

Kaggle Team Name: Justin

















Team Members: Justin Maines

Email: jtm5948@psu.edu

Kaggle Competition

Competition Description: For the second project of DS 310, we partook in a competition hosted on Kaggle.com. For the competition we were provided with data tables that included various information related to publicly traded companies. This data included things like revenue, revenue growth, expenses, and other important quantitative values that are important in identifying the value of a stock. Using these datasets, the goal of the competition was to create a classification model that determines if a trader should buy or sell a given stock. The model, which is built on the training data provided to us in a .csv file, classifies a stock with 1 for a buy or 0 for a sell. The predictions had to be outputted into another .csv file, which were then submitted to Kaggle for scoring. The competition was scored based on F1 score, with a 0.43 score being required for full points.

Final Leaderboard Position: 5/21

#	Team	Members	Score	Entries	Last	Code	Join
1	Yeman&Aoran		0.52873	27	10h		
2	Double B		0.52298	18	1h		
3	anonYmous		0.52023	12	1d		
4	xaviour007		0.51685	13	3d		
5	Justin		0.51412	2	4h		
 Your Best Entry! Your most recent submission scored 0.51412, which is an improvement of your previous score of 0.48467. Great job!							Tweet this
6	thebeannnnnn		0.51190	9	4h		
7	ns0806		0.51162	12	3d		
8	srio		0.51000	4	2d		
9	Nick Mangione		0.50887	10	3d		
10	ajk6604		0.50761	19	3h		
11	AkashKoneru		0.50434	4	3d		
12	Evie & Alex		0.50295	5	3d		
13	i5204		0.50140	10	8h		
14	Kelly Wolfe		0.49710	26	3d		
15	ZiyaoYang3000		0.48680	7	2m		

Solution Details

Pre-processing: In the pre-processing step of my project, I started by mounting my Google Drive for easy access to all the .csv file data I needed to use. With this drive mounted, I was able to navigate to the location the files were held and importing them to **train_data** and **test_data** pandas data frames. With the .csv files added into my Jupyter notebook as data frames, I used the **.head()** function on each table to get a feel for the dataset, then used the **.describe(include='all').T** function to see how many of the columns have null data that needs to be fixed.

Feature Engineering: To begin feature engineering, I used the knowledge I gained from pre-processing to write a short function that filled the null values with the average of its column.

```
for a in train_data:
    if train_data[a].count() < 3459:
        train_data[a].fillna(train_data[a].mean(), inplace=True)

for a in test_data:
    if test_data[a].count() < 1488:
        test_data[a].fillna(test_data[a].mean(), inplace=True)
```

To accomplish this, I wrote for loops that iterated over each column in both the training and testing data frames. I used an if statement to find columns that had null values, then used the **.fillna()** and **.mean()** functions to replace all the null values with the mean from the column. After this, I dropped the **Name** and **Sector** columns from each table, making the data frames in a format acceptable for classification. Next, I found the count of each class in the training data in order to downbalance the data. To accomplish this balancing, I found a random sample of 961 cases with a class of 0. After joining the new values together, I separated the training table into a vector of classes and a table with everything except the class. Finally, I used the **test_train_split** function to get **X_train**, **X_test**, **y_train**, and **y_test** variables to use in my models.

Model Selection: To find the best model for this task, I wanted to try many different models and get a base score for each. The scoring method I used was a combination of F1 score and Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) score from scikit learn. I tested five different models: **K-Nearest Neighbors**, **MLPClassifier**, **XGBClassifier**, **ExtraTreesClassifier**, and **RandomForestClassifier**. I first tested using K-Nearest Neighbors, which gave me a score of around 0.64. MLPClassifier had a score of around 0.62, XGBClassifier had a score of around 0.65 as did ExtraTreesClassifier. RandomForestClassifier gave me the highest score, about 0.68, so I decided to use this model going forward.

Hyperparameter setting and tuning: After determining I was going to use RandomForestClassifier for my model, I started working on hypertuning the parameters. To do this, I used a grid search cross validation, testing the attributes shown below.

```
params = {
    'n_estimators': [100, 200, 500],
    'criterion': ['gini', 'entropy'],
    'min_samples_split': [1,2,4,5],
    'min_samples_leaf': [1,2,4,5],
    'max_leaf_nodes': [4,10,20,50,None]
}
```

I used roc_auc as the scoring method to determine which parameters would be best, and tested this over 3 folds. Upon the completion of the grid search, I found that the parameters listed below would provide the best model

```
hyper_rfc = RandomForestClassifier(
    n_jobs = -1,
    criterion = 'gini',
    max_leaf_nodes = None,
    min_samples_leaf = 4,
    min_samples_split = 2,
    n_estimators = 100
)
```

DS 310 – Project 2

Using these values, I outputted the data to a .csv file as describe in the sample code for this project, and received an F1 score of 0.51412, which was better than my original submission of 0.48467

Lessons Learnt: By working on this project, I was able to learn a lot about machine learning. While class, labs, and homework assignments have been able to help me gain a baseline knowledge of machine learning and classification methods, working on a project without many guidelines or restrictions was very helpful in my learning. The main thing I learned was that testing different models before hyperparameter tuning can be very useful in determine what the best model to use going forward is.