91      0.88      0.89      3083
```



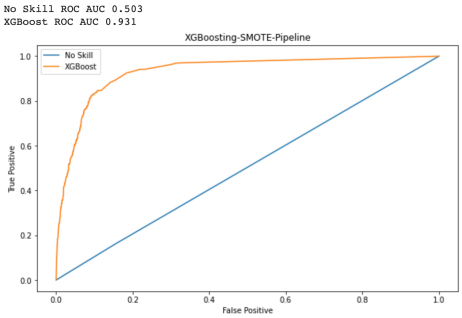


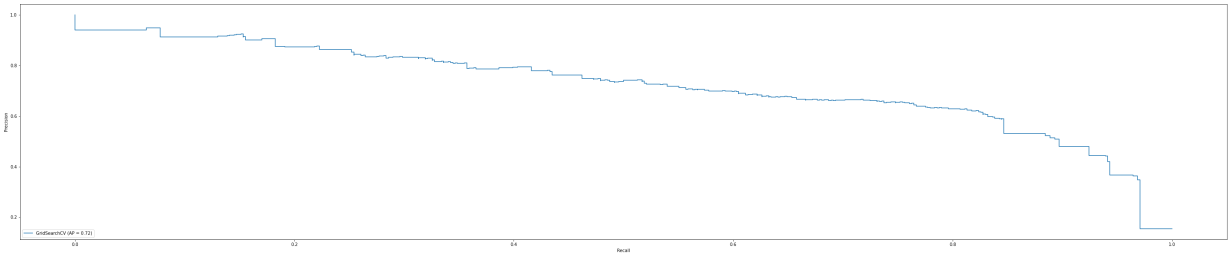
- XGBoosting-SMOTE-ImbalancePipeline
-

```
Validation Recall: 0.7527972027972029
Recall Test: 0.8235294117647058
Precision Test: 0.6182965299684543
F2 Test: 0.7722616233254531
F1 Test: 0.7063063063063063
[[2365  242]
 [  84  392]]
              precision    recall  f1-score   support

       False       0.97      0.91      0.94      2607
        True       0.62      0.82      0.71       476

    accuracy                           0.89      3083
   macro avg       0.79      0.87      0.82      3083
weighted avg       0.91      0.89      0.90      3083
```



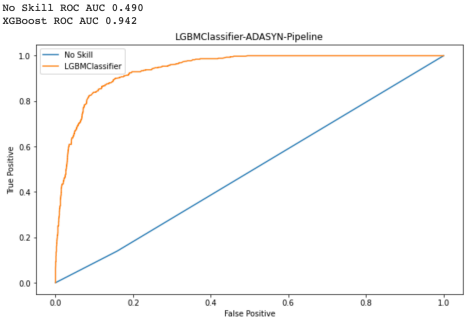


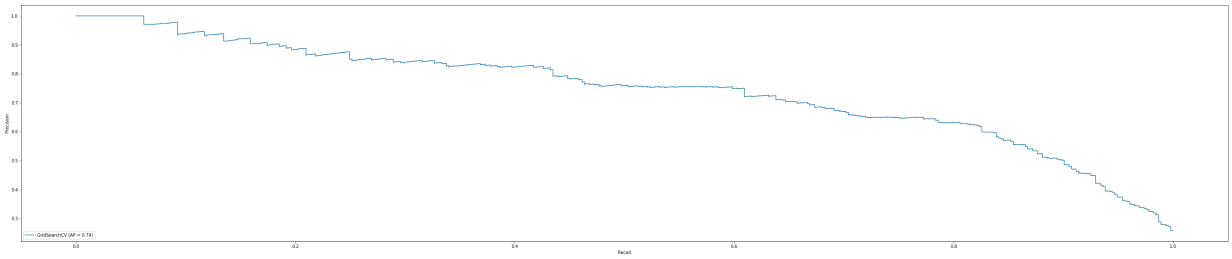
- LigthGBM-ADASYN-ImbalancePipeline

```
Validation Recall: 0.7711289560699103
Recall Test: 0.8388520971302428
Precision Test: 0.5846153846153846
F2 Test: 0.7717303005686434
F1 Test: 0.6890299184043518
[[2360  270]
 [  73  380]]
              precision    recall  f1-score   support

       False       0.97      0.90      0.93      2630
        True       0.58      0.84      0.69       453

    accuracy                           0.89      3083
   macro avg       0.78      0.87      0.81      3083
weighted avg       0.91      0.89      0.90      3083
```



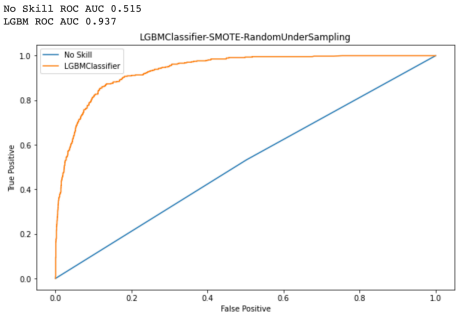


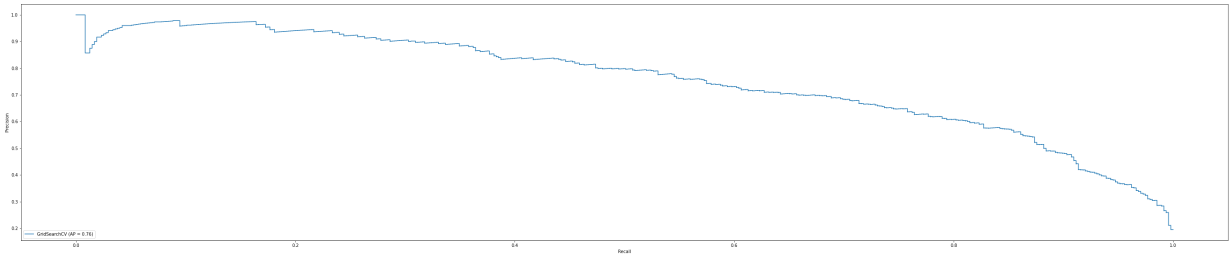
- LigthGBM-SMOTE-RandomUnderSampler-FeatureSelection

```
Validation Recall: 0.8814827201783724
Recall Test: 0.8610526315789474
Precision Test: 0.5602739726027397
F2 Test: 0.7775665399239544
F1 Test: 0.678838174273859
[[2287  321]
 [  66  409]]
              precision    recall  f1-score   support

       False       0.97      0.88      0.92      2608
        True       0.56      0.86      0.68       475

    accuracy                           0.87      3083
   macro avg       0.77      0.87      0.80      3083
weighted avg       0.91      0.87      0.88      3083
```



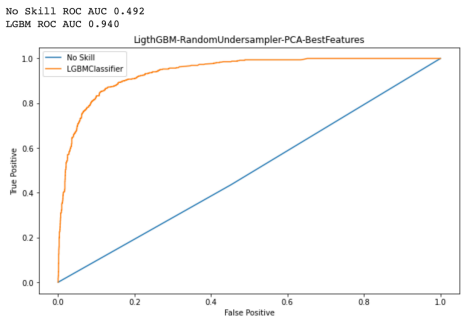


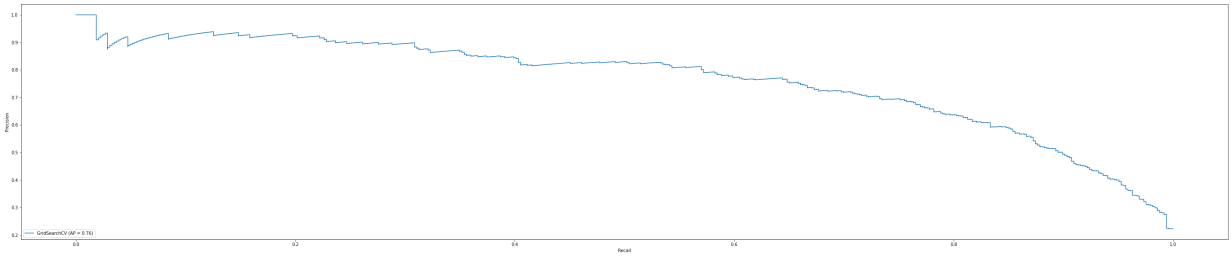
- LigthGBM-RandomUnderSampler-PCA -FeatureSelection

```
Validation Recall: 0.823520141830001
Recall Test: 0.8580246913580247
Precision Test: 0.5704514363885089
F2 Test: 0.7794392523364486
F1 Test: 0.685291700903862
[[2283  314]
 [  69  417]]
              precision    recall  f1-score   support

       False       0.97      0.88      0.92      2597
        True       0.57      0.86      0.69       486

    accuracy                           0.88      3083
   macro avg       0.77      0.87      0.80      3083
weighted avg       0.91      0.88      0.89      3083
```





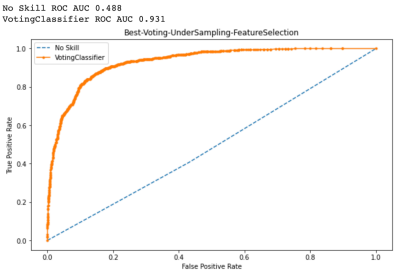
5- Voting Classifier

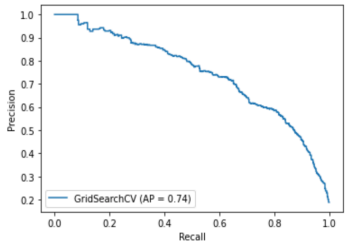
- Voting-UnderSampling-FeatureSelection

```
Validation Recall: 0.8136685823754789
Recall Test: 0.8322440087145969
Precision Test: 0.5584795321637427
F2 Test: 0.757936507936508
F1 Test: 0.668416447944007
[[2322  302]
 [  77  382]]
              precision    recall  f1-score   support

       False       0.97      0.88      0.92      2624
        True       0.56      0.83      0.67       459

    accuracy                           0.88      3083
   macro avg       0.76      0.86      0.80      3083
weighted avg       0.91      0.88      0.89      3083
```



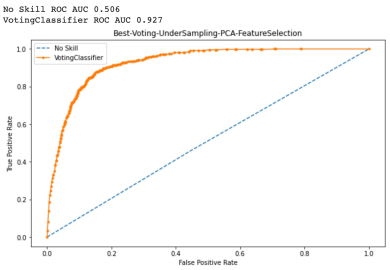


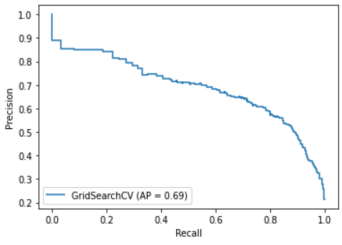
- Voting-UnderSampling-PCA-FeatureSelection

```
Validation Recall: 0.8163120567375886
Recall Test: 0.8273092369477911
Precision Test: 0.5659340659340659
F2 Test: 0.7573529411764705
F1 Test: 0.6721044045676999
[[2269  316]
 [  86  412]]
              precision    recall  f1-score   support

       False       0.96      0.88      0.92      2585
        True       0.57      0.83      0.67       498

    accuracy                           0.87      3083
   macro avg       0.76      0.85      0.80      3083
weighted avg       0.90      0.87      0.88      3083
```



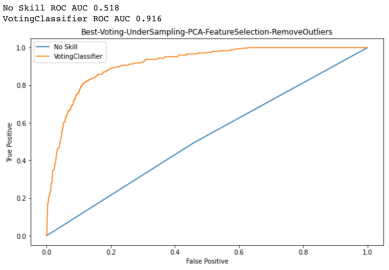


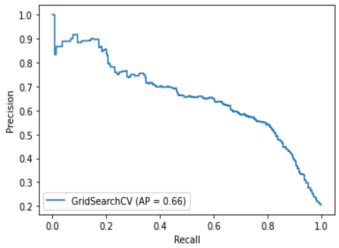
- Voting-UnderSampling-PCA-FeatureSelection-RemoveOutliers

```
Validation Recall: 0.801718098415346
Recall Test: 0.8158640226628895
Precision Test: 0.5207956600361664
F2 Test: 0.732824427480916
F1 Test: 0.6357615894039734
[[1890  265]
 [  65  288]]
              precision    recall  f1-score   support

       False       0.97      0.88      0.92      2155
        True       0.52      0.82      0.64       353

    accuracy                           0.87      2508
   macro avg       0.74      0.85      0.78      2508
weighted avg       0.90      0.87      0.88      2508
```



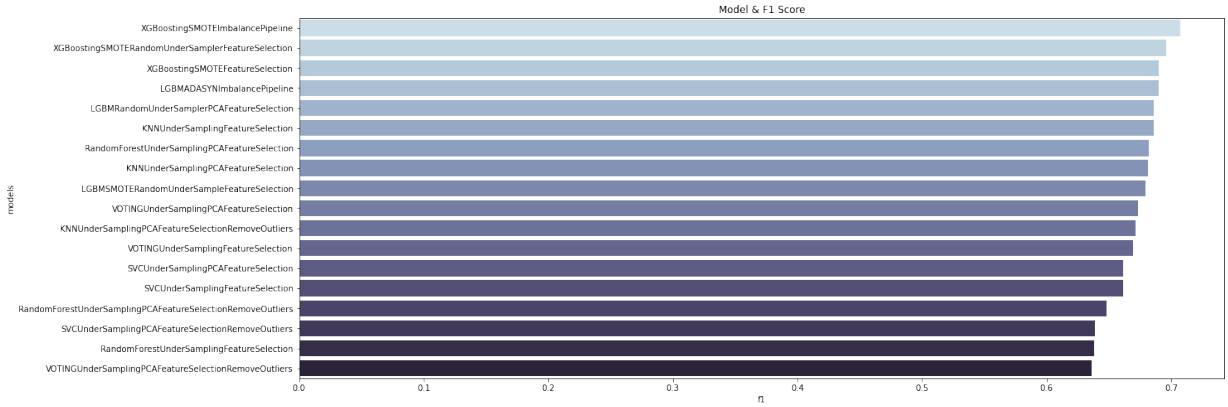


Evaluation

This dataset is imbalanced. In imbalanced datasets, accuracy is not a good metric to evaluate. To evaluate my models, I calculated F1, F2, AUC and AreaUnderPrecisionRecallCurve.

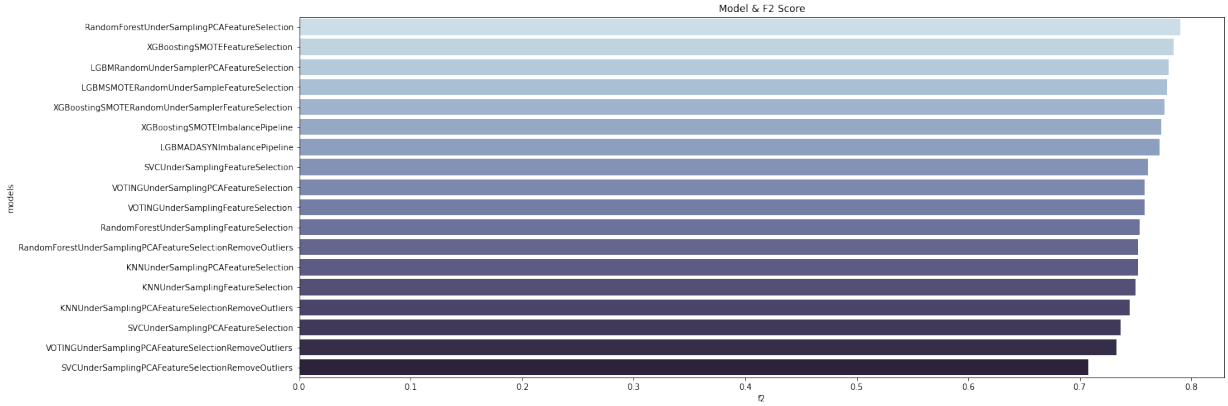
F1 Score is calculated from precision and recall and it is a good metric if true and false observations both have the same importance.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

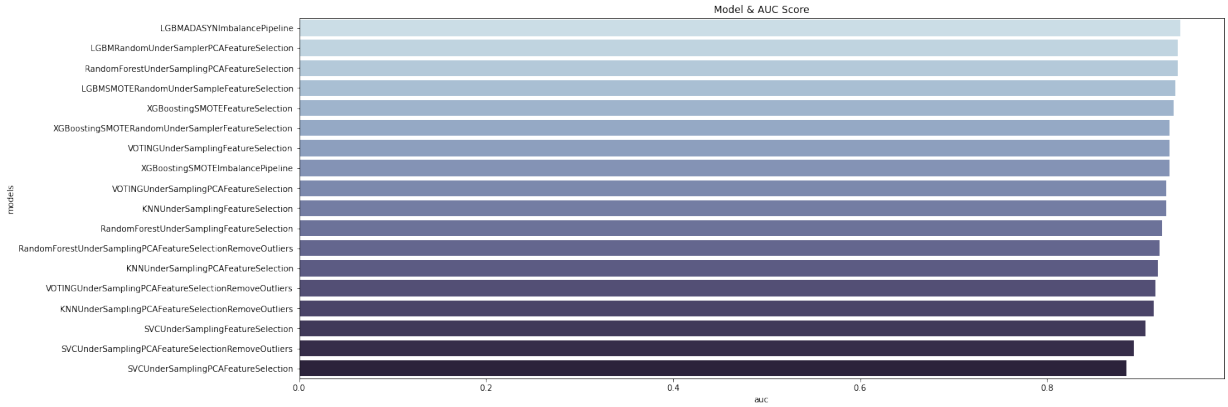


F2 Score is a kind a modified version of F1 and it gives more importance to recall. If positive observations have more importance than False observations, using an F2 score is better than F1.

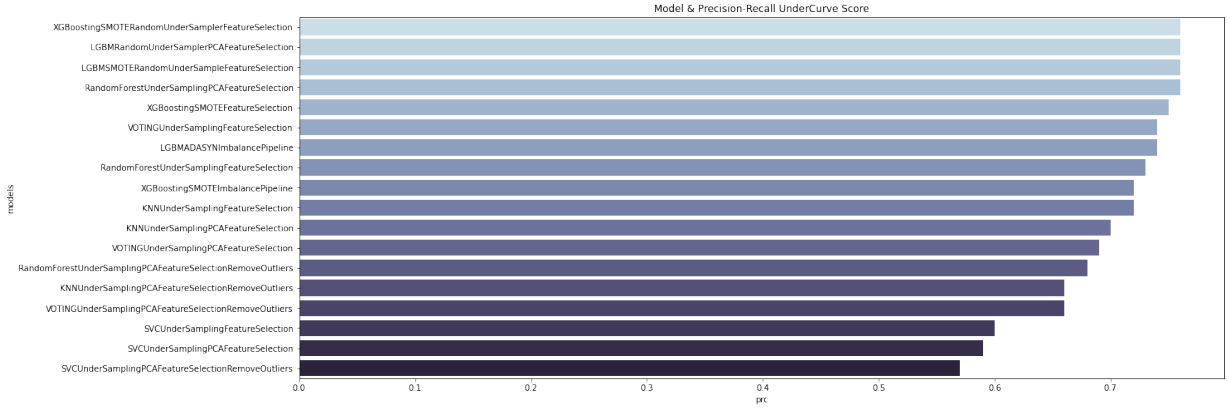
$$F_\beta = \frac{(1+\beta^2)\cdot(\text{precision}\cdot\text{recall})}{(\beta^2\cdot\text{precision}+\text{recall})}$$



AUC Score is a good criterion to evaluate the classification quality of the model. Better AUC means, the model can separate True and False better.



PR Curve Under Score gives insights into the prediction quality of positive (minority) observations of the model.



4- RESULT ANALYSIS AND DISCUSSION

In the table below, it can be seen that, val_recall and recall scores are very close. We can interpret it like there is no overfitting problem.

models	val_recall	recall	precision	f1	f2	auc	prc
KNNUnderSamplingFeatureSelection	0.771	0.799	0.602	0.686	0.750	0.927	0.72
KNNUnderSamplingPCAFeatureSelection	0.763	0.807	0.590	0.681	0.752	0.918	0.70
KNNUnderSamplingPCAFeatureSelectionRemoveOutliers	0.756	0.803	0.576	0.671	0.745	0.914	0.66
SVCUnderSamplingFeatureSelection	0.840	0.847	0.542	0.661	0.761	0.905	0.60
SVCUnderSamplingPCAFeatureSelection	0.797	0.799	0.564	0.661	0.737	0.885	0.59
SVCUnderSamplingPCAFeatureSelectionRemoveOutliers	0.776	0.763	0.549	0.639	0.708	0.893	0.57
RandomForestUnderSamplingFeatureSelection	0.856	0.859	0.556	0.638	0.754	0.923	0.73
RandomForestUnderSamplingPCAFeatureSelection	0.846	0.884	0.564	0.682	0.790	0.940	0.76
RandomForestUnderSamplingPCAFeatureSelectionRemoveOutliers	0.834	0.843	0.526	0.648	0.752	0.920	0.68
XGBoostingSMOTEFeatureSelection	0.864	0.860	0.576	0.690	0.784	0.935	0.75
XGBoostingSMOTERandomUnderSamplerFeatureSelection	0.886	0.839	0.595	0.696	0.776	0.931	0.76
XGBoostingSMOTEImbalancePipeline	0.753	0.834	0.620	0.707	0.773	0.931	0.72
LGBMADASYNImbalancePipeline	0.772	0.839	0.585	0.690	0.772	0.942	0.74
LGBMSMOTERandomUnderSampleFeatureSelection	0.882	0.862	0.561	0.679	0.778	0.937	0.76
LGBMRandomUnderSamplerPCAFeatureSelection	0.824	0.859	0.571	0.686	0.780	0.940	0.76
VOTINGUnderSamplingFeatureSelection	0.814	0.833	0.566	0.669	0.758	0.931	0.74
VOTINGUnderSamplingPCAFeatureSelection	0.817	0.828	0.561	0.673	0.758	0.927	0.69
VOTINGUnderSamplingPCAFeatureSelectionRemoveOutliers	0.802	0.816	0.521	0.636	0.733	0.916	0.66

KNN models have a great precision score which proves they can detect the majority very well but they have relatively lower recall which means not very good to detect “True”(minority) labels. Overall, they have an average AUC score.

SVC models performed the lowest performance than other models. The reason behind this might be poor hyperparameter tunning.

RandomForest models have high recall but precision is low. This means they are very good to detect True labels.

XGBoost & LigthGBM models perform best overall. The difference is XGBoost models have better precision, LigthGBM has a better recall.

PCA was very beneficial to handle with correlated columns and in every model, it raised the performance. On the other hand, removing outliers decreased the results, which means, in this model, outliers contain valuable information to classify labels.

If both True and False labels have the same importance, the F1 Score is best to evaluate them and XGBoost models have the best F1 scores. In my opinion, predicting True observations is more valuable than False ones. Therefore, using the F2 score and AUC score is better than F1 to evaluate.

To sum up, “RandomForestUnderSamplingPCAFeatureSelection” and “XGBoostingSMOTEFeatureSelection” models are my best models.