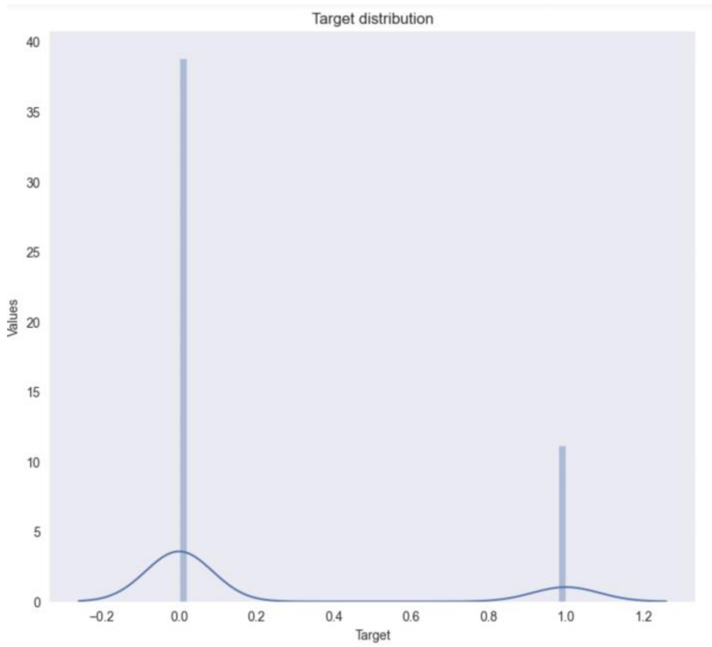
Kaggle Report by Rithu Anand Krishnan

1. **What have I tried?**
This was my first official Kaggle competition, it was daunting at first but now in retrospect, I am very proud of my efforts even if I wasn't on the top 10 list. During the course of the competition, I had the opportunity to try multiple things, few things worked and multiple of them failed. Here is a brief list of what worked and what didn't.

    a. Performed a distribution check on target value. Found that the dataset had more entries for 0 when compared to 1. This helped me understand that the dataset wasn't equally divided between both values.

    b. Ran describe function on both train and test dataset. Found that multiple columns were constants. Here is a list:
    ['6','30','35', '40', '60','80', '90', '102', '118', '129', '138', '142','163', '206', '216', '239', '322', '323', '333']
    I thought this was jackpot, that I found some columns that could be removed and there by my score would increase but I was wrong. I learnt that if the variance is zero, it means that the feature is constant and will not improve the performance of the model. It was supposed to make the models run faster but the time difference wasn't significant, so I let these columns be.

    c. Preprocessing data: With the help of describe function, I found that multiple columns had values with a negative to positive range. It wasn't evenly distributed. I applied MinMaxScaler processing with range (-1, 1) on the following columns.
    columns = ['4', '119', '180','383', '384', '379']. For columns within positive range columns = ['63','132','382','146','198'], I used MinMaxScaler with default parameters.
    This increased the AUC score a little but not significantly.

    I also performed RobustScaler preprocessing but that wasn't much of a help to me. I moved on to finding the skewness of these columns. I used from skew library from scipy.stats. The threshold was 0.75, I performed log1p on the values of these columns to reduce the skewness and ran my model.

I was disappointed to not see better performance. I performed PCA even though it wasn't advised because I just wanted to see how it would affect my score, turns out it's true. I can't just use PCA for every problem. This was double assurance.

Later after trying multiple things, I figured that preprocessing these columns actually brought my accuracy down, so I stopped doing it.

d. Different models and their accuracies.
   LogisticRegression with liblinear as solver, AUC = 66.22
   RandomForest with n_estimators=150, AUC = 74.89
   SVC with C 0.0.25, AUC = 57.21
   KNeighborsClassifier with n_neighbors as 5, AUC = 55.73
   XGBClassifier with n_estimators=100, AUC = 83.26
   CatBoostClassifier with iterations=500, learning_rate=0.03, depth=6, AUC = 88.001 (Best one)

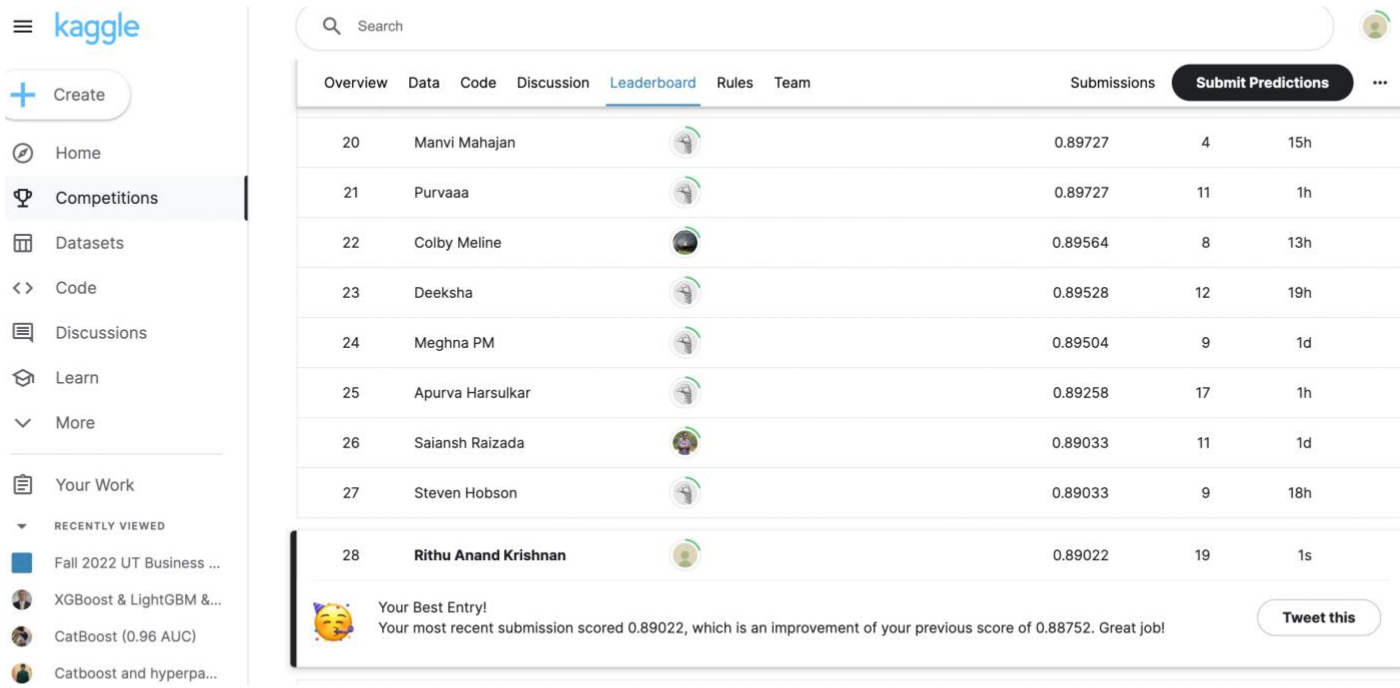
   Stacking these together with StackingCVRegressor with CV as 12, AUC = 68.09

   I didn't get a good score with these; I was able to get a decent position in my first try but later everyone seemed to have improved their models significantly and I felt lost. No matter what I tried my score wouldn't reach 89.

2. **What worked for me?**
   a. Hyperparameter tuning! I picked catboost because that performed better than all the other models I had tried.

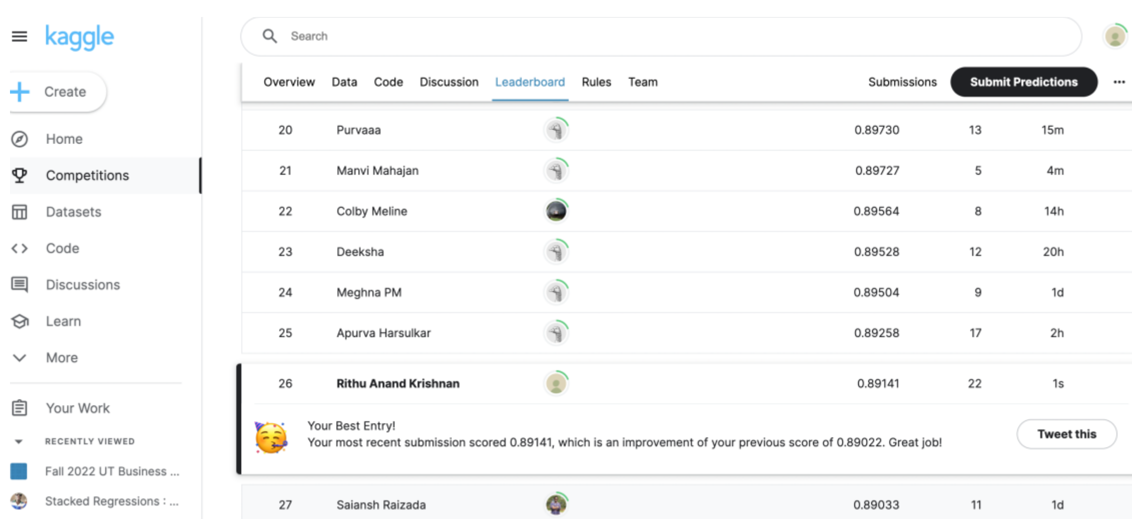
I used RandomizedSearchCV to get the best parameter values. I mainly focused on the following.
parameters = {'depth' : [2, 3, 4, 5,6,7, 8],
'learning_rate' : [0.019,0.042, 0.03,0.034, 0.045, 0.04, 0.036, 0.038, 0.032],
'iterations' : [850, 1000, 950, 1350, 1200, 1100, 1300, 1250, 1150]
}
Since I knew the target values of 0 was greater than 1. I set my class_weights=[0.55, 0.45].

And found the best parameters for Catboost to be
para = { "iterations":713, "learning_rate":0.019, "verbose":0, "max_depth":6, "class_weights":[0.55,0.45], "l2_leaf_reg":1 }. I moved up to leader board with these values.



But this wasn't enough, now my target was to get to 90 from 89.

b. After looking at the dataset closely I realized that the training set size wasn't large. I was splitting them further to make them into test and train set for my models. I got to know that I could use the entire train set to train my model without splitting for test set. I had never taken this approach before because my dataset size has always been big enough in the past. Now that I know I could leverage the entire dataset I am never going back to the old ways again. I immediately had better score by using entire train set to train the model. But the jump wasn't significant enough.



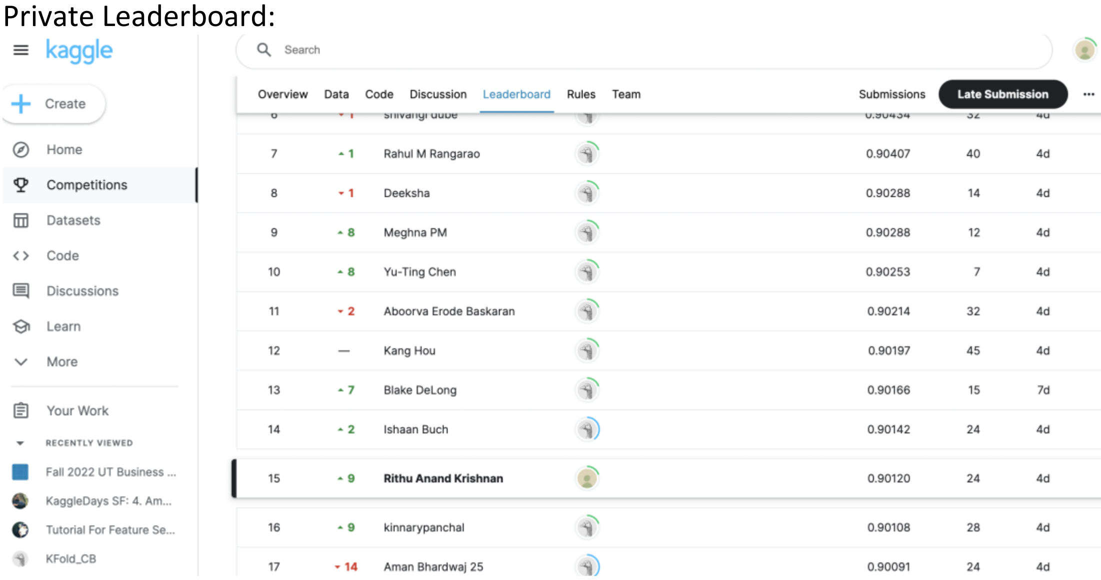
c. While actively searching the internet for help, I fumbled up on StratifiedKFold. It would let me pick different subsets of my train data, thereby letting me use all my train data in the most effective way. I used 10 as the value for my n_splits and high random state 2021 with shuffle as true.
I used catboost as my model with parameters learning_rate = 0.01, depth = 11.

It got my score up to AUC=89.89

d. I had given up on the idea of stacking because the first time I tried it, my auc values was lower than most individual model values. Later I realized that I could use stacking in a better way. I did this by first getting my base models to their best self and them clubbing them.
I used XGBoost and CatBoost.
xgb.XGBClassifier(n_estimators=7,objective="binary:logistic", random_state=42)
CatBoost with para = { "iterations":1100, "learning_rate":0.019, "max_depth":6, "class_weights":[0.55,0.45]}

I used StackingClassifier with these two models and I made catboost as the final estimator. With this I got my best leadership score.

Private Leaderboard:



3. Post-closing of Kaggle competition

a. Trying the parameters shared during the class. I stacked randomforest along with xgb and catboost. With further tuning as follows.
catboost = { 'learning_rate': 0.03, 'n_estimators':400, 'max_depth': 4, "random_state":35 ,"verbose":0, "border_count":70, "boosting_type":"Ordered", "class_weights":[0.55,0.45]}
xgb1 = xgb.XGBClassifier(n_estimators=7,objective="binary:logistic", random_state=42)
rf = RandomForestClassifier(n_estimators=50,max_depth=9, max_features='auto', bootstrap=True)

This gave me my best score ever.