# Stock Market Prediction for Companies Working in the Biotechnology Industry

Shabnam ZareShahraki

August 22, 2022

## 1 Introduction

This project is the final project of my bachelor's degree in Computer Engineering, done under supervision of Dr. Nastooh Taheri Javan. As a sophomore student, I was introduced to the exciting and rapidly-growing field of Machine Learning. Ever since then, I have been studying the fundamentals of Python programming, Machine Learning, Statistics, and Mathematics hoping to gain deep knowledge. A few of my study materials and courses I used during my learning process include Machine Learning Specialization by Stanford University and DeepLearning.io, Intro to Machine Learning by Kaggle, Intermediate Machine Learning by Kaggle, and Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. Applying the knowledge I had gained, I chose this project and have been working on it for some time.
In the following sections, a detailed explanation of each step of the process has been written. In addition, the code can be found here.

## 2 Problem Definition

This project aims to compare the performance of various supervised learning approaches to stock market prediction. In this project, both classification, as well as regression algorithms are applied. Limiting the boundaries of the project, only stock market data of companies active in the biotechnology field are taken as input. The target variable is the Close feature, which specifies the closing price of a stock. While in the case of the classification algorithms, the purpose is to predict whether the Close feature increases or decreases on the following day, for the regression algorithms the motive is to forecast the exact closing price.

## 3 Data Collection

To perform any data-related task, the initial step is to acquire the data. Firstly, a data set containing all tickers and information of companies in the U.S. stock market was found on Kaggle. The original data set can be accessed by clicking on the link below: View Initial Data set on Kaggle

These companies are categorized by multiple factors, one of which is their industry. There are 491 indices whose industry is recorded as biotechnology. Therefore, the names and tickers of those were extracted and saved into a separate .csv file. Next, stock market data of the previously extracted companies were retrieved from reliable sources, namely Yahoo Finance and IEX Cloud. Unfortunately, 119 of 491 companies do not have any data recorded available for free, thus these companies' data are yet to be found. The retrieved data is stored as .csv files to be used as input for the rest of the project.
All notebooks relating to this section are accessible through the link below:
Data Collection Notebooks

## 4 Returns Analysis

Returns Analysis consists of analyzing each company's accumulative return, daily return, comparing their percent return throughout the period, Sharpe ratio, Sortino ratio, and Probabilistic Sharpe Ratio.

## 4.1 Initial Exploration

Initial exploration is done in order to study all the features and their impact on the target or prediction.

As was seen in the code, our data is a Time series, whose index is of type DatetimeIndex, and columns are of type float. Additionally, the most number of observations recorded on a single company is 1258, although not all of them have as many indices.

## 4.2 Examining Correlations

Correlations among the features needed to be studied. Therefore, a heatmap indicating the correlations was plotted. In all illustrations, a high correlation among 4 features, Close, Open, High, and Low was seen. In most cases, another feature, Adjacent Close, also revealed a high correlation with the Close feature.

This visualization confirms the initial suspicion that four features are highly correlated, and thus should be eliminated, since it may result in "Multicollinearity". Multicollinearity can lead to misleading results in some algorithms. At the end of this phase, there are 3 features remaining, Close, Adj Close, and Volume. The reason Adj Close will not be dropped at this point is that it will be used in some computations later on.

All photos showing the correlations among each company's features are stored here.

## 4.3 Daily and Accumulative Returns Analysis

Firstly, each company's accumulative return was computed and a line diagram was generated and stored. Here is an example showing a company's accumulative return:

All the plots can be viewed here. Secondly, each company's daily return throughout the time period was computed and a respective line diagram was generated and stored. The companies' daily returns were diverse. All the plots can be viewed here. Next, the percentage change of all companies was computed. The result of this phase was also surprising. Although some indices demonstrated great positive changes, there were some others that showed highly negative numbers.

## 4.4 Sharpe Ratio

The Sharpe ratio can be used to evaluate the total performance of an aggregate investment portfolio or the performance of an individual stock. The Sharpe ratio indicates how well an equity investment performs in comparison to the rate of return on a risk-free investment, such as U.S. government treasury bonds or bills.

$$\text{SR} = \frac{E[R_a - R_f]}{\sigma_a}$$

SR $\rightarrow$ Sharpe Ratio
$R_a \rightarrow$ Asset Return
$R_a \rightarrow$ Risk Free Return
$\sigma_a \rightarrow$ Standard Deviation of Asset Return
$R_f \approx 0 \implies$

$$\text{SR} = \frac{E[R_a]}{\sigma_a} \rightarrow \text{SR} = \frac{Mean\ Asset\ Return}{Standard\ Deviation\ of\ Asset\ Return}$$

## 4.5 Sortino Ratio

In contrast to Sharpe ratio, Sortino ratio only punishes returns falling below a user-defined threshold. Additionally, instead of inspecting the complete standard deviation, Sortino ratio only computes the downside risk, i.e the standard deviation of all returns which are smaller than the target.

## 4.6 Probabilistic Sharpe Ratio

The problem of Sharpe ratio is, that it is calculated via historical data and thus it only yields an estimation and not the true Sharpe ratio.

To tackle this problem you can use the Probabilistic Sharpe Ratio. It is defined as

$$\text{cdf}\Big(\frac{(\widehat{\text{SR}} - SR^*)}{\hat{\sigma}(\widehat{SR})}\Big)$$

where cdf is the cumulative density function of the normal function, $\widehat{SR}$ is the original Sharpe ratio, $SR^*$ is the benchmark Sharpe ratio (often 0), and $\hat{\sigma}(\widehat{SR})$ is the standard deviation of the estimated Sharpe ratio which is computed as follows:

$$\hat{\sigma}(\hat{SR}) = \sqrt{\frac{1}{n-1}\Big(1 + \frac{1}{2}\widehat{\text{SR}}^2 - \gamma_3\hat{\text{SR}} + \frac{\gamma_4}{4}\widehat{\text{SR}}^2\Big)}$$

Here, $\gamma_3$ and $\gamma_4$ correspond to skew and fisher kurtosis.

In words, Probabilistic Sharpe ratio computes the probability that the true Sharpe ratio is $\leq$ the estimated Sharpe ratio ($PSR = P(SR \leq \widehat{\text{SR}})$) given the benchmark Sharpe ratio.

All notebooks relating to this phase are available through the link below:
Returns Analysis Notebooks

## 5 Preprocessing

This phase is done to prepare the data for the next phase, which is modelling. The implementation of this step is here.

### 5.1 Lag Features

To make a lag feature, the observations of the target series are shifted so that they appear to have occurred later in time. In this project, 5 lag features are created. Lag features let us model serial dependence. A time series has serial dependence when an observation can be predicted from previous observations, which is the primary goal.

### 5.2 Other Features

In addition to lag features, I assumed other factors affect stock market prices, as well. Therefore, not only lag features but also Volume, Daily Return, Monthly Moving Average, and Quarterly Moving Average were also added to the Dataframes before being input to models. This step resulted in tremendous performance improvement.

### 5.3 Feature Scaling

#### 5.3.1 Z-Score Normalization

This step is done in order to prepare our data for the next phases. In this project, Z-score Normalization is performed on the Close feature in order to make the values comparable. The equation of this method of normalization is shown below:

$$\text{Normalized} = \frac{\big(X - \mu\big)}{\sigma}$$

$\sigma \rightarrow$ Standard Deviation
$\mu \ \rightarrow$ Average(Mean)

#### 5.3.2 Min Max Normalization

Another way of normalizing the values is to apply Min-max normalization. The equation is as below:

$$\text{Normalized} = \frac{\big(X - \textbf{Min}\big)}{\textbf{Max - Min}}$$

# 6 Models

Simply put, a Time series is a set of observations collected over time. In this project, data from 21-08-2017 to 18-08-2022 has been recorded.
Lag Feature $\rightarrow$ to make a lag feature we shift the observations of the target series so that they appear to have occurred later in time.

## 6.1 Classification

### 6.1.1 Logistic Regression

Logistic Regression estimates the probability of that an instance belongs to a particular class, making it a Binary Classification algorithm. If the estimated probability is greater than a threshold (usually 0.5), then the model predicts that the instance belongs to the *Positive Class*, otherwise it is classified as belonging to the *Negative Class.*
This model computes a weighted sum of the input features, X, plus a bias term, b, then outputs the *logistic* of this result. The logistic is a *sigmoid function* which outputs a number between 0 and 1. The Equation is shown below:

$$g(x) = \frac{1}{1 + e^{-(w.x+b)}}$$

The cost function over the whole training set is the average cost over all training examples. The equation is as follows:

$$J(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} log(f_{w,b}(x^{(i)})) + (1 - y^{(i)})(log(1 - f_{w,b}(x^{(i)}))\right)$$

Regularized:

$$J(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} log(f_{w,b}(x^{(i)})) + (1 - y^{(i)})(log(1 - f_{w,b}(x^{(i)}))\right) + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

To optimize the algorithm, *Gradient Descent* is utilized, which tweaks the parameters attractively to minimize the cost function. The equation is shown below:

$$w_j = w_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}w_j$$

$$b = b - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})$$

The Logistic regression model applied in this project can be accessed through this link.

### 6.1.2 Gradient Boosting

Gradient boosting is a method that goes through cycles to iteratively add models into an ensemble. It begins by initializing the ensemble with a single model, whose predictions can be pretty naive. The cycle begins:

- The current ensemble is used to generate predictions for each observation in the data set. Adding the predictions from all models in the ensemble, a prediction is made.
- These predictions are used to calculate a loss function.
- As done in Logistic Regression, Gradient Descent is used to optimized the parameters.
- A new model is added to the ensemble.
- These steps are repeated!

XGBoostClassifier model has been implemented and is available here.

|  | Predicted |  |
|---|---|---|
| Actual | 1 | 0 |
| 1 | True Positive | False Positive |
| 0 | False Negative | True Negative |

Table 1: Confusion Matrix

#### 6.1.3 Neural Networks

A sequential neural network model is used to predict whether the close price inclines or declines the next day. Here, an 8-layered neural network is constructed, the first 7 layers(Input and Hidden layers) of which have "relu" activation functions and the final layer has a "linear" activation unit. The number of units in each layer is 70, 60, 50, 40, 30, 20, 10, and 2, respectively.
The final layer of the neural network generates 2 outputs. One output is selected as the predicted answer. In the output layer, a vector **z** is generated by a linear function which is fed into a softmax function. The softmax function converts **z** into a probability distribution as described below. After applying softmax, each output will be between 0 and 1 and the outputs will sum to 1. They can be interpreted as probabilities. The larger inputs to the softmax will correspond to larger output probabilities. The softmax function can be written:

$$a_j = \frac{e^{z_j}}{\sum_{k=0}^{N-1} e^{z_k}}$$

Where $z = \mathbf{w} \cdot \mathbf{x} + b$ and N is the number of feature/categories in the output layer. The implementation of this model is accessible here.

## 6.2 Regression

### 6.2.1 Gradient Boosting

Extreme Gradient Boosting algorithm, which has already been explained the previous section, operates the same way for regression problems. Therefore, I will not repeat the details once more.
XGBoostRegressor model has been implemented and is available here.

### 6.2.2 Neural Networks

Same as before, a sequential neural network model is used to predict whether the close price, however, in this case, our incentive is to predict the exact close price of each stock. Here, an 10-layered neural network is constructed, the first 9 layers(Input and Hidden layers) of which have "relu" activation functions and the final layer has a "linear" activation unit. The number of units in each layer is 90, 80, 70, 60, 50, 40, 30, 20, 10, and 1, respectively. The final layer containing a single linear unit outputs the prediction. The related notebook can be viewed here.

## 7 Model Evaluation

To Evaluate the classification models, a confusion matrix is generated for each company's data. Additionally, Precision, Recall, and F1 Score are also computed.
Evaluation of regression models, on the other hand, cannot be done using the means explained in the last paragraph. Here, Mean Squared Error is calculated, a common method for regression models.

## 7.1 Evaluating Classification Models

### 7.1.1 Confusion Matrix

A confusion matrix is a simple table that is often used to evaluate the performance of a classification algorithm. A sample confusion matrix is demonstrated below:

### 7.1.2 Precision and Recall

Precision answers the question "what fraction of the predictions are accurate?". In other words, it is the total number of true positives divided by the number of predicted positives. The equation is shown below:

$$\text{Precision} = \frac{\textbf{True Positives}}{\textbf{True Positives+False Positives}}$$

On the other hand, Recall determines the fraction of all positives correctly detected, i.e., the total number of true positives divided by the number of actual positives. The equation is shown below:

$$\text{Recall} = \frac{\textbf{True Positives}}{\textbf{True Positives+False Negatives}}$$

Both Precision and Recall used to measure the effectiveness of a learning algorithm.

### 7.1.3 F1 Score

F1 Score combines Precision and Recall to measure the accuracy of the algorithm. It takes the average with an emphasis on the lower value. The equation is as follows:

$$\text{F1 Score} = \mathbf{2} \times \frac{\textbf{Precision} \times \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}}$$

## 7.2 Evaluating Regression Models

### 7.2.1 Mean Squared Error

To evaluate the performance of a regression model, the value of each prediction made is subtracted from the actual value, then is squared to avoid negative values. The sum of all these losses is computed and divided by the number of examples, culminating in the mean squared error. The complete equation can be seen below:

$$\text{Mean Squared Error} = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y} - y\right)^2$$

## 8 Results

To make this report concise, all results are saved in separate files, accessible here

## References

[1] Stanford University, DeepLearning.io (2022) *Machine Learning Specializtion.*

[2] Kaggle (2022) *Intro to Machine Learning.*

[3] Kaggle (2022) *Intermediate Machine Learning.*

[4] Kaggle (2022) *Timeseries.*

[5] Aurélien Géron (2019) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, O'Reilly Media, Inc., 2nd Edition

[6] Jose Portilla (2021) *Python for Finance and Algorithmic Trading with QuantConnect*, Udemy