Capstone Project **Machine Learning Engineer Nanodegree**	Dmytro Morosov April 30st, 2018

Definition

Project Overview

The essential part of market stock trading is stock prices prediction. Stock market players are trying to determine the future value of a company stocks traded on an exchange. The accurate stock price prediction allows get profit from operations on trading on stock market. The problem is that stock prices depends on large number of factors and process of price prediction is very sophisticated and can't be described with some mathematical formula and some set of steps which can be combined in algorithm. Experienced stock market players make their decisions based on historical price data, news about companies, company's activity statistic and they personal previously gained experience and knowledge.

The issue is the market is very volatile and influenced by a large number of factors. Market often does have concrete trends that can be analyzed, and therefore may still be able to be reasonably predicted in short periods of time.

The idea of the project is to apply Machine Learning concepts to predict individual stocks from American Stock Markets in short term (5 day period). All data sets were downloaded through API from Yahoo Finances service.

Problem Statement

The goal of the project to develop service for price prediction of individual stocks in American Stock Market over 5-day period in future with accuracy +/- 5% of actual value, on average). Provide user friendly interface with ability check historical pricing data and show predicted data for individual stocks in the American Stock Market .To archive mentioned goals next steps are proposed to be implemented:

1. Download through Yahoo Finance service API historical pricing data of 4 individual stocks and the S&P 500.

2. Process downloaded data, check missing points, clear unused features, if required implement extra features (for example calculated from base features with using statistical methods).
3. If required normalize data.
4. Visualize and analyze received features and stock prices. Split data on train and test sets for cross validation.
5. Use several Machine Learning regression algorithms, train them on historical data with different look back periods. For each look back period split on train and test set are required. Check accuracy of each algorytm on test set. Choice algorithm with best score.
6. Implement web application which allow users select any of of four stocks, look through historical data and see predicted values for next 5 days.

Metrics

The *r2_score* function computes R2, the coefficient of determination. It provides a measure of how well future samples are likely to be predicted by the model. Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R2 score of 0.0.
If is the predicted value of the -th sample and is the corresponding true value, then the score R^2 estimated over is defined as

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_{\text{samples}}-1} (y_i - \bar{y})^2}$$

where $\bar{y} = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} y_i$.

The best possible score for a R2 metric evaluation is 1, and the score can also be negative. A negative score can indicate that a particular model is unusable for the problem, and if R2 is consistently 1 then the model is disregarding the input features (Sklearn.metrics.r2_score, N.d).
[http://scikit-learn.org/stable/modules/model_evaluation.html#r2-score]

The R2 value represents how well observed outcomes are reproduced by a model, compared to the total variance explained by the model.

Analysis

Data Exploration

For future stock price predictions historical data must be obtained. For completing task, proof model concept historical data was downloaded manually in CSV format from Yahoo Finance Service (for example https://finance.yahoo.com/quote/CSV/history?p=CSV). Downloaded files were used in Jupyter notebooks files which according to Capstone project rules were excluded from repository. Final project implementation gets data in JSON format automatically from Yahoo Finance service API and makes all calculation based on received data. In order to ensure good application performance and avoid possible issues with exceeding limit for API calls for financial services all loaded data saved to application database (NoSql solution MongoDB) and database is used as data source for all next calculations on real application.

For purpose of this project dataset was used for which contained historical values of stock indexes that represent a company traded in the American Stock Market (NYSE or NASDAQ) from April 28th, 2016 – April 28th, 2018. Dataset contains 2 years of historical data with indexes: Google (GOOG), Apple (AAPL), IBM (IBM), Nvidia (NVDA), and the S&P 500 (SPY). Google, Apple, IBM, and Nvidia where chosen for personal interest as one of largest IT companies of the USA and the world. The interesting and challenging index is also S&P 500 (SPY). The S&P 500 is the stock index, which includes 500 selected US stock companies with the largest capitalization. Challenged task with S&P 500 index is that stock not always traded and there is null values in dataset when stocks were not traded.

The technique of data preprocessing will be discussed more detail later in methodology chapter.

Downloaded CSV file from Yahoo Finance provides multiple features for analysis: “Open”, “High”, “Close”, “Adjusted Close”, “Volume”.

- *Date* – day for which stock prices provided
- *Open* – price of stock when trade was opened
- *High* – highest price on stock for the particular day

- *Close* – price of stock by the end of day when trade was closed
- *Adjusted Close* – average stock price for the day
- *Volume* – number of stock traded on stock market on the particular day

Example of how dataset looks provided bellow:

```
        Date         Open         High          Low        Close  \
0  2017-03-27  2329.110107  2344.899902  2322.250000  2341.590088
1  2017-03-28  2339.790039  2363.780029  2337.629883  2358.570068
2  2017-03-29  2356.540039  2363.360107  2352.939941  2361.129883
3  2017-03-30  2361.310059  2370.419922  2358.580078  2368.060059
4  2017-03-31  2364.820068  2370.350098  2362.600098  2362.719971

     Adj Close      Volume
0  2341.590088  3240230000
1  2358.570068  3367780000
2  2361.129883  3106940000
3  2368.060059  3158420000
4  2362.719971  3354110000
```

Provided data example represents dataset for S&P 500 over a 5-day period.
Statistic information about whole dataset:

```
count     505.000000
mean     2388.149999
std       216.057368
min      2000.540039
25%      2179.979980
50%      2385.260010
75%      2564.979980
max      2872.870117
```

Dataset contains 505 points with 2000.54 as minimum value and 2872.870117 as maximum value. Mean values is 2388.15 with standard deviation 216.06.
As requested in rubric for finance & trading capstone only 2 features are going to be used: Date and Adjusted Close. “Adjusted Close” is a feature which must be predicted by application. Other features provided by Yahoo Finance service will be removed and not used in model implementation. “Adjusted Close” quite complex feature for making accurate calculation and prediction, so we will extend features set with engineered features calculated base on basic statistics and historical values of Adjusted Close feature:

Exploratory Visualization

The following graph (**Fig. 1**) is a visualization of the Rolling Mean. The visualization is useful to see how they relate to the Adjusted Close values over a given period of time

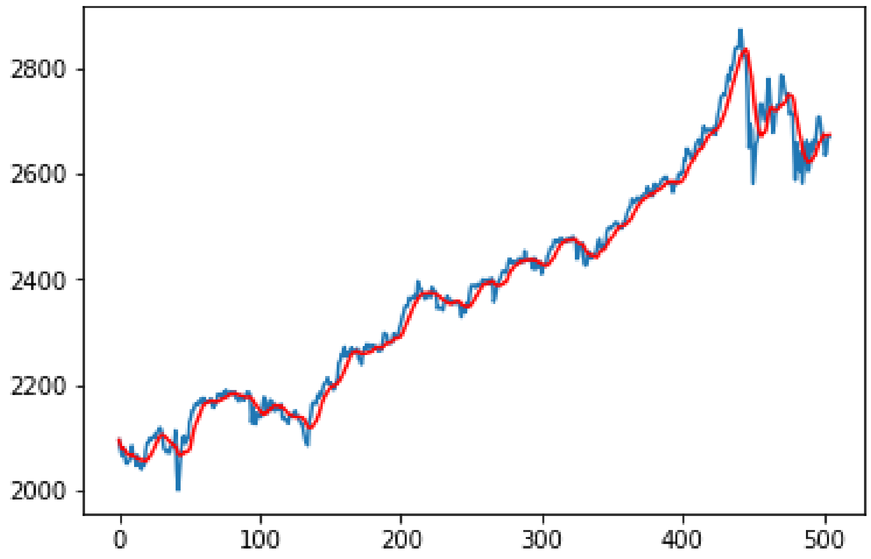

Fig. 1 - Rolling mean for S&P 500 stock prices over 2 year period (on graph number of days on X axes, stock price on Y axes)

Algorithms and Techniques

The problem is a Supervised Learning problem given we are attempting to predict future stock prices based on learning and training from historical data. The continuous nature of the data makes it particularly suitable for regression type algorithms. The problem need to be solved in project is to predict continuous value of “Adjusted Close” value based on its historical values. Soa after evaluating the types of algorithms available I have decided to use an Linear Regression algorithm, the SV Regression algorithm, the K-Nearest Neighbor (KNN) Regressor algorithm, and a Random Forest Regressor algorithm to evaluate which returns the best results. Each of these algorithms are available in the Scikit Learn Python library. Another technique that will be utilized for all mentioned models is Time Series Splitting for cross-validation.

So main idea behind used algorithms:

Linear regression is a linear approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X.
In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data [https://en.wikipedia.org/wiki/Linear_regression].
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. [https://en.wikipedia.org/wiki/Random_forest]. Decision trees are a logic-based algorithm and are generally applicable to problems with discrete or categorical features, since the tree structure is based on learning simple decision rules from the features. Strengths of DTs are the speed of training and classification, relatively high tolerance to missing or irrelevant attributes and the relative ease with which the model and the model parameters can be understood and interpreted. Weaknesses of DTs include relatively low accuracies compared to other models, a tendency to overfit and the possibility of instability, since small variations in the data might result in a completely different tree being generated.
K-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output of k-NN regression is the property value for the object. This value is the average of the values of its k nearest neighbors. k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms. [https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm]

Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated. [http://www.saedsayad.com/support_vector_machine_reg.html]

Benchmark

The main goal of this project is to build model able to predict data with error not higher than 5% work week ahead (5 days) and predict already know historical stock prices with score more than 95%.

Methodology

Data Preprocessing

Preprocessing stage includes pulling data (for report reviews let's go through the S&P 500, and used it as a benchmark for finding null values in other stocks). The following steps were taken:

1. Decide on a historical date range to explore. The dates April 28th, 2018 – April 28th, 2018 where used in the analysis process.
2. Gather data on the S&P 500 based on the chosen dates.
3. Gather data on 4 other stocks based on the chosen dates. GOOG, NVDA, AAPL, and IBM were used during this phase.
4. Drop all data values except for the Adjusted Close values.
5. Drop all dates in all chosen stocks where the S&P 500 is null. These represent dates the stock market was closed.
6. Locate null values in the other stocks and fill in values forward. This means taking the last known value that was not-null, and filling that value forward until you reach the next non-null value.
7. Locate any more null values in the other stocks and fill backwards. If a particular stock has null values at the start of the date range, find the first non-null value and use that value to fill back to the start of the date range.

At this point the data is ready to be explored. There is no need to search for outliers, as all historical data values represent the nature of the stocks.

Implementation

Once the data was fully processed and features were evaluated I decided to run the data through 3 different models to determine which gave the best results. The algorithms chosen were a simple Linear Regression, SV Regressor, A KNN Regressor, and a Random Forest Regressor.

In web application each of this methods was implemented in separate model. And was invoking as for prediction unknown future values and making prediction of historical values. For prediction of historical values predictions were made iteratively from first day after look back period to current day. Look back period was chosen according to tests performed on models to gain best accuracy. Because of sophisticated and variable forms of stock price graphs accuracy was getting worse on quite small number of history points. As optimal value 30 points were chosen, because it was shown good result for stock with high fluctuation (price was going dramatically down and the recovering after some time) on short time periods. One important note affected on implementation was that fact that implemented algorithms were going to be used on real web servers with limit budget, so heavy methods of optimisation were not allowed due limited resources of web server. Estimated configuration of web server for test project purpose: CPU with cors and 2 GB of RAM (usual configuration for start of small projects).

Linear Regression

As usual to avoid overfirting data for regression was splitted on train and test dataset.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

Random state 42 was used to ensure reproducible results of the regression method. 25% of data of look back period was used for testing of predicted model and 75% of data was used for training prediction model. GridSearchCV optimization was used on jupyter notebbok on project on manually downloaded data. As parameters for optimization were used:

- fit_intercept - whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (e.g. data is expected to be already centered). *Used values:* [True, False]
- normalize - If True, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm. *Used values:* [True, False]

GridSearchCV didn't show better performance that standart parametrs provided by sklearn. On real web application on other stock prices GridSearchCV was not used due to high performance load on web server.

SVRegression

Like in previous steps data was splitted on train and test sets with train_test_split method.

GridSearchCV optimization was used on jupyter notebook on project on manually downloaded data. Parameters used for optimization [1, 10] for C, as well as [1, 0.1, 0.01, 0.001] for epsilon. The parameters were derived from the recommendations in the documentation. An R2 scorer is set up and used as the scoring function to find the best model parameters. On real web application on other stock prices GridSearchCV was not used due to high performance load on web server.

Random Forest Regressor

Regression method was used with the same splitted data as previous methods (25% test set and 75% train set). Random state 0 was used to gain reproducable results. GridSearchCV optimization was used on jupyter notebook on project on manually downloaded data. As parameters for optimization were used:

- n_estimators - The number of trees in the forest. *Used values:* [1, 10, 100, 500, 1000]
- criterion - The function to measure the quality of a split. Supported criteria are “mse” for the mean squared error, which is equal to variance reduction as feature selection criterion, and “mae” for the mean absolute error. *Used values:* [‘mse’, ‘mae’]
- max_depth - the maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than

min_samples_split samples. *Used values:* [None, 100, 1000, 5000, 10000]

GridSearchCV didn't show better performance than standart parameters provided by sklearn. On real web application on other stock prices GridSearchCV was not used due to high performance load on web server.

KNN Regressor
As usual regression method opearates with splitted test and train sets. Regressor was run with parameter n_neighbors = 2. As in previous example on proof of concept stage GridSearchCV was used. Parameters for optimization were used:

- n_neighbors - number of neighbors to use by default for kneighbors queries. *Used values:* [1, 2, 3, 4, 5, 10]
- weights - Weight function used in prediction *Used values:* ['uniform', 'distance']
- algorithm - Algorithm used to compute the nearest neighbors. *Used values:* ['auto', 'ball_tree', 'kd_tree', 'brute']

According to GridSearchCV results for real application parameters used: n_neighbors = 2 as optimal value with best result.

Refinement
Initial runs was showing poor results for most regression models, especially on parts where stock were significantly went down in price. Linear regression was showing one of the most worst result on graphs part with significant fluctuation of price values. SVR was reacting too slowly on stock price trend and was given prediction values with delay. KNN and Random Forest Regression was showing quite optimistic results. This were only methods able to work with sophisticated pattern of data. Available options for method improvements were tuning parameters of each method and play around of look back period of historical data for future data predictions. Obviously all few years data set can't be used for future period, because of very sophisticated data pattern and high computational resources required for this computation. So for prediction was used limited interval from past right before prediction point. As number points for look back period were used values: 10, 15, 30, 45, 60, 100. Short look back

period (10, 15 points) was leading to biased model for each regression model, definitely this was too small number of points to make accurate prediction. Optimal result was 30 points. This points number was enough to make prediction and was contained parts of dataset with quite simple pattern, which was essential for LinearRegression algorythm and SVR. For KNN and Random Forest Regression 45 points was good choice too, but for more points (60, 100) result was getting worse, sophisticated data pattern on this points number was getting a problem event for KNN and RNN. According to previous result look back period was chosen as 30 points. To tune parameters of each individual regression model GridSearchCV method was used. Only few method was improved their result given GridSearchCV in other cases score values were lower than standard parameters. Chosen parameters was described before in "Implementation" section.

Results

Model Evaluation and Validation

To estimates model results R2 score was used for 2 year data period (except interval from start with length of look back period) and calculated based on historical data and predicted data calculated for all period (except interval from start with length of look back period (30 days)).Best model was chosen based on the highest value of calculated R2 score.

Before starting this project expected that *LinearRegression* would shown best results. But very noisy data on some intervals was decreasing accuracy of the model. Final R2 score values is: 0.97. Real value is lower due to look back period which is gives R2 score equal 1, where predicted values copied real values for better visualization. On next Fig.2 red line represents real stock price values, while blue line predicted by LinearRegression model values. As we can see blue line looks smoother than red line and its values has a bit delay after red ones. Thats means than proposed model can't react quickly on stock market changes and definitely can't predict future changes. That's why model is not recommended to be used in real life.

Worse results were shown by *SVR* method. Method couldn't pick stock prices trend in time and always reacts with significant delay (see Fig. 3). As previous method it also was not able to get stock price changes in time and looks like biased on graph. As in previous example figure shows red line as real stock price values and blue line as predicted by SVR model values. Definitely model cannot be used for prediction values in real life. Due to sophisticated data pattern Ensemble model has showed best results (Fig. 4 and Fig. 5). *Random Forest* Regressor has R2 value equal to 0.98 which is higher 0.95 (meat requirement of this project).

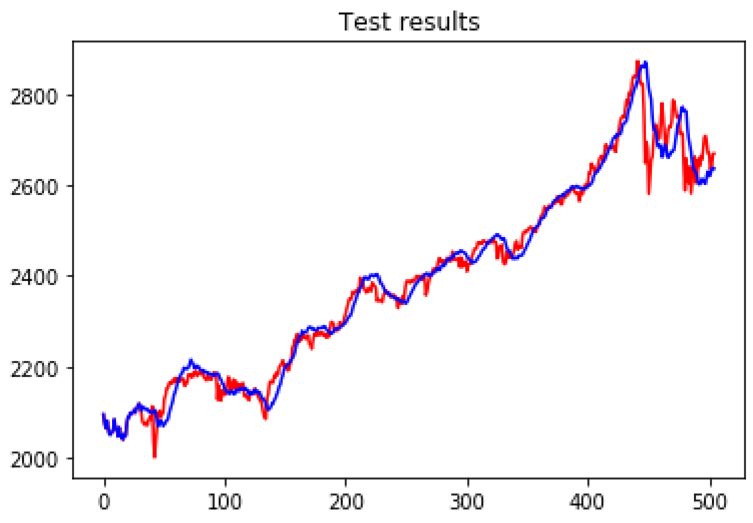

Fig. 2 - Linear Regression algorithm performance.

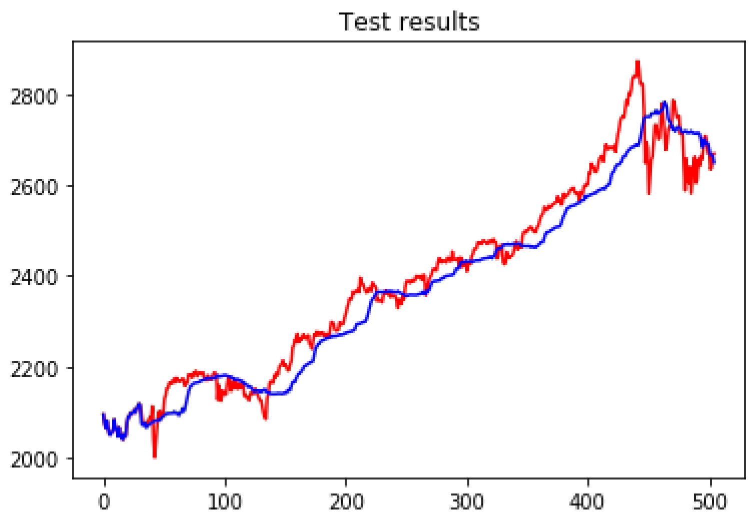

Fig. 3 - SVR algorithm performance.

As we can see on figure model good generalizes data, but still has some delay after real values. Customers can loss money's on price drop happened before model predict it. It must be kept in mind in case of model use in real life.

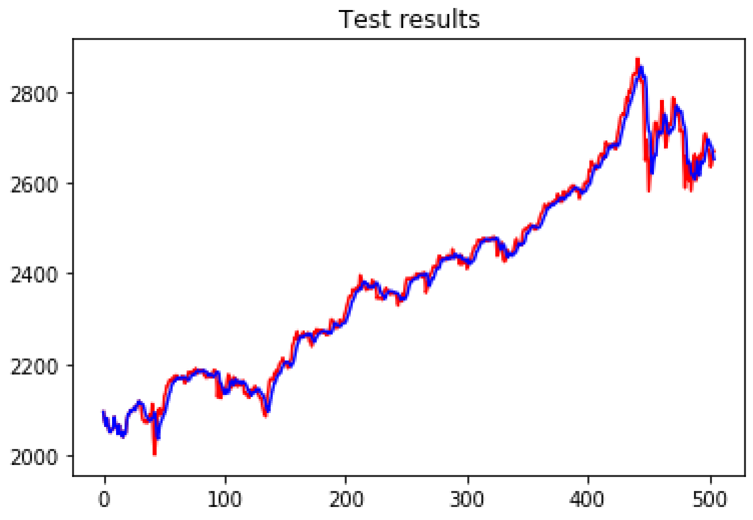

Fig. 4 - Random forest algorithm performance.

The best result was shown by *KNN* regression. Reported score is 0.98. Model a bit better generalizes data, but still has some delay after real values.

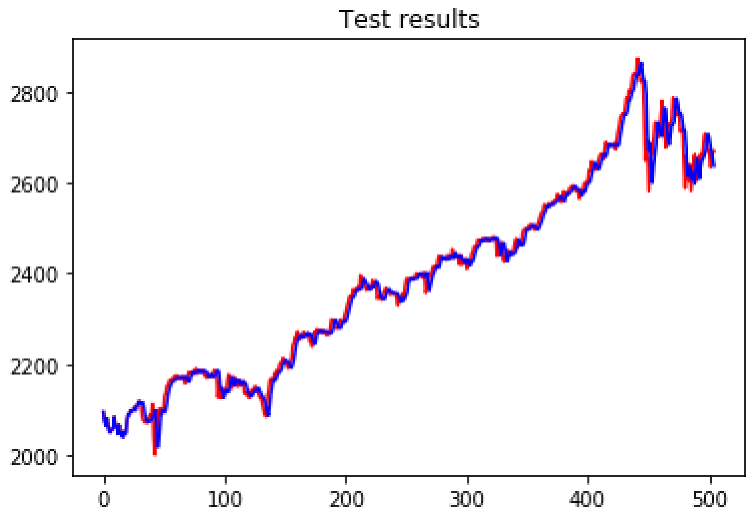

Fig. 5 - *KNN* algorithm performance.

As with Random Forest regression it must be kept in mind that customers can lose money on price drop happened before model predict it.
From models above KNN regression has shown best results and can be used for stock price prediction. Results on real application with other historical stocks data shows almost the same performance and accuracy of the model. Regarding

prediction of unknown values KNN regression model shows good results on first days after look up period and results getting worse for next days. That's why model doesn't good fit requirement of the project to predict values with at least accuracy 95% for 5 days. To meet this requirements possibly that more powerful algorithms are required: like neural networks, recurrent neural networks or ARIMA model.

Conclusion

Free-Form Visualization

As we can see from next slides gained model from project is able to predict different stock prices with accuracy 5%.

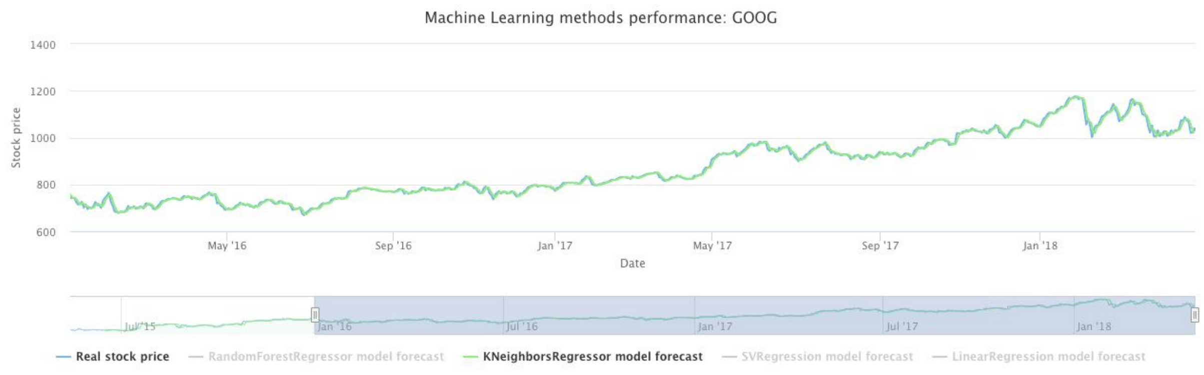

Fig. 6 - Comparison of predicted values with real prices of GOOG stocks.

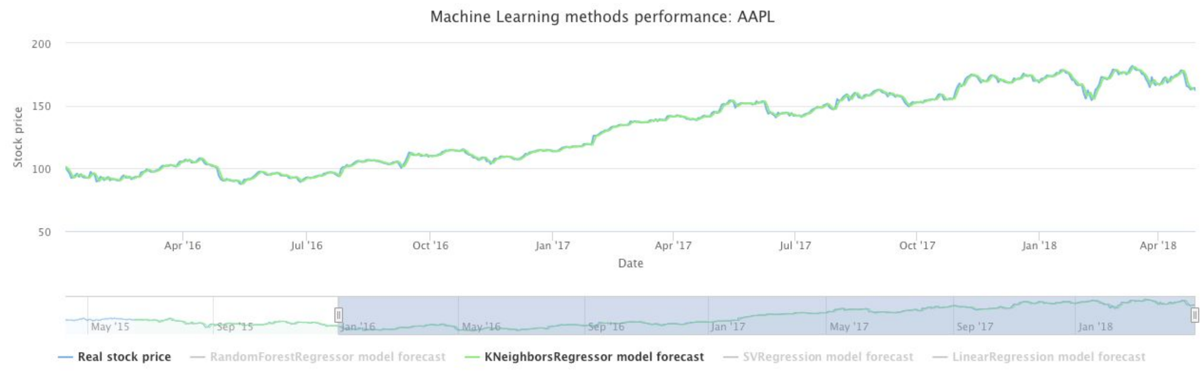

Fig. 7 - Comparison of predicted values with real prices of AAPL stocks.

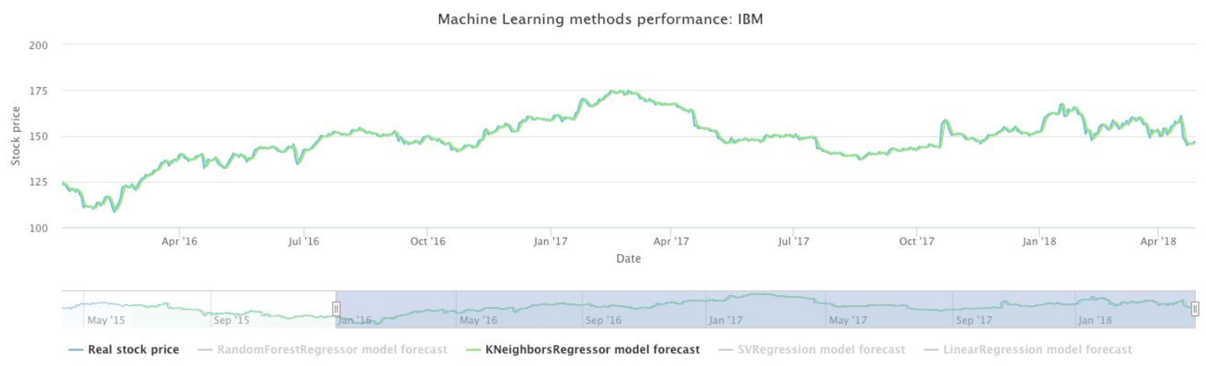

Fig. 8 - Comparison of predicted values with real prices of IBM stocks.

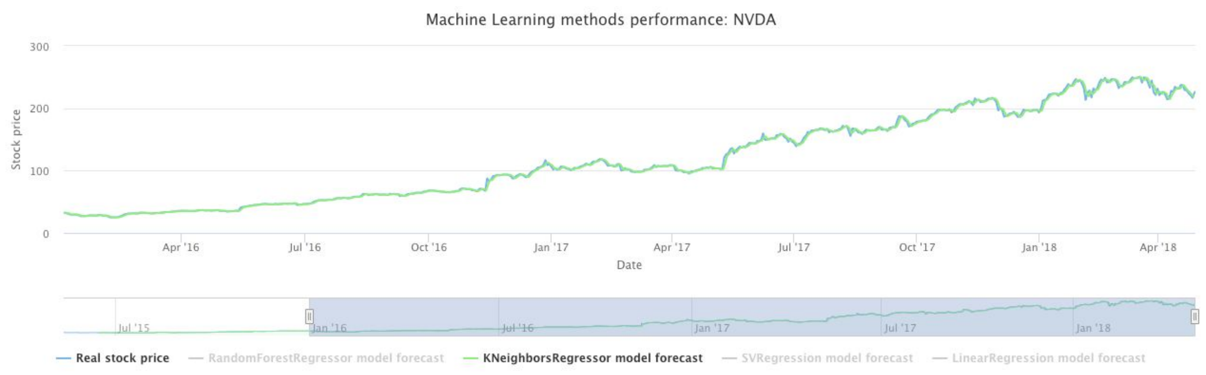

Fig. 9 - Comparison of predicted values with real prices of NVDA stocks.

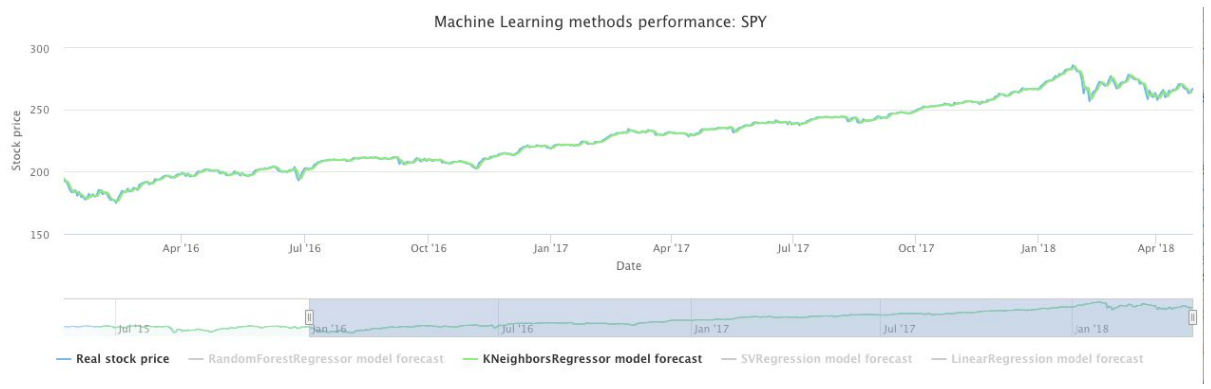

Fig. 10 - Comparison of predicted values with real prices of SPY stocks.

Reflection

As we can see stock price prediction problem can be presented as regression problem. For solving this problem machine learning regression algorithm can be used. Based on stock price behavior and math behind regression method optimal solution can be found for common stock prices and if required individually for

specific stock price. The important thing to remember that result depends on data, how it was obtained, preprocessed and spited for cross-validation to get best result.
Difficult point of this project was absence of any other data which could affect stock price (features), so stock price prediction was more like data pattern detection. And it was interesting to implement with real web infrastructure and provide prediction as a service with set of stock. Final model has shown good result, but as expected there is deviation between real and predicted values. This difference can be reduced by improving model, which will be discussed next.

Improvement

As discussed before accuracy and performance improvement can be gained by using other algorithms. Regression machine learning algorithms like AdaBoostRegressor, DecisionTreeRegressor doesn't seem show better results. At the same time seems very hard to gain better accuracy for selected regression algorithms. The main problem behind mentioned algorithms that they can't catch and predict quick changes and some long seasonal trends. From this perspective more sophisticated algorithms can improve results. For example neural networks can be used with different architectures. For more advanced case to catch seasonal changes ARIMA model can be used or recurrent neural networks. That's models definitely improve result of final model.