

Machine Learning Engineer Nanodegree

Capstone Proposal

Mincent Lee January 31st, 2018

Proposal

Domain Background

This project will develop a stock price predictor by machine learning. The proposal is historically simplified from the [Project Description ~ Investment and Trading](#)^[1] and based on the [Course ~ Machine Learning for Trading](#)^[2] for my first solid step to study machine learning for trading. Because the risk free rate of return from a bank account or a very short-term treasury bond is about 0 lately, folks have put so much money into the stock market^[2:1]. The stock prediction can help us to understand market behaviour and trade profitable investments according to the wealthy information in the stock history and company data which is suitable for machine learning process^[1:1]. There are lot related academic research support the stock prediction^{[2:2][3]} while there are also opponent Efficient-Market Hypothesis^[4].

Problem Statement

For reality and accuracy^[4:1] concerns, the target problem of my first stock study is simplified to predict whether the *Adjusted* (for stock splits and dividends) *Closing price* rises or falls. The stock price predictor is inputted a certain range of daily trading data and outputs whether the *Adjusted Closing price* rises or falls (might ignore the rare flat cases at the first step) next to the certain range. This is quantifiable, measurable, and replicable. The relevant potential solution are the Classifiers of the [scikit-learn](#)^[5], e.g., the [ensemble Gradient Boosting Classifier](#)^[6].

Datasets and Inputs

The datasets used in this project is obtained from the [Yahoo! Finance](#)^[7] by the python module [yahoo-finance](#)^{[8][9]}. The target stock might be the [S&P 500 Index](#)^[10] that might be the best

representation of the U.S. stock market^[10:1]. The inputs include *Opening price*, *Highest price*, *traded Volume*, *Adjusted Closing price*, and so on.

Solution Statement

The potential solution is training a Classifier by daily trading data within specific ranges of days to predict *Rising* or *Falling* of the *Adjusted Closing price* following the range. The daily trading data are obtained from the [Yahoo! Finance](#)^[7:1] by the python module [yahoo-finance](#)^[8:1]. The machine learning libraries and Classifiers might come from [scikit-learn](#)^[5:1], e.g., the [ensemble Gradient Boosting Classifier](#)^[6:1], and the parameter [random_state](#)^[11] will be recorded. Therefore, the solution is quantifiable, measurable, and replicable.

Benchmark Model

The predicted *Rising* and *Falling* results will be evaluated in fact with the exact benchmark of specific daily prices from the [Yahoo! Finance](#)^[7:2] by the python module [yahoo-finance](#)^[8:2]. Might also build a [Naïve Predictor](#)^[12] which always predict *True/False* (*Rising/Falling*), if necessary.

Evaluation Metrics

The solution model will be evaluated with the exact benchmark of specific daily prices from the [Yahoo! Finance](#)^[7:3] by the $F_\beta - score$ ^[13] with the [fbeta_score](#) function of [scikit-learn](#)^[14]. The mathematical representations is:

$$F_\beta = (1 + \beta^2) \frac{precision \cdot recall}{\beta^2 precision + recall}$$

The β might be 1 for balanced precision and recall^[13:1].

Project Design

- Analysis
 - Data Collection
 - Will try [S&P 500 Index](#)^[10:2] first
 - Data include *Opening price*, *Highest price*, *traded Volume*, *Adjusted Closing price*, and so on
 - Will try the python modules
 - ["yahoo-finance 1.4.0," PyPI - the Python Package Index](#)^[8:3]

- "fix-yahoo-finance 0.0.21," *PyPI - the Python Package Index*^[9:1]
- Data Structure
 - Format data to the *DataFrame* of *pandas*^[15]
- Data Cleaning
 - Basic abnormal trading data handling^[2:3]
 - *Imputation of missing values*^[16]
- Feature-set Exploration
 - Feature-set include *Opening price, Highest price, traded Volume, Adjusted Closing price*, and so on
 - Data Mining
 - *pandas.DataFrame.describe*^[17]
 - *NumPy*^[18]
 - Exploratory Visualization
 - *matplotlib*^[19]
 - *seaborn ~ statistical data visualization*^[20]
- Benchmark
 - Models
 - Exact benchmark of specific daily prices from the *Yahoo! Finance*^[7:4] by the python module *yahoo-finance*^[8:4]
 - Might build a *Naïve Predictor*^[12:1] which always predict *True/False* (*Rising/Falling*), if necessary
 - Evaluation Metrics
 - $F_{\beta} - score$ ^[13:2]
 - *fbeta_score* function of *scikit-learn*^[14:1]
 - The β might be 1 for balanced precision and recall^[13:3]
- Methodology
 - Data Pre-processing
 - Outlier detection
 - *Methods/algorithms* of *scikit-learn*^{[21][22]}
 - Normalizing Numerical Features
 - *Scaling/Standardization*^[23] *scalers* of *scikit-learn*^[24]
 - Encode Stock Price Changings for Classification
 - Encode the predicted *Adjusted Closing price* to *True* if it is *Rising* than the previous trading day (skip the non-trading days) and vice versa (*False* if it is *Falling*)
 - Shuffle and Split Data
 - Apply *sklearn.model_selection.train_test_split*^[25] with recorded *random_state*^[11:1] for replicability

- Supervised Learning Models
 - [Classifiers in scikit-learn^{\[5:2\]}](#), e.g., the [ensemble Gradient Boosting Classifier^{\[6:2\]}](#)
- Training and Predicting Pipeline
 - Build normalizing, training, predicting, scoring functions
 - Make the normalizing [Scaler^{\[24:1\]}](#) and [Classifier^{\[5:3\]}](#) in a [Pipeline^{\[26\]}](#)
- Initial Model Evaluation
 - Apply default and coarse-grained parameters
 - Record all the available parameter `random_state[11:2]` for replicability
 - The specified day range of each daily trading data might be a week initially
- Refinement
 - [Fine-tune the hyper-parameters^{\[27\]}](#) ([Exhaustive Grid Search^{\[27:1\]}](#) with [Cross-validation^{\[28\]}](#) at least)
 - Tune the day range of each input might by [Feature importances^{\[29\]}](#) or [Feature selection^{\[30\]}](#)
- Results
 - Model Evaluation and Validation
 - Evaluate with the $F_1 - score^[13:4]$
 - Validate with [Cross-validation^{\[28:1\]}](#) and long period of days
 - Justification
 - Compare with the exact benchmark of daily prices^[8:5] in fact
 - Might compare with the [Naïve Predictor^{\[12:2\]}](#), if necessary

1. "MLND Capstone Project Description - Investment and Trading," [Udacity](#) ↔ ↔

2. Tucker Balch, "Machine Learning for Trading," [Georgia Tech](#) and [Udacity](#) ↔ ↔ ↔ ↔

3. "Stock market prediction," [Wikipedia](#) ↔

4. "Efficient-market hypothesis," [Wikipedia](#) ↔ ↔

5. "Choosing the right estimator," [scikit-learn.org](#) ↔ ↔ ↔ ↔

6. "sklearn.ensemble.GradientBoostingClassifier," [scikit-learn.org](#) ↔ ↔ ↔

7. Yahoo! Finance ↔ ↔ ↔ ↔ ↔

8. "yahoo-finance 1.4.0," [PyPI - the Python Package Index](#) ↔ ↔ ↔ ↔ ↔ ↔

9. "fix-yahoo-finance 0.0.21," *PyPI - the Python Package Index* ↔ ↔
10. "Standard & Poor's 500," *Wikipedia* ↔ ↔ ↔
11. "Utilities for Developers," *scikit-learn.org* ↔ ↔ ↔
12. udacity, "Project: Finding Donors for CharityML," *github.com* ↔ ↔ ↔
13. "F1 score," *Wikipedia* ↔ ↔ ↔ ↔ ↔
14. "sklearn.metrics.fbeta_score," *scikit-learn.org* ↔ ↔
15. "pandas.DataFrame," *Python Data Analysis Library*, *pandas.pydata.org* ↔
16. "Imputation of missing values," *scikit-learn.org* ↔
17. "pandas.DataFrame.describe," *pandas.pydata.org* ↔
18. NumPy, *SciPy.org* ↔
19. matplotlib.org ↔
20. seaborn: statistical data visualization, *seaborn.pydata.org* ↔
21. "Outlier detection with several methods," *scikit-learn.org* ↔
22. "Comparing anomaly detection algorithms for outlier detection on toy datasets," *scikit-learn.org* ↔
23. "Standardization, or mean removal and variance scaling," *scikit-learn.org* ↔
24. "Compare the effect of different scalers on data with outliers," *scikit-learn.org* ↔ ↔
25. "sklearn.model_selection.train_test_split," *scikit-learn.org* ↔
26. "Pipeline: chaining estimators," *scikit-learn.org* ↔
27. "Tuning the hyper-parameters of an estimator," *scikit-learn.org* ↔ ↔
28. "Cross-validation: evaluating estimator performance," *scikit-learn.org* ↔ ↔

29. "Feature importances with forests of trees," scikit-learn.org ↩

30. "Feature selection," scikit-learn.org ↩