Machine Learning Engineer Nanodegree

Capstone Proposal: Investment and Trading Capstone Project

Igwebuike Onyeka Daniel
February 10th, 2021

Background

In very recent times, finance industries and firms have used trading models to predict possible future outcomes, and study market movements for better and more profits. These financial activities have also risen in numbers in terms of amount of transactions with the rapid global economic developments, and thus making their trend more complex to monitor.

"Forecasting time series is a long-standing research problem that is applicable to econometrics, marketing, astronomy, ocean sciences, and other domains. Similarly, networks are the subject of active research with broad relevance to areas such as transportation, internet infrastructure, signaling in biological processes, and online media. In this paper, we are concerned with forecasting among a network of time series with mutual influences. Tools for tackling this problem will help answer questions about complex systems evolving over time in the above application domains and beyond." Alasdair Tran, 2021[1]

Leveraging on the immense amount of data available, machine learning models can use knowledge of past sequences to predict possible future results, using one of many machine learning algorithms available. Just as in many other fields of science, research and development, machine learning has proven also to be a great tool for financial modelling and analysis and more include portfolio optimization, stock betting and predictions.

"Before the emergence of efficient machine learning algorithms, researchers in China and elsewhere generally used various statistical and econometric methods to build prediction models for research. Conventional statistical and econometric models require linear models and cannot be used to predict and analyse financial products before transforming nonlinear models to linear models. As an important branch of machine learning algorithms, neural networks (NNs) have the following advantages compared to conventional statistical methods: they are numeric, data-driven and adaptive." Pengfei Yu, 2019[2]

Problem Statement

The challenge with stock market prediction and forecast, as with most Time Series, is its sporadic non-linear, 'unpredictable' nature. No one can say with a probability of one (1) at what price any Company would close the next day -or the current day also. This uncertainty, which does not completely depend on several independent features or known variables is what the machine learning models, and training algorithms would try to solve.

First, the magnitudes of TS (Time Series) are largely differing, second, the distribution of these magnitudes are highly skewed.

Datasets and Inputs

The first step is data acquisition, and a larger amount of data makes it better for the training model and thus results. These data sources include (but not limited to) Google, Yahoo, Bloomberg etc. However, for this project, I have used Yahoo data source.

I chose the Oracle data and used same data across the models I created, this was so I could compare models across same data. The data was daily trading activities for a twelve (12) years period -2008 to 2020. This dataset is imported (from Yahoo) as Pandas DataFrames, with seven columns ['High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close']. I chose the 'Adj Close' column as my data column, on which the prediction is done, and the target prediction are the next day values of the 'Adj Close' column.

Labelling was not needed as this is a Time-Series the predictions were simply the next day values.

Benchmark Model

For a Benchmark Model, I used an xgboost stock prediction model, Stock Price Prediction - 94% XGBoost | Kaggle. Although, this example has multiple models from different algorithms, I based my comparism on the one similar to the algorithms I used. This benchmark xgboost model used a classification approach, with a label output column which indicates if there was an increase/rise in closing price (indicated as a 1), or a decrease/fall in closing price (indicated as a 0).

My target is based on the precision, recall and accuracy scores of the benchmark xgboost model, however, I will be using a regression approach rather than a classification approach, thus, I would be measuring the values of metrics like RMSE (Root Mean Squared Error).

Solution Statement

The best approach would be to use a few algorithms and compare their outputs (predicted values) with the test data, and against a known benchmark, and the run diagnosis to calculate for RMSE, and further testing for measure of performance. R2 can also be calculated; R-squared (R2), which is the proportion of variation in the outcome that is explained by the predictor variables. In multiple regression models, R2 corresponds to the squared correlation between the observed outcome values and the predicted values by the model. The Higher the R-squared, the better the mode

The best approach is to build a model using one of the known algorithms, split the data into train and test groups, and train this model with the input train-group data, and then use the model to make predictions, and compare these predictions with the test data.

Evaluation Metrics

Major metrics which would be used for evaluation include the confusion matrix, and RMSE (Root Mean Square Error), and the R2 or Adjusted R2.I noticed some of these errors are outputted during training of the model, another option would be to write functions which calculates the metrics.

Project Design

- The first step is Data Acquisition:

    - Download data from the online Yahoo data source, this is done using the Pandas datareader module.

- The second step is Data Cleaning and Pre-processing:
    - I need only the 'Adj Close' column, thus, dropping all other columns.
- Preparation of Data for inputs to the model:
    - First is splitting the data into Train and Test groups. A Validation group could also be added to avoid oversampling.
    - Depending on which algorithm used, there is the need to convert data into necessary formats, e.g. csv-format for xgboost models, and json-format for deepar models.
    - There is also the need to arrange the columns in format needed by the either AWS Sagemaker (when using in-built algorithms), or when using external libraries.
- Model building, and setting training Parameters and Hyperparameters tuning:
    - This is the pre-training phase, when the model is set up with the basic parameters needed for training.
- Training and Deployment:
    - The '*.fit()*' method is used to run the model
    - Model building and training might take more than two repetitions, since we train and re-train to compare outputs when different parameters are used, that means keeping other hyperparameters constant and changing one, and then compare what changes are evident.
    - After training, the model is deployed by running the '*.deploy()*' method, to create endpoints which can be used to conduct predictions into the future
- Evaluation and Analysis:
    - Using statistics, precision and accuracy are calculated and compared to the benchmark, or other well-known standards to determine how well the model has done.

References:

[1]Alasdair Tran, A. M. (2021). *Radflow: A Recurrent, Aggregated, and Decomposable Model.* Australia: Creative Commons Attribution 4.0 International.

[2]Pengfei Yu, X. Y. (2019). *Neural Computing and Applications.* Hubei.