Lorenzo Schneiderman

Stock Market Technical Analysis

Pattern Recognition on Price Data using K-Nearest Neighbors

02 February 2018

Purpose:

The incentive in making a predictive model of stock price change is usually to integrate it into a deployable trading strategy, which consists of at least one rule or model. However, the scope of this research project is not to devise and evaluate a complete trading strategy, but evaluate only a single model's ability at predicting one-day price action (percent change) of each stock in the S&P 500 during a period of time, in which this model 'trains' only using price data for every stock in the S&P 500. The assessed model generates predictions uses a type of pattern recognition using a similarity function (measurement) between two price series and searches for similar fixed-length price series patterns within a period of market price-history. The model then predicts the following day's price-action by averaging the actual outcomes measured following the matching price series patterns. This algorithm of finding the most similar examples from a labeled dataset and averaging their labels, known as k-nearest neighbors, is a form of instance based learning and is computationally expensive, but it can offer certain benefits like simplicity and transparency, which may be found crucial to identifying and distinguishing phenomena in the chaotic and noisy stock market (more on this inside "Further Discussion"). In my study I sought to find predictive power, measured by correlation, between the 24-hour price change of the stocks in the S&P 500 and the outputted predictions generated by the k-nearest neighbors algorithm training only on the price history of stocks from the S&P 500.

Method:

Initially historical stock prices are downloaded for every S&P 500 stock individually from finance.yahoo.com/ inside **Datafeed.py** via its function make_data_set(). Then, at run-time, the KNNRegressor class (inside of **StocksKNN.py**) executes the preprocessing computations required to compile and store 'readily comparable' stock-price patterns. This step involves the KNN-model recording a new example-label pair for each day observed and each stock. After pricing data for every stock in the S&P 500 has been downloaded between the dates 2017/11/16 -2018-02-05, the data undergoes preprocessing. In order for the KNN model to readily compare the similarity of a price series across all stocks, the following two steps must be made to normalize and then store each normalized price series example:

1. Subtract the 'present' day's price from every previous day's price in the example's time window (here the 'present' day precedes the outcome observation of the next day's change). For example, if a stock's value has consistently been increasing over n-days, then it's example-vector will have all-negative values for each day that is below the current day's price. During my evaluation N=20 so that twenty days are recorded in every example.
2. Then, normalize the example vector into a proportion change by dividing every day's price by the the 'present day' price.

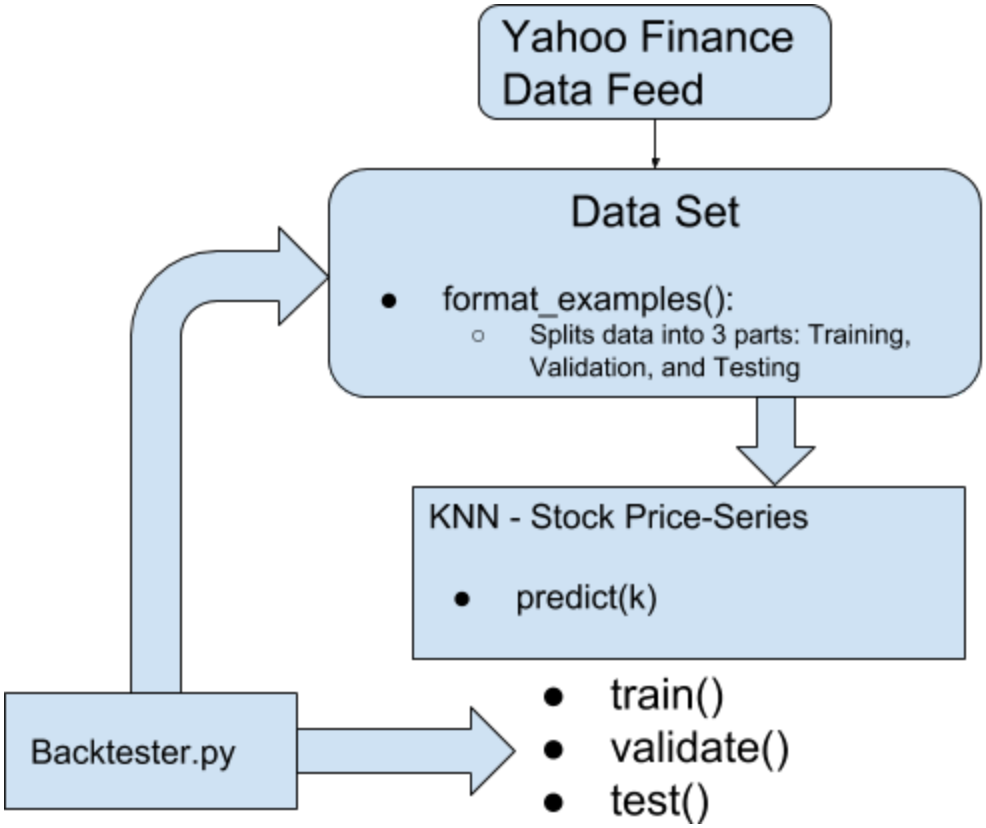
The label, or outcome, of that respective example would then be the percent change observation on the following day. This procedure can be repeated each day of a stock's history to record a new price example-outcome pair.

After using this transformation, comparing the price series of any stock for any fixed-length period is as simple as taking the euclidean distance of the two normalized price series vectors. That is, its difference can be expressed by the root of the sum of the squared differences between each examples respective-day's percentage change. With this as its measure of similarity between price patterns, the KNNRegressor can identify the most similar historical price series and average the examples' empirical outcomes to generate a prediction. The output from KNN was simply measured as the mean of the similar patterns' outcomes, without assigning any measure of certainty from the outcomes distribution, yet the distribution of k-outcomes can be analyzed beyond just its mean.

Methods called to train, validate, and test following data storage and preprocessing are also found inside the class KNNRegressor. Also inside of StocksKNN class is the helper method predict(x, k), which takes as its argument an array of inputs representing one or more 'normalized' price series and returns outputs generated by the KNN algorithm.

For evaluation the data is partitioned in a 4:1:1 training, validation, testing ratio and examines 200 trading days in total. During validation the model compares its predictions and the theoretical outcomes for every validation example to find the optimal integer value for K--choosing whichever K maximizes the outcome-output correlation. Then the data from the validation set is added to the KNN model's memory, and the optimal K is used to evaluate the algorithm in the testing set.

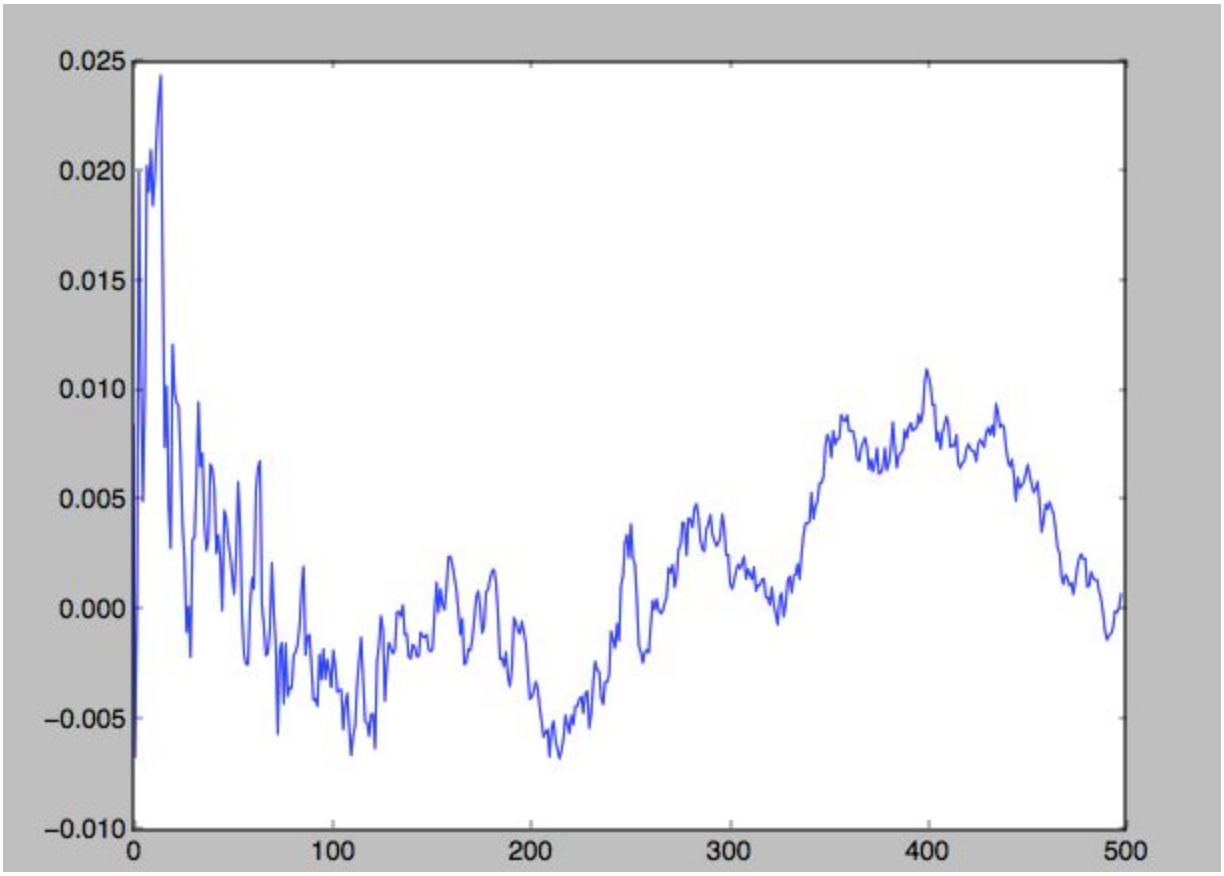
A Flowchart of Python Files



Findings:

Testing the KNN model's performance on days 2017/11/16 -2018-02-05 I found a correlation of 0.01976 for sample size 2808 (p=0.1474935) using K = 15.

Here is a graph of the validation performance of each K (on the x-axis) and the resulting correlation coefficient (y-axis).



Further Discussion:

The incentive in making a predictive model of stock price change is usually to integrate it into a deployable trading strategy, which consists of at least one rule or model. Typically trading strategies may operate using a combination of simple and complicated models and inputs--such as ‘Only buy Tesla stock on sunny days’, and ‘Minimize diversifiable risk among correlated assets.”--because deriving models independently of each other has a tendency to lower the chance of overfitting complicated models to the noisy and chaotic world in the financial market. Due to its simple nature and transparency at every step, pattern-recognition with KNN is a good place to begin exploring stock market behavior models, especially in conjunction with other technical analysis strategies.

Another benefit of using KNN over standard statistical regressors is the inherent measure of confidence the distribution of outcomes signifies. In the application of trading where multitudes of opportunities are available, most opportunities presented can be ignored without any cost to the trader. So extensive and diverse are the global financial markets that confused/conflicting trading opportunities can be (hopefully) overlooked entirely in favor of unanimously agreed upon opportunities where all models' predictions align. In this way a selective-ensemble of predictors is made even more powerful than the best of all its models (with the assumption that each model's performance is random, at worst) by limiting the domain of its predictions to inside the examples which all component models forecasts agree. With the ultimate goal in mind of setting up multiple "weak" predictors into an actionable trading strategy I hypothesized that I would want to use k-nearest neighbors because its mechanism of prediction provides easily derivable confidence metrics. The standard k-nearest neighbors algorithm works by searching a labeled dataset using a similarity function to rank the similarity of every known example to a given query example (one it is trying to make a prediction for). Then, the k-most similar examples of the training set are referenced and their labels averaged to generate an output. While merely averaging the labels for each of the known examples can be powerful, it neglects to consider the spread of the distribution of observed outcomes (attributing a value of certainty), as well as the similarity between the queried example and each of its derived similar examples from its memory (which can be thought of as the "discrepancy" between the input and the query objects). Both these metrics are intuitively useful at determining a models confidence estimate for its prediction, and additional features in a modified KNN predictions may be useful

in retaining a large domain of stock opportunities, while utilizing as many models synergistically as possible.