Capstone Project

Peter Liu

Machine Learning Engineer Nanodegree

June 5, 2018

Stock Market Correction Prediction using Machine Learning Methods

Definition

Project Overview

People in finance have always been interested in making market prediction. In 1910, a polish economist Pawel Ciompa published the first paper that describe using statistic methods on economic data to model future trends. Given enough data and analysis, models can be built to predict near future events with certain degree of accuracy. For many reasons financial modelling remains inaccessible to most people. Analyzing large amount of data requires fast and costly computing power. The algorithms tend to be complex and difficult to understand. But most of all, since the knowledge is so valuable monetary, it is not easily shared. Financial computer modelling remains one of the best kept secret in finance. With the continuing advances of computer resources both hardware and software, and financial data readily available online, building a financial computer model is more accessible than before. In this project I will apply machine learning technique to make market prediction.

The US stock market has been on a steady rise since early 2009. Many investors believe the market cannot sustain this trend forever. They believe the market should fall slightly from time to time to maintain the long term health. The slight bumps of market corrections are more desirable than a large market crash event. In this project, I will build a model to predict when a correction is going to occur. With this knowledge, an investor can choose to sell their investments before a correction, and buy it back during the correction at a lower price.

Problem Statement

The goal of this project is to create a model to predict when the rising US stock market will sustain market correction. A correction is defined as a 10% drop from the 52 week high. In this model, the Dow Jones Industry Average (DJA) will be an alias for the US stock market.

The following tasks is to be considered:

- Download and preprocess publicly available financial data from sources such as Yahoo Finance, Bureau of Labor and Statistics, and the US Treasury Department.
- Perform data analysis to determine periods of market corrections.
- Create a neural network model using Python and Keras on top of Tensorflow to train and learn to predict market corrections.
- Test the model

The model will output three variables corresponding to short term correction, medium term correction, and longer term correction. The variable ranges from 0 to 1 with 0 meaning low probability of correction, and 1 meaning high probability of correction.

Metrics

Accuracy of will be measured by the errors achieved by different version of the model. By varying the number economic data as inputs, model architectures, and hyper parameters, different accuracy can be achieved.

There will be three versions of the model. A simple model with minimal inputs and layers, a default model with all inputs and layers, and a tuned model where the hyper parameters are fine tuned to achieve better results.

Analysis

(approx. 2-4 pages)

Data Exploration

The dataset need to represent a full picture of the US economy. I've chosen to start with the benchmark indices of the two major markets of Equity and Fixed Income, as well as important economic indicators from the Bureau of Labor and Statistics (BLS). This dataset consists of historical prices download from three sources. Equity pricing data from Yahoo Finance, fixed income yield from the US Treasury, and economic statistics from Bureau of Labor and Statistics (BLS).

The historical prices of the major market indices such as DJA, S&P, Nasdaq Composite, and the volatility index (VIX) were all available from Yahoo Finance for download. Except for the VIX, the daily open, high, low, close, adjusted close, and volume are available from 1985. The VIX data is available from 1/2/1990.

The US Treasury provides daily yields of the benchmark securities (1/3/6 Months, 1/2/3/5/7/10/20/30 Year) dating back to 1990.

The economic statistics such as unemployment, productivity, consumer price index (CPI), and producer price index (PPI) comes from BLS. The BLS publish these statistics at different intervals and starting date.

Statistic	Start Time	Interval
Unemployment	1948	Monthly
Productivity	1947	Quarterly
CPI	1913	Monthly
PPI	1986	Monthly

This project will focus on the timeline from January 1, 1987 to March 1, 2018. Therefore data that lies outside of this timeline will be discards. The data will also be normalized.

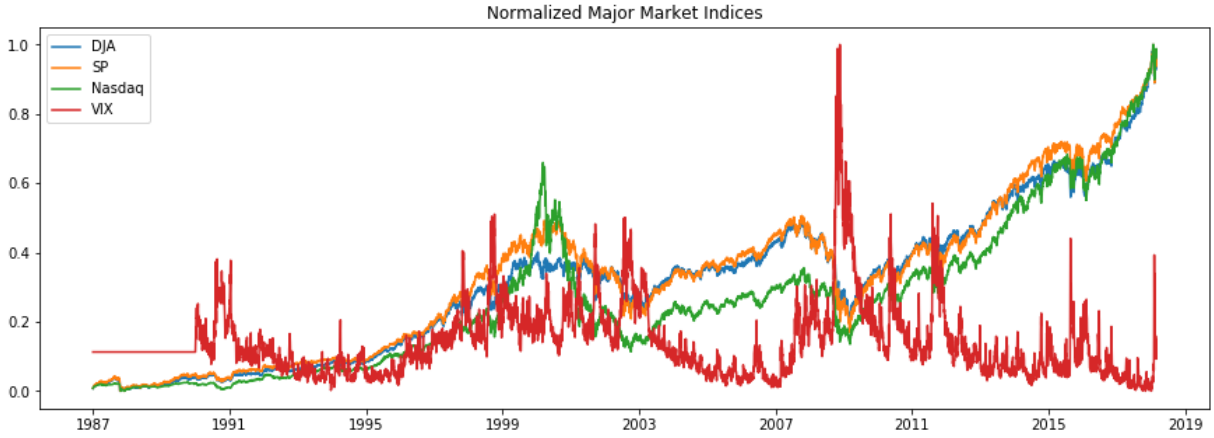
The model deals with daily price changes. The economic indicators are not published daily but instead is monthly or quarterly. They will have to be handled during the data processing.

After the dataset was created and back filled, it will be split into training and testing set. The split will be of two intervals from the beginning to 90% of present time as the training set, and the rest as the testing set.

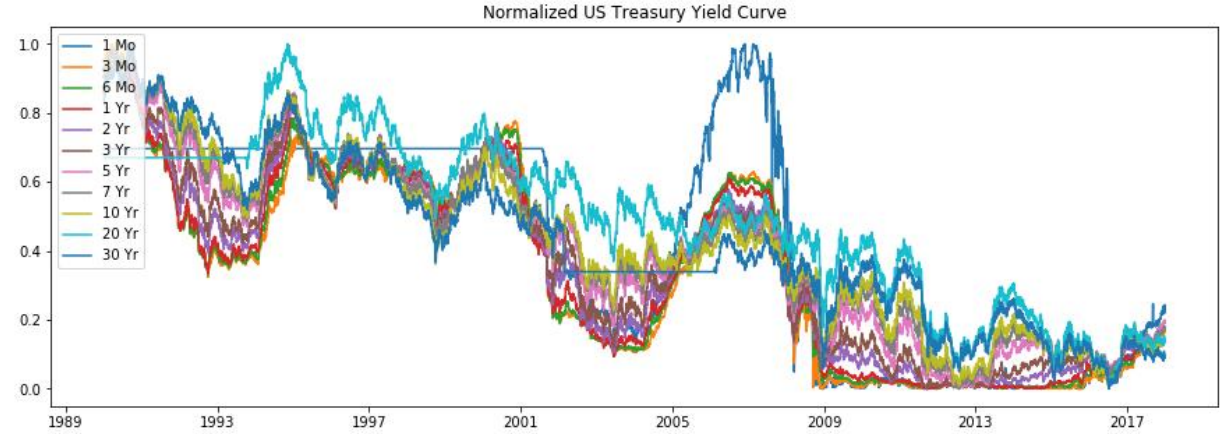
A Python module dataset.py was created to handle Excel and CSV format files, as well as interpolating monthly and quarterly data into daily data, back filling missing data, and other miscellaneous data related tasks.

Exploratory Visualization

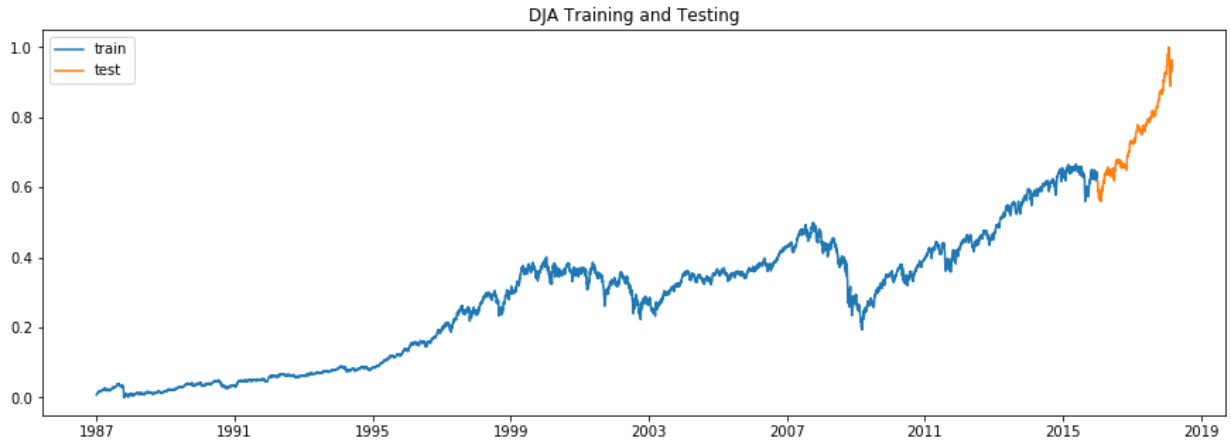
Here is a graph of the adjusted closing prices of normalized market indices.



Here is a graph of the normalized US Treasury Yield Curve.



Here is a graph of the split between training set and testing set.



The dataset consists of 21 columns. There are 7865 data points each representing one trading session in the time frame between 1/1/1987 and 1/1/2018. There are 2067 days where the DJA is in correction.

Training Set: 6507 trading days, 2031 days of corrections.

Testing Set: 543 trading days, 36 days in corrections.

Algorithms and Techniques

Deep learning neural network models will be constructed. The models will be programmed in Python with the Keras / Tensorflow framework. Different configurations of the model will be tested in order to achieve optimize results. Each configuration may have different architecture layers, hyper parameters, loss functions and optimizers, batch sizes, and training iterations.

The input will consists of prices from equity indices and treasuries, as well as economic statistics. The goal for the model is to predict correction indicators. There will be 5 such indicators indicating correction probability in 1, 7, 30, 90, 180 days into the future. In this project we can have a generalist model where 1 model is trained to predict all 5 indicators, having an output of 5 dimensions. Alternatively, we can have 5 specialist model who is trained only to predict a specific correction. Intuitively, it seems the specialist model should outperform the generalized model. The results of both approaches will be compared.

Benchmark

We will use two modelling approaches; generalist and specialist. The generalist model will predict all 5 correction indicators. The specialist models will consist of 5 models each predicting one correction indicator. We start using an untrained model to predict 5 outputs, and then 5 models to predict one output each. This will be the baseline accuracies. The accuracies should be close to random selections.

Each model will be trained using training data. Through the process of trial and error of adjusting the configuration, we should arrive with better accuracies than the benchmark models.

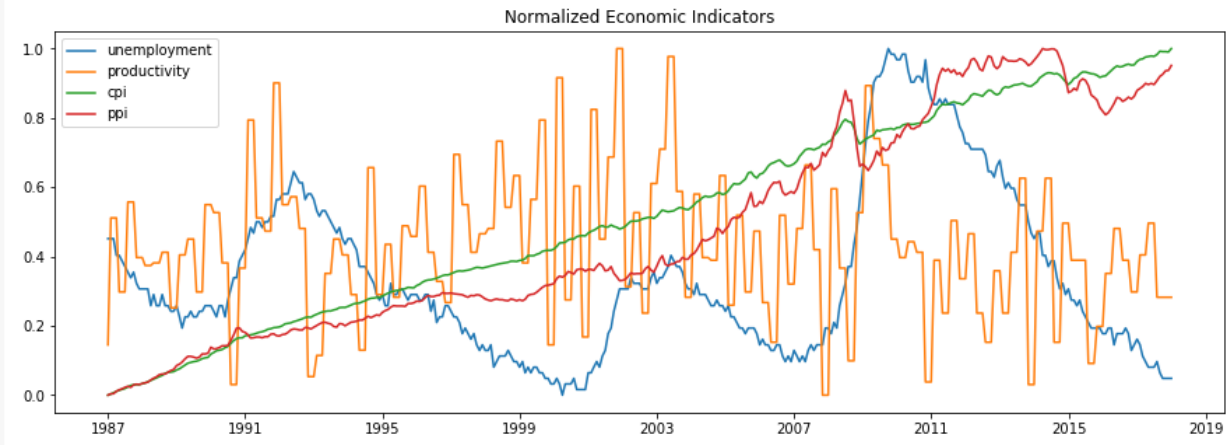
III. Methodology

(approx. 3-5 pages)

Data Preprocessing

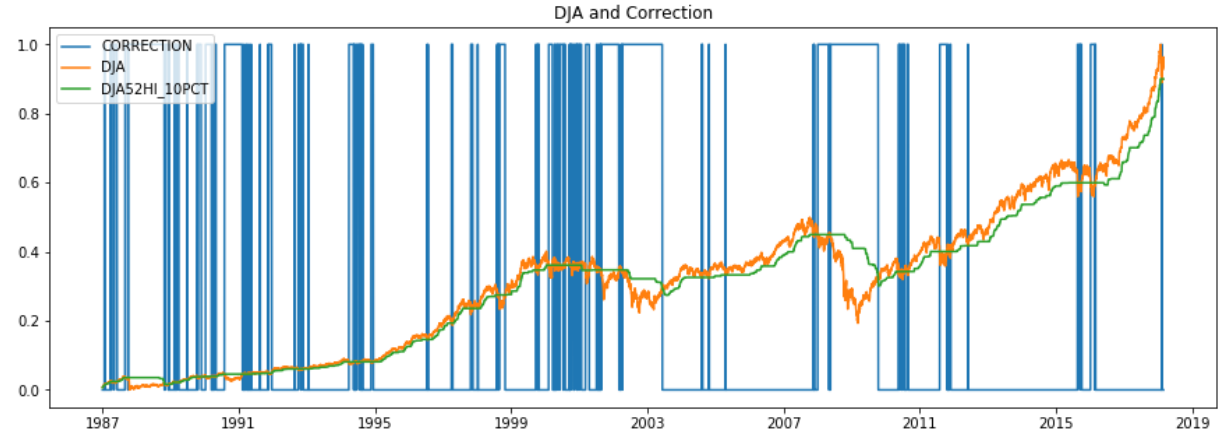
Since the economic indicates consists of monthly and quarterly intervals, they will need to be interpolated and back filled to property align in the dataset.

Here is a graph of the economic indicators with interpolated daily prices.



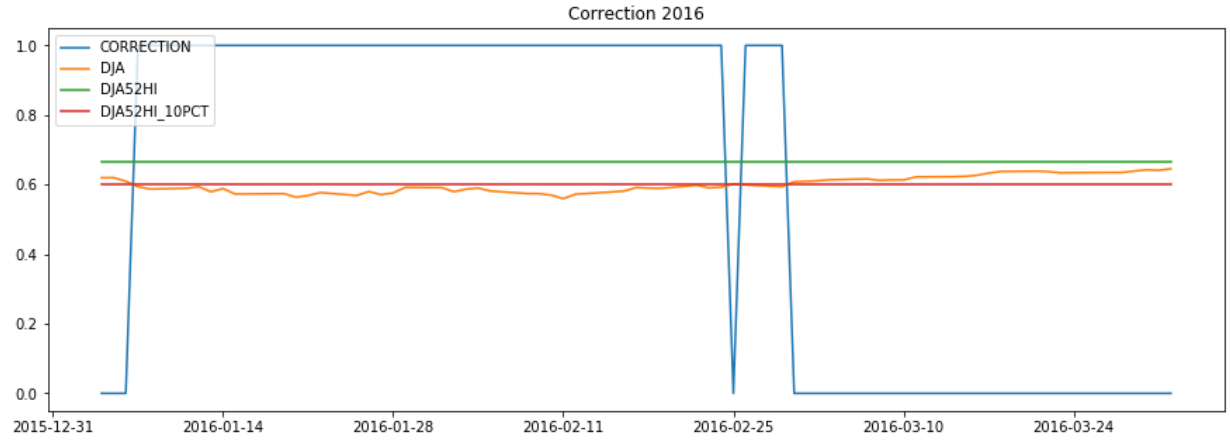
The correction indicator is derived using the following rule: it is set to 1 if the daily price movement is 10% below the 52 week high, otherwise 0.

Here is a graph of the correction indicator overlaying the market indices. When DJA is below DJA 52 high – 10%, the correction is at 1.



Here is a close up graph of the correction in early 2016. After spending 35 in correction, the market resumed the rising trend for the rest of 2016.

```
Number of days in correction 35
```



The correction indicator is a feature which is easily created by using the below 365 day moving average. In addition, we will transform this feature to create future corrections by shifting this indicators by desired days. For example, the 7 days correction is created by shifting the correction indicator by 7 days back. We will then have the 7 day correction indicator which tells us that in exactly 7 days there will be a correction.

For this project we created a 1 day, 7 days, 30 days, 90 days, and 180 days leading indicators.

Here is a graph of the correction indicators overlaying the DJA

Implementation

A python utility module dataset.py was created to handle to handle the various input data formats. The module parses both csv and Excel formatted data into a data frame. Some numeric fields were annotated with characters and need to be handled specifically. The module also handles the Excel data format. Since the data needs to be in daily format, the module back / front fill to convert quarterly and monthly data into daily format. The modules then lines up all the different feeds into a single data frame ready in preparation for the next step.

The next step is to create a number of correction indicators. First the 365 moving average of the DJA is calculated. Then the correction indicator is calculated. After that the indicator is shifted back to create the 1, 7, 30, 90, 180 correction indicators.

A starting and ending time is set for the dataset. Then it is split into two sets, training and testing. Two intervals are set. Training is from 1987 to 2015. Testing is 2016 – 2018.

Three types of models will be built each with its own architecture. They are the Benchmark, Generalist, and Specialist. The benchmark model consists of a basic three layers architecture. This model will not be trained and will produce 5 outputs that of being correct by random chance. The Generalist consists of 5 layers and will be trained to adjust its weights. This model will product 5 outputs. The Specialist is a set of 5 models each consists of 5 layers. This model will produce one output each.

Model Architecture Diagram

Basic
Input
Dense
Dense
Output

Generalist
Input
Dense
Dense
Dense
Dense
Output

Specialist x 5
Input
Dense
Dense
Dropout
Dense
Dropout
Dense
Output

Refinement

Each of the models was executed against the test set. The output of the model and the target label was used to compute an accuracy score. The activation functions, the model parameters, and loss function, and the optimizer will be tuned to achieve the best accuracy. Here is a list of parameters.

Dense Layer parameter	8, 16, 32
Dropout Layer parameter	0.2, 0.3, 0.4
Loss Functions	binary_crossentropy, categorical_crossentropy
Optimizer	rmsprop, adam
Epochs	10, 25, 50
Batch size	8, 16, 32

Here are the results of the accuracy of running the models:

	Basic	Generalist	Specialist	Optimized
Initial Training Loss	NA	0.5810	0.5709	0.4515
Final Training Loss	NA	0.3465	0.2898	0.1979
Initial Training Accuracy	NA	0.7099	0.7216	0.7966
Final Training Accuracy	NA	0.8549	0.7626	0.9134
Testing Accuracy	48.7069	85.6291	88.8009	92.4265

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

Optimized	Adam	Rmsprop	Epochs=60 No Dropout, adam, batch_sizes=8	Dropout / epochs 60
Initial Training Loss	0.4515		0.4698	0.5063
Final Training Loss	0.1979	0.2154	0.1820	0.1964
Initial Training Accuracy	0.7966		0.7760	0.7414
Final Training Accuracy	0.9134	0.9084	0.9217	0.9162
Testing Accuracy	92.4265		92.104%	

The Benchmark model was not tuned to provide a baseline of a random chance results.

The optimized model uses 3 additional Dense layers, epochs=30, and batch_size=8, and optimizer=rmsprop.

Does not make the model better:

1. Changing optimizer from adam to rmsprop
2. Dropout

Makes the model better

1. Increasing Epochs from 10 to 30
2. Decreasing batch_sizes from 32 to 8

Baseline random chance model achieved 45% accuracy. Specialist does slightly better than Generalist 85% vs 88%. Optimized model achieved the highest accuracy of 91%.

Since the project started, 2018 turns out to be a year of many corrections. The model was used for data from 3/2018 to 7/2018. The resulting accuracies were as followed.

Justification

The Benchmark model was able to achieve just below 50% accuracy. The generalist and specialized model achieved 85% and 88% accuracy respectively. The optimized model achieved 92% accuracy.

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

Here is a chart of the prediction 1 day correction vs the actual correction. The model is very cautious. It gives many false positives. It does have general knowledge.

Here is a chart of optimized model 30 days correction vs actual.

Here is a chart of optimized model 90 days correction vs actual.

Reflection

Predicting 1 day into the future is difficult. However, it is much harder to predict 7 days into the future. It is nearly impossible to predict 30, 90, 180 days. This makes intuitive sense.

The trial and error optimization is a difficult task. This process is best left to using the Keras API.

Improvement

The model uses daily closing prices from major indices to predict broad base market correction. Using less granular data such as real time ticker prices, and using more market indices should improve the model.

The optimization could have been done using Keras API which test the accuracy of various model configurations.

Before submitting, ask yourself. . .

- Does the project report you've written follow a well-organized structure similar to that of the project template?
- Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
- Would the intended audience of your project be able to understand your analysis, methods, and results?
- Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
- Are all the resources used for this project correctly cited and referenced?
- Is the code that implements your solution easily readable and properly commented?
- Does the code execute without error and produce results similar to those reported?

Proposal Text

For a 0 and 1 classification problem, the benchmark should be 50%. The accuracy of the model should be above 50%.

The preprocessing of the data may require filling out missing data, and extrapolating intervals. Normalizing the data will also be useful. The data will be visualized in charts and graphs.

Historical inflation, unemployment, productivity, CPI, treasury yield curve, and individual stock prices, and market indices will be used as input to the model.

The time series will be split into training and testing data. Training data will be the entire dataset minus one year of data. Suppose the dataset is the historical data up to the yesterday. Then the training data will be the dataset up to one year ago. The testing data will be the dataset from one year ago to yesterday.

The correction indicator can be computed. Using the rule of 10% down from 52 high to indicate a correction. This is the output label the model is trying to guess during prediction / testing.

These are all important data to determine the economic state. They illustrate current sentiments and future expectation of the economic performance of the US economy. They affect corporate earnings and profits and therefore the stock market. Investors often base their investment decisions on the current level and the outlook of these indicators.

Although the available economic data goes back to the 70s (unemployment) and 40s (inflation), this project will narrow down the time frame between 1981 and 2018. The following seven market correction events that happened during that time will be the main focus[1].

While the Dow is a key measurement of when a correction is occurring (10% down from 52 high), the rest of the above economic indicator determine the investment climate and future expectation of corporate earnings and profits.

Initially the DJA, S&P 500, and the Nasdaq Composite will be included in the dataset. Two other indices, Finance and Technology Services may be more sensitive to the market events mentioned above. Only publically available data will be used as input to the model. Stock indices historical data can be obtained online from finance.yahoo.com. Economic data such as inflation, unemployment, productivity, and CPI can be obtain from the Bureau of Labor Statistic www.bls.gov. The US Treasury Yield Curve data is available from www.treasury.gov .

Black Monday	10/19/1987
1990s Recession	7/1990
Dot-com Bubble	3/10/2000
Financial Crisis of 2007	9/16/2008
Flash Crash	5/6/2010
Market Selloff 2015-2016	8/18/2015
Market Selloff 2018	2/5/2018

1 https://en.wikipedia.org/wiki/List_of_stock_market_crashes_and_bear_markets

There are many publically available datasets. Using the browser, the following datasets were downloaded as csv / txt files
finance.yahoo.com - DJA, S&P, and Nasdaq Composite, CBOE Volatility Index (VIX) www.bls.gov - Unemployment, Inflation, CPI, Productivity www.treasury.gov - Treasury Yield Curve
And they were examined in Python using ggplot / numpy
Under historical data in finance.yahoo.com
Dow Jones 30 - ^DJI Jan 29, 1985 to Mar 01, 2018 S&P 500 - ^GSPC Jan 03, 1950 to Mar 01, 2018 Nasdaq - ^IXIC Feb 05, 1971 to Mar 01, 2018 VIX - ^VIX Jan 02, 1990 to Mar 01, 2018
From Bureau of Labor Statistics
Unemployment Rate (Seasonally Adjusted) - LNS14000000 Output Per Hour - Non-farm Business Productivity - PRS85006092 CPI for All Urban Consumers (CPI-U) 1982-84=100 (Unadjusted) - CUUR0000SA0 PPI industry group data for Total manufacturing industries, not seasonally adjusted - PCUOMFG--OMFG--
From www.treasury.gov under Daily Treasury Yield Curve Rates

create a correction indicator

Establish a 52 week high column. Then compare the daily price with the 52 week high. If it is less than 10%, the ticker is in correction.