Stock Market Predictions with Artificial Intelligence

Jennifer Sissi Lange
Department of Information Systems
Hanyang University
Seoul, Republic of Korea
9207120238
Jenni-sissi.lange@debeef.de

Jade Lu
Department of Computer Science
Hanyang University
Seoul, Republic of Korea
9206220239
Jlu23@student.gsu.edu

Kerk Zong Zhe
Department of Electronic Engineering
Hanyang University
Soul, Republic of Korea
9206820235
Kerk0013@e.ntu.edu.sg

Gin Lo Hoi Ching
Department of English Language and Literature
Hanyang University
Seoul, Republic of Korea
S216213@hsu.edu.hk

This is a blog in process!!!

***Abstract*—This paper describes an approach on how to use artificial intelligence to predict a possible rise or fall of the stock market value in the near future.**

Keywords—Artificial Intelligence, Decision Tree, Random Forest, Receiver Operating Characteristic, Dataset Modification

I. Introduction

The stock market is a dynamic system that presents the economic well-being of companies. Investors and market participants tend to make decisions based on the traditional econometric models and technical analysis indicators. Due to the complexity of the past performance of the stock, the accuracy of the traditional method is not highly reliable. This uncertainty and the unpredictability of the stock market means investors might face serious financial risks in optimizing their investment strategy. 61% of Americans own stock because they are more open to buying stock (Caporal, 2023). There is a need for a sophisticated and data-driven solution that can provide more reliable and actionable predictions.

In this project, an AI model is proposed to predict the stock market trend and to capture the intricate patterns and subtle relationships within financial data. An ensemble learning model utilizing Random Forest and Decision Tree algorithms to analyze historical stock market data and price movements will be developed. The model will provide labels to indicate the trend of the stocks. The user could make better decisions based on our model.

By providing more accurate and reliable predictions, this model can help investors make informed investment decisions, optimize portfolio management strategies, and mitigate financial risks. Financial institutions can benefit from improved risk assessment and asset allocation, leading to enhanced profitability and stability. Moreover, a more precise understanding of market trends can contribute to overall market efficiency and investor confidence.

II. Dataset

A. *TESLA Stock Dataset (TSLA)*

The TESLA dataset includes approx. 10 years of the TSLA stock data from June 29, 2010 - February 3, 2020. The original dataset was slightly modified removing Adj Close for consistency reasons. So the features that are included in the modified dataset are as follows: Date, Open, High, Low, Close, Volume, and Rate of Change. The prices are based on USD.

B. *SAMSUNG Stock Dataset (SSNLF)*

The SAMSUNG dataset includes decades of SSNLF stock data from January 4, 2000 - May 23, 2022. The prices are also based on USD. It's also modified as described in the TESLA Stock Dataset description.

C. *TWITTER Stock Dataset (TWTR)*

The TWITTER dataset was chosen to be used because of its interesting history. Due to Elon Musk's purchase of Twitter back in 2022, it became a private company, was delisted from the New York exchange, and is no longer in the stock market. So this dataset contains its history on the stock market from November 7, 2013 - October 27, 2022. The prices are also based on USD. It's also modified as described in the TESLA Stock Dataset description.

III. Methology

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections A-D below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

A. *Original Dataset*

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the

abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

Date - Includes the year, month, and day in that order (YYYY-MM-DD)

Open - This is the opening price of the day.

High - The highest price of the day.

Low - The lowest price of the day.

Close - This is the closing price of the day.

Volume – The total amount of shares traded in the day.

Adj Close - This means the adjusted closing price of the day. It's adjusted to better reflect the stock's value after anything such as corporate actions would affect the stock price after the market closes. However, this was taken out for consistency reasons.

B. Modified Dataset and Features

The dataset is extended with modified data to make the dataset bigger and create more data for the AI to train on. With the additional created data, possible patterns and relationship between different information will be recognized by the AI. In the following it is described how the data is going to be modified and extended.

Slope/rate of change - It is calculated using the formula above. Its purpose is to help train the AI to take into consideration the influence the stock prices on the days before have on the stock prices on the day of. It helps the AI recognize certain patterns happening in the given datasets. For instance, when calculating the rate of change for the opening price, y2 would be the opening price of the current day, y1 would be the previous day's opening price, and x2 - x1 would be the number of days. The slope is calculated as follows:

$$slope = \frac{y_2 - y_1}{x_2 - x_1}$$

Then this calculation would be repeated based on the fixed number of days. The following figure illustrates the calculation of the slope

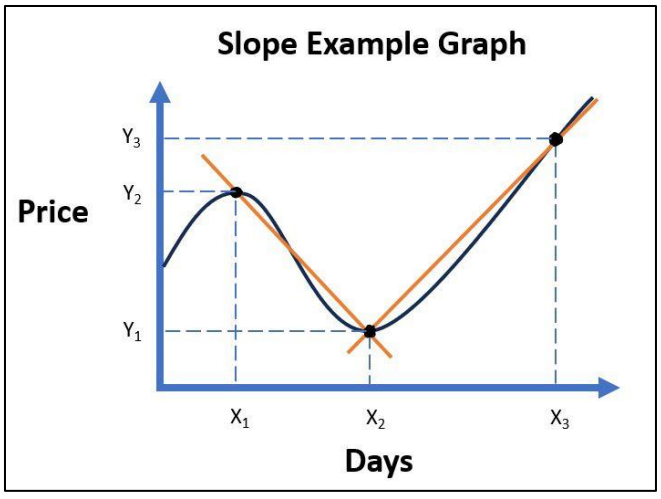

Figure 1 - Calculation of the slope

Average slope - Using the outcome of the calculation of the slopes, the average of those single slopes can be calculated, allowing one to determine whether the price will rise or fall. If the number is positive, the value increases. If the number is negative, the value falls.

Number of rises and falls - In this new feature, the individual calculated slopes are taken into account again. The number of slopes that have a positive or negative sign are counted. This means that the number of times the value of the share rises or falls in a certain time frame is counted. In this way, it is intended that the artificial intelligence in this work may be able to recognize patterns of behavior on the stock market in order to predict the further course. In the following illustration, the rising and falling of the price is shown with arrows.

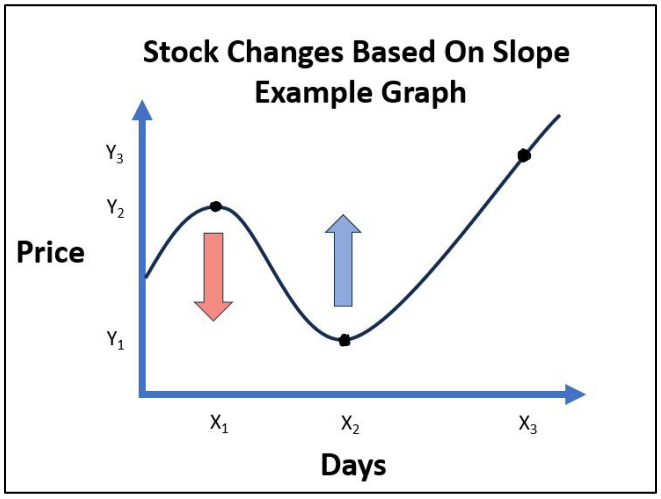

Figure 2 - Stock price changes based on slope

Difference between High and Low - Here the difference between the High and Low prices are calculated. This is done for the current day and for all past days of a certain time frame. The difference shows how much the price fluctuates within a day. This is intended to illustrate further behavior in the stock market in order to identify certain patterns.

Average difference between High and Low - The average difference between high and low is calculated. This summarizes whether the difference in the days is generally high.

Difference between Open and Close - Here the difference between the High and Low prices are calculated. This gives us more information based on the stability of the price. This is intended to provide further information that may be useful in exploring patterns for the artificial intelligence.

Average difference between Open and Close - Similar to the high and low values, an average difference between open and close is also calculated here over a specific time window.

C. Creating a label

The labels 1 and 0 are used to predict a rising or falling price. The labels are determined on the basis of the average rate of the low value. A positive slope is labelled 1 and a negative slope is labelled 0. The low value, which represents the lowest price on a day, is used to forecast based on a worst-case scenario.

D. Distribution of the dataset

In terms of training and testing the created model of artificial intelligence and the modified dataset will be split up 70% of the data will be used for training and 30% used for testing. The amount of training data is distributed as such because there needs to be as much data to train the AI. On the other hand, there must be enough testing data available to ensure an accurate testing result.

E. Implementation of the code for modifying and labelling the dataset

Several functions were declared for the realization of the project and subsequently applied. Their use is described below. The exact code can be found in the Jupyter notebook file, which is included in the appendix.

def combine_data(trade_datas):

This function is used to merge different datasets. A list of DataFrames is passed, which are combined into one at the end. Since datasets from different companies is used in this project, it is necessary to merge them in a standardized way.

Def modify_data(trade_data,t_previous_days, t_label_days)

This function is used to modify a data set. The purpose of modifying the data set is to extract more information from the given data set and use it as an additional feature. The new features are used, for example, to recognize temporal patterns. The label is also created in this function. To execute the functions, the dataset is required as well as the number of days immediately before and after the respective data point that are to be analyzed.

def plot_label_visualization(data):

This function displays the result of the automatically created labels. The progression of a price is displayed graphically. For each label, 3 data points are selected at random and displayed in the graph in the form of dots in different colours. A vertical line with the respective colour is then drawn for each data point, making it clear which point in time is being considered for a forecast.

F. Function of a Decision Tree

G. Function of a Random Forest

H. Implementation of the code to create the model

In this project, the decision tree and random forest are both created with the library sklearn and programmed in scratch. As the models are each trained and compared with different parameters, functions are defined for the executions. The following functions have been created to summarize all the individual steps involved in creating the respective models. The general procedure of the functions is very similar. To execute the function, a previously modified dataset is provided. Firstly, the dataset is split into test and training data. Then the model is created and trained with the training data. At the end, the trained model is tested with the test data and a value for the accuracy is created. At the end, the model, the predictions made during testing, the labels and the accuracy are returned. The declared functions of the respective models are presented in the following.

def run_decision_tree_classifier_by_sklearn(modified_ trade_data)

This function is about the creation of a decision tree classifier with the help of the module of the Sklearn library. The following function by Sklearn is used: ....

def run_random_forest_classifier_by_sklearn(modified_ trade_data):

This function is about the creation of a decision tree classifier with the help of the module of the Sklearn library. The following function by Sklearn is used: ....

def run_decision_tree_from_scratch(modified_trade_ data)

This function involves the creation of a decision tree classifier whose algorithm was implemented within the scope of this project. The exact algorithm will be explained in more detail later.

def run_random_forest_from_scratch(modified_trade_ data)

This function involves the creation of a random forest classifier whose algorithm was implemented within the scope of this project. The exact algorithm will be explained in more detail later.

The implementation in scratch is described below, starting with the decision tree.

class Tree_Node():

The class Tree_Node() represents a Node in a Decision Tree. It contains information about the splitting of the tree including the feature index, a threshold, and a subtree. Additionally, it contains the information gain resulting firm a splitting of a tree. Furthermore, it contains the label if it is a leaf node.

class Decision tree():

The Decision_tree class represents a Decision tree classifier which is implemented from scratch in the purpose of this project. It contains the parameters like mind_datas_branching and max_depth which describe how often a tree will be split and the maximum depth of the split. Furthermore, it contains the root of the previous branches.

def create_tree(self, dataset, curr_depth=0)

The function create_tree builds a Decision Tree recursively based on the given dataset. It uses values like min_datas_branching and max_depth as conditional values to decide whether conditions are met to calculate

the best split and build another subtree recursively. If the conditions aren't met, a leaf node will be created.

def get_best_branching(self, dataset, number_of_datas, number_of_features)

This method searches for the best split in a decision tree based on the dataset. It iterates through the features and their possible thresholds to calculate the best branching. It calculates the potential threshold for every feature, divides the dataset and calculates the information gain using whether the Gini index or the entropy. It updates the dictionary of the best branching based on the highest possible improvement in the information gain value.

def branch_tree(self, dataset, feature_index, threshold)

The function branch_tree divides a dataset based on a feature and a threshold. It creates two arrays in which the data smaller or bigger than the thresholds are separated.

def information_gain(self, parent, l_child, r_child, mode="entropy")

The information gain function is used to calculate the information gain after a splitting of a branch. For the calculation it uses ether the Gini index or the entropy, determined by the 'mode' parameter.

def entropy(self, y):

With the entropy function, disorders and randomness within a set of labels can be calculated. By examining the different types of labels and quantifying how unpredictable they are, it calculates a probability.

def gini_index(self, y):

With the function Gini index, similar like in the entropy function, the impurity or disorder is going to be measured. By iterating through the labels, it calculates the Gini index based on the probability of the occurrence of each of the labels. After applying a specific mathematical formula, it returns the Gini index.

def calculate_leaf_value(self, label):

The function calculate_leaf_value detects the label of a leaf note. It predicts the value by doing a majority voting of the labels that the leaf contains. The most frequent label will be returned as it's label.

def fit(self, data, label):

Withe the fit function, the Decision Tree is going to be trained. For this, the dataset and the labels are getting merged before using the function create_tree to establish the nodes.

def predict(self, data):

The function predict is used to create predictions based on the given dataset. It iterates through each data point and creates a prediction with the function make_predictions function.

def make_prediction(self, x, tree):

The make_prediction function is used within the predict function. It uses the Decision Tree structure and evaluates a single data point against the nodes of the tree. It compares the feature values with the nodes and finds the correct root through the tree. Finally, it finds the predicted label when reaching the leaf note and returns it.

The implementation of the random forest in scratch is now presented below. It should be mentioned that the decision tree programmed in scratch presented above is used to create the forest.

class RandomForest():

The class RandomForest is used to build a Random Forest model. It is implemented from scratch and create decision trees that are, as described above, implemented in scratch as well. The RandomForest class uses parameters like n_trees to describe the number of trees within the decision forest. Furthermore, it uses the parameters like max_depth and min_datas_branching which are necessary to create a decision tree. Furthermore, it contains a parameter which describes the number of features in the dataset and another which contains all trees in an array.

def fit(self, X, y):

The Random Forest gets trained by the function fit. It creates multiple Decision Trees for the forest. When creating the Decision tree, it uses the functions that are explained in the Decision Tree class section. After the creation of the individual decision trees the function samples() is going to be used to create a diverse subsets for the input data of each tree. This way, the Random Forest model has a higher variability among a tree which leads to more robustness and a higher predictive performance.

def samples(self, X, y):

After each Decision tree is created, the function samples() creates different training datasets for each tree. These datasets are created randomly out of the given dataset. This way, unique training datasets are created which leads to an individual training of the trees.

def identify_most_common(self, y):

This function detects the most frequently occurring label within a set of labels. These labels are created by all of the Decision trees in the Random Forest. The most common label will be returned.

def predict(self, X):

With the predict() function, the created Random Forest model will be used to predict the label of a given test dataset. The data will run through all created Decision trees in the forest. Each Decision tree will predict the label of the data. Using the predictions of all trees, the most common predicted label is going to be taken as the final predicted label.

IV. Evaluation and analysis

A. Receiver operating characteristic

B. Results

C. Comparison of the results

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

V. Related Works

VI. Conclusion

References

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi TIFF or EPS file, with all fonts embedded) because, in an MSW document, this method is somewhat more stable than directly inserting a picture.

To have non-visible rules on your frame, use the MSWord "Format" pull-down menu, select Text Box > Colors and Lines to choose No Fill and No Line.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.