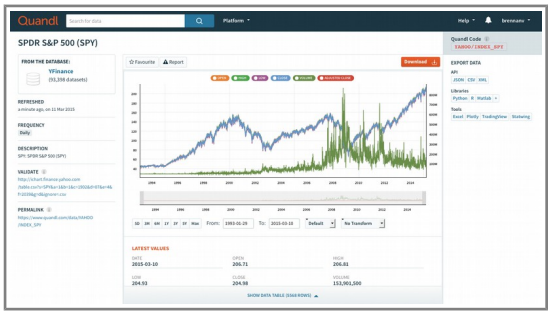
Predicting Movements in the Stock Market

At the start of our data science course (DAT4) at General Assembly we were presented with the prospect of using machine learning to creativity solve any problem we could imagine. I chose to try to predict the stock market. What follows is a log of my findings and follies. Feel free to follow along on my ipython notebook or my presentation.

For the purposes of this effort, the S&P 500 is generically refereed to as the market. Using the data mining workflow I'll attempt to predict up or down movements in the market.

Data Mining Workflow (from DAT4)
1. Define the problem / question
2. Identify and collect data
3. Explore and prepare data
4. Build and evaluate model
5. Communicate results



Lets's get started.

1. Defining the Problem

For my problem I wanted to look one day into the future and answer the question: Will the stock market would go up or down? To simplify the effort, I made it a binary problem: 0 if stocks go down, 1 if stocks go up.

Disclaimer: I am not an economist.

2. Identify and Collect Data

Using Quandl, I gathered data that I thought might be associated with determining market behavior.

My “forces” are daily indexes representing changes in the dollar, gold, oil and effective federal interest rates. My “indicators” are monthly or even quarterly indexes showing changes in inflation, federal recession estimates, labor productivity and unit costs.

Locate daily indices representative of market forces:

```
#see Quandl.com
gold = "FRED/GOLDPMGBD228NLBM" #since 1968
oil = "DOE/RWTC" #since 1986
rates = "FRED/DFF" #federal interest rates since 1954
stock = "YAHOO/INDEX_GSPC" #since 1950
dollar = "FRED/DTWEXM" #since 1973
force = [rates, dollar, gold, oil]
```

```
recession = "FRED/RECPROUSM156N"
inflation = "RATEINF/INFLATION_USA"
productivity = "FRED/OPHNFB" #new economy theory
costs = "FRED/PRS88003203" #consider outputPerHour / unitCosts
indicator = [recession, inflation, productivity, costs]
```

Load dataframe, check row length and clean up column names

```
token = ""
stock = Quandl.get(stock, authtoken=token, trim_start='1987-01-02')
forces = Quandl.get(force, authtoken=token, trim_start='1987-01-02')
indicators = Quandl.get(indicator, authtoken=token, trim_start='1987-01-02')
```

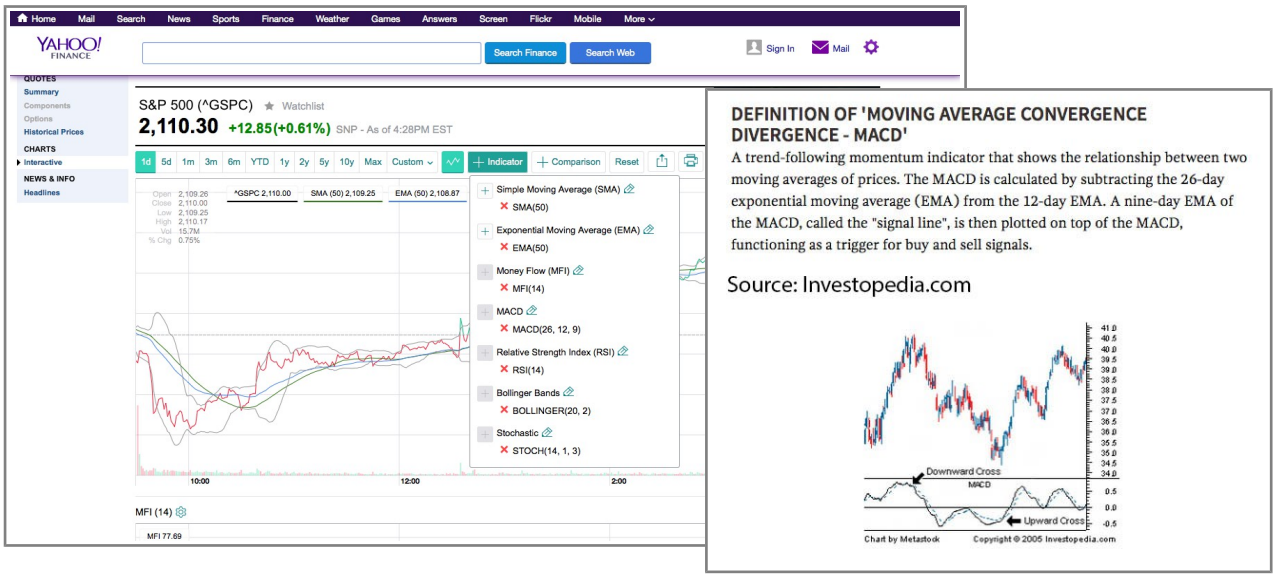
```
print 'stock rows: %d, force rows: %d, indicator rows: %d' % (len(stock), len(forces), len(indicators))
```

```
stock rows: 7090, force rows: 10272, indicator rows: 669
```

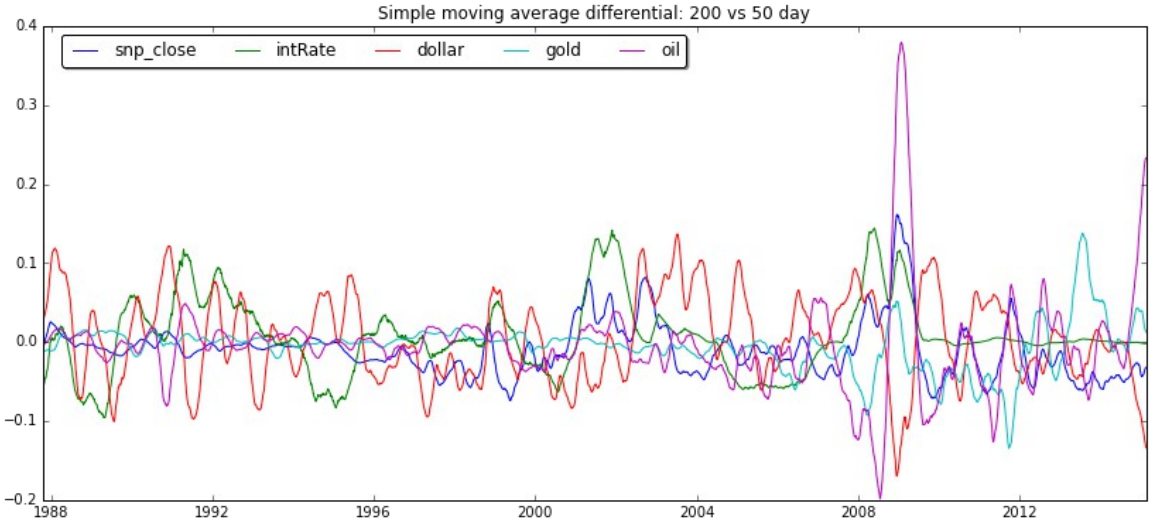
3. Explore and prepare the data

All data was loaded in pandas data frames and indexed as a time series. Lower frequency data sets were forward filled so that each row is an accurate representation of the economic forces measured on each day. Care was taken to maintain time series locality of information. Non-trading days were excluded from the data set.

Using Yahoo Finance and Investopedia as authoritative guides, a series of differential variables were calculated fin order to expand my list features or columns in my training set.



Calculated features included simple moving averages, exponential moving averages, the divergence and convergence of averages across different time spans, i.e.; 200 day averages versus 50 day averages, and 26 day averages versus 12 day averages.



Could this be a signal?

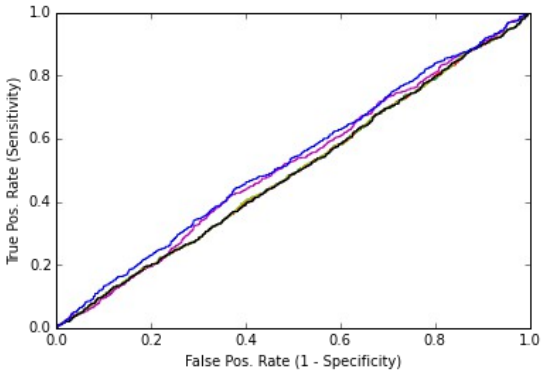
Calculated features also included Bollinger Bands which is essentially a measure of relative volatility using changes in standard deviations.

Supplemental features also included year, quarter and month tagging for the purposes of adding seasonality, and a progressive run count to indicate periods of successive up or down movement in the S&P 500.

Lastly as simplified calculation of money-flow index was included to measure the volume of trading that occurred daily with regards to the daily shift in the price of the S&P 500.

4. Build and evaluate model

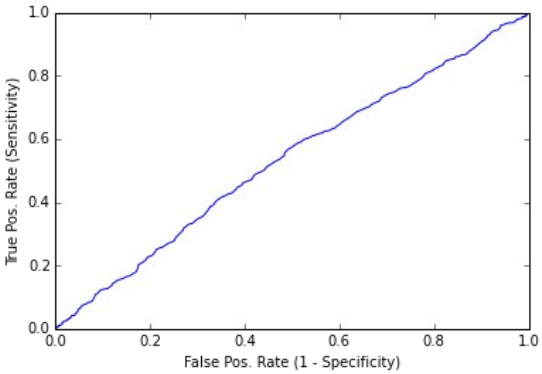
Several models were prepared and evaluated using logistic regression with limited success. Models were evaluated using train-test-split methods to produce out of sample ROC-AUC scores, confusion matrices and feature importance scores. The highest overall ROC-AUC score achieved using this method was 0.528.



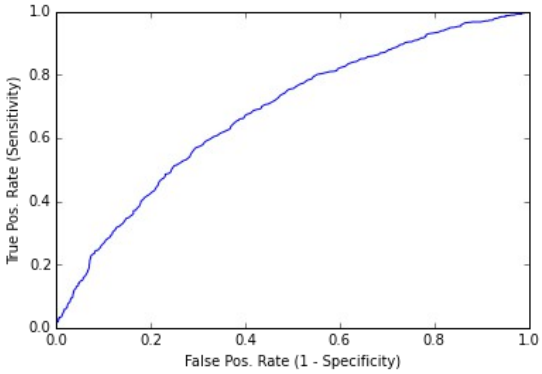
Logistic Regression ROC AUC

Random Forest Classifier (RFC) methods also initially appeared to be unsuccessful. With ROC-AUC scores only slightly better at 0.533.

After several days of non-productive work, I tried something seemingly counterintuitive:
I tried using the same RFC to predict two days into the future. The resultant ROC AUC score was 0.680.



RFC ROC AUC Looking One Day Out



RFC ROC AUC Looking Two Days Out

After rechecking my models for errors, it appears that my features are tuned for larger time periods. Given the granularity of my data being daily, I'm cautiously optimistic that I might be able to tune my features to give me a slightly better one day response.

For now, the model indicates a ROC AUC of 0.90 at seven days out. I am happy with these results even though it doesn't give me the original predictions for which I was looking.

Final Thoughts

If I was to evaluate my model based on the knowledge that's been imparted to me by my wise and all-knowing instructors at General Assembly, I'd say I still have a ways to go before I hack the stock market.

Q: How can we make a model that generalizes well?

1) split dataset
2) train model
3) test model
4) parameter tuning
5) choose best model
6) train on all data
7) make predictions on new data

Regardless, it's been an enjoyable and enlightening journey.

I'd like to thank my data science instructors at General Assembly for their extremely hard work, time and energy. You guys rock.