MLND Capstone Project Description - Investment and Trading

Investment and Trading Capstone Project

Build a Stock Price Indicator

Description

Investment firms, hedge funds and even individuals have been using financial models to better understand market behavior and make profitable investments and trades. A wealth of information is available in the form of historical stock prices and company performance data, suitable for machine learning algorithms to process.

For this project, your task is to build a stock price predictor that takes daily trading data over a certain date range as input, and outputs projected estimates for given query dates. Note that the inputs will contain multiple metrics, such as opening price (Open), highest price the stock traded at (High), how many stocks were traded (Volume) and closing price adjusted for stock splits and dividends (Adjusted Close); your system only needs to predict the Adjusted Close price.

You are free to choose what form your project takes (a simple script, a web app/service, Android/iOS app, etc.), and any additions/modifications you want to make to the project (e.g. suggesting what trades to make). Make sure you document your intended features in your report.

Setup

Recommended setup for a simple script or web app/service:

- Python
- NumPy, SciPy, Pandas
- (Optional) Python API for data access (see below)

Android/iOS app:

- Android/iOS SDK
- Native library/API for data access (see below)

Data

There are several open sources for historical stock price data which you are free to use:

- Yahoo! Finance: You can directly query for a stock through the web API, or download a dump of .csv files and use them.
- Bloomberg API: Multiple APIs available, including Python.
- Quandl: Also multiple APIs, including Python.

Look for an API endpoint/library function that lets you obtain daily stock values such as Open, High, Low, Close, Volume and Adjusted Close. Remember that Adjusted Close is what you are trying to predict.

Tasks

Implement stock predictor

For your core stock predictor, implement:

- A training interface that accepts a data range (start_date, end_date) and a list of ticker symbols (e.g. GOOG, AAPL), and builds a model of stock behavior. Your code should read the desired historical prices from the data source of your choice.
- A query interface that accepts a list of dates and a list of ticker symbols, and outputs the predicted stock prices for each of those stocks on the given dates. Note that the query dates passed in must be after the training date range, and ticker symbols must be a subset of the ones trained on.

Test and measure performance

A basic run of the core system would involve one call to the training interface, and one or more calls to the query interface. Implement a train-test cycle to measure the performance of your model. Use it to test prediction accuracy for query dates at different intervals after the training end date, e.g. the day immediately after training end date, 7 days later, 14 days, 28 days, etc.
(Note: Pick the training period accordingly so that you have ground truth data for that many days in the future.)

Build user interface

Once you're iterated on your stock predictor a few times, and it is giving results you are happy with (say, predicted stock value 7 days out is within +/- 5% of actual value, on average), implement a more user-friendly interface that lets you specify stock(s) you are interested in and provides predictions at some pre-defined intervals.

You can extend the system to suggest good stocks to buy or sell, and when. You could also maintain a portfolio of stocks for the user to make these suggestions more concrete. Document these enhancements in your report, with diagrams, screenshots, etc.

Learning Resources

Courses

- Machine Learning for Trading, Tucker Balch (Georgia Tech and Udacity)
- Stocks and Bonds, Khan Academy

Books

- Python for Finance, Yves Hilpisch