Stock Predictions Using Machine Learning & Python

Xuan He

Alex Koynoff

Prathyusha Challa

Zelalem Kebede

Itunu Oyeyipo

Project Overview

- Use Machine Learning to predict stock prices
- ML models selected:
  - scikit-learn(sklearn) - Linear Regression
  - scikit-learn(sklearn) - Support Vector Regression (SVR)
- Picked 10 random stocks and loaded 12 months of historical prices
- Predicted 30 trading days and compared to the real prices
  - Simulations for each model ran over 200 times. Took the average of all simulations for the visualization
- Tools used:
  - Excel
  - Pandas
  - Matplotlib
  - Tableau

- Review and visualize the results
- Summary and commentary

Machine Learning Models Used

- **Sklearn - Linear Regression (LR) (from sklearn.linear_model import LinearRegression)**

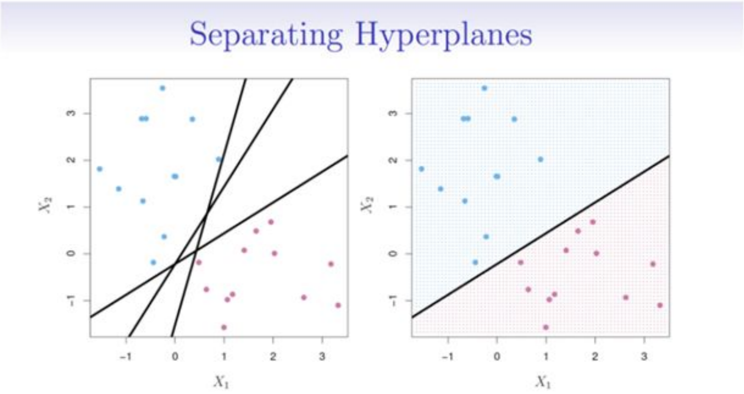
  Overview
  - Searching for a relationship among variables
    - Dependent variables (y) or responses
    - Independent variables (x) or predictors

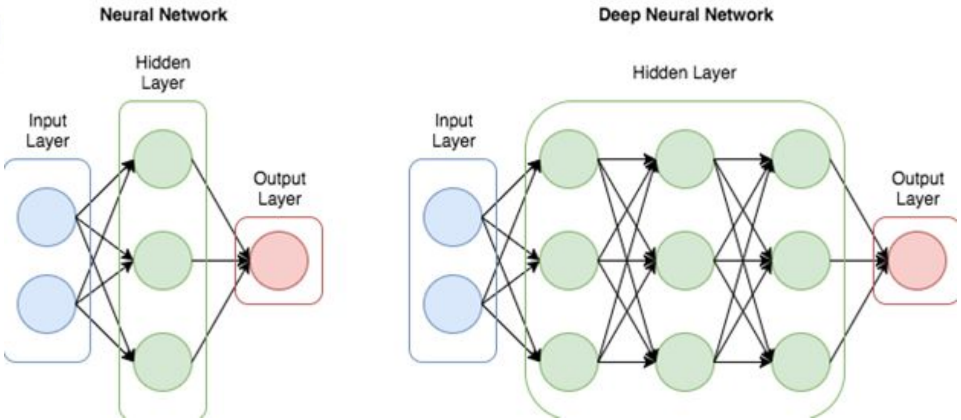
  Pros
  - One of the most widely used as it is easier to interpret the results
  - Works on any size of dataset
  - Gives information about relevance features

  Cons
  - Conversely, it assumes there is a straight-line relationship between dependent and independent variables which is sometimes incorrect
  - It can be very sensitive to outliers or anomalies
  - The algorithm assumes data is normally distributed in real when they are not.
  - It is not recommended for most practical applications because it oversimplifies real world problems.

Machine Learning Models Used

- **Sklearn - Support Vector Machine (from sklearn.svm import SVR)**
  Overview
  - A linear model for classification and regression problems
  - SVM algorithm for pattern recognition
  - The algorithm creates a line or hyperplane which separates the data into classes
  - The basic idea of SVM:
    - Just like 1-layer or multi-layer neural nets



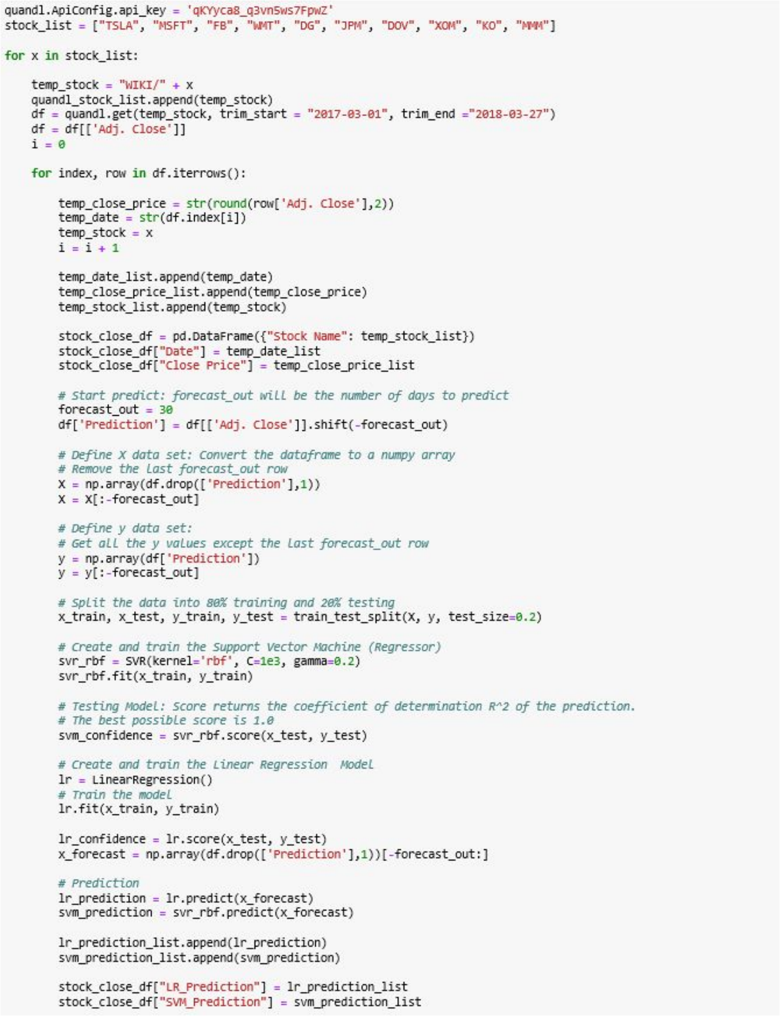


Machine Learning Models Used

- **Sklearn - Support Vector Machine (from sklearn.svm import SVR) cont:**

  Pros
  - Works well on smaller cleaner datasets
  - It can be more efficient because it uses a subset of training points
  - Helps analyse data and identify trends.
  - Compute the multiplication of independent variables
  - Require less training when dealing with smaller datasets
  - No distribution requirements

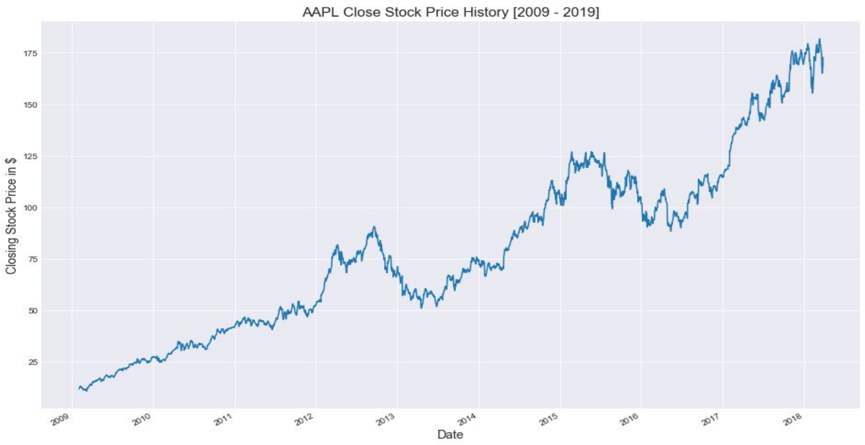
  Cons
  - Isn’t suited to larger datasets as the training time with SVMs can be high
  - Less effective on noisier datasets with overlapping classes
  - Can be inefficient to train, memory intensive and annoying to run and tune

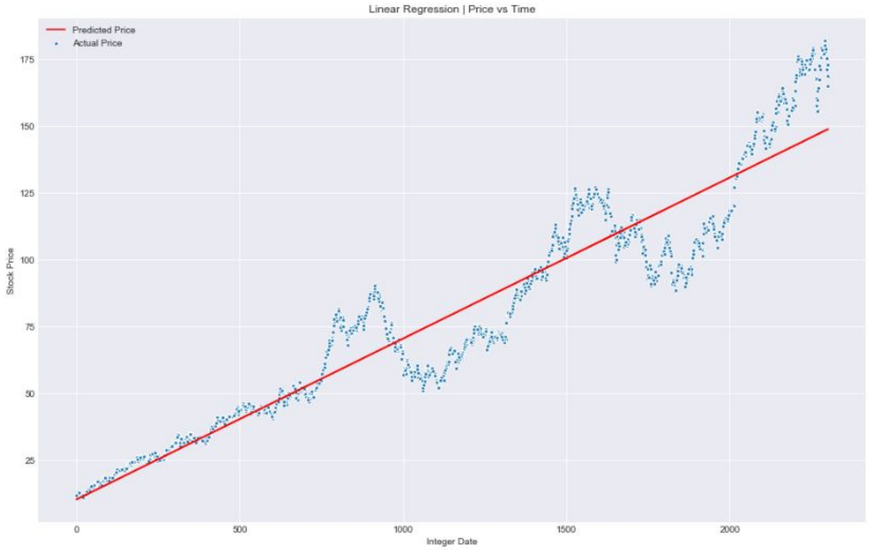


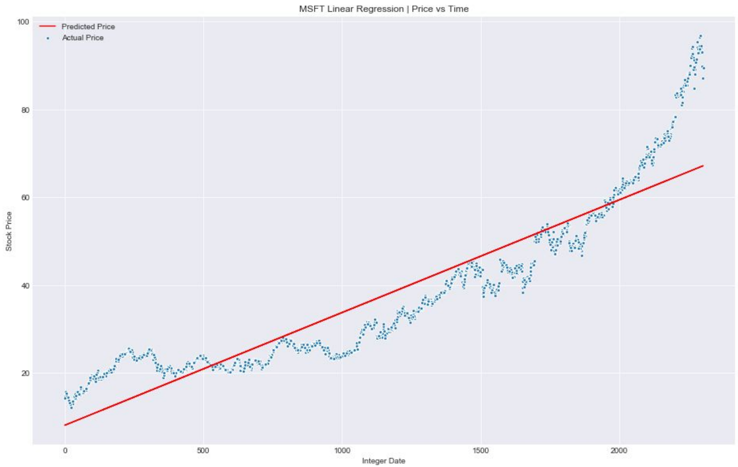
```python
quandl.ApiConfig.api_key = 'qKYyca8_q3vn5ws7FpwZ'
stock_list = ["TSLA", "MSFT", "FB", "WMT", "DG", "JPM", "DOV", "XOM", "KO", "MMM"]

for x in stock_list:

    temp_stock = "WIKI/" + x
    quandl_stock_list.append(temp_stock)
    df = quandl.get(temp_stock, trim_start = "2017-03-01", trim_end ="2018-03-27")
    df = df[['Adj. Close']]
    i = 0

    for index, row in df.iterrows():

        temp_close_price = str(round(row['Adj. Close'],2))
        temp_date = str(df.index[i])
        temp_stock = x
        i = i + 1

        temp_date_list.append(temp_date)
        temp_close_price_list.append(temp_close_price)
        temp_stock_list.append(temp_stock)

        stock_close_df = pd.DataFrame({"Stock Name": temp_stock_list})
        stock_close_df["Date"] = temp_date_list
        stock_close_df["Close Price"] = temp_close_price_list

        # Start predict: forecast_out will be the number of days to predict
        forecast_out = 30
        df['Prediction'] = df[['Adj. Close']].shift(-forecast_out)

        # Define X data set: Convert the dataframe to a numpy array
        # Remove the last forecast_out row
        X = np.array(df.drop(['Prediction'],1))
        X = X[:-forecast_out]

        # Define y data set:
        # Get all the y values except the last forecast_out row
        y = np.array(df['Prediction'])
        y = y[:-forecast_out]

        # Split the data into 80% training and 20% testing
        x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

        # Create and train the Support Vector Machine (Regressor)
        svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.2)
        svr_rbf.fit(x_train, y_train)

        # Testing Model: Score returns the coefficient of determination R^2 of the prediction.
        # The best possible score is 1.0
        svm_confidence = svr_rbf.score(x_test, y_test)

        # Create and train the Linear Regression  Model
        lr = LinearRegression()
        # Train the model
        lr.fit(x_train, y_train)

        lr_confidence = lr.score(x_test, y_test)
        x_forecast = np.array(df.drop(['Prediction'],1))[-forecast_out:]

        # Prediction
        lr_prediction = lr.predict(x_forecast)
        svm_prediction = svr_rbf.predict(x_forecast)

        lr_prediction_list.append(lr_prediction)
        svm_prediction_list.append(svm_prediction)

        stock_close_df["LR_Prediction"] = lr_prediction_list
        stock_close_df["SVM_Prediction"] = svm_prediction_list
```

Code

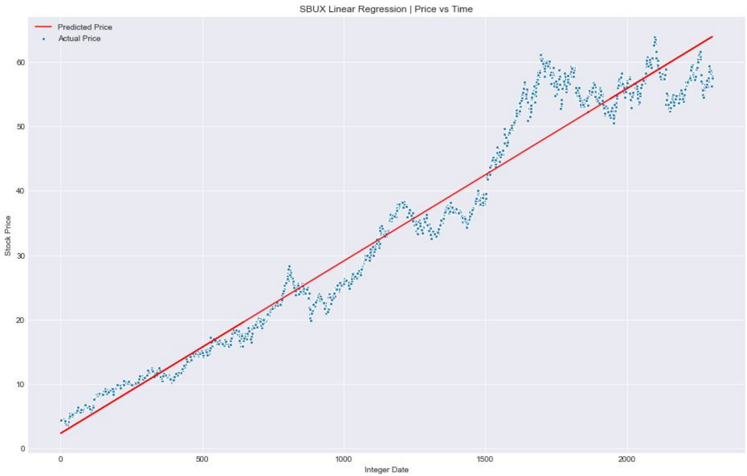
- Selected 10 random stocks
- Ran 12 months of closing prices from Quandl API
- Looped through the list of stocks
    - Selected 30 days of prediction
    - Each model included within the loop
        - Linear
        - SVR (Support Vector Regression)
    - Information put into lists and exported to csv for further manipulation and Tableau visuals.


AAPL Close Stock Price History [2009 - 2019]
Closing Stock Price in $
Date
25
50
75
100
125
150
175
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018

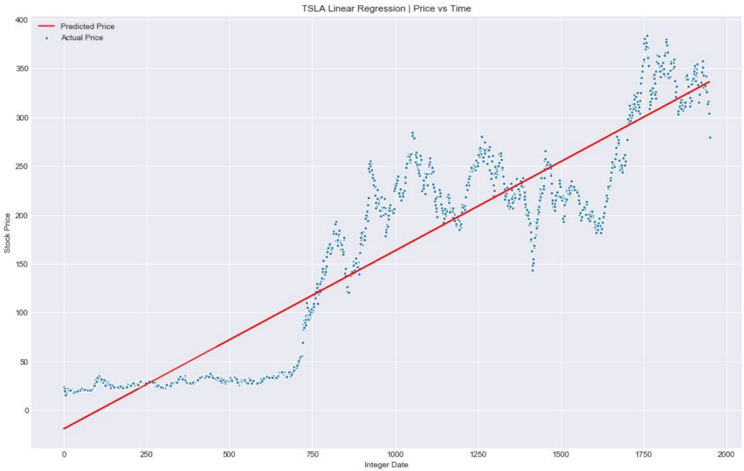

Linear Regression | Price vs Time
Predicted Price
Actual Price
Stock Price
Integer Date
25
50
75
100
125
150
175
0
500
1000
1500
2000



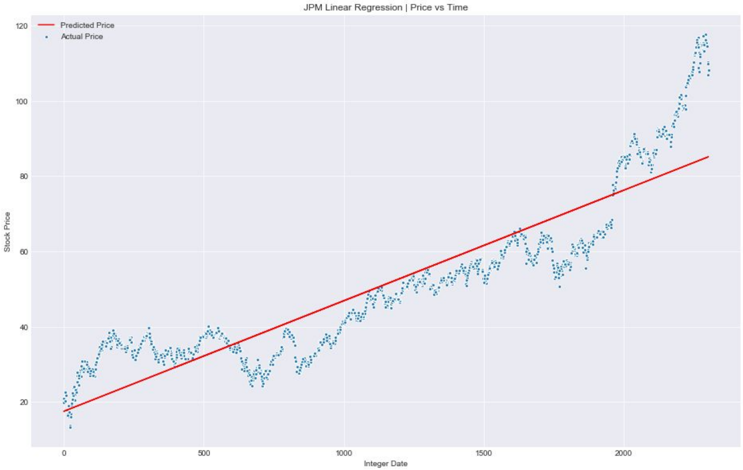
MSFT Linear Regression | Price vs Time



SBUX Linear Regression | Price vs Time



TSLA Linear Regression | Price vs Time



JPM Linear Regression | Price vs Time

Visualization

CLICK ME!!!

Limitations and Future Considerations

- Limitations:
    - Only one variable used (historical closing prices)
    - Probably neither Linear Regression or SVR is a good enough model to predict stock prices

- Future Considerations:
    - Add other factors that impact stock prices to be built into the model
        - News
        - Industry specific trends
        - Company specific information
        - Fundamental analysis
            - Company financials, ratios, etc
        - Technical analysis
            - Moving averages, price patterns, charting techniques, etc
        - Build and test with other models and compare results
            - Deep Learning - Long Short Term Memory Network (LSTM)

Summary

- **<u>Do not</u> depend on our model to trade stocks! You will lose money!**
- **<u>Do not</u> totally trust any prediction online! Make your own decision!**
- Add more variables for consideration in models to make them more accurate
- Use different models and compare results
- Use at your own risk!

Q & A

(or we can take a look at what ML can do and see few top trending stocks suggested by ML)

What Machine Learning Can and Cannot Do ?

- Learning a function that maps well-defined inputs to well-defined outputs
- Large (digital) data sets exist or can be created containing input-output pairs
- No long chains of logic or reasoning that depend on diverse background knowledge or common sense
- The task provides clear feedback with clearly definable goals and metrics
- No need for detailed explanation of how the decision was made
- A tolerance for error and no need for provably correct or optimal solutions
- The phenomenon or function being learned should not change rapidly over time

THE WALL STREET JOURNAL.

Five of the Best Stocks to buy based on **US News**

- Advanced Micro Devices (AMD)
- Chipotle Mexican Grill (CMG)
- Nike (NKE)
- Canopy Growth Corp. (CGC)
- Microsoft Corp. (MSFT)

*** Disclaimer : If you happen to make money buying the suggested stocks, Venmo here ⇒***