# QuantFy: Predictive Market Behavior App using Machine Learning

Balagangadhar Bathula

March 12, 2017

**Abstract**

Investment firms, hedge funds and even individuals have been using financial models to better understand market behavior and make profitable investments and trades. To maximize return of investment these firms use large amounts of historical data and apply machine learning algorithms to the process. QuantFy is an application (app) that would build predictive/optimized models using machine learning models such as linear regression, random forests and $k$-nearest neighbour.

# Contents

This is description document for the Capstone project for Machine Learning Nanodegree program of Udacity.

# 1 Domain Background

Many financial, high frequency trading (HFT), and hedge fund companies are analyzing trading strategeis including algorithmic steps from information gathering to market orders using machine-learning approaches. This project proposes to apply these probabilistic techniques to make trading decisions. Using approaches such as linear regression, Q-learning, $k$-NN and regression trees and apply them to the actual trading situations.

This application software will be build using Python tools [Hil]. The book [RB] provides a comprehensive overview of domain knowledge that would be required to understand the financial market world. Also the speicifications given [Bal16] will be used as guidelines to build this application.

# 2 Problem Statement

Using the historical data for prices and performance statistics as features, we can get predicted future price of the stock. For instance we can set of performance statistics (such as Bollinger bands, price-to-earning (P/E) ratio) as features of the present date mapped to price (say one week forecast which is an input) of price. This creates a data set $< X, Y >$, where $X$ is the set of features and $Y$ as trading days. Features used are the measurable quantities that a particular stock could use in the predicting things such as change in price, market relative change in price or simply future stock price. Few questions that one would ask, before getting started are-

1. Breath and depth of data: How much historical data one would like to consider?
2. What ticker symbols are you going to use?

General machine learning models that can be used for this problem are:

1. Regression
2. $k$-nearest neighbors
3. Random Forests
4. Time-series analysis: This analysis will be used to estimate the commodity prices such as oil/gas etc. For instance we can observe long term trend in the time-series (such as exponential) and fit an exponential model to predict the future price. Using the concepts of drift and seasonality, build models for commodity stocks.

# 3 Datasets and Inputs

There are many open APIs to extract data:

1. Yahoo Finance:
   (a) Using the http://ichart.finance.yahoo.com/table.csv?s={YOUR_SYMBOL} we can query and get historical data.
   (b) We can also use yahoo_finance python library or,
   (c) We can also use pandas_datareader
2. Quandl: Using quandl python API we can get the historical data
3. Bloomberg API

Current version of QuantFy application proposes to uses Yahoo Finance source and Quandl to get historical data. One can extract more than 10 years of historical stock prices for many ticker symbols. This data is obtained using appropriate API calls to the sources (listed below).

For instance user can the date range for viewing the stock prices of ticker symbols. Then an request call will be made and accquired data will be converted into a `pandas` dataframe. This data will also be stored locally, using python libraries such as `ediblepickle`, so when the same request is made (no change in the parameters), data is retrived locally. Since the data is generated online (rather than locally avaiable data), we are not particularly concerned about the storing the data. At least from the past experinces, the stock price data can be easily stored in the dataframe without any memory errors. We anticipate that the historical prices for the each ticker symbol will be in the order of less than 1 MB.

1. Close price and/or Adjusted close price
2. Start price and/or Adjusted start price,
3. Opening price and/or adjusted opening price

The data from these sources consists of features like:

Snapshot of the plot prices for IBM, TSLA (Tesla) and AAPL (Apple) are shown in Figure 1, 2, and 3 respectively. For more detailed refer to the application website **QuantFy** [Bat17c].

Along with these features, one could compute other features like:

1. Daily returns: How much the price will go up or down on a particular instance of time:

$$r(t) = \frac{p(t)}{p(t-1)} - 1$$

   where $p(t)$ is the price at time $t$. Similarly, we can compute

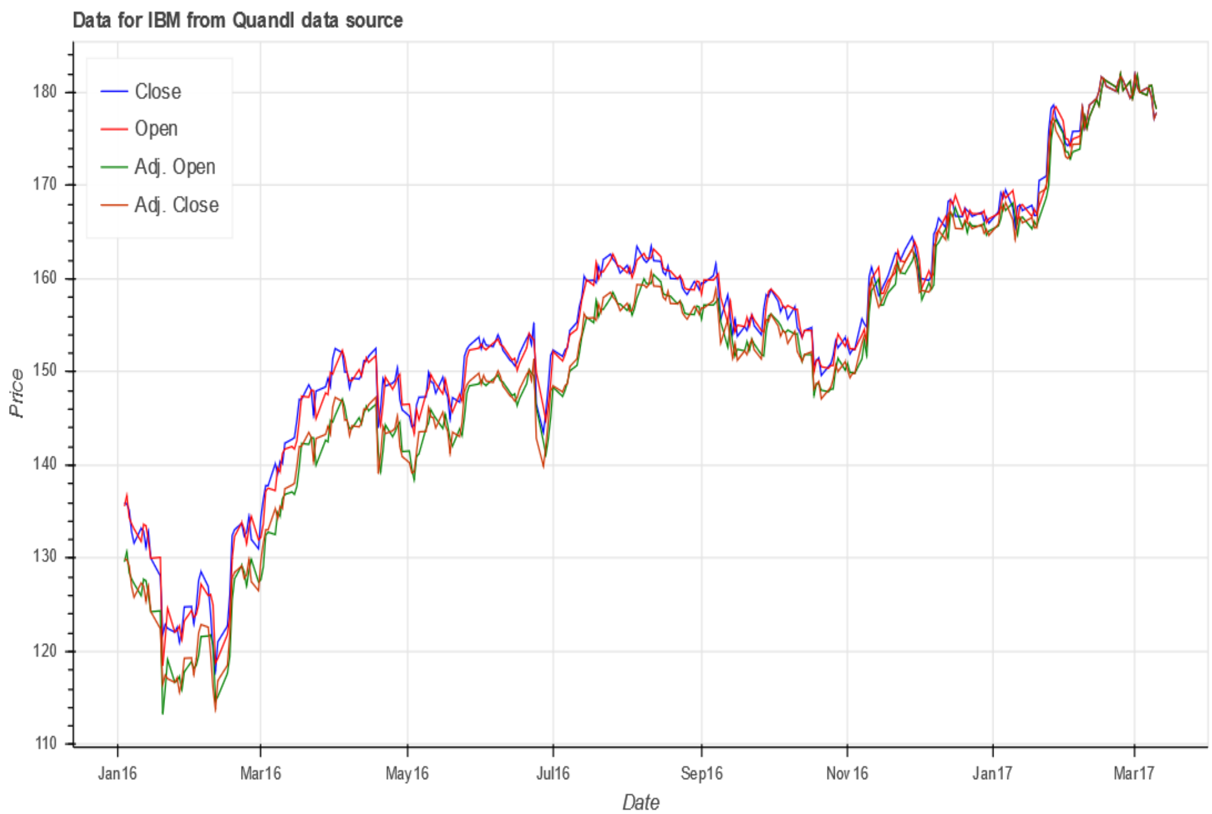


**Figure 1:** IBM prices from Jan 1 2016, from Quandl data source

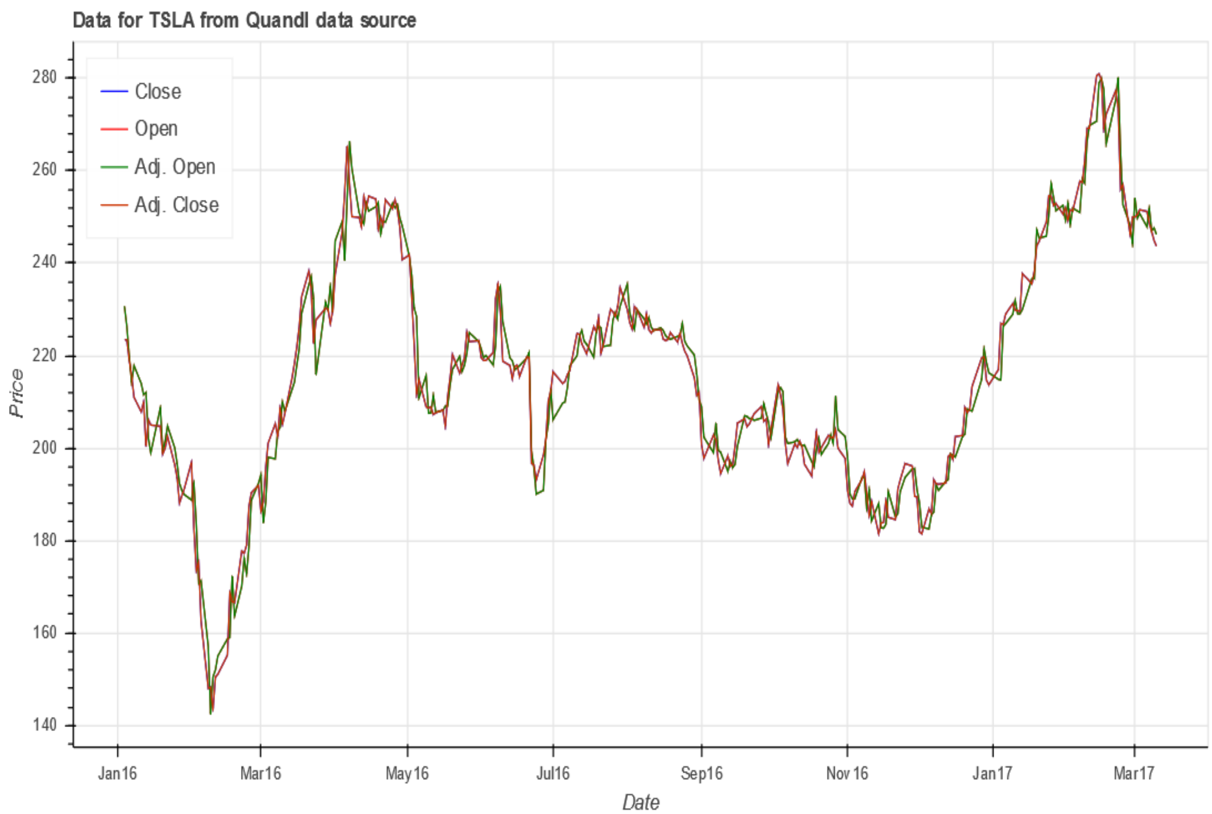


**Figure 2:** Tesla prices from Jan 1 2016, Quandl data source

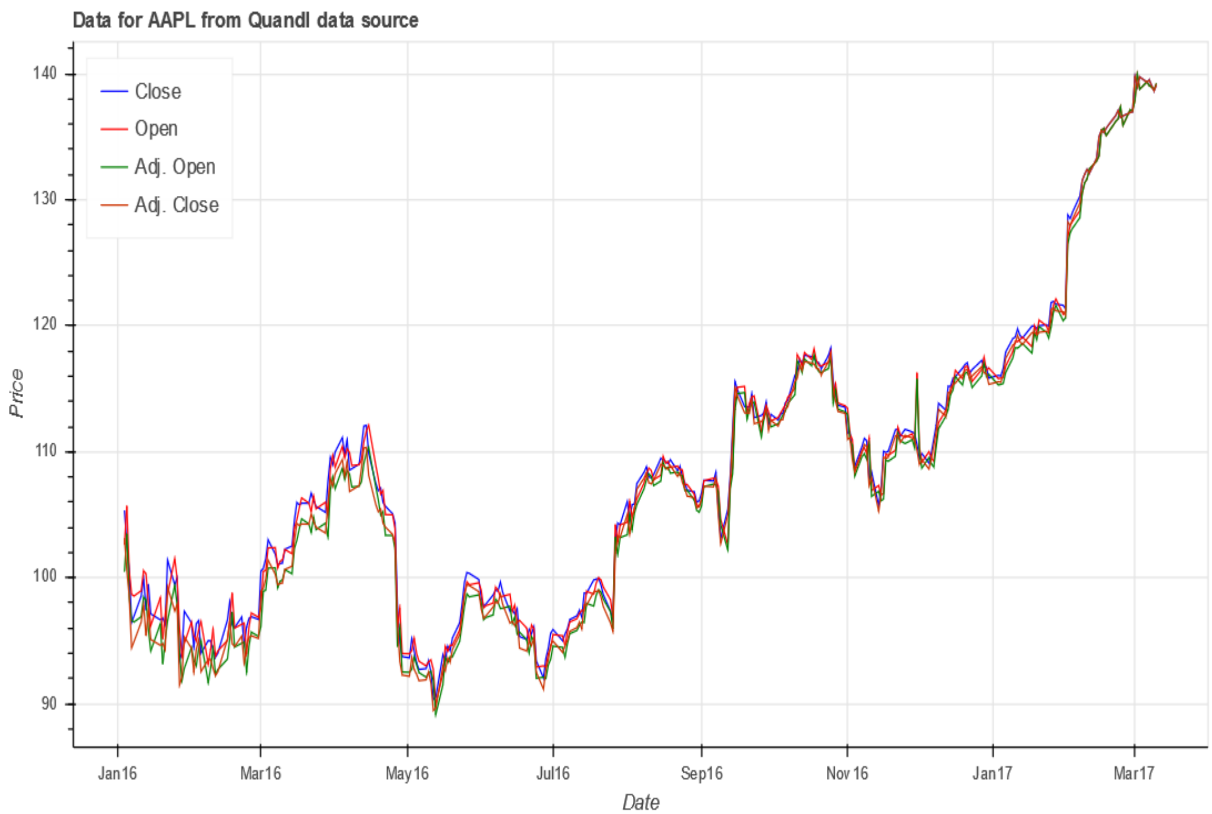


**Figure 3:** Apple prices from Jan 1 2016, Quandl data source

2. Cumulative return is given by,

$$cr(t) = \frac{p(t)}{p(0)} - 1$$

3. Simple moving average (SMA): Also know as moving average

```
rolling_df=pd.DataFrame(index=df.index)
rolling_df =pd.Series.rolling(df,window=window).mean().to_frame()
rolling_df.ix[:window, :] = 0
```

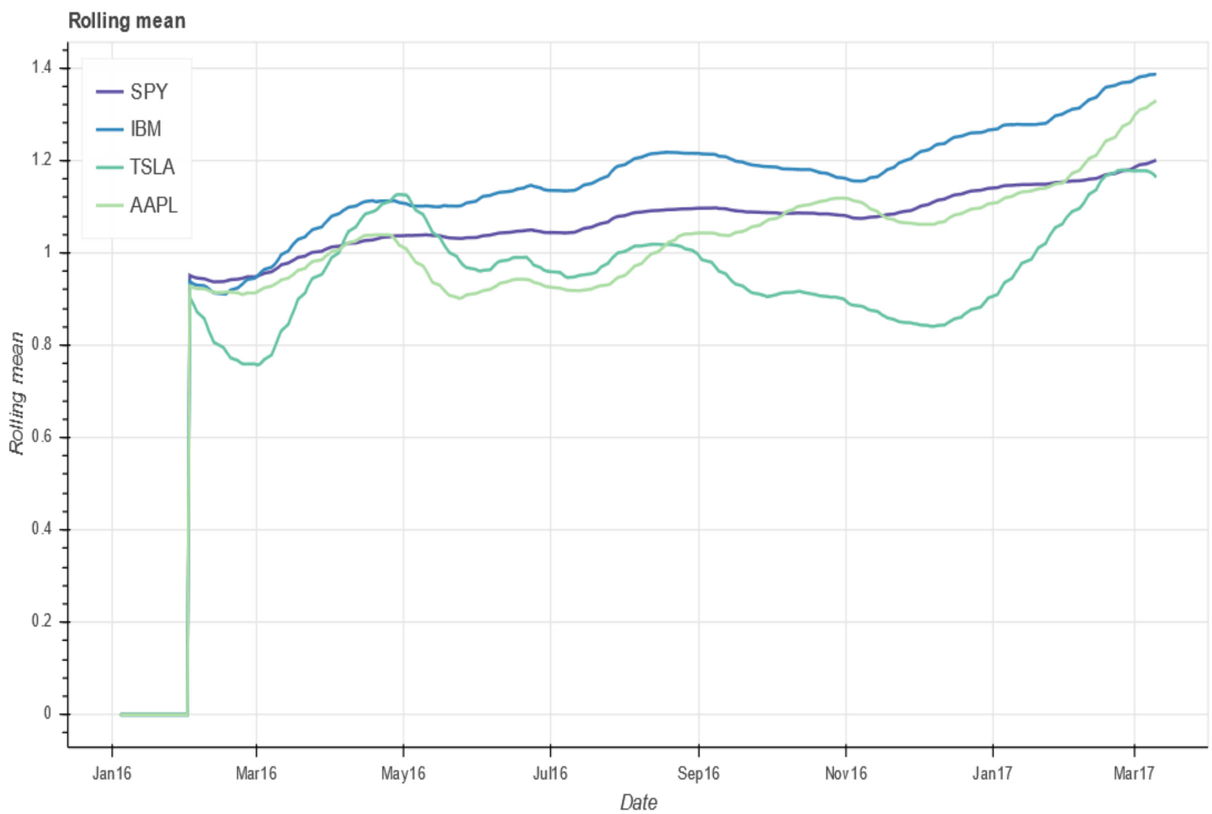


**Figure 4:** Rolling mean for the symbols IBM, TSLA and AAPL

4. Bollinger bands:

$$U_{band} = m_r + 2 \times \sigma_r,$$

$$L_{band} = m_r - 2 \times \sigma_r$$

5. Sharpe ratio This metric adjusts the return for risk, also know as risk adjusted reward.

$$sr = \frac{R_p - R_f}{\sigma_p},$$

where $R_p$ is the portfolio return, $R_f$ is risk-free return and $\sigma_p$ is the standard deviation of portfolio return.

$$sr = \frac{E[R_p - R_f]}{\sigma[R_p - R_f]}$$

Sharpe ratio varies depending on how frequently you sample the data. $sr_f = k \times sr$, where $k = \sqrt{\# \text{ samples/year}}$

6. Violatility is nothing but the standard deviation of the daily returns.

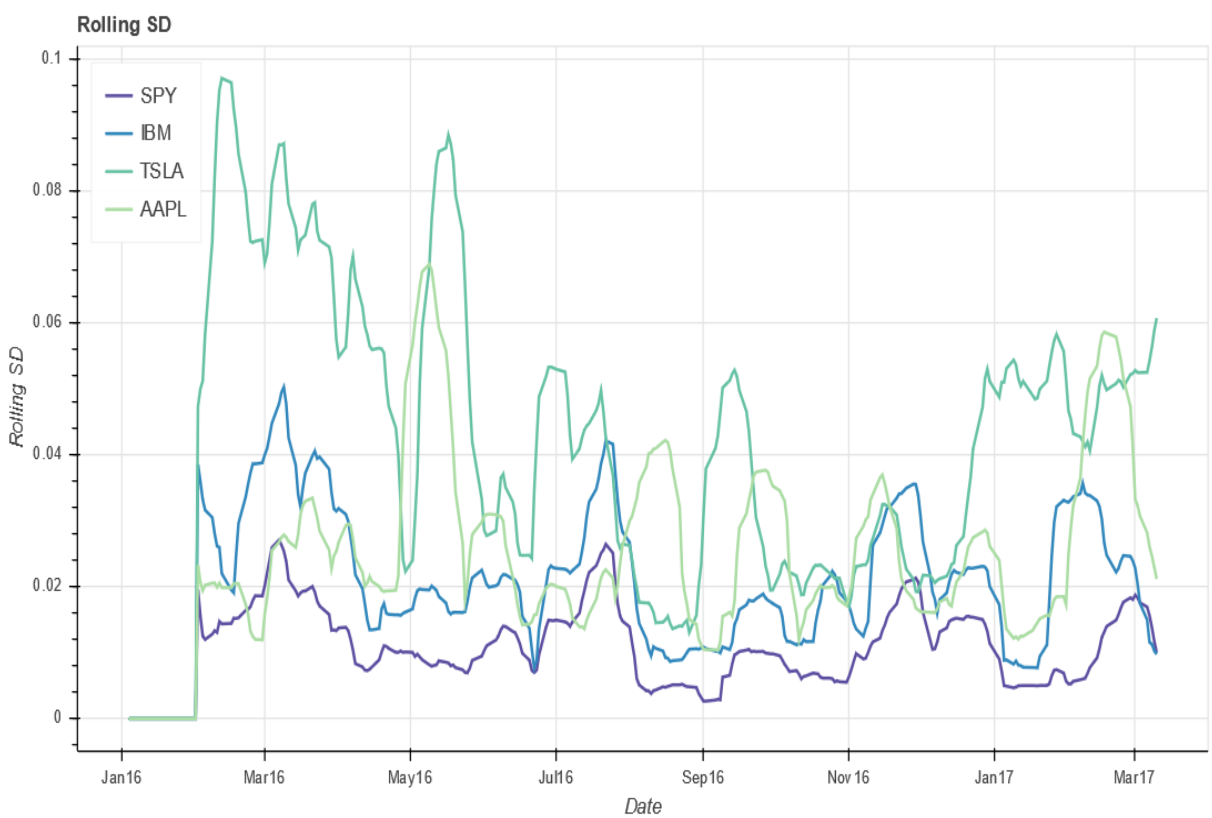


**Figure 5:** Rolling standard deviation ($\sigma$) for the symbols IBM, TSLA and AAPL

7. Momemtum is defined as:

$$r(t) = \frac{p(t)}{p(t-N)} - 1$$

where $N$ is the size of the window.

Below are some of the plot obtained for IBM, TSLA and AAPL are shown in Fig. 4, 5, and 6 respectively. For more details, refer to the application website [Bat17c]

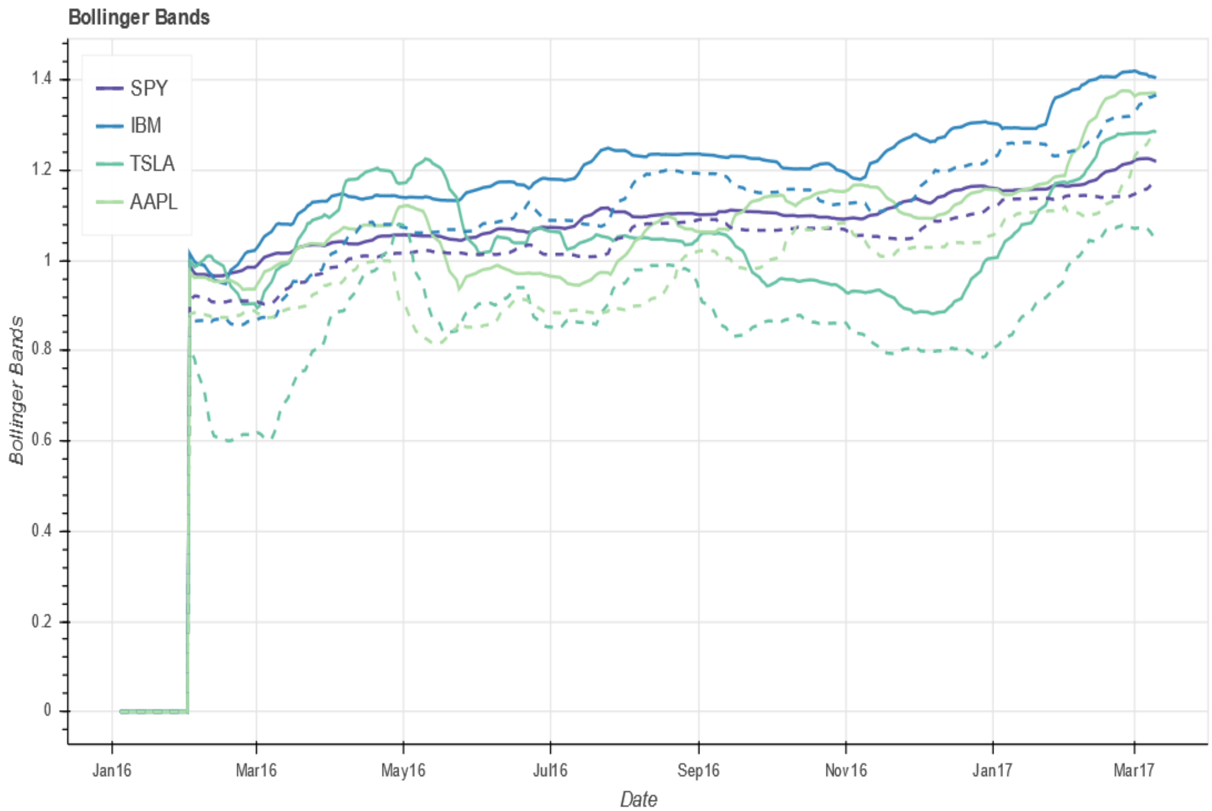


**Figure 6:** Bollinger bands IBM, TSLA and AAPL; upper are shown in solid lines and lower bands are shown in dotted lines

# 4 Portfolio optimization

- Given set of assets and time-period, find allocations of funds to assets that maximizes performance.
- Performance metrics can be:
  - $\max cr(t)$,
  - $\min \sigma(t)$, i.e., minimizing volatility.
  - $\max sr$
- Framing the problem:

1. Provide a function $f(X) = cX + b$ to minimize.
2. Provide an initial guess for allocations.
3. Call the optimizer.

- Constraints: We know that allocations for each stock symbol in the portfolio should be $\leq 1$, i.e.,
$$\Sigma_{i=0}^{k}|X_i| = 1.0,$$
where $X_i$'s are allocations for each stock.

This problem is a convex optimization problem and can be solved using the numerical optimization algorithms. Below is the snippet of the optimization code in `scipy`.

```
import scipy.optimize as spo
initial_alloc=[(1.0/len(sym))]*len(sym)
bounds=((0,1),)*len(sym)
opt_allocs=spo.minimize(sharpe_function*-1, initial_alloc,
        args=df, method='SLSQP',bounds=bounds,
        constraints=({'type':'eq','fun':lambda
        opt_allocs: 1-np.sum(np.abs(opt_allocs))}
        )
        );
optimal_allocs=opt_allocs.x
```

Portfolios in comparision with SP&500 for the symbols, IBM,TSLA and AAPL for duration between Jan 1 2016 to March 12 2017, for both unoptimized and optimized are shown in Fig. 7 and 8 respectively.

# 5 Market Simulator

Based on the orders file, the portfolio value can be analyzed. For instance if BUY or SELL order comes, depending on the stock price value of the order, the portfolio value changes. Using the leverage threshold, orders can be executed or fail to execute if the leverage value exceeds the threshold.

Leverage value is defined as:

$$leverage = \frac{\Sigma|sp|}{(\Sigma(sp) + cash)}$$

,

where $sp$ is defined as all-stock positons [Bal16].

The Fig. 9 is generated using orders file located at **Orders-demo**


**Figure 7:** Optimized portfolio for IBM, TSLA and AAPL in comparision with SP&500


**Figure 8:** Unoptimized portfolio for IBM, TSLA and AAPL in comparision with SP&500


**Figure 9:** Market simulator for a order file. The green vertical lines indicate a BUY order and RED vertical lines indicate a SELL order

## 6 Solution Statement

Predicts the stock prices and also measures the accuracy of these forecasts. Using backtesting a method where we roll back time and measure the accuracy of these forecasts. Slice the sample data from (historical) training set and apply the machine learning models to get the forecast. By comparing the predictive results of the model against the historical results, backtesting can determine whether the model has predictive value.

## 7 Benchmark Model

As a benchmark model, one could use a simple mean-model. For instance, a paticular stock price can be predicted using mean of the ajusted close price. Also, algorithms such as $k$-nearest neighbours etc., can be applied using default parameters in `scikit-learn` and then using `GridsearchCV` one could obtain optimal parameter setting. In other words, benchmark models can be based on simple parameter settings. Performance of advanced algoithms such as ensemble approaches can be tested against these benchmark models.

## 8 Evaluation Metrics

Here we choose two evaluation metrics:

Evaluation metrics can include root-mean-square error (RMSE), correlation coefficient.

- **RMSE**: Mean Absolute Error (MAE), is calculated as the ratio of sum of absolute values of difference between actual value and the predicted value to number of samples. Clearly this metric indicates the average deviation of a predicted value from its actual value. This is a linear function and all errors are weighted equally.

  On the other hand mean-square-error (MSE), is ratio of sum of squared deviations to the number of samples, i.e., it is the mean of the squared deviation. This is a quadratic function, and if the errors are large, they are weighted more while the small errors are weighted less (say if absolute error for a sample point is less than 1, then square of the number is much smaller). Thus in a MSE based estimator, we can clearly see how the predicted values compared to the true values. MSE would be preferred performance metric over MAE especially if you want to have a model, where large errors are particularly undesriable. Also by taking the square-root of the MSE value we get the Standard deviation, which is again helps to predict the variablity of the data.

  For this problem, minimizing the metric RMSE makes the stock price prediction accurate.

- **Correlation**: This metric can be used to evaluate ML algorithm is to look at the relationship between predicted value vs. actual value. Let say we have a test data, and we run our model again test data to get $Y_{predict}$, which is the stock price in this case. We can now compare the $Y_{true}$ against $Y_{predict}$. By looking at the relationship between $Y_{true}$ and $Y_{predict}$ we say if the model is correctly predicting the values. This metric is the correction function $\rho$ and it ranges between $-1 \leq \rho \leq +1$. If $\rho = 1$, then they are strongly correlated; if $\rho = -1$ they are negatively correlated and if $\rho = 0$, they are not correlated.

# 9 Project Design

Design phase of the QuantFy application consists of following phases:

- Time-series plots for the various stocks can be visualized at this link: Price plot [Bat17b]. This link also computes various other technical indicators such as the SMA, Bollinger bands, etc.

- Portfolio optimizer (First release): Analyzing using stock price data, and using numerical optimization techniques, this has already been implemented in the app. Link to the app is **Portfolio optimizer** [Bat17a]

- Investment and Trading using machine learning (Second release): This was be built using the machine-learning stratagies such as (1) $k$-nearest neighbours (2) Random forests (3) Linear regression and (4) Ridge regression. One can compare the performance of any two algorithms in the application. For more details, visit the application at **ML models**

  Left side of the RED dotted line indicates the past values, while the right side are the predicted values of the stock.

- Commodity stocks (as future work)

- Reinforcement Learning (future work)

**Software**

1. Front-end: Python `flask` based web application

2. Back-end: Python libraries, such as `scikit-learn, pandas, numpy, scipy, bokeh`.

3. Hosting platform: Heroku

The first version of the application can be found at: **QuantFy** [Bat17c]. The cuurent version of the source code for this application can be found at this GitHub link: **Source code** [Bat17d]

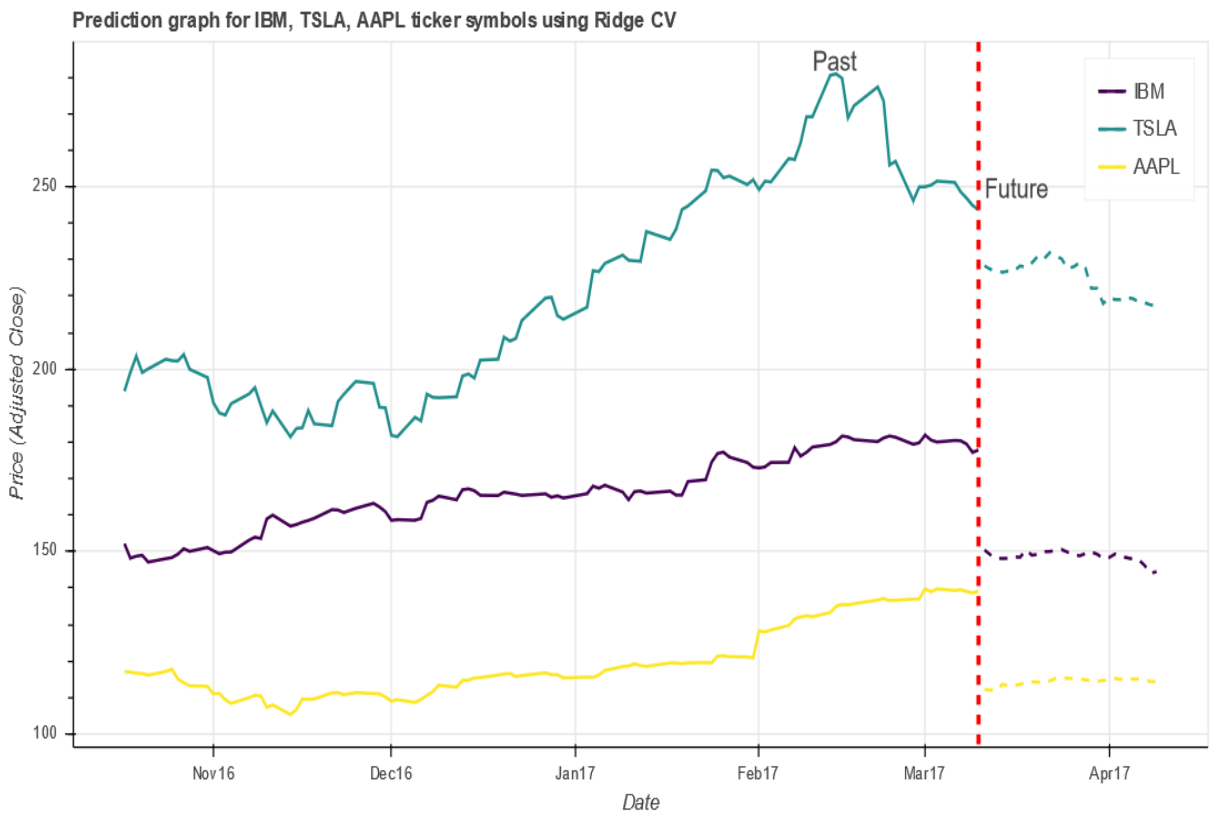


**Figure 10:** Prediction values for 30 days for IBM, TSLA and AAPL using Ridge regression

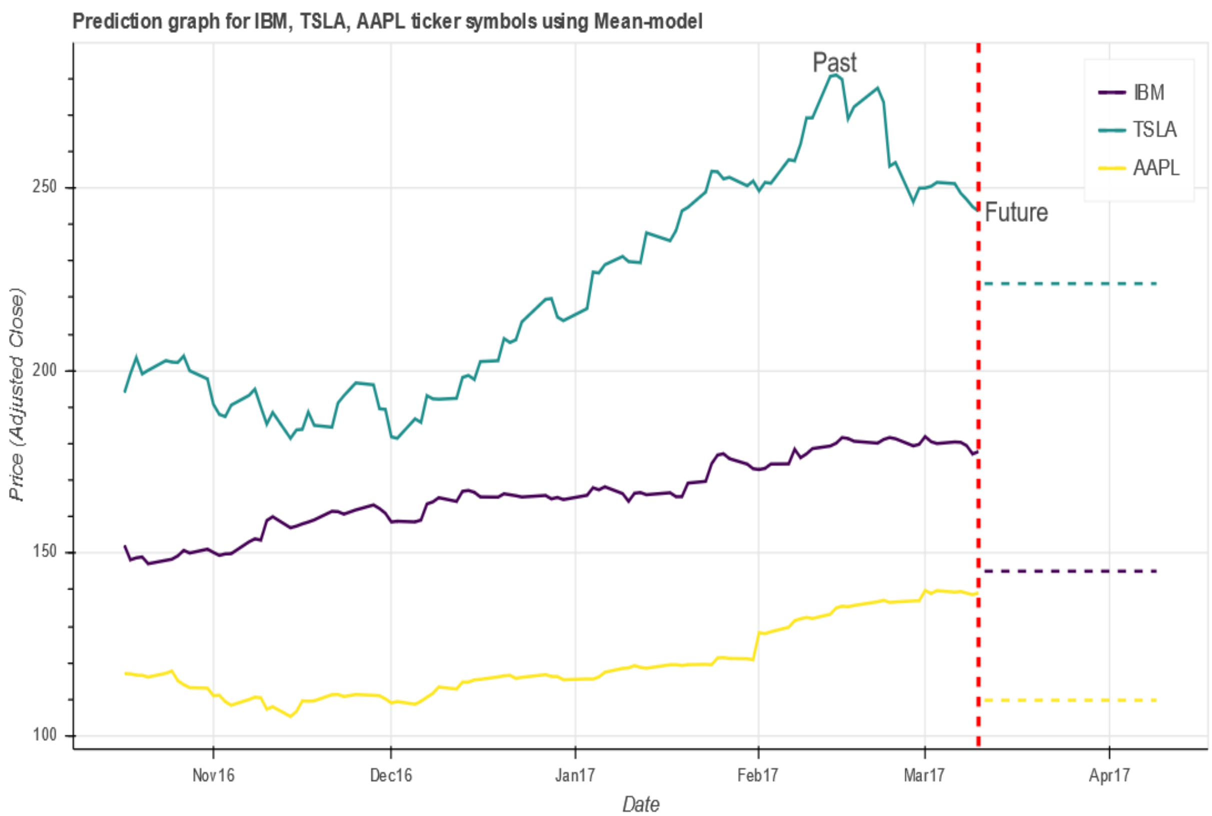


**Figure 11:** Prediction values for 30 days for IBM, TSLA and AAPL using the benchmark Mean model.

# References

[Bal16] Tucker Balch. Machine learning for trading. Udacity course, 2016.

[Bat17a] Balagangadhar Bathula. Portfolio optimizer: QuantFy. http://quantfy.herokuapp.com/portfolio, 2017. [Online].

[Bat17b] Balagangadhar Bathula. Price Plot: QuantFy. http://quantfy.herokuapp.com/price_plot, 2017. [Online].

[Bat17c] Balagangadhar Bathula. QuantFy. http://quantfy.herokuapp.com/, 2017. [Online].

[Bat17d] Balagangadhar Bathula. Source code: QuantFy. https://github.com/beegeesquare/QuantFy, 2017. [Online].

[Hil] Yves Hilpisch. *Python for Finance*. O'Reilly.

[RB] Philip J. Romero and Tucker Balch. *What Hedge Funds Really Do: An Introduction to Portfolio Management*. Business Expert Press.