Business Data Science Kaggle Competition Report

by

Mounika Tarigopula

This report contains the brief description of the steps which I followed on the given dataset to determine the predictions. The given problem is a binary classification. All my predictions are soft labels.

Data Exploration:

Observe red how the given dataset looks like 24 features, training data set contains 16383 samples, testing data set contains 16385 samples.

```
[ ] train_data_set = pd.read_csv("/content/drive/My Drive/Kaggle/train_final.csv", index_col='Id')
    train_data_set.head()
```

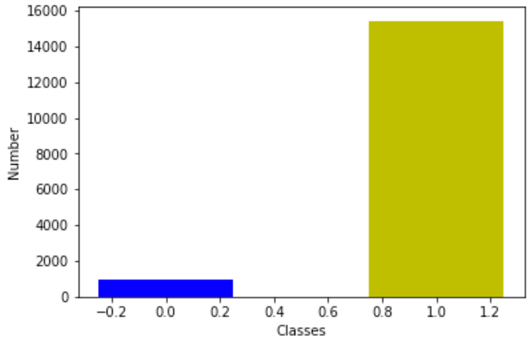
Id	Y	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19
1	1	25884	1	33.63	118596	1	0	118595	125738	1	3	1	2	121374	-2.266430	1945	118450	119184	1	121372
2	1	34346	1	10.62	118041	1	0	117902	130913	1	1	1	23	118943	-0.305612	15385	117945	292795	1	259173
3	1	34923	1	1.77	118327	1	0	117961	124402	1	2	1	1	118786	2.015561	7547	118933	290919	1	118784
4	1	80926	1	30.09	118300	1	0	117961	301218	1	0	1	1	118332	-3.172501	4933	118458	118331	1	307024
5	1	4674	1	1.77	119921	1	0	119920	302830	1	0	1	2	128231	0.573767	13836	142145	4673	1	128230

```
test_data_set.head()
```

	Id	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f
0	16384	37733	1	1.77	118603	1	0	118602	118097	1	0	1	1	117888	2.453740	13881	117941	117887	1	117
1	16385	312129	1	3.54	118052	1	0	117961	290919	1	4	1	43	118322	-0.012317	14638	118992	290919	1	118
2	16386	24884	1	23.01	118300	1	0	117961	302830	1	0	1	1	128231	1.000000	770	119181	4673	1	128
3	16387	4674	1	1.77	119091	1	0	119062	118036	1	9	1	1	117908	1.000000	16752	143531	290919	1	117
4	16388	68725	1	3.54	118300	1	0	117961	171056	1	0	1	6	118639	-0.503250	4945	118360	118638	1	118

The Y label contains imbalanced number of 1’s and 0’s as shown below:

```
Count when Y = 0:  948
Count when Y = 1:  15435
```



Then I have marked the ‘Id’ variable as index in training and test data set because it doesn’t not help in predicting the values.

I have checked for null values in training data set features but there are no null values.

```
# Verifying whether there are any NULL values
train_data_set.isnull().sum()
```

```
Y      0
f1     0
f2     0
f3     0
f4     0
f5     0
f6     0
f7     0
f8     0
f9     0
f10    0
f11    0
f12    0
f13    0
f14    0
f15    0
f16    0
f17    0
f18    0
f19    0
f20    0
f21    0
f22    0
f23    0
f24    0
dtype: int64
```

Described the training dataset for better understanding of the dataset based on mean, std etc...

```
[ ] # Getting mean and other statistics of each column
    train_data_set.describe()
```

	Y	f1	f2	f3	f4	f5	f6	f7	
count	16383.000000	16383.000000	16383.000000	16383.000000	16383.000000	16383.000000	16383.000000	16383.000000	16383.0000
mean	0.942135	43007.775865	1.044375	11.770938	118323.581456	1.044436	0.050052	117089.674113	169730.1786
std	0.233495	33611.182771	0.264806	353.187115	4518.059755	0.265601	0.293892	10261.292970	69396.6778
min	0.000000	-1.000000	1.000000	1.770000	23779.000000	1.000000	0.000000	4292.000000	4673.0000
25%	1.000000	20311.000000	1.000000	1.770000	118096.000000	1.000000	0.000000	117961.000000	117906.0000
50%	1.000000	35527.000000	1.000000	1.770000	118300.000000	1.000000	0.000000	117961.000000	128130.0000
75%	1.000000	74240.500000	1.000000	3.540000	118386.000000	1.000000	0.000000	117961.000000	234498.5000
max	1.000000	312152.000000	7.000000	43910.160000	286791.000000	9.000000	10.000000	311178.000000	311867.0000

Logistic Regression:

My initial step is applying the logistic regression to the training data set and predict the values on test data. As this is one of the first technique which I learned in class, so I tried this to the training set. The area under AUC score was very low i.e. 0.5

```
[ ] logistic = LogisticRegression()
    logistic.fit(train_X, train_Y)
    predictions = logistic.predict(test_X)
    test_predictions=logistic.predict(test_data)
    auc_score = roc_auc_score(test_Y, predictions)
    auc_score

    0.5
```

```
[ ] solution=pd.DataFrame({"id":test_data_set.Id,"Y":test_predictions})
    solution.to_csv(f"{dir}/logistic.csv", index=False)
```

logistic.csv 11 days ago by Mounika Tarigopula add submission details	0.51359	0.52258

Random Forest:

I have applied Random forest by tunning the hyper parameters on small training data set and got the best hyper parameters, which are used on complete training data set and which gave me better AUC score when compared to logistic regression. I have predicted the soft labels on the test dataset. The score obtained is 0.88

```
RF_model = RandomForestClassifier()
parameters = {
            'criterion':('gini','entropy'),
            'max_depth': (9,10),
            'n_estimators' : [7000]}
cv = GridSearchCV(RF_model, param_grid=parameters, cv=5)
cv.fit(X_s,y_s)
```

```
GridSearchCV(cv=5, error_score=nan,
             estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                              class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              max_samples=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators=100, n_jobs=None,
                                              oob_score=False,
                                              random_state=None, verbose=0,
                                              warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'criterion': ('gini', 'entropy'), 'max_depth': (9, 10),
                         'n_estimators': [7000]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

```
[ ] cv.best_params_
```

```
{'criterion': 'gini', 'max_depth': 9, 'n_estimators': 7000}
```

```
[ ] hyper_parameters = {
     'n_estimators': 7000,'max_depth': 10}
```

```
[ ] RF_model = RandomForestClassifier(**hyper_parameters)
    RF_model.fit(X,y)
    predictions = RF_model.predict_proba(test_data)[:,1]
    #test_predictions=RF_model.predict(test_data)
```

```
[ ] solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictions})
    solution.to_csv(f"{dir}/random.csv", index=False)
```

random.csv 10 days ago by Mounika Tarigopula add submission details	0.88077	0.85597

XGBoost:

I have applied XBGClassifier to the training data set and got the AUC score as 0.86. Then I have started turning the hyper parameters on small data set and got the best hyper parameters, which are used on complete training data set and which gave me better AUC score i.e. 0.87 when compared to default XGB classifier.

```
[ ] model_xgb = xgb.XGBClassifier()
    model_xgb.fit(X, y)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

```
[ ] predictions1 = model_xgb.predict_proba(test_data)[:,1]
```

```
[ ] solution=pd.DataFrame({"id":test_data_set.Id,"Y":predictions1})
    solution.to_csv(f"{dir}/xgb.csv", index=False)
```

```
[ ] xgboost_model = xgb.XGBClassifier(objective='multi:softmax', num_class=10,eval_metric='auc')
```

```
xgb_parameters = {
                      'max_depth':                    [5],
                      'learning_rate':                [0.1],
                      'n_estimators':                 [728, 735, 742],
                      'objective':                    ['binary:logistic'],
                      'booster':                      ['gbtree'],
                      'tree_method':                  ['auto'],
                      'reg_alpha':                    [0.0011],
                      'reg_lambda':                   [0.9, 1.2]
}
```

```
[ ] xgb_cv = GridSearchCV(xgboost_model, xgb_parameters, cv=5, n_jobs=-1)
    xgb_cv.fit(X_s,y_s)
```

```
[ ] xgb_cv.best_params_

    {'learning_rate': 0.05, 'max_depth': 5}
```

```
[ ] hyper_parameters = {'base_score': 0.5, 'booster':'gbtree', 'colsample_bylevel':1,
          'colsample_bynode':1, 'colsample_bytree':0.3, 'gamma':0.0,
          'learning_rate':0.0491, 'max_delta_step':0, 'max_depth':11,
          'min_child_weight':1.5, 'missing': None, 'n_estimators': 255,'n_jobs': 1,
          'nthread': None, 'objective':'binary:logistic', 'random_state': 0,
          'reg_alpha': 0.0011, 'reg_lambda': 1, 'scale_pos_weight':1, 'seed':None,
          'silent': None, 'subsample': 1, 'verbosity': 1}
```

```
[ ] xgboost_model = xgb.XGBClassifier(**hyper_parameters)
```

```
[ ] xgboost_model.fit(X,y)

    XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                  colsample_bynode=1, colsample_bytree=1, gamma=0,
                  learning_rate=0.05, max_delta_step=0, max_depth=5,
                  min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
                  nthread=None, objective='binary:logistic', random_state=0,
                  reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                  silent=None, subsample=1, verbosity=1)
```

```
[ ] predictions = xgboost_model.predict_proba(test_data)[:,1]
    solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictions})
    solution.to_csv(f"{dir}/xgb.csv", index=False)
```

xgb.csv 10 days ago by Mounika Tarigopula add submission details	0.86497	0.84418	☐
xgb.csv 10 days ago by Mounika Tarigopula add submission details	0.87002	0.84990	☐

LightGBM:

Even after tunning the hyper parameters in XGBClassifier, my AUC score didn't increase more than 0.87. Then I have applied the LightGBM to my training data set and got the AUC score of 0.89 which is high when compared to XGBClassifier.

```
[ ] lgbm_model = lgb.LGBMClassifier()
    lgbm_model.fit(X,y)
    predictio = lgbm_model.predict_proba(test_data)[:,1]
    solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictio})
    solution.to_csv(f"{dir}/lgbm.csv", index=False)
```

lgbm.csv 9 days ago by Mounika Tarigopula add submission details	0.89668	0.87313

Outlier Removal:

For better scores, thought of identifying and removing the outliers in the data set. An outlier is nothing but an observation that is unlike the other observations. It doesn't fit in the same way. I have applied below methods to remove the outliers.

```
for x in train_data_set.columns:
        print(x," : ", train_data_set[x].unique())

#f2,f5,f6,f9,f11,f18,f20,f21,f22,f24
```

```
Y  :  [1 0]
f1  :  [25884 34346 34923 ... 43426 34326  6706]
f2  :  [1 2 4 3 5 6 7]
f3  :  [3.363000e+01 1.062000e+01 1.770000e+00 3.009000e+01 3.540000e+00
 2.301000e+01 5.310000e+00 1.770000e+01 3.540000e+01 2.832000e+02
 7.080000e+00 8.850000e+01 1.823100e+02 2.655000e+01 8.319000e+01
 4.796700e+02 1.593000e+01 1.947000e+01 7.257000e+01 1.416000e+01
 1.239000e+01 2.832000e+01 4.071000e+01 2.478000e+01 2.513400e+02
 7.611000e+01 5.664000e+01 1.221300e+02 1.185900e+02 3.894000e+01
 2.124000e+01 6.372000e+01 1.062000e+02 4.425000e+01 3.186000e+01
 5.133000e+01 2.301000e+02 1.150500e+02 3.327600e+02 8.142000e+01
 6.903000e+01 3.717000e+01 4.779000e+01 6.018000e+01 4.956000e+01
 3.486900e+02 6.761400e+02 1.283250e+03 6.726000e+01 3.150600e+02
 3.752400e+02 6.549000e+01 1.097400e+02 1.274400e+02 1.522200e+02
 5.398500e+02 1.239000e+02 9.381000e+01 1.309800e+02 8.460600e+02
 2.230200e+02 3.433800e+02 1.840800e+02 1.451400e+02 7.080000e+01
 5.841000e+01 4.060380e+03 5.487000e+01 8.496000e+01 9.558000e+01
 6.195000e+01 1.416000e+02 1.007130e+03 2.725800e+02 4.248000e+01
 1.327500e+02 8.673000e+01 1.947000e+02 2.463840e+03 5.310000e+01
 2.478000e+02 1.446090e+03 2.247900e+02 1.079700e+02 2.548800e+02
 2.371800e+02 5.841000e+02 1.044300e+02 2.424900e+02 1.557600e+02
 2.212500e+02 1.008900e+02 1.876200e+02 1.610700e+02 1.805400e+03
 4.602000e+01 8.850000e+01 1.699200e+02 1.929300e+02 9.027000e+01
 1.290330e+03 2.796600e+02 2.460300e+02 1.026600e+03 4.548900e+02
 1.203600e+02 1.752300e+02 7.788000e+01 9.912000e+01 4.902900e+02
 6.442800e+02 3.097500e+02 3.221400e+02 2.265600e+02 1.345200e+02
 7.965000e+01 2.601900e+02 2.743500e+02 7.129560e+03 1.770000e+02
 1.026600e+02 2.088600e+02 1.132800e+02 9.204000e+01 2.030190e+03
```

* **Z-Score**: It is used to describe any data point by finding their relationship with standard deviation and mean of the group of data points.

```
[ ] from scipy import stats
    import numpy as np
    z = np.abs(stats.zscore(train_data_set))
    print(z)
```

```
[[0.24782827 0.50948217 0.16758185 ... 0.17005468 0.01027273 0.16927037]
 [0.24782827 0.25771304 0.16758185 ... 0.17005468 0.01027304 0.16927037]
 [0.24782827 0.24054561 0.16758185 ... 0.17005468 0.01027304 0.16927037]
 ...
 [0.24782827 0.43584372 0.16758185 ... 0.17005468 0.01027304 0.16927037]
 [0.24782827 1.05521006 0.16758185 ... 0.17005468 0.01026959 0.16927037]
 [0.24782827 0.86672537 0.16758185 ... 0.17005468 0.01027273 0.16927037]]
```

```
[ ] threshold = 3
    print(np.where(z > 3))
```

```
(array([    1,     3,    16, ..., 16375, 16379, 16380]), array([19, 19,  2, ..., 16, 18, 21]))
```

```
data_clean = train_data_set
data_clean = data_clean[(z < 3).all(axis=1)]
```

```
[ ] train_data_set.shape
```

```
(16383, 25)
```

```
[ ] data_clean.shape
```

```
(9525, 25)
```

```
[ ] z = np.abs(stats.zscore(train_data_set))
    threshold = 3
    print(np.where(z > 3))
```

* **<u>IQR (Interquartile range) Score:</u>** It is a measure of statistical dispersion which is used to identify the outlier i.e. the difference between 75th and 25th percentiles.

```
[ ] Q1 = train_data_set.quantile(0.25)
    Q3 = train_data_set.quantile(0.75)
    IQR = Q3 - Q1
    print(IQR)
```

```
Y           0.000000
f1      53929.500000
f2          0.000000
f3          1.770000
f4        290.000000
f5          0.000000
f6          0.000000
f7          0.000000
f8     116592.500000
f9          0.000000
f10         1.000000
f11         0.000000
f12         1.000000
f13      1116.000000
f14         1.704562
f15     34348.000000
f16      2144.000000
f17    172521.000000
f18         0.000000
f19      1732.000000
f20         0.000000
f21         0.000000
f22         0.000000
f23         8.000000
f24         0.000000
f25         0.000000
dtype: float64
```

```
[ ] boston_iqr_clean = train_data_set[~((train_data_set < (Q1 - 1.5 * IQR)) |(train_data_set > (Q3 + 1.5 * IQR))).any(axis
```

```
[ ] boston_iqr_clean.shape
```

```
(2397, 25)
```

* **Based on Standard Deviation:** If any row contains more than five featured which are away from mean with two standard deviations then we are removing that record. But we got only one record as outlier in this method.

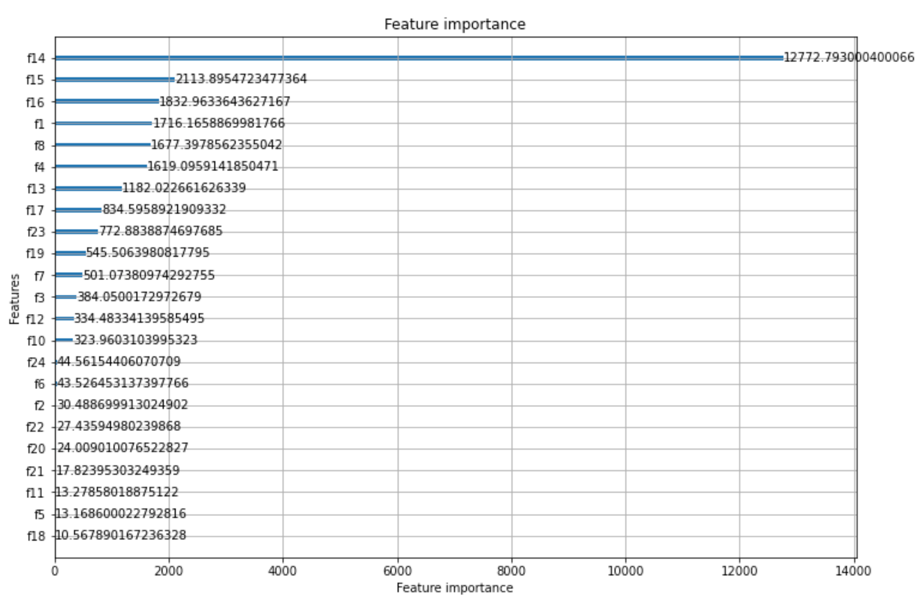
```
#Outliers
outliers=set()
X_outlier=X.copy()
Y_outlier=Y.copy()
for idx, r in X_outlier.iterrows():
    outlier_count=0
    for i in X_outlier.columns:
        c_mean=X_outlier[i].mean()
        c_std=X_outlier[i].std()
        if np.abs(r[i]-c_mean) > (2*c_std):
            outlier_count+=1
    if outlier_count>5 and Y_outlier[idx]==1:
            outliers.add(idx)
print(outliers)
X_outlier.drop(outliers, axis=0, inplace=True)
Y_outlier.drop(outliers, axis=0, inplace=True)
print(X_outlier.shape)
```

To my surprise I have received very less or no impact on the score for all these techiniques.

Feature Importance:

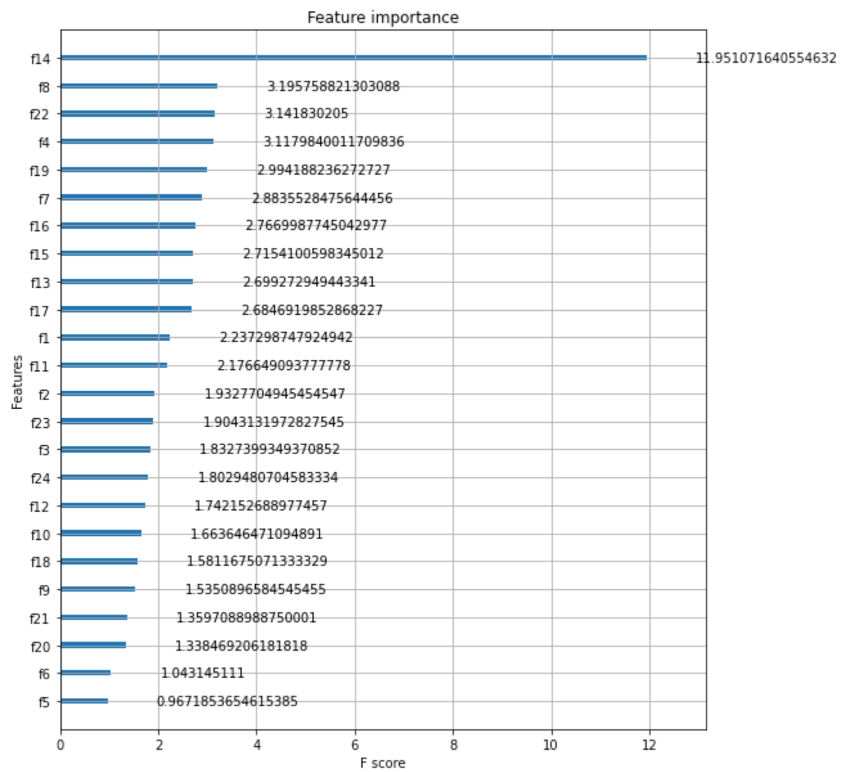
Till now I got the highest score via LGBMClassifier model. So, I have taken the gain-based plot of feature importance on this model, which helps in knowing the importance of each feature and we can eliminate the noisy features.

```
# Feature importance as per LGBM
lgbm_model=lgb.LGBMClassifier(random_state=1)
lgbm_model.fit(X,y)
ax = lgb.plot_importance(lgbm_model, importance_type='gain',max_num_features=50)
fig = ax.figure
fig.set_size_inches(12, 8)
```



I have taken the gain-based plot of feature importance on this XBG model, which helps in knowing the importance of each feature and we can eliminate the noisy features. Below plot shows the importance features.

```
xgb_model=xgb.XGBClassifier(learning_rate=0.13,n_estimators=500,max_depth=4, base_score=0.8, eval_metric='auc')
xgb_model.fit(X,y)
ax = xgb.plot_importance(xgb_model, importance_type='gain',max_num_features=50)
fig = ax.figure
fig.set_size_inches(9, 10)
```



Based on this feature importance plots I have selected the top 18,15,10 features and applied the LGBMClassifier and XGBClassifier model to that.
LGBM gave good score on top 18 features when compared to the LGBM score on all features.
XGBoost gave good score on top 10 features when compared to the XGBoost score on all features.

```
def columns(X,lists):
    Modified_X=X[lists]
    return Modified_X

column_10=columns(X,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19'])
column_15=columns(X,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24'])
column_18=columns(X,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24','f6','f2','f

test_data_10=columns(test_data,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19'])
test_data_15=columns(test_data,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24'])
test_data_18=columns(test_data,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24','

lgbm_model = lgb.LGBMClassifier()
lgbm_model.fit(column_10,y)
predictio = lgbm_model.predict_proba(test_data_10)[:,1]
solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictio})
solution.to_csv(f"{dir}/lgbm_10.csv", index=False)

lgbm_model = lgb.LGBMClassifier()
lgbm_model.fit(column_15,y)
predictio = lgbm_model.predict_proba(test_data_15)[:,1]
solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictio})
solution.to_csv(f"{dir}/lgbm_15.csv", index=False)

lgbm_model = lgb.LGBMClassifier()
lgbm_model.fit(column_18,y)
predictio = lgbm_model.predict_proba(test_data_18)[:,1]
solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictio})
solution.to_csv(f"{dir}/lgbm_18.csv", index=False)
```

```
def columns(X,lists):
    Modified_X=X[lists]
    return Modified_X

column_10=columns(X,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19'])
column_15=columns(X,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24'])
column_18=columns(X,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24','f6','f2','f

test_data_10=columns(test_data,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19'])
test_data_15=columns(test_data,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24'])
test_data_18=columns(test_data,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24','

lgbm_model = xgb.XGBClassifier()
lgbm_model.fit(column_10,y)
predictio = lgbm_model.predict_proba(test_data_10)[:,1]
solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictio})
solution.to_csv(f"{dir}/xgm_10.csv", index=False)

lgbm_model = lgb.XGBClassifier()
lgbm_model.fit(column_15,y)
predictio = lgbm_model.predict_proba(test_data_15)[:,1]
solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictio})
solution.to_csv(f"{dir}/xgm_15.csv", index=False)

lgbm_model = lgb.XGBClassifier()
lgbm_model.fit(column_18,y)
predictio = lgbm_model.predict_proba(test_data_18)[:,1]
solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictio})
solution.to_csv(f"{dir}/xgm_18.csv", index=False)
```

lgbm_18.csv 9 days ago by Mounika Tarigopula add submission details	0.89569	0.87922
top_10_xgm.csv 6 days ago by Mounika Tarigopula add submission details	0.90581	0.88815

Ensemble:

I have read an article in towardsdatascience regarding ensemble technique which stated that take a mean of three different model predictions in order to improve the score. I have tried the same but with two models i.e. LGBM and XGBoost.

```
def columns(X,lists):
    Modified_X=X[lists]
    return Modified_X

column_10=columns(X,['f14','f8','f22','f4','f19','f7','f16','f15','f13','f17'])
test_data_10=columns(test_data,['f14','f8','f22','f4','f19','f7','f16','f15','f13','f17'])
xgm_model = xgb.XGBClassifier(random_state=7,learning_rate=.15,n_estimators=300,max_depth=5,base_score=0.8)
xgm_model.fit(column_10,y)
predictio = xgm_model.predict_proba(test_data_10)[:,1]
solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictio})
solution.to_csv(f"{dir}/top_10_xgm.csv", index=False)

xgb_model = xgb.XGBClassifier(random_state=7,learning_rate=.15,n_estimators=300,max_depth=5,base_score=0.8)
xgm_model.fit(column_18,y)
pred = xgm_model.predict_proba(test_data_18)[:,1]

final_pred = (predictio+pred)/2
final_pred.shape
solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictio})
solution.to_csv(f"{dir}/avg_10_18.csv", index=False)
```

```
xgb_model = xgb.XGBClassifier(random_state=7,learning_rate=.15,n_estimators=300,max_depth=5,base_score=0.8)
xgb_model.fit(column_18,y)
pred = xgb_model.predict_proba(test_data_18)[:,1]

final_pred = (predictio+pred)/2
final_pred.shape
solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictio})
solution.to_csv(f"{dir}/avg_lgbm_xgm.csv", index=False)
```

avg_lgbm_xgm.csv 6 days ago by Mounika Tarigopula add submission details	0.89022	0.87412

Resampling Approaches:

I have used the below resampling techniques in order to balance the classes in the training data set and get the better scores.

*<u>Random Under Sampler:</u> It randomly removes the samples from the majority class. It gave less score when compared to the same model with all the features data. Random cutting of training data set led to the decrease in score.

```
[ ] # instantiating the random undersampler
    rus = RandomUnderSampler()
    # resampling X, y
    X_rus, y_rus = rus.fit_resample(X.values, y.values)
    lgbm_model = lgb.LGBMClassifier()
    lgbm_model.fit(X_rus,y_rus)
    predictio = lgbm_model.predict_proba(test_data)[:,1]
    solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictio})
    solution.to_csv(f"{dir}/randomundersample.csv", index=False)
```

randomundersample.csv 0.89440 0.87660
8 days ago by Mounika Tarigopula
add submission details

*Tome Links: It is an effective technique in under sampling which removes the samples in pairs one from each class which are close to the decision boundary. It removes the majority class samples making the decision boundary more prominent in auto mode. It gave me good score when compared to the model with original data set.

```
def columns(X,lists):
    Modified_X=X[lists]
    return Modified_X

column_18=columns(X,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24','f6','f2','f
test_data_18=columns(test_data,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24','

ros = TomekLinks()
x_tls, y_tls = ros.fit_resample(column_18.values, y.values)

lgbm_model = lgb.LGBMClassifier()
lgbm_model.fit(x_tls,y_tls)
predictio = lgbm_model.predict_proba(test_data_18)[:,1]
solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictio})
solution.to_csv(f"{dir}/TomekLinks.csv", index=False)
```

TomekLinks.csv 0.89440 0.87660
8 days ago by Mounika Tarigopula
add submission details

* SMOTE: It is an oversampling technique where it generates the synthetic samples from the minority class. It is used to obtain the class balanced training data set. It gave me very less score when compare to applying the model with original training data set. Looks like there is overlapping of features between the classes so it didn't work well.

```
oversample = SMOTE()
X_o, y_o = oversample.fit_resample(X.values, y.values)
lgbm_model = lgb.LGBMClassifier()
lgbm_model.fit(X_o,y_o)
predictio = lgbm_model.predict_proba(test_data)[:,1]
solution= pd.DataFrame({"id":test_data_set["Id"],"Y":predictio})
solution.to_csv(f"{dir}/SMOTE.csv", index=False)
```

SMOTE.csv 0.86741 0.85471
8 days ago by Mounika Tarigopula
add submission details

Grid Search CV:

Till now I am tunning the hyper parameters manually and cams to know that estimators and learning rate are indirectly linked with each other. I have tried so many numbers and finally listed some values for estimators and learning rate which helped me to get the good score.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
train = X_train.loc[:,('f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24','f6','f2',
test = X_test.loc[:,('f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24','f6','f2','f
d_train = xgb.DMatrix(data=train,label=y_train)
d_valid = xgb.DMatrix(test,label=y_test)
learning_rate =[0.04, 0.045, 0.05, 0.055, 0.06, 0.065, 0.07, 0.075, 0.08, 0.085, 0.09, 0.095, 0.1]
max_depth = [9,10,11,12,13,14,15,16]
n_estimators = [100,110,120,130,140,150,160,170,180,190,200,210,220,230,240,250]
alpha=[0.0009, 0.001, 0.0011]

for a in learning_rate:
    for b in max_depth:
        for c in n_estimators:
            for d in alpha:

                params = {"objective":"binary:logistic",'colsample_bytree': 0.3,'learning_rate':a,
                    'max_depth':b,'min_child_weight':1.5,'alpha':d}
                model = xgb.train(params, d_train, c)

                predictions=model.predict(d_valid)

                print(a," ",b," ",c," ",d," ",roc_auc_score(y_test.to_numpy(), predictions))
```

```
0.04    9    100    0.0009    0.8864486129554179
0.04    9    100    0.001    0.8875053440834387
0.04    9    100    0.0011    0.8875068547784037
0.04    9    110    0.0009    0.8856747594595943
0.04    9    110    0.001    0.887047225835301
0.04    9    110    0.0011    0.8870460928140773
0.04    9    120    0.0009    0.8886927503259324
0.04    9    120    0.001    0.890046710688318
0.04    9    120    0.0011    0.8900459553408354
0.04    9    130    0.0009    0.8886957717158623
0.04    9    130    0.001    0.8902831344503414
0.04    9    130    0.0011    0.8902827567765998
0.04    9    140    0.0009    0.8893423491608845
0.04    9    140    0.001    0.8903892607716327
```

The best parameters which have the best results are:
learning_rate = 0.075,0.080,0.085
max_depth = 15,16
n_estimators = 120,200,222
alpha = 0.001,0.0011

I have taken the top 18 features from the XGBoost feature importance plot and applied XGBoost model in which as hyperparameters provided the result which we got on the top and predicted the results which gave me the best score.

```
[ ] def columns(X,lists):
        Modified_X=X[lists]
        return Modified_X

    column_18=columns(X,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24','f6','f2','f
    test_data_18=columns(test_data,['f14','f15','f16','f1','f8','f4','f13','f17','f23','f19','f7','f3','f12','f10','f24','
    dtrain = xgb.DMatrix(data=column_18,label=y)
    dtest = xgb.DMatrix(test_data_18)
```

```
[ ] params = {"objective":"binary:logistic",'colsample_bytree': 0.3,'learning_rate':0.085,
                  'max_depth':16,'min_child_weight':1.5,'alpha':0.001}
    model = xgb.train(params, dtrain, 120)
    preds=model.predict(dtest)
    solution= pd.DataFrame({"id":test_data_set["Id"],"Y":preds})
    solution.to_csv(f"{dir}/feature_mod_1.csv", index=False)
```

```
[ ] params = {"objective":"binary:logistic",'colsample_bytree': 0.3,'learning_rate':0.080,
                  'max_depth':16,'min_child_weight':1.5,'alpha':0.0011}
    model = xgb.train(params, dtrain, 200)
    preds=model.predict(dtest)
    solution= pd.DataFrame({"id":test_data_set["Id"],"Y":preds})
    solution.to_csv(f"{dir}/feature_mod__2.csv", index=False)
```

feature_mod_1.csv 6 days ago by Mounika Tarigopula add submission details	0.91566	0.89861
feature_mod__2.csv 6 days ago by Mounika Tarigopula add submission details	0.91712	0.90314

Based on the above results I have applied the learning rate = 0.080 and alpha = 0.0011, max_depth = 16 and n_estimators = 200 for the top 10 features and got the best score.

```
def columns(X,lists):
    Modified_X=X[lists]
    return Modified_X

column_10=columns(X,['f14','f8','f22','f4','f19','f7','f16','f15','f13','f17'])
test_data_10=columns(test_data,['f14','f8','f22','f4','f19','f7','f16','f15','f13','f17'])
params = {"objective":"binary:logistic",'colsample_bytree': 0.3,'learning_rate': 0.080,
              'max_depth': 16,'min_child_weight':1.5,'alpha':0.0011,'n_estimators': 200}
xgb_las = xgb.XGBClassifier(**params)
xgb_las.fit(column_10,y)
ypred = xgb_las.predict_proba(test_data_10)[:,1]
solution= pd.DataFrame({"id":test_data_set["Id"],"Y":ypred})
solution.to_csv(f"{dir}/xgm_toppp_10.csv", index=False)
```

xgm_toppp_10.csv 5 days ago by Mounika Tarigopula add submission details	0.91595	0.89719

In Conclusion, This Kaggle competition was a great learning experience as it helped me to apply the theoretical techniques which I learnt in class, I have applied it in our dataset in order to get best AUC Score. Along with that I have researched the techniques, read the documentation about the models which helped me to gain more knowledge. Along with that the Kaggle leaderboard was a great motivator it helped to do best so that I get the best score. My best scores worked very nicely in private leader boarder also with increase in the AOC scores.