Machine Learning Engineer Nanodegree

Capstone Project

Predicting stock prices in Turkish Stock Market

Görkem Göknar

July 2, 2018

I. Definition

Project Overview

A stock price in a trading day can either increase, decrease according to previous trading day. If a trader is able to predict the direction of stock before making a movement this would yield a significant advantage and increase profit.

Predicting performance of western stock markets (US or EU) is a well-tested area, as these stocks are the ones defining global trend and not big major price fluctuations happen everyday (exception is global crisis periods). Predicting a stock market that depends on foreign investment is not a well worked area as price movements are sometimes unpredictable due to either global exchange rates, or local economic factors.

Turkish stock market is one of the unpredictable areas where any big USD or EURO movement (hence global money flow in country) directly affects stocks.

In this project one of the reliable stocks (which are in BIST30 index) in Turkish Stock Market will be used to predict next day's movement.

Problem Statement

This projects objective is to predict next day price direction of a given stock, given the historic prices of the stock plus foreign exchange rates (due to local currency) data, over a time period using concepts and techniques in technical analysis and machine learning. Direction will be defined as 1 if increase in market value of stock and 0 for a decreased value. For example, if previous trading day close price is 8.70 and todays close price is 8.5, this would be decrease and directionality value (target) would be 0.

Multiple supervised learning classification techniques will be used some of them are namely, Gaussian Naïve Bayes, Multinomial Naïve Bayes, Support Vector Machines (SVM) and also neural network based approach will be tested.

Main stock data (Close, High, Low, Volume) will be get by an API from web and also foreign exchange rate prices will be fed as input. Technical indicators will be calculated for stock and exchange rates. Target would be to predict if base stock closing price will increase or decrease next trading day.

Base stock for the implementation will be AKBNK (Akbank) which is a high volume stock in BIST index, implementation will be systemized for other stocks to see fore testing performance.

Additional Google trend data for keywords “faiz” (interest rate) and “debt” (borç) are included as features as Turkish economy is directly affect by foreign exchanges and any big rumour on interest rate or increasing debt may affect stock.

Twitter sentiment analysis was opted out as it is nearly impossible to find good independent voices giving correct economic reviews.

Metrics

Since target would be to predict increase and decrease this is a classification problem. Also given the 5 years of daily data, we do not have many training data so it would be better to reach this problem as classification.

Model will be evaluated on F-1 score and Accuracy.

Accuracy is defined by following:

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

simply said as ratio of True positives and True Negatives over total number of data.

		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

F-1 score is a metric derived from precision and recall and defined as:

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

Where:

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Precision shows how precise/accurate our model is; out of those predicted positive how many of them are actually positive (correctly classified).

Recall shows how many of the actual positives our model captured through labelling it as a True positive

In this project for a precision and recall calculation for a buy signal (next day increase):

True Positive: Model classifies a buy signal (next day increase) really as a buy signal.

False Positive: Model classifies a sell signal as a buy signal

False Negative: Model classified a buy signal as a sell signal

True Negative: Model classifies a sell signal as a sell signal

We will use one of the base benchmarks with buy and hold strategy where where model always assumes an increase (which corresponds to a buy and hold strategy).

Also K Nearest Neighbourhood score will be used as benchmark to beat.

Model results will be satisfactory if F-1 score and accuracy is greater than 55% which are our base performance metrics (better than buy and hold and K Nearest Neighbours). Model will be tested with fore testing and simulated profit gains will be compared with buy and hold strategy (which is common base for financial gains)

II. Analysis

Data Exploration

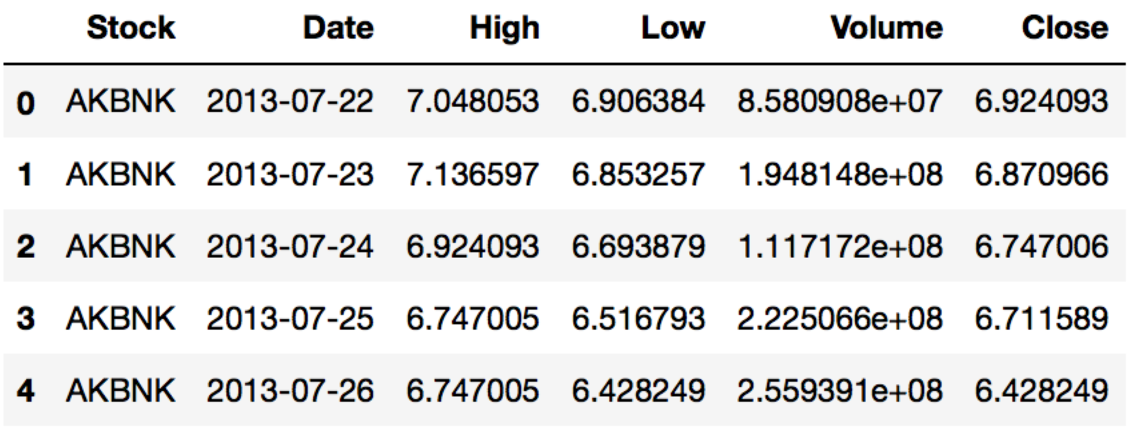
Datasets and Inputs

Data used in this project was historical market data from 22.07.2013 to 22.06.2018. This 5 Year data was found to be more reliable than older data, as also this data shows a multiple ups and downs on stock prices due to fluctuations in Turkish economy (previous 5 year data would mostly show upward trend due to after 2008 crisis).

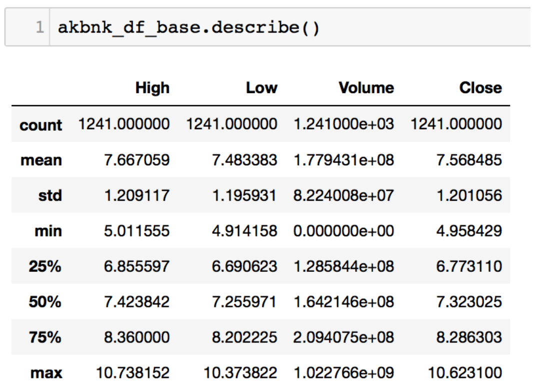
Raw data generator is available in the source codes and dataframe can be generated by following example code:

```
akbnk_df_base = get_stock_from_mynet("AKBNK",stock_range="yillik_5")
```

This would yield data with following columns with 1241 rows (depends on time data was pulled):

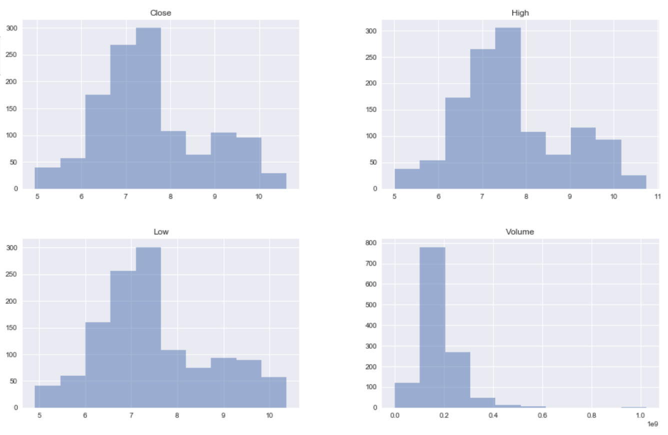


	Stock	Date	High	Low	Volume	Close
0	AKBNK	2013-07-22	7.048053	6.906384	8.580908e+07	6.924093
1	AKBNK	2013-07-23	7.136597	6.853257	1.948148e+08	6.870966
2	AKBNK	2013-07-24	6.924093	6.693879	1.117172e+08	6.747006
3	AKBNK	2013-07-25	6.747005	6.516793	2.225066e+08	6.711589
4	AKBNK	2013-07-26	6.747005	6.428249	2.559391e+08	6.428249



```
akbnk_df_base.describe()
```

	High	Low	Volume	Close
count	1241.000000	1241.000000	1.241000e+03	1241.000000
mean	7.667059	7.483383	1.779431e+08	7.568485
std	1.209117	1.195931	8.224008e+07	1.201056
min	5.011555	4.914158	0.000000e+00	4.958429
25%	6.855597	6.690623	1.285844e+08	6.773110
50%	7.423842	7.255971	1.642146e+08	7.323025
75%	8.360000	8.202225	2.094075e+08	8.286303
max	10.738152	10.373822	1.022766e+09	10.623100



It can be seen that on 5 year date range minimum of stock close price was 4.958429 and maximum is 10.6232, mean of the close price is 7.5684 with standart deviation of 1.2010. Note that most of

Raw data fields are explained as below:

- **Stock :** Stock Name
- **Date :** Date of the trading day (day only)
- **High :** Highest value of stock in trading day
- **Low :** Lowest value of stock in trading day
- **Volume:** Total volume of buys and sells on stock in trading day

Note that Open value at trading day is not available using this data source, hence simulation forecast will be based on final close price next day.

Other data to be used are foreign exchange rates and BIST30 index. Since BIST is mainly invested by foreigners (more than 70%) USD/TRY movement directly affects market.

Brazil, Russia and Turkey are considered developing countries and often show similar trends to global movement so BRY/TRY and RUB/TRY are also used as features.

Japanese yen should show any change in Eastern market trend (JPY/TRY), who is actually opening the trading day.

BIST30 Index of Turkish market is included to show trend in local economy. BIST30 index contains 30 of the most frequently traded and highest market value stocks.

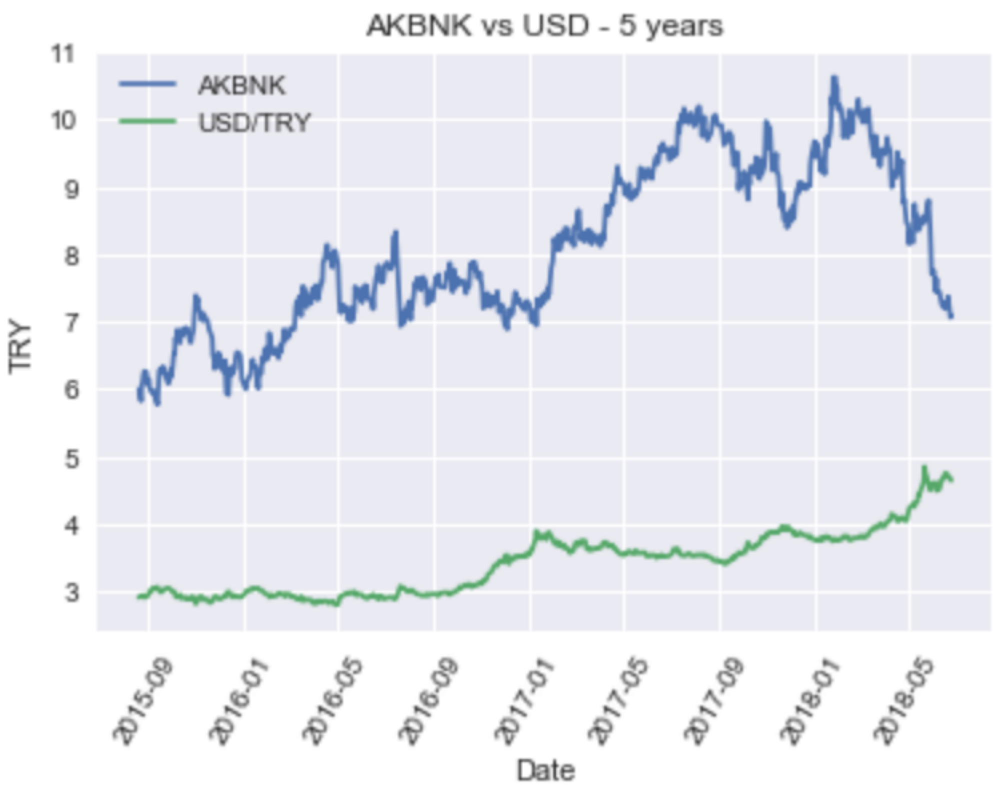
Instead of using global market indices foreign exchange rates were used, each foreign market has different trading days due to national holidays and including this data reduced the training dataset (by %15-20 percent).

Each of the exchange rate and index has similar column format as the stock used.

Note: Data used in this project is available in data folder and can be used by following base code (detail available on notebook):

```
akbnk_df_base = pd.read_csv('data/akbnk_df_base.csv').drop("Unnamed: 0",axis=1)
```

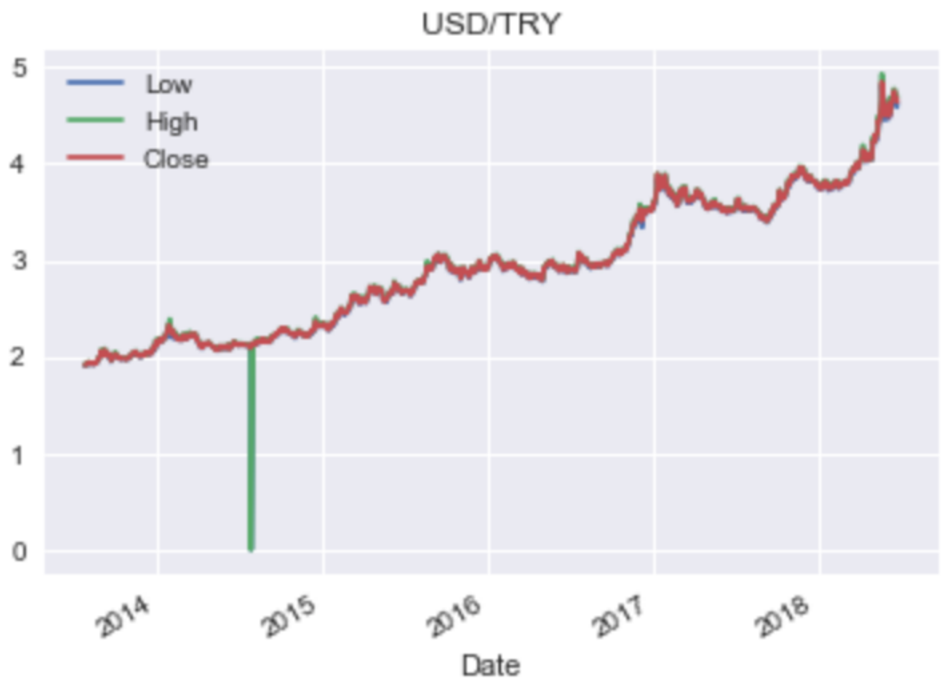
Exploratory Visualization



From the above visualization it is seen that general movement trend is AKBNK stock value decreases when there is a sharp increase in USD/TRY. Last 4 months is especially important as USD/TRY rose very sharp and if one is able to predict this trend (even if there are some short term buy/sell gains), money can be at least saved by not making a false movement.

By checking the plot one can see that there seems to be trends in data for short periods that shows upward movement as well as downward movement, it may be hard to predict a big decrease in price (as can be seen on May 2018, which USD/TRY rose very sharp in a day) but if one can predict the trend, a better profit can be made.

Also it is seen data data’s lowest and highest points (both for AKBNK and USD/TRY) are not oscillating, it has an overall increasing trend, so feeding directly values to the algorithm may give false information. Raw data needs conversion to ratio for all the features.



Some of the features (USD/TRY and BIST30) shows zero points in data, these may be due to errors in data source, we need to handle these points and either drop or replace with rolling mean values.

Additional financial technical indicators will show known financial tricks to predict movement direction and will be engineered. More will be explained in methodology section.

Algorithms and Techniques

Stock predictions systems need to be fast and generalize well as online learning is needed, each day system may require retraining with new data before market opens.

Scikit-Learn provides simple api to test multiple supervised learning classification algorithms, with this tool we are able to asses training performances of various algorithms to select the best model for the problem. Having only 1241 data points with less than 1000 used as training with labelled categorical data, SVC and ensemble methods seem to be better selection for base algorithms.

SVC and ensemble methods are found out popular in stock prediction literature with relatively low volume of training dataset, SVC especially is fast and generalizes well. Ensemble methods like Random Forest, Gradient Boosting or XGboost require very detailed parameter optimization and may be slow on training.

Nowadays deep learning using Neural networks are mainly research. Neural networks tend to perform better with high volume of training dataset, due to our data number being low for NN a basic deep learning model was tested out.

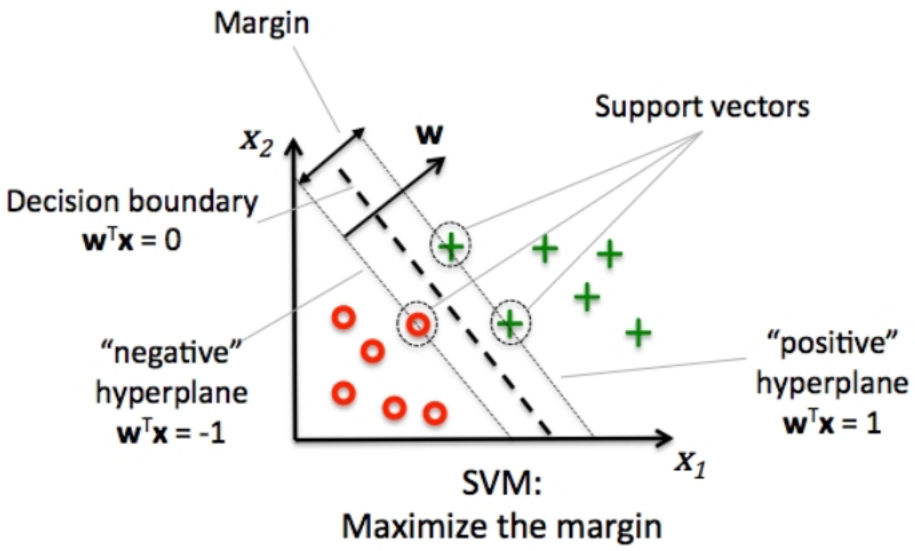
Support Vector Machines

For a simplified two feature space scenario best line splitting features is found such that distance from closest point from two groups is farthest away. Additionally, using the kernel trick can create non-linear decision boundaries in order to capture complex patterns in data, where other linear models may fail to capture. SVCs may not perform well on large datasets, yet dataset used in this project should suffice. Hyper Parameters for SVC are:

Kernel: Type of function to separate hyperplanes, can be linear or radial based or sigmoid.

C: Cost of misclassification. Large C gives low bias and high variance, small C gives higher bias and lower variance

Gamma: For non-linear classification controls shape of the peaks where planes are transformed to higher dimension. Small gamma will give low bias and high variance while large gamma will give higher bias and low variance.



K Nearest Neighbourhood

Stores all available cases and classifies new cases by a majority vote of its k neighbours using a distance function. Since this is a basic classification algorithm implementation it would be basis for the benchmark.

Random Forest

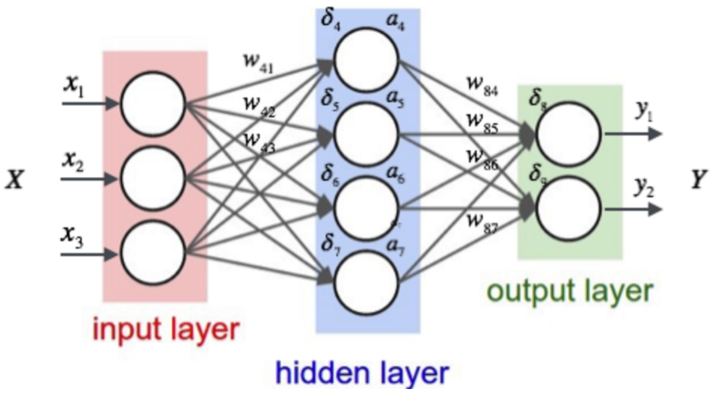
Ensemble of decision trees (Forest). Each tree gives a classification and votes for class. Forest chooses classification having the most votes in all the trees in the forest. Several studies show promising results with Random Forest and ensemble methods.

Gradient Boosting

The idea is to use the weak learning method several times to get a succession of hypotheses, each one refocused on the examples that the previous ones found difficult and misclassified

Artificial Neural Networks

Input data is processed by multiple hidden layers and multiple nodes and error is back propogated until a suitable training loss value is found after several timesteps. Keras with tensorflow backend is used to see a simple deep learning performance with grid search.



Benchmark

Base benchmark will be simple buy and hold, which is equal to always predicting an increase and always generates a buy signal. Second benchmark for model will use K-Nearest neighbourhood classifier with no optimization.

Following scores are the benchmark scores to beat:

Benchmark	F1-Score Validation	Accuracy Score Validation
Buy and Hold	0.3211	0.4890
K-Nearest Neighbours	0.5494	0.5494

Target to beat would be an accuracy score of 55% and F1 score of 55%, yet in order to make a profit we are aiming for more than 60% accuracy as around 50% accuracy means just flipping a coin.

Data will be splitted as Training , Validation and Test. Final model will use Test data to do a real world trading simulation of the prediction system.

III. Methodology

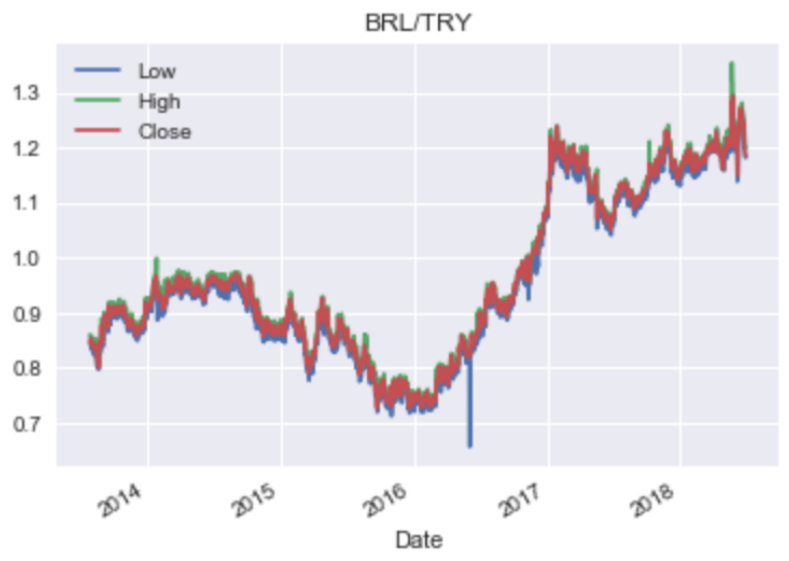
Data Preprocessing

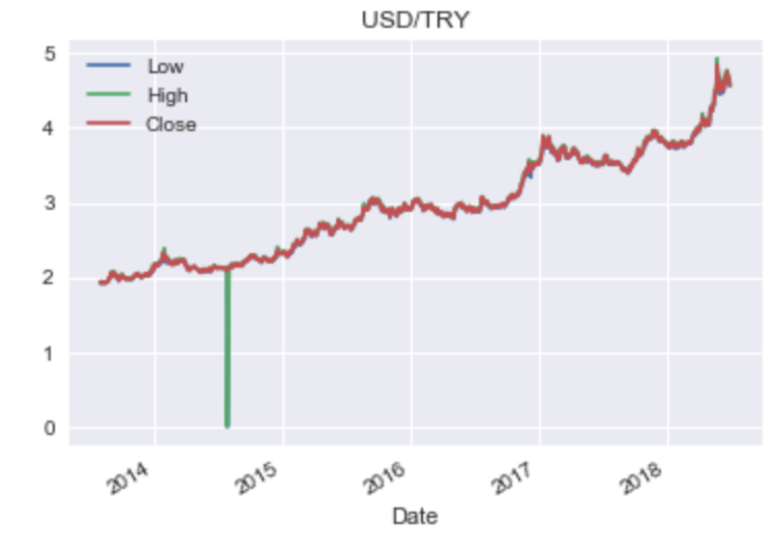
Following preparation steps are done before training:

- Handle Outliers and missing/incorrect data
- Transform raw data to ratios
- Calculate target data
- Calculate Technical Indicators
- Merge data
- Remove unneeded data rows
- Normalize feature data
- Split data for Training and testing

1- Handling Outliers and Missing/Incorrect data:

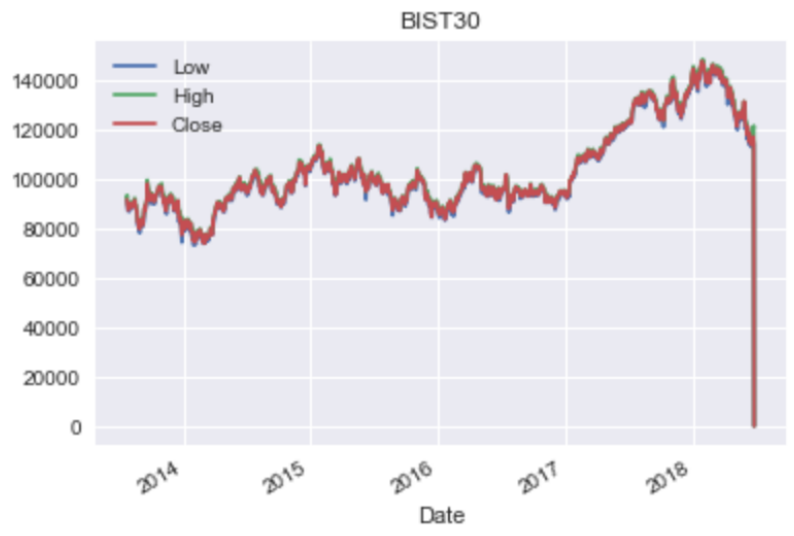
While checking the data we see that BRL/TRY and USD/TRY has outlier values in the data.





With BRL/TRY we see that it has a Low/High value on May 2016 but we see that it is not far beyond a rolling mean of a 5 day (instead of 0.8 it is 0.7) so this data is assumed correct as similar also happens on June 2018.

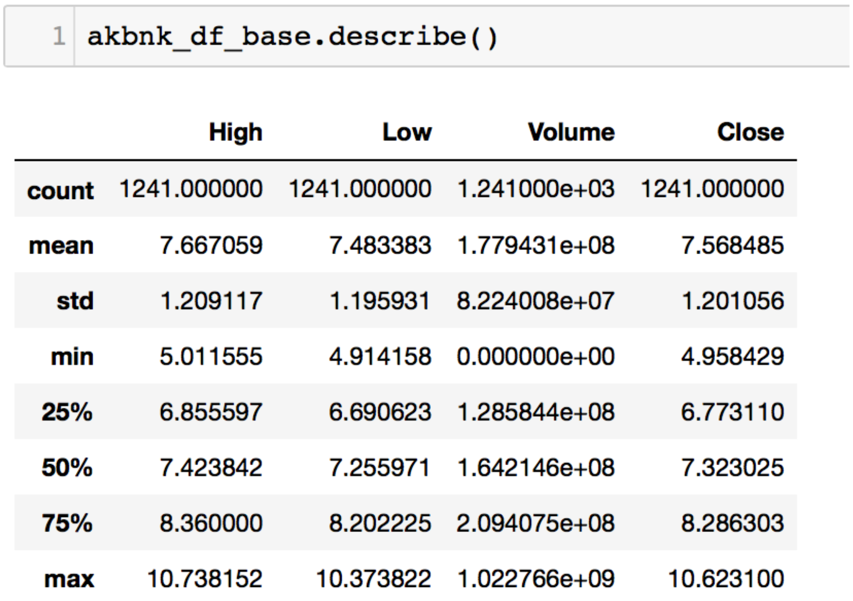
On USD/TRY case there is a 0 value data for a day, deleting this row would also mean stock's data on this day would be deleted during merger, so we replace this data with previous days data.



Due to data sources online nature when data is pulled mid-day BIST30 data may have zero valued data, since our aim is actually predict next day we remove todays data (system is supposed to do prediction before market is open).

2- Transform raw data to ratios

Checking daily Close, High, Low and Volumes we can see that overally target stock has an increasing trend. For example first quarter of Close values are below 6.85 and last quarter are above 8.36.



```python
akbnk_df_base.describe()
```

	High	Low	Volume	Close
count	1241.000000	1241.000000	1.241000e+03	1241.000000
mean	7.667059	7.483383	1.779431e+08	7.568485
std	1.209117	1.195931	8.224008e+07	1.201056
min	5.011555	4.914158	0.000000e+00	4.958429
25%	6.855597	6.690623	1.285844e+08	6.773110
50%	7.423842	7.255971	1.642146e+08	7.323025
75%	8.360000	8.202225	2.094075e+08	8.286303
max	10.738152	10.373822	1.022766e+09	10.623100

Since first 3 years will be trained, 1 year will be used for validation and last year will be our test data mean and standart deviation of test and target will not be same and this may give false results. So High Low Volume and Close are converted to ratios using previous day.

Following features are derived:

- **Day_of_week** : Day of the week (0-4 , no weekends for stock data, can be up to 6 for exchanges)

- **Days_from_last_open :** Engineered feature; how many days market is open from last open. Monday would usually give 3 as weekend market is closed, and longer holidays (can be as well as 9 in Turkey) can be tracked, this may show traders mood in long breaks or any big market moving events.

- **Low_Close_perc : **Percent difference of low and close showing daily trend
- **Prev_Day_Close_perc :** Close percent difference between previous day and this trading day, shows performance on close

- **Prev_Day_Volume_Change:** Volume percent difference between previous da y and this trading day , shows volatility

3- Calculate target data

We want to predict if Close value will increase or decrease next day so using next trading days close price we calculate if stock increased above 0.0005 percent (1 as buy signal) or decreased or holding (0 as sell signal).

- **Next_day_prediction :** Target label , 1 for buy, 0 for sell
- **Next_day_buy_price :** Will be used for simulation and will be dropped before training

4- Calculate Technical Indicators for data and add as feautures

Following technical indicators are used for trend and volatility direction calculation. For stock all of the indicators are used, for additional exchange and BIST30 only base indicators are used.

- **RSI_indicator :** Momentum indicator from calculating RSI value (based on Close). An RSI above %70 usually means stock is overbought (and should act to sell) and below %30 means stock is undervalued and soon increase should come. Reference

- **ATR :** Average True Range, measure of volatility Reference

- **DON_HBAND_IND :** Donchian Channel High , highest price over n periods
- **DON_LBAND_IND :** Donchian Channel Low, , lowest price over n periods Reference

- **ADX :** Average Directional Index, shows strength of trend (Reference

- **CCI_indicator:** Commodity Channel Index, buy signal above 100 and sell signal below -100 Reference

- **MACD_Signal and MACD:** Moving Average Convergence divergence, trend following momentum indicators, shows relationship between two moving average prices Reference

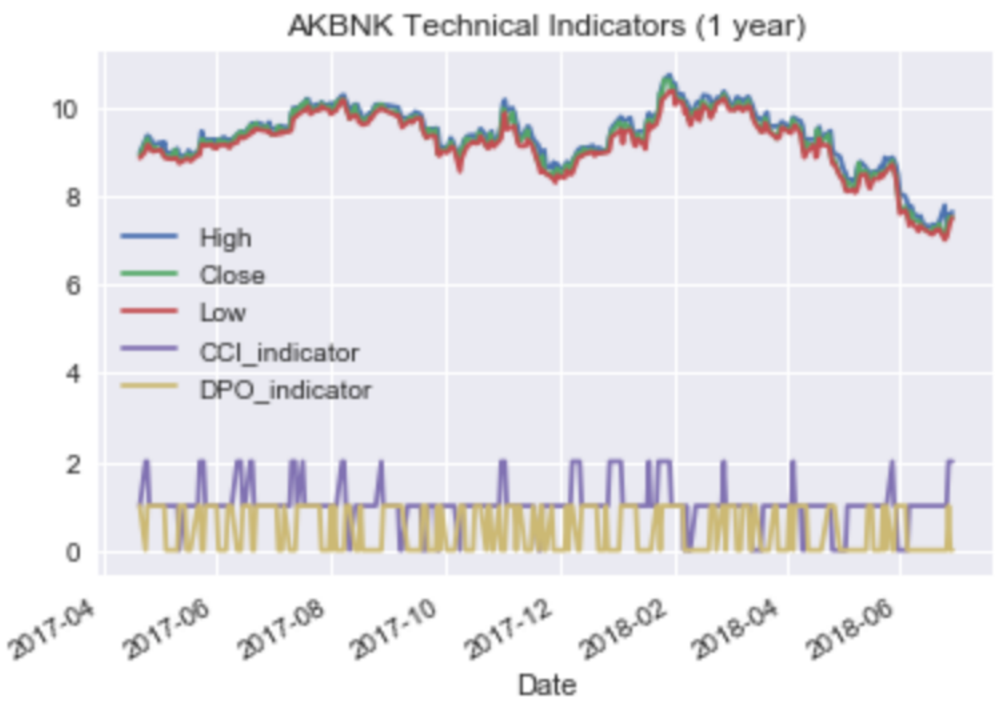
- **DPO_indicator :** Detrended Price Oscillator, estimates length of price cycles from peak to peak Reference

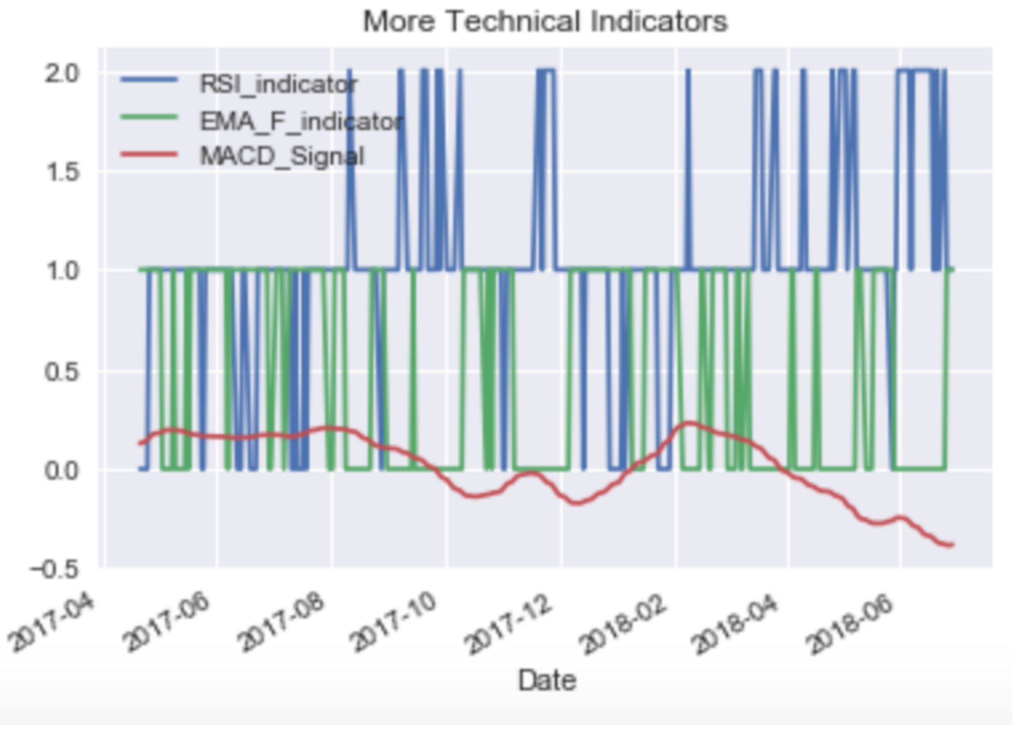
- **EMA_F_indicator :** Exponential Moving Average in n days Reference
- **EMA_S_indicator :** Exponential Moving Average in n*2 days

- **ICH_A :** Ichimoku Cloud High, shows support and resistance Reference
- **ICH_B :** Ichimoku Cloud Low, shows support and resistance

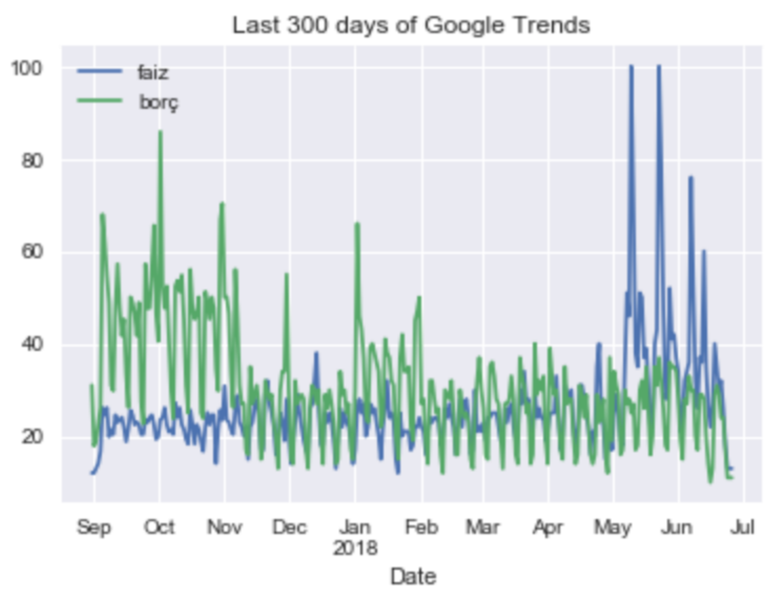
- **Trix :** Triple Exponential Moving Average, more smoothed EMA Reference

Some of the technical indicators are visualized below:





Additionally google trend data is pulled with keywords “**faiz**” (interest rate) and “**borç**” (debt) in Turkey’s results to give additional information on peoples searches. These two keywords show local interest in economy independent of stock.



5- Merge data

All the data is merged on "Date" column using inner join. Base data is stock data so we make sure our stock data information is there.

6- Remove unneeded data rows

While calculating technical indicators (like Moving average) some of the initial data will have NaN values (28 rows) we drop these initial rows.

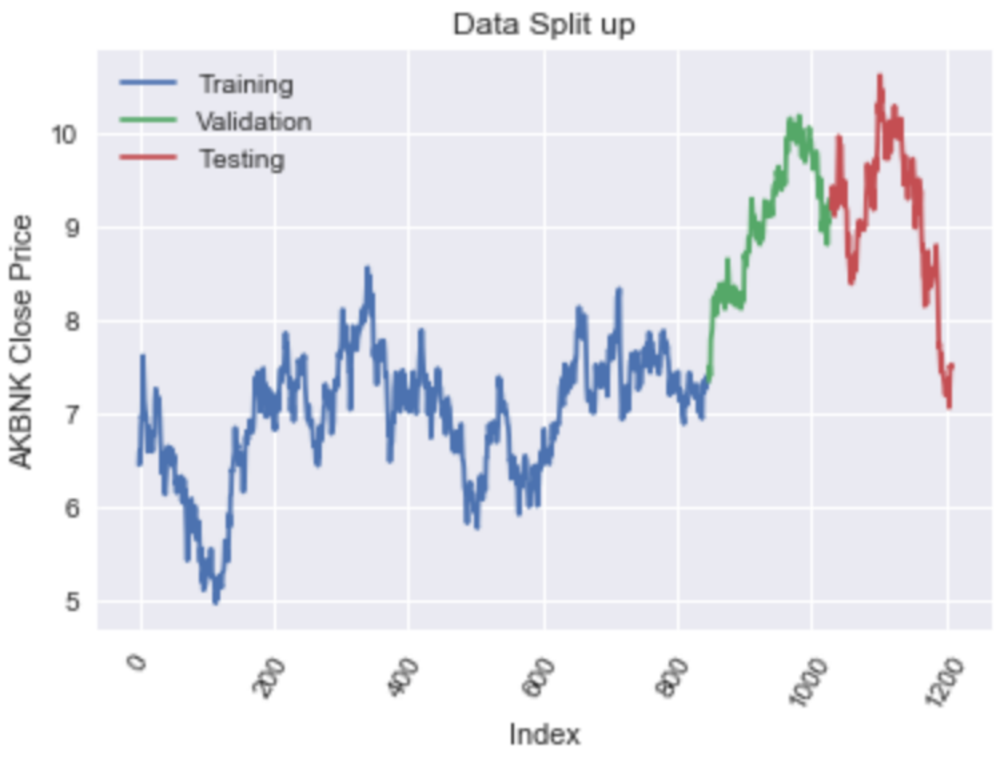
Also feature data and target data are separated from each other. A copy of data is made before separation final simulation phase (to keep date and close columns and calculate profit).

7- Split data for Training and testing

We use 70% of final merged data as training , 15% as validation and 15% for testing. Out of all 5 year data, more than 3 years will be used for training, near 1 year as validation and another 1 year as testing. Model validation will be done by validation data while testing data will only used for final testing of the model chosen.

Out of 1210 rows 846 is used for training and 182 is used for validation and 182 for testing.

Data Type	Start Date	End Date
Training	2013-09-11	2017-01-24
Validation	2017-01-25	2017-10-16
Testing	2017-10-17	2018-06-29



8- Normalize feature data

Each feature in data is normalized to 0 – 1 range for training purposes. Normalization is also done in feature data of validation and test.

Implementation

Most of the coding work was done during web scraper api implementation for pulling data and generating technical indicators.

Api to pull data from investing.com was implemented and data was first trained with this api. After seeing unusually high performance and further investingation revealed that investing.com data is unsuitable for the problem as daily exchange data output time is using U.S. timezone but stock data is using Turkish timezone. So a stock closing price on a day is 17:00 Turkish timezone but closing price on exchange (like USD/TRY) on 23:59 is actually future day in Turkey. Although this data included Open prices as well further processing is needed to get correct time offset data. Mynet.com api was chosen to be reliable as it is from local datasources, drawback is close data is refreshed some 30 min to 1 hour before market opens in Turkey, yet it is more reliable.

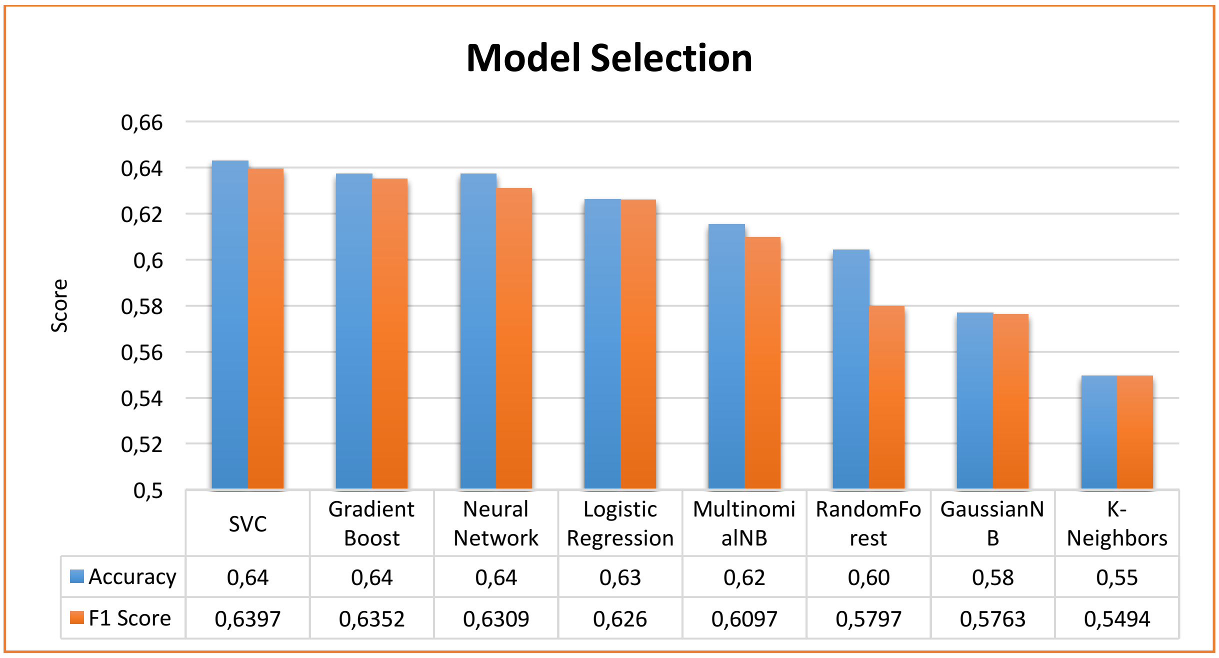
Convolutional Neural Network implementation and LSTM Neural Network implementations were tried out. We were unable to achieve good success rate with CNN networks as computing power and time is needed to find best network with initial and hidden nodes. Additionally LSTM NN was tried out but was unable to implement a successful batch and time series split with this model, so it is not included in results. These implementations were converted to basic dense layer approach and results are provided, details can be found in the notebook.

Having training and test data after preprocessing, scikit-learn algorithms are used with default parameters to extract best model.

Following scikit-learn classfiers were used:

```
sklearn.dummy.DummyClassifier

sklearn.naive_bayes.GaussianNB

sklearn.naive_bayes.MultinomialNB

sklearn.neighbors.KNeighborsClassifier

sklearn.ensemble.RandomForestClassifier

sklearn.svm.SVC

Neural Network: Using Keras , 15x5x2 dense layers
```

Model selection is done on training and validation data only. This step reveals that Support Vector Machine with default parameters is suited for our problem with best accuracy and F1 Score.

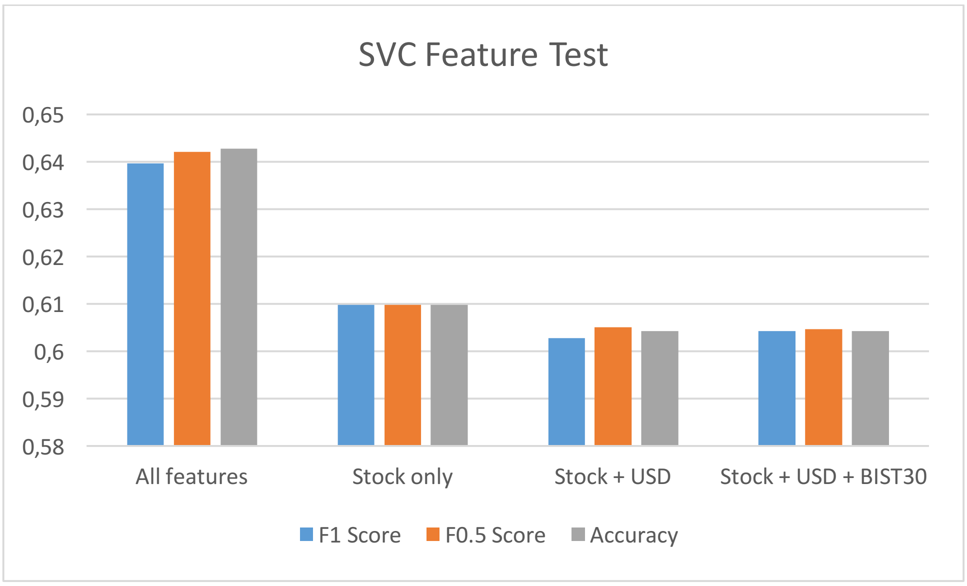


	SVC	Gradient Boost	Neural Network	Logistic Regression	MultinomialNB	RandomForest	GaussianNB	K-Neighbors
🟦 Accuracy	0,64	0,64	0,64	0,63	0,62	0,60	0,58	0,55
🟧 F1 Score	0,6397	0,6352	0,6309	0,626	0,6097	0,5797	0,5763	0,5494

Default hyper parameters for SVC is C=1.0, gamma=”auto”, kernel=”rbf”.

Before going on further optimization we want to check if the features involving stock, exchange rates and BIST30 index are really good and if we need a feature reduction.

For that just using default SVC shows that indeed using all features generated better performance. A %3 improvement of accuracy and F1 score with using all features instead of just stock features is critical on this project.



Additonal model evaluation was performed with a basic Neural network with 1 hidden layer.0.6

Refinement

Support Vector Machine was chosen as having the best base accuracy and F1_score. Note that Gradient Boosting and Neural Networks also gives promising results and may be fine tuned for better performance on future.

After SVC is chosen as the best model Grid search is done on SVC to find optimal hyper parameters.

Important to note is while making a grid search Time Series Split cross validation with 4 splits was done to get the best model for generalization. “C”, “kernel”, “gamma” values were fed.

Optimization with Grid Search shows that there is a 0.01 decrease in F1 Score and Accuracy, yet this is done using cross validation.

After optimization kernel with “**rbf**” is chosen with C parameter as 3 and gamma as 0.0178. Accuracy score is found as 0.6318 in validation data.

Some of the grid search results ranked by accuracy are below. Note that selected model has also the highest recall among other candidates.

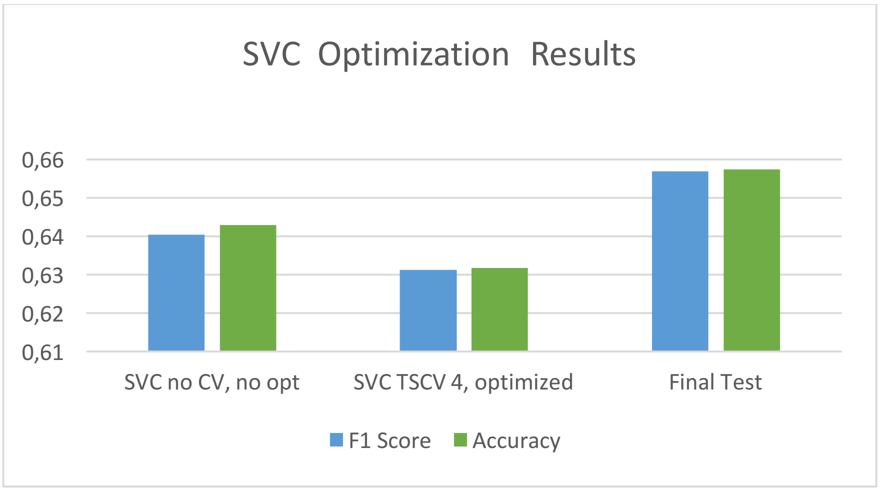
Rank	params	Mean test recall	Mean train recall	Mean test accuracy
1	{'C': 3, 'gamma': 0.017857142857142856, 'kerne...	0.572513	0.630474	0.591716
2	{'C': 10, 'gamma': 0.015, 'kernel': 'rbf'}	0.558041	0.659462	0.590237
2	{'C': 2, 'gamma': 0.017857142857142856, 'kerne...	0.524198	0.651713	0.590237
2	{'C': 2, 'gamma': 0.001, 'kernel': 'linear'}	0.524198	0.651713	0.590237
2	{'C': 2, 'gamma': 0.3, 'kernel': 'linear'}	0.524198	0.651713	0.590237
2	{'C': 2, 'gamma': 0.5, 'kernel': 'linear'}	0.524198	0.651713	0.590237
2	{'C': 2, 'gamma': 0.1, 'kernel': 'linear'}	0.524198	0.651713	0.590237
2	{'C': 2, 'gamma': 0.15, 'kernel': 'linear'}	0.524198	0.651713	0.590237
2	{'C': 2, 'gamma': 0.015, 'kernel': 'linear'}	0.524198	0.651713	0.590237
2	{'C': 2, 'gamma': 0.0015, 'kernel': 'linear'}	0.524198	0.651713	0.590237
11	{'C': 0.2, 'gamma': 0.0015, 'kernel': 'linear'}	0.555409	0.634112	0.587278
11	{'C': 0.2, 'gamma': 0.017857142857142856, 'ker...	0.555409	0.634112	0.587278

IV. Results

Model Evaluation and Validation

Using multiple scikit algorithms and a base Neural networks implementation it is shown that Support Vector Machines gives the best performance, and while after tuning hyper parameters we do see a slight decrease in accuracy after testing we see that we have good results with testing data.

Reduction on after performance optimization can be explained by using cross validation to fine tune the model.



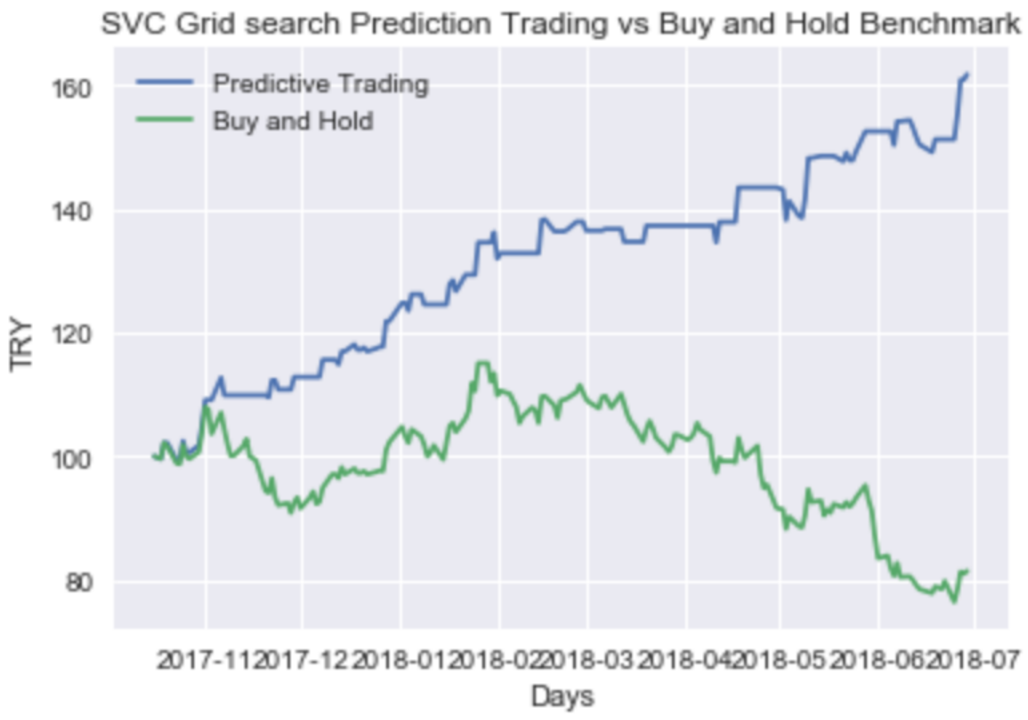
Justification

Optimized SVC model has lower accuracy score and F1-Score than its base non optimized cousin. This is due to Time series Split with 4 folds made on the SVC algorithm. Validation score of the final model outperforms both buy and hold and K-Nearest neighbourhood algorithms. Final testing score gives promising results as Model was finally tested with out of sample test data and performance is seen as satisfactory.

Model	F1-Score Validation	Accuracy Validation	F1- Score Test	Accuracy Test
SVC Optimized with TSCV (final)	0.6313	0.6318	0.6568	0.6574
SVC no CV no opt	0.6404	0.6428		
Buy and Hold (benchmark)	0.3211	0.4890		
KNN (benchmark)	0.5494	0.5494		

Final model was simulated using a custom trader simulation system.

Out of sample testing data from 2017-10-16 and 2018-06-29 was used (was never used in validation or training), %61.76 increase was generated while with buy and sold strategy - %18.55 loss was done.

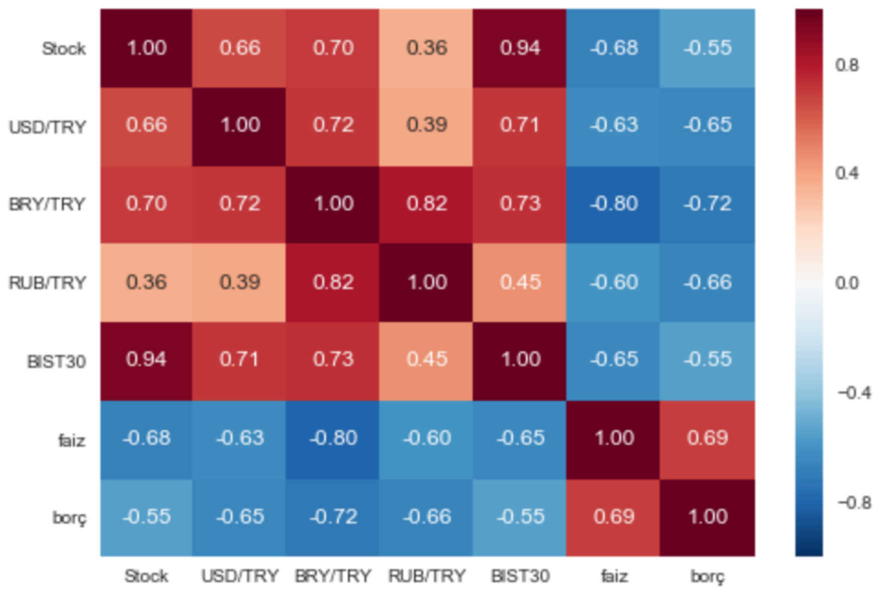


Results are satisfactory as with this fluctuating market (May 2018 saw sharp increase in USD/TRY and it was election month in June 2018) system was able generate profit while USD increases and stock market is on decreasing trend.

V. Conclusion

Free-Form Visualization

Feature Correlation

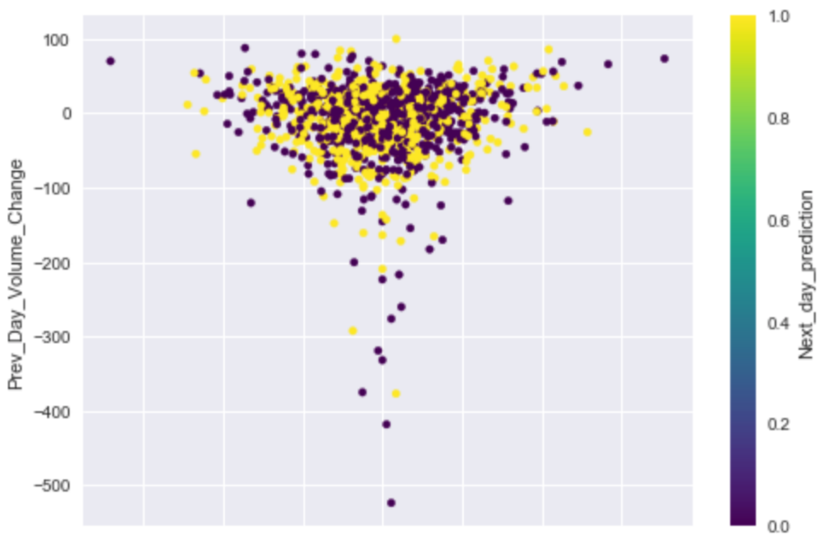


	Stock	USD/TRY	BRY/TRY	RUB/TRY	BIST30	faiz	borç
Stock	1.00	0.66	0.70	0.36	0.94	-0.68	-0.55
USD/TRY	0.66	1.00	0.72	0.39	0.71	-0.63	-0.65
BRY/TRY	0.70	0.72	1.00	0.82	0.73	-0.80	-0.72
RUB/TRY	0.36	0.39	0.82	1.00	0.45	-0.60	-0.66
BIST30	0.94	0.71	0.73	0.45	1.00	-0.65	-0.55
faiz	-0.68	-0.63	-0.80	-0.60	-0.65	1.00	0.69
borç	-0.55	-0.65	-0.72	-0.66	-0.55	0.69	1.00

Correlation matrix shows that our features selected all are either positively or negatively correlated with our target value.

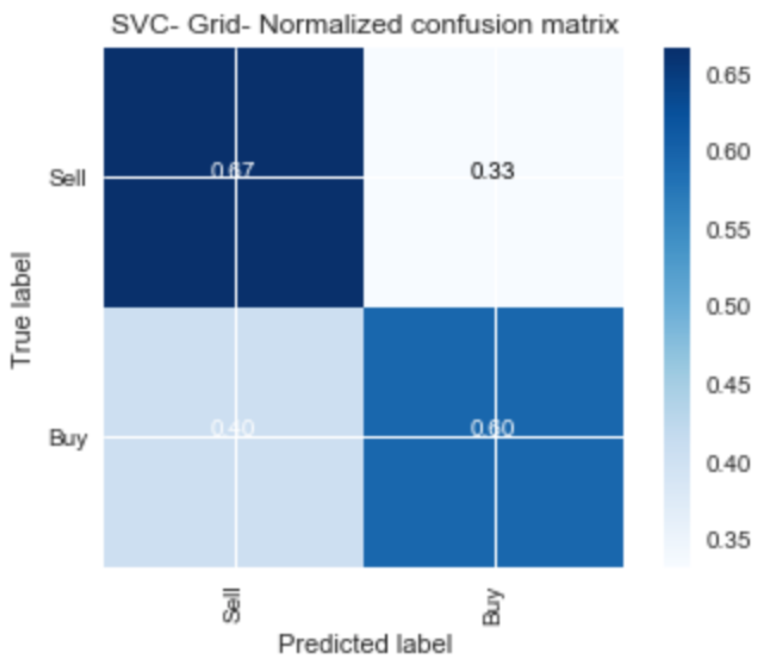
Feature Relation

Comparing Previous day volume change and close percentage change we can see that there seems to be a pattern but can not be separated linearly.



Confusion Matrix

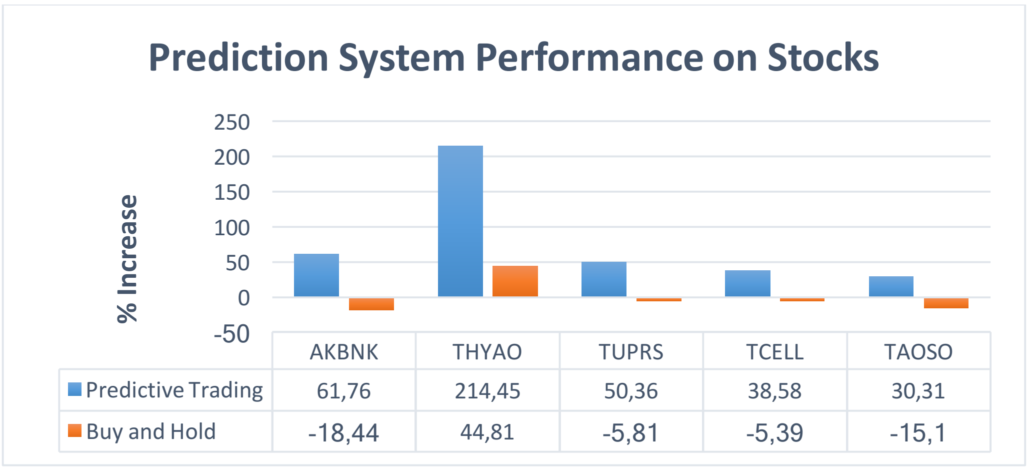
In order to visualize performance following confusion matrix is generated:



We can see that system mostly confuses buys as sell, with confusion value higher than sells identified as buys. It is more important to get more sell signals correct as this directly affects profit loss, while buy signals affect profit generation.

Alternative Target Stocks

Since model and feature generation is actually base, implementation is systemized and evaluated on several other stocks. Even on stocks with fluctuating performance vs USD prediction system is able to beat buy and hold strategy. On a overall trending stock (due to EURO income) THYAO does not have many decreases yet our system is able to cumulate a 214% gain while buy and hold got 44% gain.



	AKBNK	THYAO	TUPRS	TCELL	TAOSO
🟦 Predictive Trading	61,76	214,45	50,36	38,58	30,31
🟧 Buy and Hold	-18,44	44,81	-5,81	-5,39	-15,1

Reflection

Most difficulties of the project were on developing feature data. Data was re-assested at in initial stages of the project I saw more than %80 accuracy which was unusual. It was found out that future data was left out in training. Another problem occurred with using dataset from investing.com as future market index data (Shangai) was being fed into network, it was found out that investing.com does not stick to times of the local time zones but uses U.S. time. Switched to local mynet.com api for more reliable data (loosing open feauture).

Another time consuming part was trying to implement a LSTM network, which should work better with time series data. I was unable to get a good and reliable implementation of LSTM network, so opted to drop out of results. Although CNN network is tried (code can be seen on notebook commented out), it is very difficult to predict good initial and hidden layer values with insight and grid search is very time consuming.

Process used to define and develop model can be summarized below:

1- Initial problem is identified and data to use is identified.
2- Webscraper is built for getting stock and exchange data. This step is crutial as we want to systemized the project, also do not want to be dependent upon single data. Mynet.com and Investing.com stock download api's were made.
3- Raw data was transformed into ratios as prices tend to increase over years.
4- Technical indicators were extracted and added as features
5- Stock, BIST30 and exchange data are all merged on date with day as reference. Target columns separated and any future date data are removed
6- Data is split into 3 parts, %70 as training, next %15 as validation and final %15 as testing. Since this is a time series data these data needs to be in order. Final test data is never used while choosing model.
7- Benchmark model performance accuracy and F1 score are found. On finance buy and hold is base benchmark. K Nearest Neighbourhood is chosen as secondary benchmark.
8- Multiple scikit-learn algorithms are tested with validation data to find best algorithm.
During this phase it was found that feature calculation was falsely done and future data was fed into system as was getting accuracy scores mode than %80. After correction it is also found out that data from investing.com api was giving future information on exchange rates relative to stock.
9- Neural network with 15x5x2 architecture was implemented to test validation performance.
In this phase CNN and LSTM Networks were also tested out but it was found that more time and computation is needed to find best optimized network, so sticked to simple solution of dense Network.
*10-*Support Vector Machine is found to have best accuracy and F1 score.
*11-*Grid search with time series cross validation is done on SVC. A slight decrease in accuracy is seen but since we used cross validation (with 4 splits) this is expected as we need a generalized model.

12-Testing is done on final SVC model and we found that %61,76 profit was made using predictive trading while buy and hold strategy makes a loss of %18.55

13-Implementation is systemized to see performance on other stocks to check if architecture is future proof.

Improvement

To further improve the model following can be done:

- Open price is done fed into network, open price may give more insight on features.
- Due to weekends and holidays we have missing data for stock. In this project close value of exchange rates were on the same day as stock, missing weekend and holiday data of stock may be filled with last close and additional parameter to indicate stock open close can be made and final exchange rate values can be fed into network. Note that simpler approach of showing how many days from last close was passed in this work.
- While generating features for stock, one may put similarly valued stocks as additional features (e.g. AKBNK as target but GARAN (Garanti Bank) as additional feature), even stocks in same sector worldwide can be added.
- Boosting classifiers may be used with more tuned parameters. Especially XGBoost is dominating Kaggle competitions.
- LSTM Neural Network can be used. Note that during development LSTM network was tried but it was not suited for this project still better implementations may give better performance.

References

[1]

https://www.duo.uio.no/bitstream/handle/10852/51275/PredictingStocksWithMachineLearning.pdf?sequence=1

https://itnext.io/2017s-deep-learning-papers-on-investing-7489e8f59487

https://iknowfirst.com/rsar-machine-learning-trading-stock-market-and-chaos

http://www.cs.umanitoba.ca/~ywang/papers/ideas14.pdf

https://www.youtube.com/watch?v=s1ubvXCu-xs

https://github.com/hayatoy/ml-forex-prediction

https://towardsdatascience.com/stock-prediction-in-python-b66555171a2

https://enlight.nyc/stock-market-prediction

[2]

http://finans.mynet.com/borsa/hisseler/akbnk-akbank/

https://trends.google.com/trends/explore?date=today%203-m&geo=TR&q=%2Fm%2F09nqf,%2Fm%2F02l6h,%2Fm%2F08l57c,%2Fm%2F025rs2z

https://itnext.io/2017s-deep-learning-papers-on-investing-7489e8f59487

https://medium.com/mlreview/a-simple-deep-learning-model-for-stock-price-prediction-using-tensorflow-30505541d877

https://www.kaggle.com/pbolleddu2320/stock-market-prediction-with-python/data

[3]

https://www.investopedia.com/terms/b/backtesting.asp

https://www.quantstart.com/articles/Backtesting-a-Forecasting-Strategy-for-the-SP500-in-Python-with-pandas

https://etd.auburn.edu/bitstream/handle/10415/5652/Application%20of%20machine%20learning%20techniques%20for%20stock%20market%20prediction.pdf?sequence=2