AlphaInsights

April 5, 2021 | COMP 3710

Nandini Patel, Niraj Patel, Lama Khalil, Carson Dickie

Code repository : https://github.com/nirajp786/AlphaInsights

Contents

Introduction

Motivation

AlphaInsights aims to provide a solution to a problem. As COVID-19 hit the world and the economy fell many people lost their jobs. Stock trading can provide another source of income; however, trading profitably is difficult. Most people lack the time, knowledge and resources needed to invest in stocks effectively.

Our Goal

Our goal is to create a user-friendly software that, when provided with historical stock market information and a stream of live data, will identify profitable entry points for stock trades.

Importance

AlphaInsights aims to be a tool for generating reliable income. As more people look to branch out their sources of income many are turning to the stock market. For one to be successful and profitable in the market they must do hours of research and training. This causes many to either quit being traders or lose money. AlphaInsights aims to remove the complications the market presents and make trading simple and profitable for everyone.

Our Model

AlphaInsights will use modern artificial intelligence (AI) and machine learning techniques to predict stock price movement and identify opportunities for short term, “long” trades.

Why Agent-Based Modeling and Simulation (ABMS)?

To identify profitable trading opportunities, an algorithm must be able to take in stock market information, interpret that information, and provide recommendations to the user. Agent based models are the ideal candidates for such a scenario.

Components

- **Agent:** a python program designed to read stock data, interpret the data, and make recommendations based on the patterns identified and new information received.
- **Environment:** a time series of stock price and trading volume data for a given company.
- **Sensor:** data is obtained directly from Yahoo Finance.
- **Percepts:** the date, the opening, high, low, and closing price for the stock, and the volume of stock traded.
- **Decision Function:** a machine learning algorithm that identifies buying opportunities.
- **Actuators:** the agent can make recommendations, buy, and sell stock.
- **Constraints:** constraints, such as limiting the scope to one stock at a time, are used to shrink the problem space.

Background & Related Work

Graphical Overview

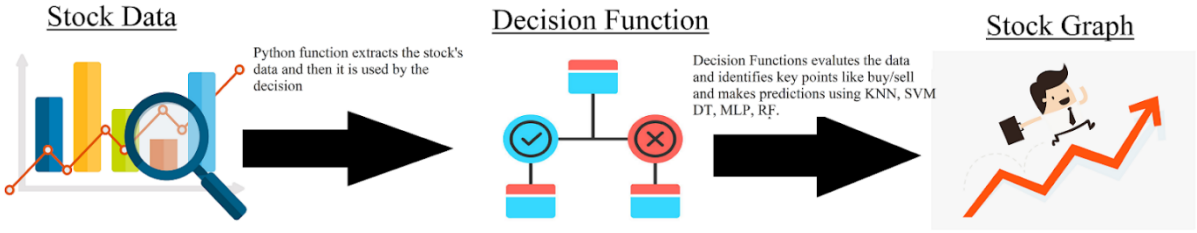


FIGURE 1 - ALPHAINSIGHTS MODEL

Machine Learning Techniques

Classification Problems

We treat stock market prediction as a supervised classification problem. Classification models identify patterns in labeled data (training set) before being exposed to unlabelled data. They approximate a mapping function $y = f(x)$, where (f) maps the input variable (x) to the output variable (y). In classification, the output domain consists of 2 or more discrete categories. For this model, the categories are "BUY = TRUE" and "BUY = FALSE".

K Nearest Neighbors (KNN)

KNN is a supervised learning technique that classifies the input variable (x) based on the identity of its K (nearest) neighbours. Data points are categorized based on their proximity to one another in the N-dimensional feature space. Both the number of neighbours (K) and the method of measuring proximity can be adjusted.

Support Victor Machine (SVM)

The main objective of this approach is to detect the hyperplane that generates the largest boundary between samples in the dataset. SVM is a powerful classifier, since it can be used with different kernels to fit the type of data that is being classified. Linear, polynomial, and RBF are some of the kernels that can be used with SVM.

Decision Tree (DT)

In this model the classification prediction is inferred from a binary tree chain of decisions. During this process, the dataset gets divided into smaller subsets, until a final decision is made.

Random Forest (RF)

The Random Forest is an extension of the Decision Tree model that uses a majority vote between several random decision trees to make predictions.

Multilayer Perceptron (MLP)

Multilayer Perceptron (MLP) is the simplest neural network architecture. It is suitable for data that is overlapping and cannot be separated using a single line. A multi-layer perceptron consists of many layers, each of which contains several neurons. MLP combines perceptron's simple architecture and neural networking complex and accurate performance.

Classifier evaluation

K-fold Cross Validation

K-fold cross validation evaluates the anticipated performance of the model. First, split the dataset into K equal subsets. One subset is selected to validate the model, while the remaining subsets are used to train the model. This process continues until all the subsets are used to validate the model.

Confusion Matrix:

A matrix that is used to compare a model's predictions to the actual identities of the training/test population. Metrics used to evaluate model performance, such as accuracy and precision, can be calculated using the matrix.

		Actual	
		+	**-**
Predicted	**+**	TPs	FPs
	-	FNs	TNs

- **True Positive:** the input variable (x) is correctly classified as a positive instance.
- **False Positive:** the input variable (x) is wrongly classified as a positive instance.
- **True Negative:** the input variable (x) is correctly classified as a negative instance.
- **False Negative:** the input variable (x) is wrongly classified as a negative instance.

Accuracy

A measure of the model's ability to distinguish between classes. Accuracy is computed as follows:

$$Accuracy = \frac{Sum\ of\ TP + Sum\ of\ TN}{Total\ numer\ of\ samples}$$

Positive Predictive Value (PPV) or Precision

A measure of the "trustworthiness" of a positive classification. Precision is calculated as follows:

$$Precision = \frac{Sum\ of\ TP}{sum\ of\ TP + Sum\ of\ FP}$$

Hyperparameters

Hyperparameters define and modify how machine learning models work. KNN, for example, has hyperparameters defining the number of neighbors to consider, the measure of proximity to use, etc. Optimal hyperparameters can be determined either by using intuition (an understanding of the data) or empirically. In this work we use grid search to optimize hyperparameters.

Grid Search

Grid Search generates a "grid" of hyperparameter combinations and tests each of them to identify the combination that produces the best results.

Related Work

Reviewing (Alkhatib et. al., 2013) helped us to determine the best hyperparameters for the KNN model. In this research The KNN algorithm with K=5 and Euclidean Distance function were applied on stock data. There was no statistical way to find the best K value in this paper. However, a decision was made

by plotting the k-values and error rates. Then, the K value that corresponds to the smallest error rate was chosen. It appears that the desired result was achieved with a small error ratio using this method. Thus, we started by trying the plotting method to obtain our best K value. Then we decided to use Grid Search, as it was more accurate.

We initially focused on using KNN for stock forecasting. However, we were later interested in including additional models in our study. For this reason, we reviewed (Chen and Hao et. al., 2017). This paper combined a modified version of K-NN, Feature Weighted KNN, with several other methods to predict future stock prices. Their additional methods included the Feature Weighted Support Vector Machine (FWSVM). It was used for stock data classification by assigning different weights to various stock features. This paper has helped us to see how useful and effective it can be to combine different classifiers. It has provided a lot of theoretical information to promote comprehension of the principles behind the classification algorithms. This allowed us to design and execute our code effectively in this project.

Finally, (Bafandeh et. al, 2013), helped us to distinguish between classification and regression methods. Further, this paper stated that KNN is a model that can be used for both classification and regression problems. The study mentioned that the classification technique is used to find out characteristics of groups to either understand initial data or to predict values. Thus, the predicted result is a discrete value. In the regression technique existing values are used to forecast future continuous values. A simple example is linear regression. It is a method allowing to summarize data between two variables using a straight line. However, many problems like this (stock prices) are complex, hence, thus, they need a much complex technique. Interestingly, the KNN algorithm can be used in both classification and regression problems.

Approach / Simulation Details

Environment

Coding Language: Python

Libraries: Scikit Learn, Matplotlib, Pandas, Numpy, ffmpeg

Project Management: Github, Microsoft Teams

Assumptions and Constraints

- Only “long” entries (buying followed by selling) are identified.
- All stock purchased is sold 10 days later, regardless of the price at that time.
- Commission fees are ignored.
- The model examines one stock at a time.

Approach

Data Retrieval

- The user is prompted to provide the stock ticker symbol for their company of choice.
- The model retrieves a time series of price data (high, low, close, volume, date) from Yahoo Finance and loads it into a Pandas Dataframe.

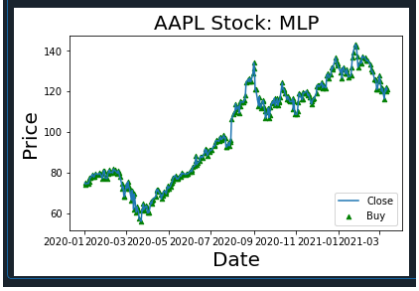
Preprocessing and Feature Engineering

- Labels are generated for each datapoint by checking whether the price is higher or lower 10 days after the date in question. If the price is higher the point is labelled as a 'BUY.'
- The user can (optionally) add moving averages of varying length for both price and volume.

Decision Function Optimization

- Five machine learning models (KNN, SVM, DT, RF, MLP) are tested using 10-fold cross validation.
- Confusion matrices are generated.
- The agent uses the model with the highest precision and accuracy to run the simulation.

```
==========================================
MLP Classifier:
[[ 12  89]
 [ 46 144]]
Precision = 0.6180
NPV = 0.2069
Specificity = 0.1188
Sensitivity = 0.7579
Accuracy = 0.5361
```




```
==========================================
Best Accuracy Among Five Classifiers
------------------------------------
Max Accuracy achieved by SVM classifier
Max Accuracy = 0.6529
==========================================
Best Precision Among Five Classifiers
------------------------------------
Max Precision Achieved By SVM classifier
Max Precision = 0.6529
==========================================
```

Simulation

- The agent identifies all "BUY" signals for a given timespan.
- The information is plotted on an animated price vs time chart; buying opportunities are identified as green arrows.
- The simulation determines the profit or loss generated assuming that the agent is given a $25, 000.00 starting balance and the ability to buy $1000.00 worth of stock each time a "BUY" signal is generated.
- Transactions are recorded in a ledger and reported to the user.

```
==========================================
SIMULATION USING BEST MODEL
----------------------------------
         Date       Balance WL     Shares    BuyPrice   SellPrice  ProfitLoss
0   2020-01-01  25000.000000  -   0.000000    0.000000    0.000000    0.000000
1   2020-01-02  25049.575441  W  13.317796   75.087502   78.809998   49.575441
2   2020-01-03  25121.188999  W  13.448543   74.357498   79.682503   71.613558
3   2020-01-06  25177.126359  W  13.342229   74.949997   79.142502   55.937360
4   2020-01-07  25241.840307  W  13.405274   74.597504   79.425003   64.713948
..         ...           ... ..        ...         ...         ...         ...
```

```
==================================
Stock Symbol:  AAPL
Number of Transactions:  201
Total Profit / Loss = 11701.54
==================================
```

Results and Discussion

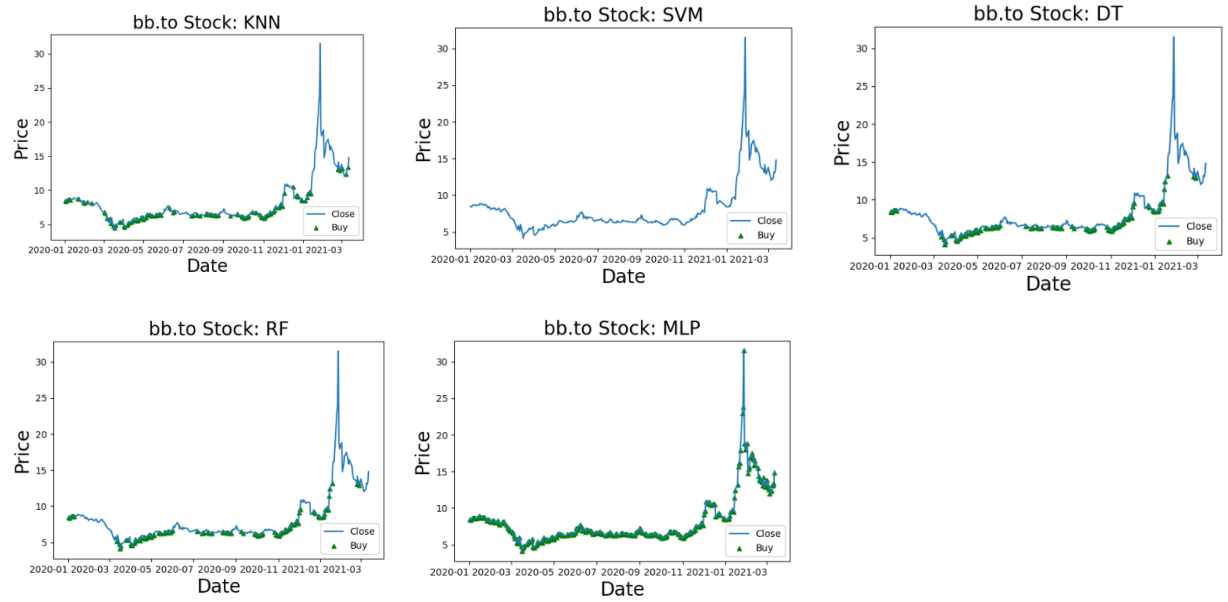
We will show the results of testing our model on the Black Berry's stock (bb.to)

Numerical Results

Classifier	Precision	NPV	Specificity	Sensitivity	Accuracy
KNN	0.5378	0.6221	0.6605	0.4961	0.8576

SVM	0.1879	0.4872	0.7037	0.0698	0.4427
DT	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
RF	1.0000	0.9939	1.0000	0.9922	0.9966
MLP	0.3862	0.5000	0.4506	0.4341	0.4433

Graphical Results



Transaction Ledger

```
========================================
SIMULATION USING BEST MODEL
----------------------------------
          Date       Balance WL      Shares  BuyPrice  SellPrice  ProfitLoss
0   2020-01-01  25000.000000  -    0.000000      0.00   0.000000    0.000000
1   2020-01-02  25052.071006  W  118.343195      8.45   8.890000   52.071006
2   2020-01-03  25097.309101  W  119.047619      8.40   8.780000   45.238095
3   2020-01-06  25108.843127  W  115.340254      8.67   8.770000   11.534025
4   2020-01-08  25112.299348  W  115.207373      8.68   8.710000    3.456221
..         ...           ... ..         ...       ...        ...         ...
125 2021-01-15  49016.639406  W   80.128205     12.48  17.959999  439.102484
126 2021-01-18  49441.639330  W   75.757576     13.20  18.809999  424.999924
127 2021-02-23  49450.813642  W   76.452599     13.08  13.200000    9.174312
128 2021-02-25  49454.570216  W   75.131480     13.31  13.360000    3.756574
129 2021-02-26  49604.531366  W   77.700078     12.87  14.800000  149.961150

[130 rows x 7 columns]
========================================
Stock Symbol:  BB.TO
Number of Transactions:  130
Total Profit / Loss = 24604.53
========================================
```

Discussion of Results

The Decision Tree (DT) classifier was best suited for this stock's dataset. Whereas the worst results were achieved by SVM. This is because Decision Trees are suitable for nominal features. DT classifiers operate by deriving the final decision from a series of data splits and consequent decision making. The gross

profit/loss of this model after 130 transactions was reported at $24,604.53. Which means that the model is making money.

It should be noted, however, that our test data was a subset of random points drawn from the same time interval as the training data, meaning that our model had information about data points occurring both before and after the unknown points. The model would only have access to data prior to the unknown point if you were truly aiming to predict future price changes. Such a model's accuracy would degrade over time, meaning that the model would have to be retrained for each new data point, which was not practical for our simulation.

We decided that a model's precision should be a key factor in determining which model performed the best. Precision quantifies the ratio of "BUY" signals that will produce a profit, which is an important metric. If the agent misses a buying opportunity it does not cost the user anything. A false "BUY" signal, however, will cost the client money.

A final observation: in all our test cases, even the models with poor accuracy and precision ultimately produced a profit when put under simulation.

Our concept was crafted for educational purposes and has not undergone the amount of testing necessary to warrant real-world implementation. No purchase or selling decisions should be based on our findings.

Future Work

None of our models were able to achieve high accuracy and precision consistently when applied to new data across a range of stocks. To improve the agent's effectiveness the decision function must be improved. This could be achieved through the engineering of better features or through the use of a more powerful model. We suspect that a deep learning model, such as a Long Short-Term Memory network (LSTM) might be best suited to the task.

References

[1] Stock price prediction using k-nearest neighbor (KNN) algorithm by Alkhatib, K., Najadat, H., Hmeidi, I., & Shatnawi, M. K. A.

[2] Chen, Y., & Hao, Y. (2017). A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction. Expert Systems with Applications, 80, 340-355.

[3] Imandoust, S. B., & Bolandraftar, M. (2013). Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background. International Journal of Engineering Research and Applications, 3(5), 605-610.