Machine Learning Engineer Nanodegree

Capstone Proposal

Oscar Martinez Rico

May 13th, 2017

Investment and Trading Capstone Project Proposal

Domain Background

As I joined my current employer, I was awarded some stocks. I find myself always trying to decide if it's a good idea to keep my stocks or sell them based on the current stock price. Having a software system that helps me determine if it is the right time to sell or keep a particular stock would be very helpful. There is available historical stock information that could be used combined with machine learning algorithms to make profitable investment decisions. There are some papers that demonstrate that using machine learning to predict stock prices is something achievable.

https://doi.org/10.1016/j.eswa.2014.07.040
https://doi.org/10.1016/j.eswa.2010.10.027

Problem Statement

Building a stock price predictor that takes daily trading data over a certain date range as input, and outputs projected estimates for given query dates. By looking at the historical data of a given stock as an input, the stock predictor application will train the model to predict the Adjusted Close value for the given stock in the future. The model will be created using regression algorithms as we are trying to predict the stock price. Having such a software system will help people like me to make more educated decisions managing their stock portfolio.

Datasets and Inputs

The stock predictor application will extract the historic stock information from Yahoo! Finance using the python library "yahoo-finance 1.4.0".

- *https://pypi.python.org/pypi/yahoo-finance*

The features used will be:

- *Open - opening price.*
- *High - highest price the stock traded at.*
- *Low - lowest price the stock traded at.*
- *Close - closing price adjusted for stock splits.*
- *Volume - how many stocks were traded.*
- *Adj Close - dividends*

The application will only predict the stock price of a stock using the historical data of the stock itself, it will not combine information of other or similar stocks.

The stocks that the application predictor will be able to predict are any stock that the Yahoo Finance API is currently supporting. The application will read the following parameters as inputs:

1. *Stock name - the stock symbol that the user is trying to predict.*
2. *Date range - (start date - end date)The date range will used to obtained the historical data and create the dataset. The bigger the the range the more data points that the application will use to create a more robust model. The user will be able to enter this rate so they can control how many data points they are interested to consider. This data will split into train and test date using:*
    a. *http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html*

The stock predictor application will predict the Adjusted Close price in the future, this means:

Predict stock price date = (enddate) + 1

Where enddate is part of date range given as input.

Solution Statement

The stock predictor application should be able to predict the stock price of a given stock by doing the following:

1. The application should receive the stock name and date range as input.
2. The application will fetch historical data of the given stock using the Yahoo Finance API.
3. Split historical data collected to have two datasets, training data and test data.
4. Training the model using the historical data, I’m planning to use the following algorithms and select the algorithm that provides a better score:
    a. Linear Regression
    b. Support Vector Regression.
    c. Nearest Neighbors Regression
5. Validate the prediction results using the test dataset.
6. Tweak the model, if necessary, to get better scores results and repeat steps 4 to 5.
7. Predict the stock price using the training model and date range given as input.
8. Show the results to users.

Benchmark Model

I will use out-of-the-box SVM for the benchmark, once the model created using of algorithms selected(Linear Regression, Support Vector Regression, Nearest Neighbors Regression) shows that significantly outperform the SVM results we can assume that the model can be useful.

Evaluation Metrics

To evaluate the model after trying with each of the algorithms selected, I’ll use the r2_score. The best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0.

Project Design

1. To obtaining the historical data I thinking to try the following:
    - ❏ Create the datasets in real-time calling:
        - ❏ `stock.get_historical('2014-04-25', '2014-04-29')`
    - ❏ To obtain the data I have two options:
        - ❏ Obtain manually the datasets and attach the corresponding files in the application.

        - ❑ *In runtime invoke the Yahoo Finance API and create the datasets using: https://pandas-datareader.readthedocs.io/en/latest/*
    - ❑ *Clean the data, making sure that all the columns are present for the date range provided. For example, if the date range it’s too big, changes are that the dataset return will contain empty columns, I need to clean this before training the model.*
    - ❑ *Split the historical dataset into test and train data.*

2. *I’m planning to use sklearn and explore the following algorithms, and plot the results after trying each of this algorithms.*
    - ❑ *Linear Regression*
    - ❑ *Support Vector Regression.*
    - ❑ *Nearest Neighbors Regression*

3. *Using each of the algorithms I will train and test each model to identify which one provides a better score; this model will have more changes to accurately predict stocks prices using unseen data.*
    a. *I’m planning to tweak the algorithms using GridSearchCV, this will help to search exhaustively over specified parameter values for an estimator.*

4. *Implement a more user-friendly interface that lets users specify stock they are interested in and provides predictions at some pre-defined intervals.*
    a. *I’ll start creating a CLI interface, if time permits I’ll create a web user interface.*