In 12 minutes: Stocks Analysis with Pandas and Scikit-Learn

Analyse, Visualize and Predict stocks prices quickly with Python

Vincent Tatan Follow

May 26, 2019 · 12 min read ★

Predicting Stocks with Data Analysis

One day, a friend of mine told me that **the key to financial freedom is investing in stocks**. While it is greatly true during the market boom, it still remains an attractive options today to trade stocks part time. Given the easy access to online trading platform, there are many self made value investors or housewife traders. There are even success stories and advertisements which boast "*Get Rich Quick Scheme*" to learn how to invest in

stocks with a staggering return of 40% and even more. Investing has become the boon for the working professionals today.

The question now are: Which stocks? How do you analyse stocks? What are the returns and risks of this stocks compared to its competitors?

The objective for this publication is for you to understand one way on analyzing stocks using quick and dirty Python Code. Just spend 12 minutes to read this article — or even better, contribute. Then you could get a quick glimpse to code your first financial analysis.

To start learning and analyzing stocks, we will start off by taking a quick look at the historical stocks prices. This will be done by extracting latest stocks data from pandas web-data reader and Yahoo Finance. Then we will try to view the data through exploratory analysis such as correlation heatmap, matplotlib visualization, and prediction analysis using Linear Analysis and K Nearest Neighbor (KNN).

Loading YahooFinance Dataset

Pandas web data reader is an extension of pandas library to communicate with most updated financial data. This will include sources as: Yahoo Finance, Google Finance, Enigma, etc.

We will extract Apple Stocks Price using the following codes:

```
import pandas as pd
import datetime
import pandas_datareader.data as web
from pandas import Series, DataFrame


start = datetime.datetime(2010, 1, 1)
end = datetime.datetime(2017, 1, 11)

df = web.DataReader("AAPL", 'yahoo', start, end)
df.tail()
```

	Open	High	Low	Close	Volume	Adj Close
Date						

Date						
2017-01-05	115.919998	116.860001	115.809998	116.610001	22103700	116.610001
2017-01-06	116.779999	118.160004	116.470001	117.910004	31577900	117.910004
2017-01-09	117.949997	119.430000	117.940002	118.989998	33387600	118.989998
2017-01-10	118.769997	119.379997	118.300003	119.110001	24420800	119.110001
2017-01-11	118.739998	119.930000	118.599998	119.750000	27418600	119.750000

Stocks Prices from Yahoo Finance

This piece of code will pull 7 years data from January 2010 until January 2017. Feel free to tweak the start and end date as you see necessary. For the rest of analysis, we will use the Closing Price which remarks the final price in which the stocks are traded by the end of the day.

Exploring Rolling Mean and Return Rate of Stocks

In this analysis, we analyse stocks using two key measurements: Rolling Mean and Return Rate.

Rolling Mean (Moving Average) — to determine trend

Rolling mean/Moving Average (MA) smooths out price data by creating a constantly updated average price. This is useful to cut down “noise” in our price chart. Furthermore, this Moving Average could act as “Resistance” meaning from the downtrend and uptrend of stocks you could expect it will follow the trend and less likely to deviate outside its resistance point.

How to Use a Moving Average to Buy Stocks

The moving average (MA) is a simple technical analysis tool that smooths out price data by creating a constantly...

www.investopedia.com

Let’s start code out the Rolling Mean:

```
close_px = df['Adj Close']
mavg = close_px.rolling(window=100).mean()
```

```
Date
2016-12-28    111.508340
2016-12-29    111.597476
2016-12-30    111.673134
2017-01-03    111.760151
2017-01-04    111.846564
2017-01-05    111.936389
2017-01-06    112.026282
2017-01-09    112.127969
2017-01-10    112.232448
2017-01-11    112.344720
```

The Last 10 Moving Average

This will calculate the Moving Average for the last 100 windows (100 days) of stocks closing price and take the average for each of the window's moving average. As you could see, The Moving Average steadily rises over the window and does not follow the jagged line of stocks price chart.

For better understanding, let's plot it out with Matplotlib. We will overlay the Moving Average with our Stocks Price Chart.

```
%matplotlib inline
import matplotlib.pyplot as plt
from matplotlib import style

# Adjusting the size of matplotlib
import matplotlib as mpl
mpl.rc('figure', figsize=(8, 7))
mpl.__version__

# Adjusting the style of matplotlib
style.use('ggplot')

close_px.plot(label='AAPL')
mavg.plot(label='mavg')
plt.legend()
```

Apple Stocks Price with The Moving Average (mavg)

The Moving Average makes the line smooth and showcase the increasing or decreasing trend of stocks price.

In this chart, the Moving Average showcases increasing trend the upturn or downturn of stocks price. Logically, you should buy when the stocks are experiencing downturn and sell when the stocks are experiencing upturn.

Return Deviation — to determine risk and return

> *Expected **Return** measures the mean, or expected value, of the probability distribution of investment **returns**. The expected **return** of a portfolio is calculated by multiplying the*

> *weight of each asset by its expected **return** and adding the values for each investment — Investopedia.*

Following is the formula you could refer to:

Formula for Returns

Based on the formula, we could plot our returns as following.

```
rets = close_px / close_px.shift(1) - 1

rets.plot(label='return')
```

Plotting the Return Rate

Logically, our ideal stocks should return as high and stable as possible. If you are risk averse(like me), you might want to avoid this stocks as you saw the 10% drop in 2013. This decision is heavily subjected to your general sentiment of the stocks and competitor analysis.

Analysing your Competitors Stocks

In this segment, we are going to analyse on how one company performs in relative with its competitor. Let's assume we are interested in technology companies and want to compare *the big guns:* Apple, GE, Google, IBM, and Microsoft.

```
dfcomp = web.DataReader(['AAPL', 'GE', 'GOOG', 'IBM',
'MSFT'],'yahoo',start=start,end=end)['Adj Close']
```


Stocks Price for Apple, General Electrics, Google, IBM, and Microsoft

This will return you a slick table of closing prices among the stocks prices from Yahoo Finance. Neat!!

Correlation Analysis — Does one competitor affect others?

We can analyse the competition by running the percentage change and correlation function in pandas. Percentage change will find how much the price changes compared to the previous day which defines returns. Knowing the correlation will help us see whether the returns are affected by other stocks' returns

```
retscomp = dfcomp.pct_change()

corr = retscomp.corr()
```


Let's plot Apple and GE with ScatterPlot to view their return distributions.

```
plt.scatter(retscomp.AAPL, retscomp.GE)
plt.xlabel('Returns AAPL')
plt.ylabel('Returns GE')
```

Scatter Plot of GE and AAPL

We can see here that there are slight positive correlations among GE returns and Apple returns. It seems like that the higher the Apple returns, the higher GE returns as well for most cases.

Let us further improve our analysis by plotting the scatter_matrix to visualize possible correlations among competing stocks. At the diagonal point, we will run Kernel Density Estimate (KDE). KDE is a fundamental data smoothing problem where inferences about the population are made, based on a finite data sample. It helps generate estimations of the overall distributions.


```
pd.scatter_matrix(retscomp, diagonal='kde', figsize=(10, 10));
```

KDE Plots and Scatter Matrix

From here we could see most of the distributions among stocks which approximately positive correlations.

To prove the positive correlations, we will use heat maps to visualize the correlation ranges among the competing stocks. Notice that the lighter the color, the more correlated the two stocks are.

```
plt.imshow(corr, cmap='hot', interpolation='none')
plt.colorbar()
```

```
plt.xticks(range(len(corr)), corr.columns)
plt.yticks(range(len(corr)), corr.columns);
```

Heatmap of Correlations among competing stocks

From the Scatter Matrix and Heatmap, we can find great correlations among the competing stocks. However, this might not show causality, and could just show the trend in the technology industry rather than show how competing stocks affect each other.

Stocks Returns Rate and Risk

Apart from correlation, we also analyse each stock's risks and returns. In this case we are extracting the average of returns (Return Rate) and the standard deviation of returns (Risk).

```
plt.scatter(retscomp.mean(), retscomp.std())
plt.xlabel('Expected returns')
plt.ylabel('Risk')
for label, x, y in zip(retscomp.columns, retscomp.mean(),
retscomp.std()):
    plt.annotate(
        label,
        xy = (x, y), xytext = (20, -20),
        textcoords = 'offset points', ha = 'right', va = 'bottom',
        bbox = dict(boxstyle = 'round,pad=0.5', fc = 'yellow', alpha
= 0.5),
        arrowprops = dict(arrowstyle = '->', connectionstyle =
'arc3,rad=0'))
```


Quick Scatter Plot among Stocks Risk and Returns

Now you could view this neat chart of risk and return comparisons for competing stocks. Logically, you would like to minimize the risk and maximize returns. Therefore, you would want to draw the line for your risk-return tolerance (The red line). You would then create the rules to buy those stocks under the red line (MSFT, GE, and IBM) and sell those stocks above the red line (AAPL and GOOG). This red line showcases your expected value threshold and your baseline for buy/sell decision.

Predicting Stocks Price

Feature Engineering

We will use these three machine learning models to predict our stocks: Simple Linear Analysis, Quadratic Discriminant Analysis (QDA), and K Nearest Neighbor (KNN). But first, let us engineer some features: High Low Percentage and Percentage Change.

```
dfreg = df.loc[:,['Adj Close','Volume']]
dfreg['HL_PCT'] = (df['High'] — df['Low']) / df['Close'] * 100.0
dfreg['PCT_change'] = (df['Close'] — df['Open']) / df['Open'] * 100.0
```


The end Data Frame Produced

Pre-processing & Cross Validation

We will clean up and process the data using the following steps before putting them into the prediction models:

1. *Drop missing value*
2. *Separating the label here, we want to predict the AdjClose*
3. *Scale the X so that everyone can have the same distribution for linear regression*
4. *Finally We want to find Data Series of late X and early X (train) for model generation and evaluation*
5. *Separate label and identify it as y*
6. *Separation of training and testing of model by cross validation train test split*

Please refer the preparation codes below.

```
# Drop missing value
dfreg.fillna(value=-99999, inplace=True)

# We want to separate 1 percent of the data to forecast
forecast_out = int(math.ceil(0.01 * len(dfreg)))

# Separating the label here, we want to predict the AdjClose
forecast_col = 'Adj Close'
dfreg['label'] = dfreg[forecast_col].shift(-forecast_out)
X = np.array(dfreg.drop(['label'], 1))

# Scale the X so that everyone can have the same distribution for
linear regression
X = preprocessing.scale(X)

# Finally We want to find Data Series of late X and early X (train)
for model generation and evaluation
X_lately = X[-forecast_out:]
X = X[:-forecast_out]

# Separate label and identify it as y
y = np.array(dfreg['label'])
y = y[:-forecast_out]
```

Model Generation — Where the prediction fun starts

But first, let's insert the following imports for our Scikit-Learn:

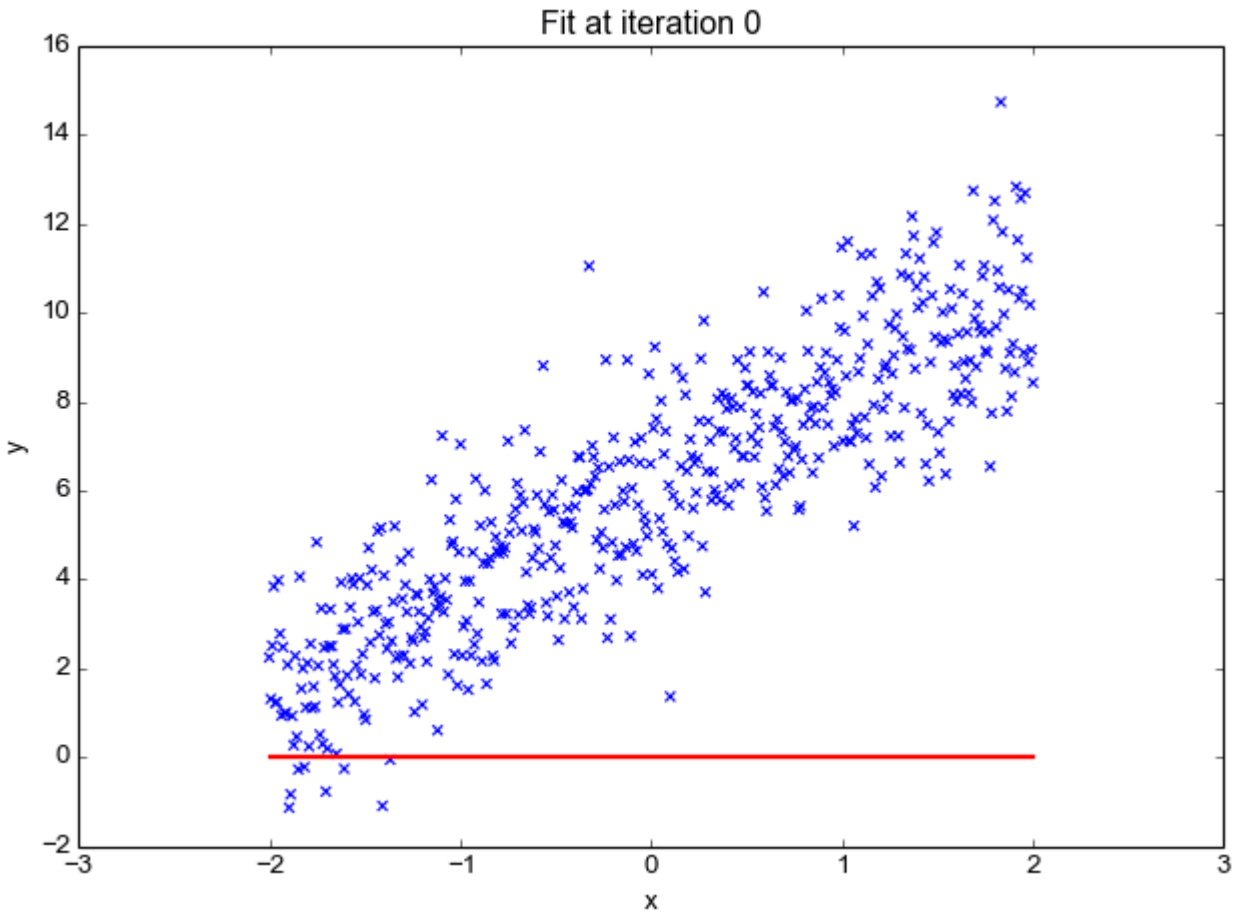
```
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor

from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
```

Simple Linear Analysis & Quadratic Discriminant Analysis

Simple Linear Analysis shows a linear relationship between two or more variables. When we draw this relationship within two variables, we get a straight line. Quadratic Discriminant Analysis would be similar to Simple Linear Analysis, except that the model allowed polynomial (e.g: x squared) and would produce curves.

Linear Regression predicts dependent variables (y) as the outputs given independent variables (x) as the inputs. During the plotting, this will give us a straight line as shown below:



Simple Linear Regression

This is an amazing publication which showed a very comprehensive review of Linear Regression. Please refer to the link below for the view.

A beginner's guide to Linear Regression in Python with Scikit-Learn

There are two types of supervised machine learning algorithms: Regression and classification. The former predicts...

towardsdatascience.com

We will plug and play the existing Scikit-Learn library and train the model by selecting our X and y train sets. The code will be as following.

```
# Linear regression
clfreg = LinearRegression(n_jobs=-1)
clfreg.fit(X_train, y_train)

# Quadratic Regression 2
clfpoly2 = make_pipeline(PolynomialFeatures(2), Ridge())
clfpoly2.fit(X_train, y_train)

# Quadratic Regression 3
clfpoly3 = make_pipeline(PolynomialFeatures(3), Ridge())
clfpoly3.fit(X_train, y_train)
```

K Nearest Neighbor (KNN)

This KNN uses feature similarity to predict values of data points. This ensures that the new point assigned is similar to the points in the data set. To find out similarity, we will extract the points to release the minimum distance (e.g: Euclidean Distance).

KNN Model Visualization where you would group the questioned element in k number of elements

Please refer to this link for further details on the model. This is really useful to improve your understanding.

Introduction to k-Nearest Neighbors: Simplified (with implementation in Python)

Note: This article was originally published on Oct 10, 2014 and updated on Mar 27th, 2018 Introduction In the four...

www.analyticsvidhya.com

```
# KNN Regression
clfknn = KNeighborsRegressor(n_neighbors=2)
clfknn.fit(X_train, y_train)
```

Evaluation

A simple quick and dirty way to evaluate is to use the score method in each trained model. The score method finds the mean accuracy of self.predict(X) with y of the test data set.

```
confidencereg = clfreg.score(X_test, y_test)
confidencepoly2 = clfpoly2.score(X_test,y_test)
confidencepoly3 = clfpoly3.score(X_test,y_test)
confidenceknn = clfknn.score(X_test, y_test)
```

```
# results
('The linear regression confidence is ', 0.96399641826551985)
('The quadratic regression 2 confidence is ', 0.96492624557970319)
('The quadratic regression 3 confidence is ', 0.9652082834532858)
('The knn regression confidence is ', 0.92844658034790639)
```

This shows an enormous accuracy score (>0.95) for most of the models. However this does not mean we can blindly place our stocks. There are still many issues to consider, especially with different companies that have different price trajectories over time.

For sanity testing, let us print some of the stocks forecast.

```
forecast_set = clf.predict(X_lately)
dfreg['Forecast'] = np.nan

#result
(array([ 115.44941187,  115.20206522,  116.78688393,  116.70244946,
        116.58503739,  115.98769407,  116.54315699,  117.40012338,
        117.21473053,  116.57244657,  116.048717  ,  116.26444966,
        115.78374093,  116.50647805,  117.92064806,  118.75581186,
        118.82688731,  119.51873699]), 0.96234891774075604, 18)
```

Plotting the Prediction

Based on the forecast, we will visualize the plot with our existing historical data. This will help us visualize how the model fares to predict future stocks pricing.

```
last_date = dfreg.iloc[-1].name
last_unix = last_date
next_unix = last_unix + datetime.timedelta(days=1)

for i in forecast_set:
    next_date = next_unix
    next_unix += datetime.timedelta(days=1)
    dfreg.loc[next_date] = [np.nan for _ in
range(len(dfreg.columns)-1)]+[i]

dfreg['Adj Close'].tail(500).plot()
dfreg['Forecast'].tail(500).plot()
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

Predictions Displayed in Plot

As we can see the blue color showcased the forecast on the stocks price based on regression. The forecast predicted that there would be a downturn for not too long, then it will recover. Therefore, we could buy the stocks during downturn and sell during upturn.

Future Improvements/ Challenges

To further analyse the stocks, here are some ideas on how you could contribute. These ideas would be useful to get a more comprehensive analysis on stocks. Feel free to let me know should there be more clarifications needed.

- Analyse economic qualitative factors such as news (news sourcing and sentimental analysis)
- Analyse economic quantitative factors such as HPI of a certain country, economic inequality among origin of company

Purpose, Github Code and Your Contributions

The purpose for this Proof Of Concepts (POC) was created as a part of investments side project that the I am currently managing. The goal of this application is to help you retrieve and display the right financial insights quickly about a certain company stocks price and predicting its value.

In the POC, I used Pandas- Web Datareader to find the stocks prices , Scikit-Learn to predict and generate machine learning models, and finally Python as the scripting language. The Github Python Notebook Code is located below.

Feel free to clone the repository and contribute whenever you have time.

Value Investing

In lieu with today's topics about stocks analysis. You could also visit my Value Investing Publication where I talked about scraping stocks financial information and displaying it in an easy to read dashboard which processes stocks valuation based on Value Investing methodology. Please visit it and contribute :).

Acknowledgments

I would like to thank you my fellow Accountancy and Finance friends who gave me constructive feedback on this publication. I really enjoyed learning that you gained much values from this publication of mine.

Finally… Reach out to me

Whew… That's it, about my idea which I formulated into writings. I really hope this has been a great read for you guys. With that, I hope my idea could be a source of inspiration for you to develop and innovate.

Please reach out to me via my **LinkedIn** and subscribe to my **Youtube Channel**

If you like it, please give me **Claps**.

Comment out below to suggest and feedback.

Happy coding :)

Disclaimer: This disclaimer informs readers that the views, thoughts, and opinions expressed in the text belong solely to the author, and not necessarily to the author's employer, organization, committee or other group or individual. References are picked up from the list and any similarities with other works are purely coincidental

This article was made purely as the author's side project and in no way driven by any other hidden agenda.