Sezer Bakay
Udacity
Machine Learning Nano Degree
1 July 2018

Predicting Stock Yield

Shares of companies provide partnership for the company's profits to its holder. For a shareholder, eventual aim is to yield a targeted return than its investment value. In 2017, 77.6 Trillion USD worth of shares were traded worldwide (The World Bank, World Federation of Exchanges Database). Investors, individuals, companies buy shares of publicly held companies with the expectation of increase in its worth, earning dividends and sell them with a expectation of decrease in its worth or to monetise its value. One profits if the value of the stock increases and exceeds its cost or make loss if not so. While it is easy to define the outcome of situations, the main challenge is to make right trading decisions which satisfy the targeted returns. In order to make profit from stock trading, investors have to give the right trading decisions.

“Big money is made in the stock market by being on the right side of the major moves. The idea is to get in harmony with the market. It's suicidal to fight trends. They have a higher probability of continuing than not.”

-Martin Zweig

In order to understand the environment of the stock trading, it would be beneficial to inspect some core elements of a stock and the nature of a stock trading. According to Lioudis in his article published at Investopedia (2017), the term “stock” refers to a general meaning which describes an ownership certificate of a company and “share” refers to the ownership certificate of a specific company. In the paragraphs of this project, we refer to the same meaning when using either term “share” or “stock”.

Shares are divided into two categories. The main difference between them is, while one grants voting right at the decisions which are taken in company’s directing board and the other one doesn’t. Our focus and scope in this project is the second category which is traded for the purpose of earning profit due to the volatility of its price.

Stocks are traded at stock markets such as NYSE (New York Stock Exchange), NASDAQ, FWB (Frankfurt Stock Exchange), BIST (Istanbul Stock Exchange). These trades are handled during the trading hours of the respective market via official agents. A stock will have 5 main data for that particular trading day. An opening price (Open) which corresponds to the price of the stock at the very beginning of the trading day. High and Low prices which correspond to highest and lowest price values during the trading day. A closing price (Close) which refers to the latest price at the end of the day and Volume data which indicates the traded amount during the particular trading day.

In our project, we will try to build a trading algorithm which decides its actions based on the information that is derived from the historical data of those 5 main fields. Due that this algorithm is an initial prototype, we will work on a single stock. Turkish Airlines (THYAO) has one of the highest trading volume in BIST. Due to its high volume, it is robust against manipulative actions and high price variances during a short (in days) period. In order to build our predicting model, we are going to benefit from the historical price and volume data of THYAO from May 2013 to May 2018.

Investors give their trading decisions based on the information that they have gathered related to the stock. Although this information might be sourced from various channels, the goal is same. The goal is to reward their investment with a targeted profit. Information to be used for analysis of the stock may be sourced from technical analysis calculations, fundamental analysis of the company or industrial/macro economic indicators and reports.

In this project, we are going to build an algorithm which benefits from machine learning techniques and use technical analysis data of the stock. Our aim is to generate a higher return than the natural performance of the stock. The algorithm will strive for buying/holding/selling actions that maximize the profit. The problem to be solved is a classification problem. We are going to try to predict if the stock will increase or not. Therefore it is a binary classification problem. In order to achieve it, we will train supervised classification models such as random forrest classifier, support vector machines, adaptive boosting classifier. These classification models will be trained with features which will be described further on.

Our initial step will be to calculate the required features to be used during training a supervised classification algorithm which will be used for making predictions. By using the 5 main data that described above, we are going to calculate 18 additional technical analysis equations (MACD signals, weighted moving average, relative strength index etc.). Besides the features, we are going to calculate our target variable in order to train the supervised classification model. The target variable is a binary classification which represents if the stock's price is going to go above 3% higher than its current price during a 10 days window. After preparing our dataset, next step will be to pre-process the data to prepare it for training our classification model. During pre-processing, we are going to scale our data and look for reducing the dimensionality of the dataset in order to increase the performance of our classification model. When we have the clean, scaled, dimension reduced dataset, we will continue with training our classification model. After achieving a satisfying model, we will build a trading algorithm which uses the predictions made by the trained model.

During the development, we will have 4 primary metrics to evaluate our performance. Precision and recall will be used to evaluate the classification model and the profit margin will be used to measure the performance of the eventual trading algorithm. Instead of comparing both precision and recall, we are going to use f0.5 score which gives a combined score of precision and recall with a double weight on precision.

Precision:

$$Precision = \frac{True\ Positivies}{True\ Positives + False\ Positives}$$

Precision is used to evaluate the classification model's positive labeling performance. While it is common to use accuracy as a performance metric at classification problems, precision

fits better to our case. In our project, we are aiming for predicting price increases which will be used as a buy signal during trading. While a false negative (the price will increase 3% but we predicted as it won't) won't cause a loss in our budget, a false positive (the price will decrease or won't increase more than 3% but we predicted reverse) may result in losing money due to wrong buying action. Therefore the ratio of True Positives matters more than the accuracy.

Recall:

Recall gives us the accuracy of the samples which are labeled as positive. The reason for taking recall into consideration is that precision itself doesn't always give the desired outcome. If the predictive model predicts most of the samples as negative, yet it may have a considerably high precision score with few positive labeled samples. Therefore additional to precision, we should also seek for the maximum recall score.

$$Recall = \frac{True\ Positivies}{True\ Positives + False\ Negatives}$$

F Beta Score (Beta: 0.5):

F Score combines both the precision and recall. Below the equation of F Beta Score is shown. In our algorithm, our first priority is precision. Therefore, we are going to assign beta as 0.5 .

$$FBeta\ Score = (1 + B^2)\frac{Precision\ x\ Recall}{(B^2\ x\ Precision) + Recall}$$

Profit Margin:

$$Profit\ Margin = \frac{Current\ Assets\ -\ Starting\ Asset}{Starting\ Asset}\ x\ 100$$

Profit margin will be used to measure the performance of our trading algorithm. At the end of the trading simulation, we will compare the profit margin of the algorithm against the margin of the natural performance of the stock (The situation when the stock is bought on the first day and sold on the last day).

Dataset

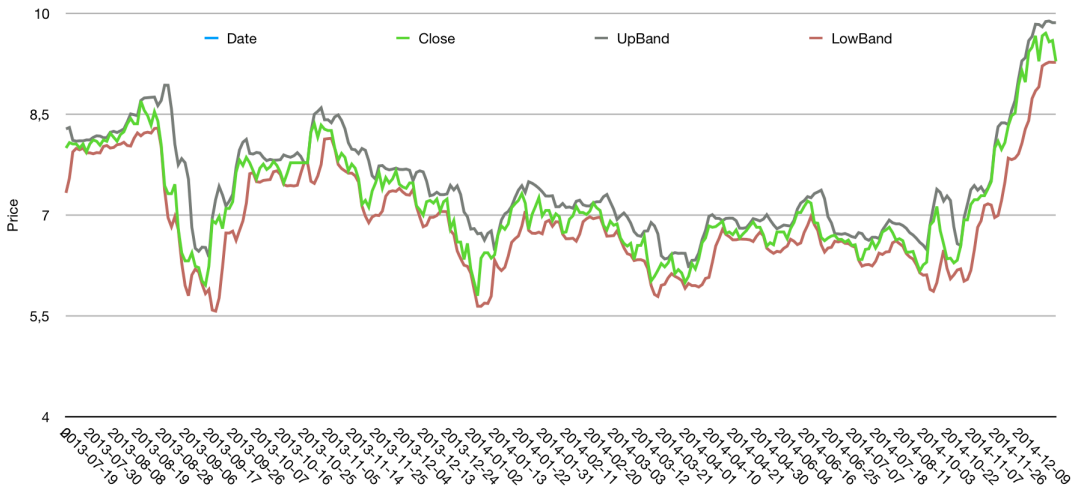
In this project we are going to use a dataset which belongs to THYAO (Turkish Airlines) that is traded in BIST (Istanbul Stock Exchange). Our dataset is a time series data from May 2013 to May 2018 and contain 1185 samples. Original data, that includes “Open”, “Close”, “High”, “Low” and “Volume”, is imported from Yahoo Finance. Additional 18 features of technical analysis values, which are listed below, are calculated by using TA-Lib library (http://ta-lib.org/).

Feature	Definition	Source
Open	Opening price of the day	Yahoo Finance
High	Highest price in the day	Yahoo Finance
Low	Lowest price in the day	Yahoo Finance
Close	Closing price of the day	Yahoo Finance
Volume	Volume of the trade during the day	Yahoo Finance
APO	Absolute Price Oscillator	Calculated by using TA-Lib Library
AROONOSC	Aroon Oscillator	Calculated by using TA-Lib Library
MACD	Moving Average Convergence/ Divergence	Calculated by using TA-Lib Library
MACD Sig	MACD Signal	Calculated by using TA-Lib Library
MACD Hist	MACD Histogram	Calculated by using TA-Lib Library
Momentum	Momentum	Calculated by using TA-Lib Library
RSI	Relative Strength Index	Calculated by using TA-Lib Library
SLowK	Stochastic Slow K	Calculated by using TA-Lib Library
SLowD	Stochastic Slow D	Calculated by using TA-Lib Library
Williams	Wlliams’ %R	Calculated by using TA-Lib Library
UpBand	Bollinger Upper Band	Calculated by using TA-Lib Library
MidBand	Bollinger Middle Band	Calculated by using TA-Lib Library
LowBand	Bollinger Lower Band	Calculated by using TA-Lib Library
P-SAR	Parabolic SAR	Calculated by using TA-Lib Library
WMA	Weighted Moving Average	Calculated by using TA-Lib Library
Chaikin	Chaikin A/D Oscillator	Calculated by using TA-Lib Library
MA28	28 days Moving Average	Calculated by using TA-Lib Library

Feature	Definition	Source
Hilbert	Hilbert Transform - Trend vs Cycle Mode	Calculated by using TA-Lib Library

The target variable is calculated according to the price performance of the stock in a 10 days window. Target variable is a binary value that classifies if the closing price of the stock has exceeded 3% of today's price. The reason behind setting the threshold as 3, besides being a satisfactory return, is balancing the count of positive and negative classes in order to prevent imbalanced class situation. After excluding the rows with n/a values, the dataset has 1185 samples.

The graph below shows the price from July 2013 to December 2014 with its Upper and Lower Bollinger Bands. According to the technical analysis approach, it is expected for the price to change its trend when it touches the boundaries of Upper or Lower Bollinger Bands. Like the Bollinger bands, other features also aim to give prediction insight for the movement of the stock. But rather than predicting the future in a deterministic approach, the algorithm will strive for extracting information from all those features by using machine learning techniques.



After calculating all the features and target variable, our data will be as below.

1	Open	High	Low	Close	Volume	APO	AROONOSC	MACD	MACD Sig	MACD Hist	Momentum	RSI
10	8.04	8.12	7.94	8.04	12399697.0	0.40168320	85.7142857142	0.22612850	0.222561298	0.0035672078	0.039999999995	56.880734
11	8.12	8.28	8.06	8.12	21237621.0	0.36025641	78.5714285714	0.22159976	0.222368992	-0.000769226	0.039999999995	58.878429
12	8.1	8.2	8.08	8.1	11012148.0	0.31256974	71.4285714285	0.21393081	0.220681356	-0.006750544	0.039999999995	58.153066
13	8.22	8.26	8.12	8.22	11629834.0	0.25423641	7.14285714285	0.21505706	0.219556497	-0.004499434	0.160000000001	61.238626
14	8.16	8.26	8.1	8.16	10810709.0	0.21628205	7.14285714285	0.20870233	0.217385665	-0.008683328	0.160000000001	58.900106
15	8.1	8.24	8.1	8.1	10576263.0	0.18423076	7.14285714285	0.19655886	0.213220305	-0.016661441	0.039999999995	56.573549
16	8.2	8.22	8.1	8.2	7212928.0	0.18012820	7.14285714285	0.19278196	0.209132637	-0.016350670	0.259999999995	59.448547
17	8.24	8.28	8.16	8.24	8681975.0	0.16666666	7.14285714285	0.19081679	0.205469469	-0.014652671	0.179999999995	60.572950
18	8.36	8.44	8.34	8.36	10718336.0	0.17243589	78.5714285714	0.19667523	0.203710622	-0.007035387	0.240000000001	63.814524
19	8.44	8.44	8.32	8.44	3655023.0	0.16871794	85.7142857142	0.20540563	0.204049624	0.0013560069	0.339999999995	65.831413
20	8.36	8.36	8.36	8.36	0.0	0.14948717	85.7142857142	0.20352311	0.203944322	-0.000421208	0.320000000001	62.103639

RSI	...	UpBand	MidBand	LowBand	P-SAR	WMA	Chaikin	MA28	Hilbert	Target
56.880734	...	8.17747509713	8.05199999999	7.9265249028	6.34783	7.80878359	-10265778.6170	7.603261071	0	1
58.878429	...	8.15299230723	8.08799999999	8.0230076927	6.34783	7.84497806	-12698930.2314	7.642329285	0	1
58.153066	...	8.15478775383	8.09599999999	8.0372122461	6.34783	7.87741750	-14875772.4162	7.678198928	0	1
61.238626	...	8.23302991070	8.11599999999	7.9989700892	6.34783	7.91502759	-12828069.3631	7.726428571	0	0
58.900106	...	8.24826637102	8.12799999999	8.0077336289	6.34783	7.94619916	-11684127.2970	7.780590000	0	1
56.573549	...	8.23121403400	8.13999999999	8.0487859659	6.34783	7.97121083	-13519132.1975	7.841304285	0	1
59.448547	...	8.25527738916	8.15599999999	8.0567226108	6.34783	8.00002804	-11510788.9722	7.885714285	0	1
60.572950	...	8.28327738916	8.18399999999	8.0847226108	6.34783	8.02807853	-8721943.12497	7.923571428	0	1
63.814524	...	8.38653939383	8.21199999999	8.0374606061	6.34783	8.05991397	-8834375.50338	7.948571428	0	1
65.831413	...	8.50746607275	8.26799999999	8.0285339272	6.34783	8.09389247	-6914283.65787	7.982142857	0	1
62.103639	...	8.49527121840	8.31999999999	8.1447287815	6.34783	8.11999999	-5500220.29763	8.009285714	0	1

In here primary statistical information related to features are shown:

	Open	High	Low	Close	Volume	APO	AROONOSC	MACD	MACD Sig	MACD Hist	Momentum	RSI
count	1185.000000	1185.000000	1185.000000	1185.000000	1.185000e+03	1185.000000	1185.000000	1185.000000	1185.000000	1185.000000	1185.000000	1185.000000
mean	8.427181	8.524481	8.319603	8.416101	4.485110e+07	0.056168	8.245931	0.059020	0.060634	-0.001614	0.074979	52.114874
std	3.267361	3.330072	3.193307	3.264557	3.846531e+07	0.348633	65.379984	0.258965	0.245620	0.068993	0.659291	12.948941
min	4.650000	4.780000	4.540000	4.630000	0.000000e+00	-1.113141	-100.000000	-0.589734	-0.504087	-0.258367	-2.449999	21.739720
%25	6.430000	6.500000	6.330000	6.410000	1.485816e+07	-0.133141	-57.142857	-0.094217	-0.089988	-0.034435	-0.270000	42.689733
%50	7.660000	7.720000	7.550000	7.630000	3.483877e+07	0.028141	14.285714	0.024034	0.021913	0.003261	0.040000	51.484957
%75	9.080000	9.140000	9.010000	9.070000	6.469398e+07	0.206474	71.428571	0.188401	0.175030	0.037363	0.420000	61.027695
max	19.870001	20.020000	19.469999	19.690001	2.879165e+08	1.432051	100.000000	1.109243	1.030194	0.192712	2.720000	85.880930

	SLowk	Slowd	Williams	UpBand	MidBand	LowBand	P-SAR	WMA	Chaikin	MA28	Hilbert	Target
count	1185.000000	1185.000000	1185.000000	1185.000000	1185.000000	1185.000000	1185.000000	1185.000000	1.185000e+03	1185.000000	1185.000000	1185.000000
mean	52.188224	52.224559	-45.595376	8.691368	8.401803	8.112237	7.383541	8.341776	-7.167822e+06	8.310584	0.820253	0.460759
std	26.733261	24.800178	31.711405	3.388848	3.243184	3.106427	2.263290	3.152359	3.696632e+07	3.111857	0.384139	0.498668
min	1.465873	2.898551	-100.000000	4.848990	4.754000	4.490106	4.540000	4.918624	-1.571597e+08	4.898571	0.000000	0.000000
%25	27.619048	29.260037	-75.000000	6.721867	6.446000	6.111957	5.600000	6.568731	-2.122138e+07	6.572143	1.000000	0.000000
%50	54.603581	54.031495	-40.677966	7.858833	7.652000	7.402627	6.347830	7.571097	-5.617987e+06	7.548214	1.000000	0.000000
%75	75.948780	73.294247	-16.577549	9.350590	9.030000	8.761366	10.350000	9.031570	8.082111e+06	9.030000	1.000000	1.000000
max	100.000000	99.176955	-0.000000	20.147953	19.518000	19.249850	10.350000	18.946839	1.575571e+08	18.873928	1.000000	1.000000

If we examine the means and standard deviations of the features, it is observed that our features are not scaled to each other. While some features have mean values around zero, there are features which have much higher scalar values for their means. With this output, we came to a conclusion that the dataset needs to be scaled before using it in a machine learning algorithm.

Algorithms and Techniques

The project is accomplished in three stages. These are preprocessing the data, machine learning and trading. Algorithms and techniques used throughout the project will be describes under these headings.

Throughout the project, the algorithm is coded in Python 3. While preparing the features of the dataset, we have used TA-Lib library to calculate the technical analysis values. For building up the data structures, appending values, splitting and doing other structural action, we have used pandas library. The dataset is constructed as a Pandas data frame. Pandas is an open source library to build high performance and easy to use data structures (https://pandas.pydata.org/).

We are going to use Sklearn library for scaling the data, dimensionality reduction and supervised learning (classification). Sklearn is an open source library built on Numpy, Scipy and Matplotlib libraries. It is used for data analysis & machine learning and data mining functions (http://scikit-learn.org/stable/index.html). The logic behind Sklearn functions is similar to each other. First of all, an object is created for a specific action. Than, the empty object is fit to the reference data (in our project, it is the training data). After the fit step, for transforming operations such as scaling, the data which is wanted to apply the transformation is transformed according to the object. If the operation is a prediction action, predict function is used to predict an outcome by using the trained object.

Preprocessing The Data:

In order to have a well performing classification model, the dataset has to be preprocessed before applying machine learning techniques.

First step will be to split our dataset into training and testing sets. Due to being a time series, it is vital to split the set in a proper way. As a result of being a time series, random sampling will cause the trained model to be overfit to the training dataset. Therefore, we are going split the first 80% of our samples as the training set and the remaining 20% (which are the samples that belong most recent dates). Additional to that, “Open”, “High” and “Low” features will be excluded because “Close” feature will be enough for referring the price information.

In order to scale the data, StandardScaler function from Scikit-Learn library will be used. Scikit Learn describes the function of StandardScaler as, it standardizes features by removing the mean and scaling to unit variance (http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html). Features are scaled and centered independently from each other.

In the dataset we have 18 features that belong to technical analysis and also “Close” and “Volume” features. The problem is, we don’t know the relativity and importance of those features. In order to reduce the dimensionality of our machine learning model and benefit

from the most relevant features, Principal Component Analysis (PCA) technique will be used. Victor Powell explained PCA as a technique to emphasize and extract strong patterns in a dataset (http://setosa.io/ev/principal-component-analysis/). The technique brings out the orthogonal components that have maximum amount of variance. In our project, PCA function from Sklearn library will be used. A scaler object will be fit to the training set and than the whole dataset will be transformed by using the scaler object.

Machine Learning:

After splitting the data and applying preprocessing steps, a machine learning model will be used to handle the classification. In order to accomplish the classification, supervised machine learning techniques will be used. In supervised ML techniques, the model is trained with features and the respective target variable. The purpose is to predict the target variable of a new sample by using the pre-trained model.

First action will be to create a classifier object to be trained. Related parameters about the classifier algorithm are defined in this step. After creating the object, it will be trained with the training data by using fit function. At the end of the fit action, we will have a trained classifier object which we can use for prediction. For evaluating our alternative models, we are going to predict our test data and compare the predicted classes to the actual targeted labels.

In the project, several supervised ML algorithm will be compared to each other. For this comparison, models will be trained with their default parameters and roughly defined heuristic parameters. They will be compared to each other based on their f0.5 scores. Hyper-parameters of the model with highest score will be tuned in order to achieve the highest metrics. Besides the hyper parameters, metrics with different PCA components will be compared in order to find the optimum component number. The scores will be evaluated by using the test dataset.

K-Nearest Neighbors (K-NN): Samples are classified according to a majority voting among their neighbor samples. While it is a very simple algorithm, the algorithm doesn't learn from the training set. Classification is done according to the distances of a sample's neighbors and their classes. This labels K-NN as a lazy-learner. Besides that, the algorithm is not robust against meshed/noisy data. For the classifier object, default parameters of Sklearn Library will be used. These are, Number of neighbors: 5, weights: uniform, algorithm: auto, leaf_size: 30 and euclidean distance as the power parameter for the Minkowski metric. The reason for including K-NN is building a simple and base classifier to compare to other classifiers as a benchmark.

Random Forrest Classifier (RFC): RFC is an ensemble method which benefits from multiple decision tree classifiers. A decision tree classifier classifies its samples according to the calculated information entropy. By using various random decision trees derived from the training dataset, the outcome will be a more generalized model. Main strength of Random Forest Classifier (RFC) is being able to classify large datasets with high accuracy (Michael Walker, 2013, Random Forests Algorithm). Besides that, the structure of RFC is

simple and easy to understand. It performs well if you take precautions agains overfitting. Just like Decision Tree Classifier algorithm, Random Forest Classifier is also sensitive to overfitting. If you let the algorithm to reach down to the deepest leaf nodes it can, it will result with an overfitting. For the classifier object, default parameters of Sklearn Library will be used. These are, number of estimators: 10, criterion: gini, max_features: auto, max_depth: None, minimum number of samples to split: 2, minimum number of samples to be a leaf node: 1. Ensemble methods are easy to implement and give satisfying performances. Due to that we have a large dataset which can't be separated linearly, random forrest classifier might give good results for this project.

Support Vector Classifier (*SVC)*: Support vector machines are supervised learning models with its algorithms for classification and regression problems. For the classification, the support vector machine handles the separation by the distances (support vectors) of samples against the class boundary. The main advantage of support vector machines is the effectiveness at high dimensional spaces. Disadvantages are, it should be avoided for over-fitting if features are greater than the samples (which is not relevant for our case) and SVM don't provide class probabilities directly. (http://scikit-learn.org/stable/modules/svm.html). For the classifier object, default parameters of Sklearn Library will be used. These are, penalty parameter C: 1, kernel: radial basis function (rbf), gamma: auto (1/number of features). Our training dataset has 20 features. We know that support vector machines give good results at high dimensional spaces. Therefore it might be beneficial to use SVC in our classification problem.

Adaptive Boosting (**Ada Boost) Classifier:** Ada Boost is an ensemble method which takes its power from ensembling multiple weak learners. As a result, it achieves a higher learning performance. The main strength of AdaBoost algorithm is that it corrects its mistakes (Ayudub, 2014, Pros and Cons of Classifiers). Also the accuracy increases as the dataset grows without overfitting. If we clean our dataset from noisy and outlier data, AdaBoost will have more accurate outcomes. The weakness of AdaBoost is being sensitive to Noisy Data and outliers (2014, http://www.nickgillian.com/wiki/pmwiki.php/GRT/AdaBoost). For the classifier object, default parameters of Sklearn Library will be used. These are: base estimator: None, number of estimator: 50, learning rate: 1.0, algorithm: SAMME.R. The reason for including Adaboost Classifier in our project is similar to random forrest classifier. Besides that, using weak learners might boost the performance at such datasets like ours without overfitting. Our aim is to build a model that generalize enough to provide a satisfying precision and recall.

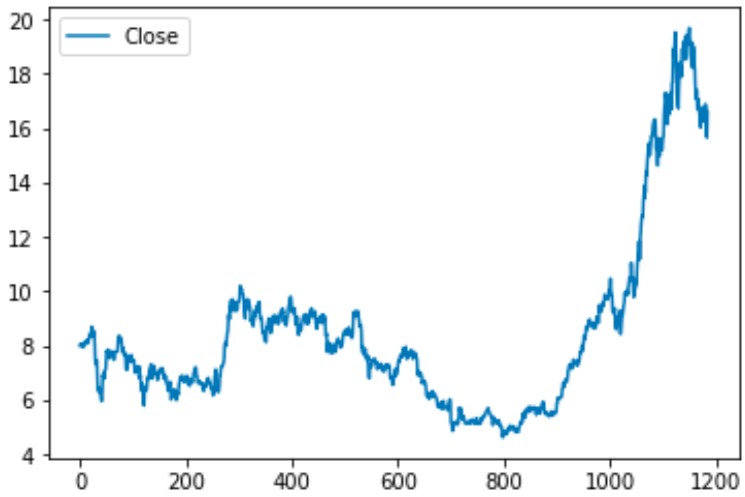
Trading:

The algorithm will give its buy decision based on the predicted class, which refers to if the stock will exceed 3% of its current price in 10 days. In the beginning of the trading day, if the agent doesn't hold any stock and the target variable is 1, it will buy a stock. In the following days, when the price increases 3%, the agent will sell its holdings. If the price decreases more than 3%, stop-loss order will execute and the agent will sell its stock.

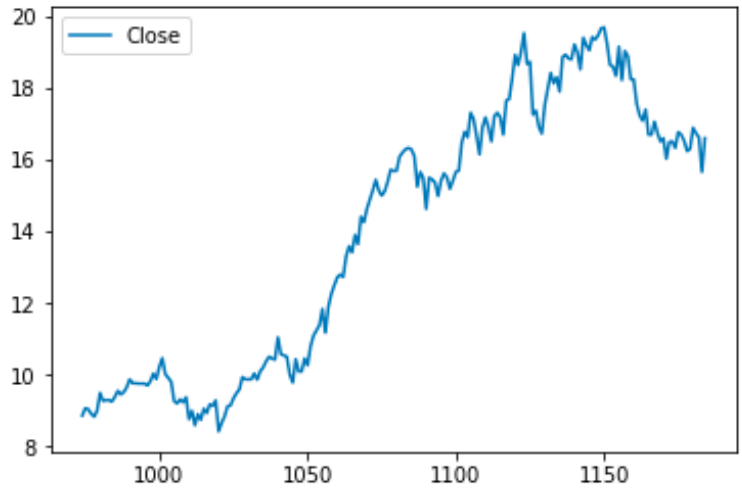
Benchmark

The benchmark to evaluate the performance of our algorithm is the natural performance of the stock. Natural performance refers to the value change of the stock from first day until the last trading. In other words we may explain it as, if the agent buys the stock in the first day and sell it on the last day, the change in the price will give the stock's natural performance. For evaluation, we will compare the profit margin calculated from natural performance. Trading costs & commission costs are not taken into account.

Price performance of THYAO for the entire dataset:



Below is the price performance of the test dataset which we aim to exceed. The price changed from 8,68 TL to 15,95 TL in 7 months. The profit margin is 83%.



After running the trading algorithm with the test dataset, we are going to calculate the earned profit in order to compare our algorithm against the natural performance of the stock.

Data Preprocessing

As described in the algorithms and techniques section, the initial preprocessing step is splitting the dataset into training and testing sets with ratios respectively 80/20. It is crucial to split first and than apply the preprocessing techniques. Our dataset is a time series, therefore samples to split can't be picked by randomly. Otherwise, test samples would be easily predictable and inhibit us if the model has overfitted or not. We have used indexing to slice the dataset as shown below.

```
85 # Split the dataset into training and testing sets. The ratio is 80/20
86 train_end = int(np.floor(r*0.8))
87 X_train = data.iloc[0:train_end, 3:c]
88 y_train = data.iloc[0:train_end, c]
89 X_test = data.iloc[train_end:r, 3:c]
90 y_test = data.iloc[train_end:r, c]
```

After splitting the dataset, it will be scaled in order to normalize means and standard deviations of the features. Scaling will done by using the StandardScaler function from Sklearn Library. For scaling the data, a standard scaler object is fit to the training dataset and than the whole dataset will be transformed by using the scaler object.

```
93 # Preprocess and standardize the data
94 from sklearn.preprocessing import StandardScaler
95 scaler = StandardScaler()
96 scaler.fit(X_train)
97 X_train = scaler.transform(X_train)
98 X_test = scaler.transform(X_test)
```

After applying scaling, all of our features will be scaled and this scaled values will be used as an input to our classification model.

Final step of preprocessing data is the feature transformation and dimensionality reduction.. Our training dataset has 20 features to classify the target variable. Due to having a high dimensional space, dimensionality reduction techniques may improve the performance of our machine learning model. The difference of applying feature transformation will be observed from the performance metrics of supervised learning models.

The technique we apply for feature transformation is Principal Component Analysis (PCA). As described in the algorithms section, this technique brings out the orthogonal components that have maximum amount of variance. First of all, we are going to execute the PCA function for maximum number of components which is equal to the number of

features in order to observe the explained variances of the components. Sum of the explained variance always equals to 100%. After that, explained variance of the components will be sorted and the number of components will be decided upon reaching to a limit where the metric of the model gives highest score.

After deciding the number of PCA components, the value of n_components will be updated to the decided number while creating the initial PCA model.

```
# Dimensionality reduction
from sklearn.decomposition import PCA
pca = PCA(n_components=None)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
components = pca.components_
```

Below, explained variances of all the components are listed:

Explained Variance of Components:

No	Explained Variance
1	0.385706
2	0.299014
3	0.105357
4	0.0539957
5	0.0470135
6	0.0327043
7	0.0221894
8	0.0163704
9	0.01342
10	0.0104182

No	Explained Variance
11	0.00619765
12	0.00352906
13	0.00223234
14	0.000971981
15	0.000753562
16	0.000106617
17	1.69241e-05
18	3.80913e-06
19	4.08096e-33
20	1.6909e-33

Supervised Learning

In order to be able to predict the class of a sample, we need an predictive classification model. In the algorithms section, we have shortlisted four supervised learning algorithms for this purpose. These shortlisted supervised learning models will be trained under some predefined scenarios in order to compare them to each other and pick the best one.

Scenarios are;

PCA: Not Applied, Max Components
Parameters: Default, 3 or more different hyper parameter sets (heuristic)

After picking up the best performing algorithm (based on f0.5 score), the model's hyper parameters will be tuned in order to reach to the highest score. During the tuning process, rather than Grid Search technique, cases will be executed and results will be noted separately.

All the supervised algorithms will be applied by using Sklearn Library. Below is a sample code for training the train data and predicting the test data. First step is to create a classifier model with defined parameters. Second step will be to train the classifier object with training and and its corresponding target variables (binary class). As a last step, we are going to predict the target variables of test data by using the trained classifier object. After the prediction, f0.5 score of the predicted classes will be compared to the actual labels of the test data in order to evaluate the performance of the classifier model.

```
# KNN
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

In the tables below, f0.5 scores of the four supervised learning models with the parameters written on the table are listed.

F0.5 Scores of K Nearest Neighbor (pca not applied):

K-NN	metric: Minkowski, distance: Euclidean
n_neighbors: 5	0,286
n_neighbors: 2	0,218
n_neighbors: 10	0,273

F0.5 Scores of K Nearest Neighbor (pca applied w/ max components):

K-NN	metric: Minkowski, distance: Euclidean
n_neighbors: 5	0,286
n_neighbors: 2	0,218
n_neighbors: 10	0,273

F0.5 Scores of RFC (pca not applied):

RFC	criterion: Gini, max_depth: None	criterion: Gini, max_depth: 10
n_estimators: 10	0,303	0,347
n_estimators: 50	0,297	0,264
n_estimators: 100	0,240	0,209
n_estimators: 200	0,301	0,297

F0.5 Scores of RFC (pca applied w/ max components):

RFC	criterion: Gini, max_depth: None	criterion: Gini, max_depth: 10
n_estimators: 10	0,555	0,523
n_estimators: 50	0,509	0,568
n_estimators: 100	0,493	0,492
n_estimators: 200	0,405	0,573

F0.5 Scores of SVC (pca not applied):

SVC	kernel: rbf, gamma: 0,04 (auto)	kernel: rbf, gamma: 0,2
C: 1	0,558	0,251
C: 0.1	0,000	0,000
C: 5	0,615	0,410

F0.5 Scores of SVC (pca applied w/ max components):

SVC	kernel: rbf, gamma: 0,04 (auto)	kernel: rbf, gamma: 0,2
C: 1	0,558	0,251
C: 0.1	0,000	0,000
C: 5	0,615	0,410

F0.5 Scores of AdaBoost (pca not applied):

ADA	learning_rate: 1.0, algorithm: SAMME.R	learning_rate: 0.1, algorithm: SAMME.R	learning_rate: 0.05, algorithm: SAMME.R
n_estimators: 50	0,406	0,184	0,026
n_estimators: 250	0,470	0,261	0,209
n_estimators: 500	0,434	0,262	0,287

F0.5 Scores of AdaBoost (pca applied w/ max components):

ADA	learning_rate: 1.0, algorithm: SAMME.R	learning_rate: 0.1, algorithm: SAMME.R	learning_rate: 0.05, algorithm: SAMME.R
n_estimators: 50	0,621	0,524	0,478
n_estimators: 250	0,433	0,588	0,583
n_estimators: 500	0,398	0,477	0,524

Result:

Supervised algorithms with different parameters are trained and their f0.5 scores are calculated. During the first implementations of this project, performances were compared only based on the precision. But this has resulted in a problem that under some conditions, there were biases for predicting most of the samples as negative yet scoring a high precision with few positive labeled samples. In order overcome this issue, recall is also included into the metrics. As a result, f0.5 has become the main metric to compare the models to each other.

If we check the results, Support Vector Classifier and AdaBoost Classifier both with PCA applied got the highest scores. According to the results, AdaBoost has a broader success at different parameters settings and besides that it does feature selection naturally (while SVM takes all the feature space). Due to that, we are going to continue with AdaBoost Classifier for the rest of the project. The next challenge will be to increase the f0.5 score of the model by tuning the hyper parameters of the classifier model.

Tuning Hyper Parameters

In order to get the highest f beta score and eventually highest profit at trading algorithm, we are going to compare results of different parameter sets for the AdaBoost classifier. At the previous section, we have observed that AdaBoost performs better when PCA is applied to our dataset. After picking the parameter set with highest score, we will calculate scores for different component numbers.

First we are going to train and test the model with different parameters which are written in the table below. The parameters that are tuned are number of estimators (n_estimators) and learning rate. Adaboost Classifier is an ensemble method. It iterates weak learners which use a simple base estimator (default decision tree), update the weights and achieves a high classification accuracy at the end of the iterations. Number of estimators sets the maximum number of this estimators/iterations. The reason for saying maximum is, the model stops iterating if it reaches to perfect fit. The amount of adjusting/updating the weights of these weak learners is set by the parameter “learning rate”.

Learning algorithm is also another parameters which can be tuned but it has been observed that the alternate algorithm “SAMME” had resulted in considerably low scores than “SAMME.R”. Therefore the scores of SAMME are not shared in the table.

PCA applied w/ max components, algorithm: SAMME.R

ADA	learning_rate: 1.0	learning_rate: 0.1	learning_rate: 0.05	learning_rate: 0.03
n_estimators: 50	0,621	0,524	0,478	0,465
n_estimators: 100	0,632	0,611	0,536	0,432
n_estimators: 150	0,554	0,591	0,583	0,496
n_estimators: 200	0,462	0,603	0,611	0,585
n_estimators: 250	0,433	0,588	0,584	0,596
n_estimators: 300	0,427	0,481	0,597	0,597

When we check the results from the table, it is seen that when the learning rate is decreased, number of estimators has to be increased to maintain accuracy. Among the combinations above, highest score (0,632) is achieved with 100 estimators and learning rate 1. Therefore, we are going to use this parameters as the final model. But still there is one more thing to reconsider apart from the parameters of the model. It is the number of PCA components for the input of our classification model.

Just like the hyper parameters, we are going to evaluate different f0.5 scores for different component numbers. In the table below, different component numbers for number of estimator: 100 and learning rate 1.0 are shown:

ADA	Explained variance	n_estimators: 100, learning_rate: 1.0
n_components: max	100	0,632
n_components: 19	100	0,632
n_components: 15	99,9	0,640
n_components: 12	99,6	0,571
n_components: 10	98,6	0,602

As a result, we have achieved highest score with 15 components. Therefore, our final supervised learning algorithm will be fed with 15 PCA components for this feature space and the parameters will be as below:

AdaBoost Classifier trained with PCA (15 components) applied dataset:
- Number of estimators: 100
- Learning rate: 1.0
- Learning algorithm: SAMME.R
- Base estimator: Decision Tree Classifier

In order to test the robustness of our model, we are going to train and test it with a slightly different dataset. We had used the first 80% of the dataset as the training set. Now we are going to flip it reverse and use the first 20% as test set and remaining 80% as the training data. After training and testing the model with dataset by using the same parameter setting, f0.5 score is resulted as 0,41. 0.41 is a low score than our initial 0,64 f0.5 score. It can be said that the model is not robust to new dataset but if we repeat this action with other parameter sets, it will be seen that there will be similar results and our final model will still have one of the highest score among them. Besides, when we try with a different supervised learning algorithm such as support vector machine, the result is similar*.

New dataset, PCA applied w/ 15 components, algorithm: SAMME.R

ADA	learning_rate: 1.0	learning_rate: 0.1	learning_rate: 0.05
n_estimators: 50	0,364	0,376	0,438
n_estimators: 100	0,406	0,367	0,400
n_estimators: 150	0,415	0,390	0,376

ADA	learning_rate: 1.0	learning_rate: 0.1	learning_rate: 0.05
n_estimators: 200	0,460	0,380	0,373
n_estimators: 250	0,430	0,384	0,398
n_estimators: 300	0,426	0,401	0,383

* If we run the model with a Support Vector Classifier (kernel: rbf, gamma: auto, C: 5), the f0.5 score is: **0,399**

The weakness against a different dataset is definitely an improvement area for the algorithm. Training the model with alternate parameter settings didn't provide a solution. More complex machine learning algorithms or combining two algorithms to improve robustness might be a solution to this situation.

Trading

Final section of the project is building a trading algorithm that gives its decision based on the predicted classes and gained value. In the figure below, logical flow of the trading mechanism is shown. The calculations are also written in Python 3.

For each trading day:
    if the agent doesn't hold stock:
        if class = 1 —> **Buy**
    if the agent holds stock:
        if price is greater 3% than the cost of stock —> **Sell**
        if price is lower 3% than the cost of the stock (stop-loss) —> **Sell**

Our agent's starting budget is 8,68 Lira (worth 1 stock). When we run this simulation for the test data that covers the last 210 samples (210 trading days, circa 7-8 months) of the main dataset, the agent did 57 trading actions. On the last day, the snapshot is shared below:

Starting budget: 8,68
Last day budget: 2,41
Last day holdings: 15,95 worth of stock (16,6 at the close).
Total asset on the last day: 18,36 Lira
Profit Margin: (18,36 - 8,68) / 8,68 = **111%**

Our benchmark to measure the success of our trading algorithm is the natural performance of the stock. The stock will be bought on the first day and it will be sold on the last day. If we calculate the profit margin for this case:

Profit Margin: (15,95 - 8,68) / 8,68 = **83%**

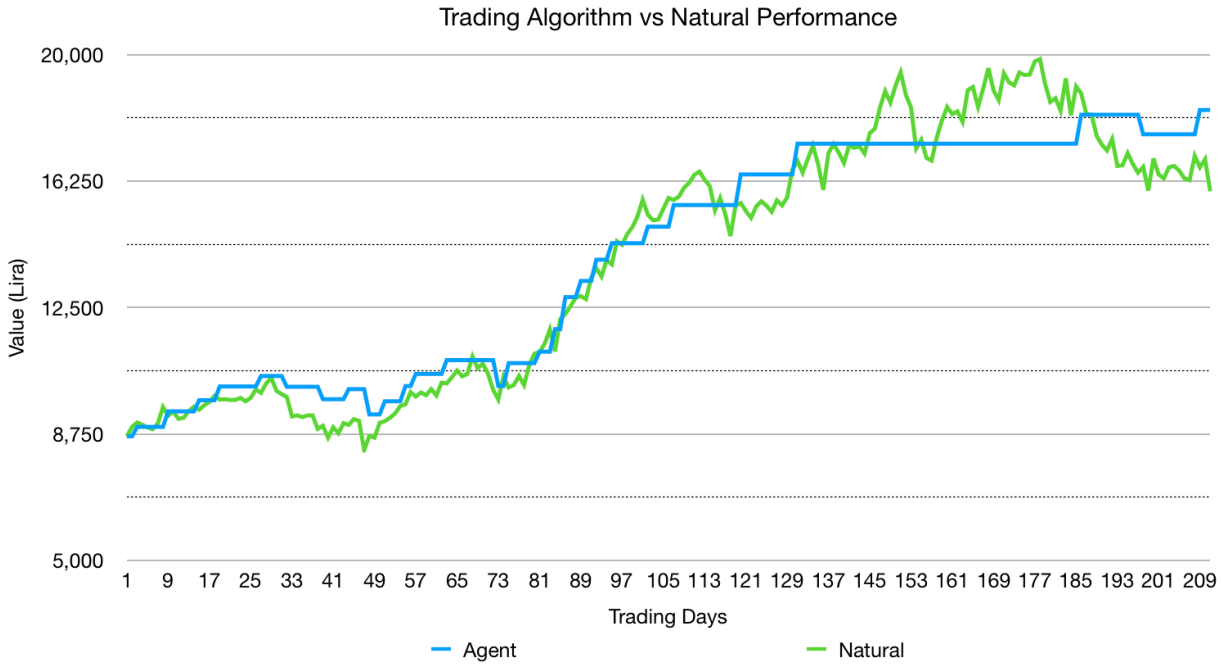
If we run the trading steps for the model which is trained with the new dataset above (last 80% of the dataset), total asset would be 7,93 Lira (starting budget 8,00 Lira) and profit margin would be - 1%. While there is a 1% loss, if we calculate the natural performance for this testing dataset, the asset on the last day would be 6,86 Lira which gives a profit margin of - 15%. While we had a lower f0.5 score, our model still have managed to exceed its natural performance.

Conclusion

In the project, first steps were preparing a dataset to be trained in a supervised learning algorithm. One of the most challenging part was to include as much as technical analysis calculations to our dataset. Thanks to the open sourced TA-Lib library, it became possible to calculate each technical analysis features with a single line of code. By using the TA-Lib library for technical analysis calculations, we have calculated technical analysis values for our historical time series dataset of THYAO stock. By calculating the required features and target variable according to a predefined price increase threshold during a 10 days period, a pandas data frames is constructed to be used at machine learning algorithm.

Next step was to evaluate different supervised classification algorithms. After comparing the f0.5 scores (precision and recall), strengths and weaknesses of algorithms, AdaBoost Classifier is chosen as the supervised learning algorithm to use for predicting the target classes. Following to that, we have evaluated the performance of our AdaBoost classifier with different parameter sets and different feature dimension reduction levels. After deciding the final model, we have executed the trading algorithm which uses the predicted classes of the machine learning model.

By applying machine learning algorithms, we have achieved 28% more profit than the natural behavior of the stock. Satisfying performance increase allows to us say that implementing machine learning to a trading algorithm has resulted in success. In order to observe the performance, graph below shows the total asset values during the trading period.



According to the graph, algorithmic agent and stock itself yield similar returns. But during the periods where price trend changes to downside, agent manages to sustain its value. This is where the positive difference results from.

When we ran our final model with a new dataset, f0.5 score was below the expectation. Different parameters or a different machine learning algorithm gave also similar low scores. This was the second challenging problem in the project and this is a clear improvement area for our machine learning model. Unfortunately, tuning the parameters didn't provide the expected improvement. More complex machine learning algorithms or combining two algorithms might be a solution to this case. XGBoost algorithm is also tried for a better result but its integration with the rest of the algorithm failed.

Besides that, machine learning model can be applied to different company stocks and with a ranking model, portfolio trading can be achieved. Advantage of portfolio trading will be being more robust to negative risks and higher profit return due to trading only with the highest ranking company shares. Commission costs can be included to create a more realistic trading environment which also leads to another improvement area to redesign the trading algorithm for minimum amount of trading actions.