Supervised Learning and the Stock Market

Jose Hernandez

Student at University of Central Florida

November 26, 2020

Author Note

This literature review will cover the algorithms and processes of using a task of machine learning known as supervised learning. I will be covering and explaining what supervised learning is and the functions used to create supervised models. Specifically, I will be covering its application in the stock market and how it could be implemented.

Table of Contents

Abstract

Machine Learning is giving computers an ability to learn without being directly programmed for a task. Combining models, mathematics and data to construct a suitable value for those that are unknown or missing. In the stock market, it's the job of an analyst/broker to use their techniques and information to predict a stock price in the next hour, month or five years. Using machine leaning, supervised learning is a way to predict the outcome while given labeled data; the test will create a model using variables x & y and their relationship to come up with a prediction ŷ. Supervised learning has its challenges as one would need a labeled dataset. However, there is another method called unsupervised learning that using unlabeled data and different techniques and models to predict variables. I will only be focusing on supervised learning and its applications in the stock market. Additionally, I have mentioned a study conducted in Malaysia using KNN to predict stock price and interpret the method used and how it could be useful in the finance industry.

Keywords: Machine Learning, A.I, Supervised Learning, stock price prediction

What is Supervised Learning

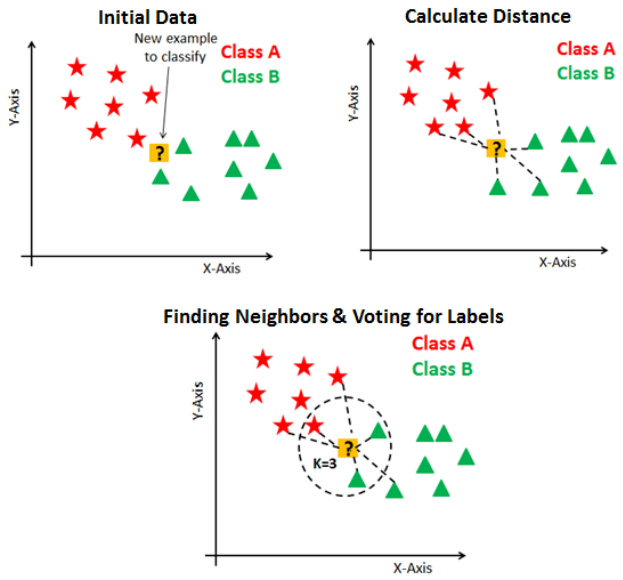
There are several branches when it comes to supervised learning, but the two I will discuss are: Regression & Classification. Clarifying these two; Regression in machine learning to sum means the output of the model one is obtaining is quantitative and Classification meaning the output is qualitative. The algorithms that are used in supervised learning obtain the types by differentiating with respect to the output variable (Quantitative or Qualitative). So, an input (data) could either be qualitative or quantitative, but the problem will only be determined depending on the output.

The learning the relationship between an input value that is qualitative or quantitative can be expressed as $\mathbf{x= [x_1\ x_2\ ...\ x_n]^T}$, then the output can also be expressed as a quantitative output variable *y.* One can't help but think about the variables the model is not able to classify/capture, we can make an additional variable (random) for error **ε**. Then when it comes to finding a model (***y*** : not yet seen output) we can represent the function as $\boldsymbol{y = f(x) + \varepsilon}$. With Regression there are a few algorithms that are used in the technique such as Lasso, Ridge, Loess, KNN, Spline and XGBoost regression algorithms. Then with Classification some algorithms are Logistic, SVM (Support Vector Machines), Random Forrest, and Hidden Markov classification algorithms. I won't go over all of these, but only the few that is widely used in data science and would be useful when trying to predict outcomes in the stock market. Ridge Regression: making an assumption that **ε** follows a Gaussian distribution with zero for mean and variance we can now write a least squares equation using a compact matrix and using vector notation as such:

$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \ \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2,$, then we can transform this into a modified minimization equation:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \ \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \gamma\|\boldsymbol{\beta}\|_2^2.$$ and $\gamma \geq 0$ is a parameter that is set by the program, also if set to zero we can originate to the basic least square problem. Ridge regression can be very like linear regression except that in creating the model we introduce a certain amount of bias; and extent to which the certain model comes up with a plot that is related to the samples that is initially inputted.

Another algorithm that is used for solving a classification and regression model is K-nearest neighbors. It can basically be described as if having a large group of points (2 groups: Green & Orange) with a new data point being created. The new data point is seeking to be assigned to one of the groups. The algorithm seeks out the nearest data points using a coordinate system and will associate the new data point with whatever group has more data points closer to it. Equations used while this test is running is finding Euclidean distances between your test points. $$x_{ij}^{\text{new}} = \frac{x_{pi} - \min(x_{ij})}{\max(x_{ij}) - \min(x_{ij})}, \qquad \forall j = 1, \ldots, p, \quad i = 1, \ldots, n.$$



The idea when using this algorithm with your input data is to choose a K (integer) that is not too large, but too small. One might choose to use k=1 to train their data as it might fit perfectly but is highly prone to overfitting. Choosing a k is difficult when conducting this test as you would like a model that is not overfit, but more ridged and has a less flexible decision boundary.

Supervised Learning Stock Market Applications

In the finance industry large banks and investment firm are gearing towards A.I and Machine Learning to improve their structure and automate transactions and investments. In a piece by efinancialcareers, “J.P. Morgan says supervised learning algorithms are provided with provided historical data and asked to find the relationship that has the best predictive power. Supervised learning algorithms come in two varieties: regression and classification methods.”. Here, J.P Morgan can use Machine Learning techniques to help predict quantitative output of certain stocks. They could use regression-based learning by implementing passed data (input) to predict perhaps a stock spike or rally that might occur (output.). Ex: Annual/Seasonal stock price stocks could be predicted on certain days/certain price % movement based on past performance with a degree of confidence. A classification method would also work well using the same concept, but the opposite way by trying to determine what category a set of given classification belong to.

Now, when it comes to a stock price and what exactly triggers its movement is quite versatile in its variables that actually effect it. Besides the known variables such as the number of banks, institutions, and independents that own a stake in a stock. Information seems to trigger the largest price fluctuations. Ex: TESLA could go its first quarter (3 months out of the year) without reporting any news or information on the company’s performance, and the price actions one would see throughout the first quarter would be cause by whomever decided to sell/buy the stock. There is only a certain amount of shares a company is issued, and a negative/positive price swing could be determined by a machine learning algorithm that would determine if a price drop

will occur (large sell of stock) or price jump will occur (amount of purchased shares is reaching the market cap).

I will be focusing specifically on the *k*-Nearest Neighbor Classifier and how it could be used to determine/predict a stock price. I will assume the price prediction will only be determined by short and medium ranged time frames. I do not think that these classifiers could determine volatile variables for stock for a time frame longer than 5 years. A company could go bankrupt, be involved in criminal acts, or executive action could have the company be bought by another and cause massive flux in price.

Now, to use the KNN classification we must map out the stocks past data and then a construct tests data that will be made into several sets of vectors. The vectors that we will create will be represented by the stock's characteristics. We will represent this variable as *N*. Our integer that we will determine '*k'* will be the number of nearest neighbors we will be seeking using Euclidian distance equations. Lastly, it will act as a poll for the class of labels from the *k* nearest neighbors and assign a prediction value. A range I believe is good for estimating the next price is next day. An equation that I believe you must use when predicting next day price is the Root Mean Square Deviation. We need to obtain a single value of measurement and the RMSE is the standard deviation of residuals that are a distance from the regression line. To sum, it tells us how concentrated the data is around a best fit line. Z_{fi} & Z_{fo} *:*are the estimated values and actual results that will become the differences squared. Another thing to note about KNN is that it's a non-parametric ML algorithm, which means that each prediction that is made is from a new instance. Hence, k will be seeking the nearest instance to apply a poll (vote). The most common distance function used with this method is Euclidean. Although, there are others one

$$\mathrm{RMSE}_{fo} = \left[\sum_{i=1}^{N} (z_{f_i} - z_{o_i})^2 / N\right]^{1/2}$$

may use such as Manhattan, Minkowski and Hamming. In this formula, we have x & y which turns out to be the data vectors, and our variable *k* is our number of attributes. A directly related

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

experiment was conducted by Lock Siew Han & MD Jan Nordin in Malaysia, "A study on a KNN approach that made use of economic indicators and classification techniques to predict the stock price trends yielded considerable precision. Four indicators were identified and calculated based on the technical indicators and their formulas. The values of the indicators were normalized in the range between -1 and +1. Accuracy and F-measure were calculated to evaluate the performance of the model. Calculation of these evaluation measures required estimating Recall and Precision which were assessed from True Negative (TN), False Negative (FN), True Positive (TP) and False Positive (FP). The performance of the KNN model was improved by using the optimal value for the *k* parameter.". To explain, here we have the F-measure which is the harmonic mean of the precision and a tests accuracy. The worst type of F-measure score is around 0.0 and the best to near perfect is 1.0. So, here we have some results on how accurate using the K-nearest neighbors test is. While using a k parameter of 25 there is a 0.82 accuracy that the data point by the model will be placed in the correct classification (group). And as we go on through to k = 45 and k = 70, we have just about the same amount of accuracy with the F-measure. I am unsure of how large the initial data set the input is, but there seems to be no issue also with overfitting with their used data.

Table 1: KNN Model Performance.

Measurement	K Parameter		
	25	45	70
Accuracy	0.8138	0.8059	0.8132
F-measure	0.8202	0.8135	0.8190

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{TP}}{\mathrm{TP} + \frac{1}{2}(\mathrm{FP} + \mathrm{FN})}$$

Conclusions

To summarize, when it comes to supervised learning one will start will input data that is filled with target values. These target values are value that are initially correct. Hence, the historical prices of a given stock would be classified as the target value (trained data) because it has been recorded to reach those prices. Next, is developing a model with high accuracy and that won't result in over fitting (deciding a k value.) Ex: picking a k value that will look for how many instances the stock has reach a price of $43 or $42, then classifying the next datapoint with the highest voted instance. Clarifying over and underfitting; overfitting is best described as the model one is using doesn't generalize the data it hasn't reached yet (too small of a *k*). Then Underfitting is quite the opposite, the model is not complex enough to capture the trends in your dataset. To use KNN with stock market data would be best for a regression problem, and to use a loss function to minimize the badness and find a best fit. The loss model as I stated above would be using the formula Root Mean Square Error (RMSE). Which correlates to the Euclidian distance equations between the data points. The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. Being more of an instance-based learning, where the function is only approximated locally, and all computation is delayed until the function is evaluated. The algorithm relies on distance for classification, also normalizing the training data can improve its accuracy dramatically. Lastly, to relate with stock market function and accuracy the study performed in Malaysia of using KNN approach in the use of economic indicators to predict stock price trends has shown incredible precision given how volatile and random the stock market can be.

References

-Lindholm Andreas, Wahlstrom Niklas, Lindsten Fredrik, Schon Thomas. (2019). Supervised Machine Learning. Pages 10-45.

-Faggella, Daniel (2020). Machine Learning in Finance-Present and Future Applications: https://emerj.com/ai-sector-overviews/machine-learning-in-finance/

-Mohri Mehryar, Rostamizadeh Afshin, Talwalkar Ameet. (2018). Foundations of Machine Learning.

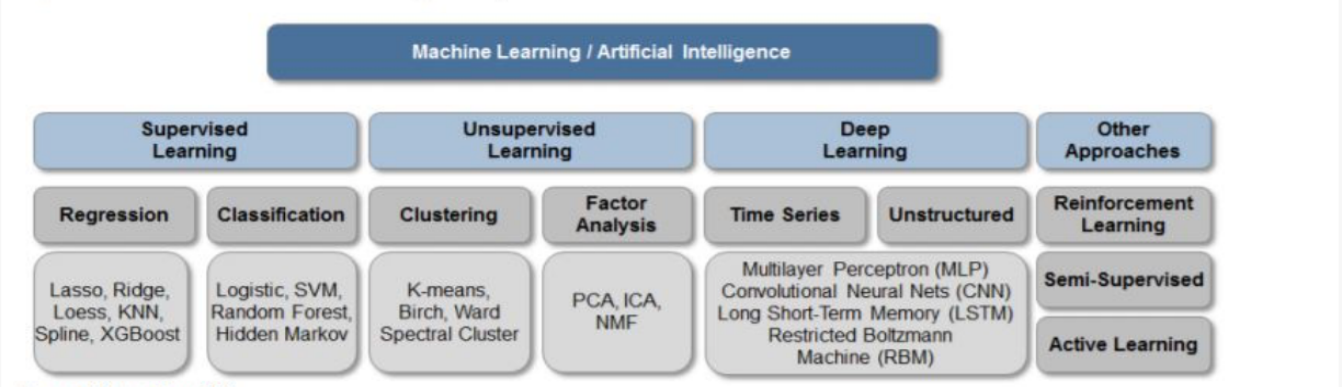
-Butcher, Sarah. (2017). JPMorgan's massive guide to machine learning and big data jobs in finance.

Lock Siew Han, MD Jan Nordin. (2005-ongoing). Predicting the Stock Price Trends Using a K Nearest Neighbors-Probabilistic Model.

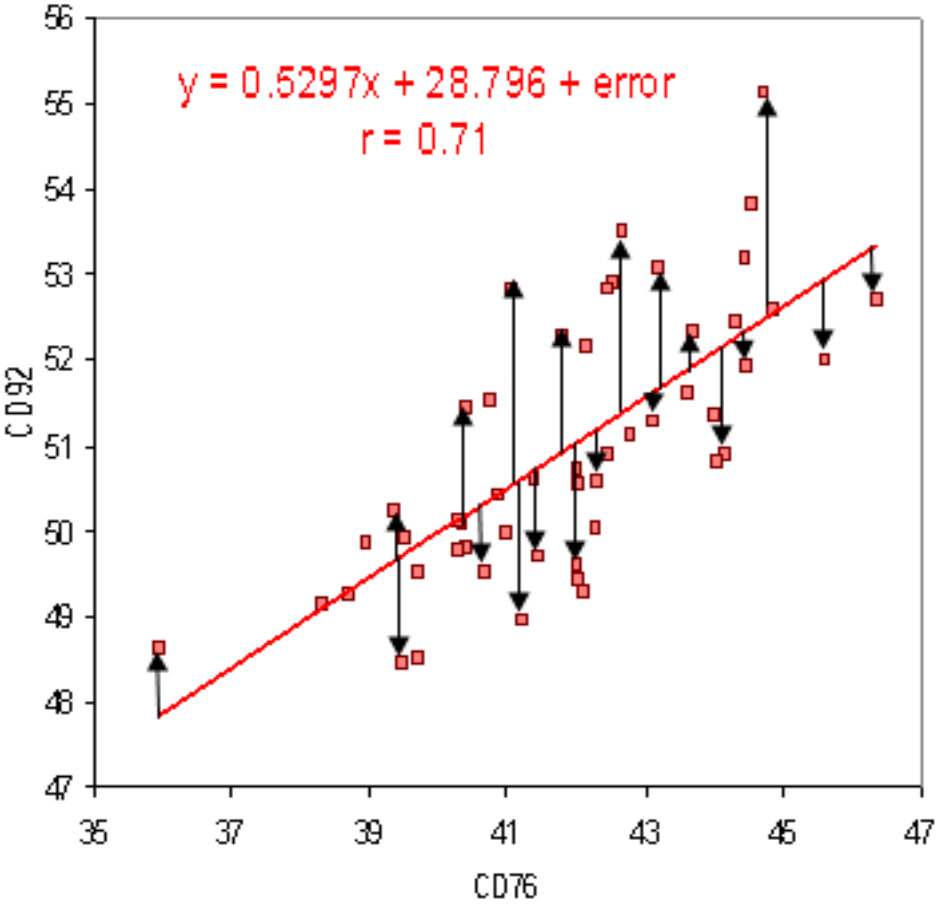
http://www.jatit.org/volumes/Vol96No18/29Vol96No18.pdf

Figures

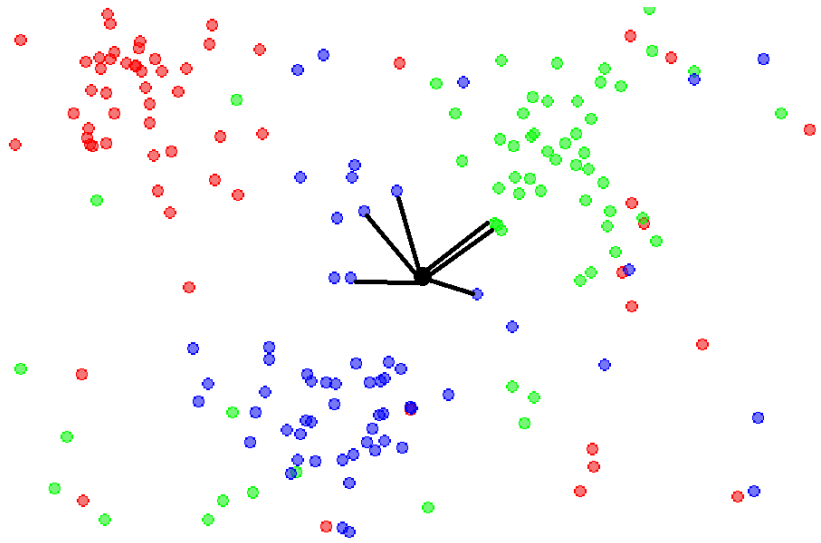
Topics in Machine Learning & Artificial Intelligence



Root Mean Square Error (Example Visualization)



How the KNN algorithm is visualized



Distance Functions used in KNN

Distance functions

Euclidean

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski

$$\left(\sum_{i=1}^{k}\left(|x_i - y_i|\right)^q\right)^{1/q}$$