STOCK PRICE PREDICTION USING LASSO REGRESSION ALGORITHM

**A PROJECT REPORT**

***Submitted by***

|  |  |
| --- | --- |
| **S. VIVEKANANTHAN** | **(920120205018)** |
| **R. MUKESH** | **(920120205301)** |
| **P. PARTHIPAN** | **(920120205302)** |

***in partial fulfillment for the award of the degree***

***of***

# **BACHELOR OF TECHNOLOGY**



***in***

**INFORMATION TECHNOLOGY**

**BHARATH NIKETAN ENGINEERING COLLEGE,**

# **AUNDIPATY, THENI**

**ANNA UNIVERSITY : CHENNAI 600 025**

MAY 2024

**ANNA UNIVERSITY : CHENNAI 600 025 BONAFIDE CERTIFICATE**

Certified that this project report **“STOCK PRICE PREDICTION USING LASSO REGRESSION ALGORITHM”** is the bonafide work of **S. VIVEKANANTHAN (920120205018), R. MUKESH (920120205301), P. PARTHIPAN (920120205302),** who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE**  Mrs. K. SARAVANASELVI, ME.,  **HEAD OF THE DEPARTMENT**  Information Technology,  Bharath Niketan Engg College, Aundipatty  Theni. | **SIGNATURE**  Dr. A. AMUDHA, ME,Ph.D.,  **SUPERVISOR**  Information Technology,  Bharath Niketan Engg College, Aundipatty  Theni. |

Submitted for the ANNA UNIVERSITY viva-voce Examination held on -- at Bharath Niketan Engineering College, Aundipatty.

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We thank the most graceful creator of the universe, our almighty GOD, Who ideally supported us ideally thought this project.

At this moment of having successfully completed our project, we wish to convey our sincere thanks to the management and our Chairman **Dr. S. MOHAN** who provide all the facilities to us.

We would like to express our sincere thanks to our Principal **Dr.P.V.ARUL KUMAR, M.E, M.B.A, Ph.D**., for letting us to do our project and offering adequate duration in completing our project.

We are also grateful to **Mrs. K.SARAVANASELVI,ME.,** our Head of the Department for his constructive suggestions during our project with deep sense of gratitude.

We extend our sincere thanks to our guide Dr **A. AMUDHA,ME.,Ph.D.,**Department of Information Technology who is our light House in the vast ocean of learning with her inspiring guidance and encouragement that helped us complete the project.

We would like to express our gratitude to all the teaching and non- teaching staff members of Information Technology Department and our friends for that kind help extended to us.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **C CCHAPTER** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** | 6 |
|  | **INTRODUCTION** | 7 |
|  | * 1. General Introduction |  |
|  | * 1. Project Objectives |  |
|  | * 1. Problem Statement |  |
|  | **SYSTEM** **PROPOSAL** | 9 |
|  | * 1. Existing System |  |
|  | * + 1. Disadvantages |  |
|  | * 1. Proposed System |  |
|  | 2.2.1 Advantages |  |
|  | * 1. Literature Survey |  |
|  | **SYSTEM DIAGRAMS** | 19 |
|  | * 1. Architecture Diagram |  |
|  | * 1. Flow Diagram |  |
|  | * 1. UML Diagrams |  |
|  | **IMPLEMENTATION** | 26 |
|  | * 1. Modules |  |
|  | * 1. Modules Description |  |
|  | **SYSTEM** **REQUIREMENTS** | 30 |
|  | * 1. Hardware Requirements |  |
|  | * 1. Software Requirements |  |
|  | * 1. Software Description |  |
|  | * 1. Testing of Products |  |
|  | **CONCLUSION** | 48 |
| **7.** | **FUTURE** **ENHANCEMENT** | 49 |
| **8.** | **SAMPLE** **CODING** | 50 |
| **9.** | **SAMPLE** **SCREENSHOT** | 53 |
| **10.** | **REFERENCES** | 58 |

**AB­­­­STRACT**

Accurate prediction of stock market returns is a very challenging task due to volatile and non-linear nature of the financial stock markets. With the introduction of artificial intelligence and increased computational capabilities, programmed methods of prediction have proved to be more efficient in predicting stock prices. The financial data: Open, High, Low and Close prices of stock are used for creating new variables which are used as inputs to the model. The models are evaluated using standard strategic indicators: RMSE and MAPE. The low values of these two indicators show that the models are efficient in predicting stock closing price. The Stock Market prediction task is interesting as well as divides researchers and academics into two groups those who believe that we can devise mechanisms to predict the market and those who believe that the market is efficient and whenever new information comes up the market absorbs it by correcting itself, thus there is no space for prediction. The artificial intelligence and increased computational capabilities, programmed methods of prediction have proved to be more efficient in predicting stock prices. In existing system, Artificial Neural Network and Random Forest techniques have been utilized for predicting the next day closing price.in this system, it doesn’t efficient for large number of dataset. The predictive result is low. In our process, we have to take the input from the input dataset. The input dataset is time series dataset. It contains the open, high, low and close price. After that we have to implement the regression algorithms. The regression such as Support Vector regression and lasso regression. The final result, we have to find the error values. In time series dataset, we should find error values. Next, we have to visualize the data.

**CHAPTER 1**

**INTRODUCTION**

* 1. **General Introduction:**

The macroeconomic environment and the financial market are complex, evolutionary, and non-linear dynamical systems. The field of financial forecasting is characterized by data intensity, noise, non-stationary, unstructured nature, and hidden relationships. Predicting financial indicators is therefore a difficult task. However, forecasting is important in the sense that it provides concrete data for investment decisions. How can we predict whether the price of a particular stock will go up or down in the upcoming year? In the modern techniques, one way is to develop a predictor based on the information in the historical data.

First of all, we should selected some major factors that may influence the performance of the stocks, we can further discover an interesting model from our dataset to predict the future performance of any stocks. That is to say, we need to learn a model that can map those factors into the class attribute which indicates the whole performance of stocks.

Support vector machine (SVM) is a machine learning technique that can be used for this purpose of classification. Established on the unique theory of the structural risk minimization principle to estimate a function by minimizing an upper bound of the generalization error, SVM is shown to be very resistant to the over-fitting problem, eventually achieving a high generalization performance.

Another key property of SVM is that training SVM is equivalent to solving a linearly constrained quadratic programming problem so that the solution of SVM is always unique and globally optimal, unlike neural networks training, which requires nonlinear optimization.

Stock market is characterized as dynamic, unpredictable and non-linear in nature. Predicting stock prices is a challenging task as it depends on various factors including but not limited to political conditions, global economy, company’s financial reports and performance etc. Thus, to maximize the profit and minimize the losses, techniques to predict values of the stock in advance by analysing the trend over the last few years, could prove to be highly useful for making stock market movements.

Traditionally, two main approaches have been proposed for predicting the stock price of an organization. Technical analysis method uses historical price of stocks like closing and opening price, volume traded, adjacent close values etc. of the stock for predicting the future price of the stock. The second type of analysis is qualitative, which is performed on the basis of external factors like company profile, market situation, political and economic factors, and textual information in the form of financial new articles, social media and even blogs by economic analyst

Now a days, advanced intelligent techniques based on either technical or fundamental analysis are used for predicting stock prices. Particularly, for stock market analysis, the data size is huge and also non-linear. To deal with this variety of data efficient model is needed that can identify the hidden patterns and complex relations in this large data set. Machine learning techniques in this area have proved to improve efficiencies by 60-86 percent as compared to the past methods.

* 1. **Objectives:**

The main objective of our project is,

* To predict or to forecast the stock price effectively.
* To implement the different machine learning algorithms for better performance.
* To enhance the overall performance for classification algorithms.

**CHAPTER 2**

**SYSTEM PROPOSAL**

* 1. **EXISTING SYSTEM:**

In existing system, accurate prediction of a stock price is a challenging task due to the complexity, chaos, and non-linearity nature of financial systems. In this brief, we proposed a multi-indicator feature selection method for stock price prediction based on Pearson Correlation coefficient (PCC) and Broad Learning System (BLS), Named the PCC-BLS framework. Firstly, PCC was used to select the input features from 35 features, including original stock price, technical indicators, and financial indicators. Secondly, these screened input features were used for rapid information feature extraction and training a BLS. The financial data: Open, High, Low and Close prices of stock are used for creating new variables which are used as inputs to the model. The models are evaluated using standard strategic indicators: RMSE and MAPE. The low values of these two indicators show that the models are efficient in predicting stock closing price.

In the realm of stock market prediction, achieving accurate forecasts is often hindered by the intricate, chaotic, and nonlinear dynamics inherent in financial systems. In our endeavor to tackle this challenge, we propose a novel framework termed the Pearson Correlation Coefficient-Broad Learning System (PCC-BLS) for enhanced stock price prediction.

At the core of our methodology lies a multi-indicator feature selection approach leveraging the Pearson Correlation Coefficient (PCC) in conjunction with the Broad Learning System (BLS). With an initial pool of 35 features encompassing original stock prices, technical indicators, and financial metrics, the PCC method serves as a discerning filter, identifying the most informative input features for prediction.

Subsequently, the selected input features undergo rapid information extraction and training within the BLS framework.

By harnessing the rich diversity of data captured by the financial variables—Open, High, Low, and Close prices of stocks—we generate new variables that serve as inputs to the predictive model. This process not only enriches the feature space but also enhances the model's capacity to discern meaningful patterns and relationships within the data.

To evaluate the efficacy of our approach, we employ standard strategic indicators such as Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). The attainment of low values for these metrics serves as a testament to the efficiency and accuracy of our predictive models in forecasting stock closing prices. By minimizing discrepancies between predicted and actual values, our framework demonstrates its effectiveness in capturing and capitalizing on underlying trends and dynamics within financial markets.

In summary, the PCC-BLS framework represents a promising avenue for advancing the field of stock price prediction, offering a systematic and data-driven approach to feature selection and model training. Through rigorous evaluation and validation, we validate the utility and reliability of our methodology in navigating the complexities of financial systems and delivering actionable insights for informed decision-making in stock trading and investment strategies.

**2.1.1 DISADVANTAGES:**

* The results is low when compared with proposed
* It doesn’t efficient for large volume of data’s
* Theoretical limits
  1. **PROPOSED SYSTEM:**

In this system, the stock price dataset was taken as input from the dataset repository. Then, we have to implement the data pre-processing step. In this step, we have to handle the missing values for avoid wrong prediction. After that, we have to implement the feature selection for selecting the best features from our dataset by using Pearson coefficient. Then, we have to split the data into test and train. In this step, test is used for predict the model and train is used for evaluate the model.we have to implement the machine learning regression algorithms such as support vector regression and linear regression .Finally, the experimental results shows that the performance metrics such as MAE, MSE, RMSE and predict or forecast the stock price based on input attributes.

In our system architecture, the initial step involves acquiring the stock price dataset from a reliable repository, laying the foundation for subsequent analysis and prediction. However, before delving into model development, a crucial preprocessing stage ensues to ensure data integrity and accuracy.

Handling missing values stands out as a pivotal aspect of data preprocessing, as overlooking these gaps can potentially skew predictions and undermine model performance. By employing robust techniques such as imputation or deletion, we strive to mitigate the impact of missing data on our predictive models, fostering more reliable outcomes.

Following data preprocessing, feature selection emerges as a critical task aimed at identifying the most informative attributes for prediction. Leveraging the Pearson coefficient—a measure of linear correlation—we discern the optimal subset of features that exhibit strong associations with the target variable, thereby enhancing the predictive power of our models.

Once feature selection is complete, the dataset undergoes partitioning into distinct training and testing subsets. This partitioning scheme ensures that our models are trained on a representative portion of the data while preserving unseen samples for rigorous evaluation. The training data serves as the foundation for model learning, while the testing data acts as an independent yardstick for assessing predictive performance.

In the subsequent phase, we embark on implementing machine learning regression algorithms—such as Support Vector Regression (SVR) and Linear Regression—to model the relationship between input features and stock prices. Through extensive experimentation and fine-tuning, we seek to harness the predictive capabilities of these algorithms to accurately forecast future stock prices.

Finally, the culmination of our efforts manifests in the form of comprehensive experimental results, wherein performance metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) provide quantitative insights into model accuracy and efficacy. Armed with these metrics, we confidently deploy our predictive models to forecast stock prices based on input attributes, empowering stakeholders with actionable intelligence for informed decision-making in financial markets.

**2.2.1 ADVANTAGES:**

* It is efficient for large number of datasets.
* The experimental result is high when compared with existing system.
* The prediction is efficient
* The process is implemented with removing unwanted data
  1. **LITERATURE SURVEY:**

# **2.3.1Stock Closing Price Prediction using Machine Learning Techniques, 2020**

# **Author*:*** Mehar Vijha , Deeksha Chandolab, Vinay Anand Tikkiwalb, Arun Kumarc**Methodology:**

Accurate prediction of stock market returns is a very challenging task due to volatile and non-linear nature of the financial stock markets. With the introduction of artificial intelligence and increased computational capabilities, programmed methods of prediction have proved to be more efficient in predicting stock prices. In this work, Artificial Neural Network and Random Forest techniques have been utilized for predicting the next day closing price for five companies belonging to different sectors of operation. The financial data: Open, High, Low and Close prices of stock are used for creating new variables which are used as inputs to the model. The models are evaluated using standard strategic indicators: RMSE and MAPE. The low values of these two indicators show that the models are efficient in predicting stock closing price

**Advantage*:***

* It prove to be highly useful for making stock market movements
* To obtain higher accuracy in the predicted price value.

**Disadvantages:**

* The noise in stock market data is usually quite high
* Complexity is high

# **2.3.2Computational Intelligence Techniques Used for Stock Market Prediction: A Systematic Review, 2020**

# **Author:** S. Zavadzki; M. Kleina; F. Drozda; M. Marques

# **Methodology:**

With the advancement of various computational techniques and the growing search for assertive predictive models, computational intelligence methods have attracted much attention. They are data-based methodologies and mainly include fuzzy logic, artificial neural networks and evolutionary computation. In the economic environment, more specifically, in the stock market forecast, where there is the challenge of the time series volatility, these methods have stood out. In this context, the objective of this paper is to present a systematic review of the literature on recent research involving forecasting techniques in the stock market, and the computational intelligence were the ones that stood out.

**Advantage**:

* Hybrid articles were collected from four large databases.
* It remove duplicated articles

**Disadvantages:**

* Prediction of stock is improper
* It occurred low accuracy

# **2.3.3 Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data; a Comparative Analysis, 2020**

# **Author:** MOJTABA NABIPOUR1, POOYAN NAYYERI, HAMED JABANI3 ,SHAHAB S

**Methodology:**

The nature of stock market movement has always been ambiguous for investors because of various influential factors. This study aims to significantly reduce the risk of trend prediction with machine learning and deep learning algorithms. Four stock market groups, namely diversified financials, petroleum, non-metallic minerals and basic metals from Tehran stock exchange, are chosen for experimental evaluations. This study compares nine machine learning models (Decision Tree, Random Forest, Adaptive Boosting (Adaboost), eXtreme Gradient Boosting (XGBoost), Support Vector Classifier (SVC), Naïve Bayes, K-Nearest Neighbors (KNN), Logistic Regression and Artificial Neural Network (ANN))

**Advantage*:***

* On the contrary, constructing over-complex trees that cause over fitting.

**Disadvantage:**

* Over-complex trees that cause over fitting is a typical disadvantage.

# **2.3.4 An Innovative Neural Network Approach For Stock Market Prediction, 2018**

# **Author**: Xiongwen Pang1 · Yanqiang Zhou1 · Pan Wang1 · Weiwei Lin2 · Victor Chang

# **Methodology:**

This paper aims to develop an innovative neural network approach to achieve better stock market predictions. Data were obtained from the live stock market for real-time and off-line analysis and results of visualizations and analytics to demonstrate Internet of Multimedia of Things for stock analysis. To study the influence of market characteristics on stock prices, traditional neural network algorithms may incorrectly predict the stock market, since the initial weight of the random selection problem can be easily prone to incorrect predictions. Based on the development of word vector in deep learning, we demonstrate the concept of “stock vector.” The input is no longer a single index or single stock index, but multi-stock high-dimensional historical data.

**Advantage**:

* The accuracy of two models is high.

**Disadvantage:**

* Encoder to predict the stock market is low.

# **2.3.5 Applications of Deep Learning In Stock Market Prediction, 2020**

**Author:** WeiweiJiang

**Methodology**:

Stock market prediction has been a classical yet challenging problem, with the attention from both economists and computer scientists.

With the purpose of building an effective prediction model, both linear and machine learning tools have been explored for the past couple of decades. Lately, deep learning models have been introduced as new frontiers for this topic and the rapid development is too fast to catch up. Hence, our motivation for this survey is to give a latest review of recent works on deep learning models for stock market prediction.

Stock market prediction represents a perennially intriguing yet formidable challenge, captivating the interest of economists and computer scientists alike.

Over the past few decades, researchers have tirelessly explored various methodologies, ranging from traditional linear models to cutting-edge machine learning techniques, in pursuit of building effective prediction models. However, it's the recent emergence of deep learning models that has sparked a renewed fervor in this field, pushing the boundaries of what's possible at an unprecedented pace.

The landscape of stock market prediction has undergone a profound transformation with the advent of deep learning, ushering in a new era of innovation and exploration. These advanced neural network architectures offer unparalleled capabilities in capturing intricate patterns and dependencies within financial data, enabling more accurate and reliable predictions than ever before.

As researchers race to harness the full potential of deep learning for stock market prediction, the field is experiencing a whirlwind of rapid development and breakthroughs. The sheer velocity of progress in this domain makes it challenging for practitioners to keep pace with the latest advancements, underscoring the need for comprehensive surveys that provide up-to-date insights into recent works and trends.

With this motivation in mind, our survey endeavors to offer a timely and comprehensive review of the latest developments in deep learning models for stock market prediction.

By synthesizing and analyzing recent research findings, we aim to provide valuable guidance and insights to researchers, practitioners, and enthusiasts navigating this dynamic and evolving landscape. Through our exploration of state-of-the-art methodologies, challenges, and future directions, we seek to contribute to the collective understanding and advancement of stock market prediction using deep learning techniques.

**Advantage**:

* Can achieve high classification rates.
* Different sparse algorithm is analysed.

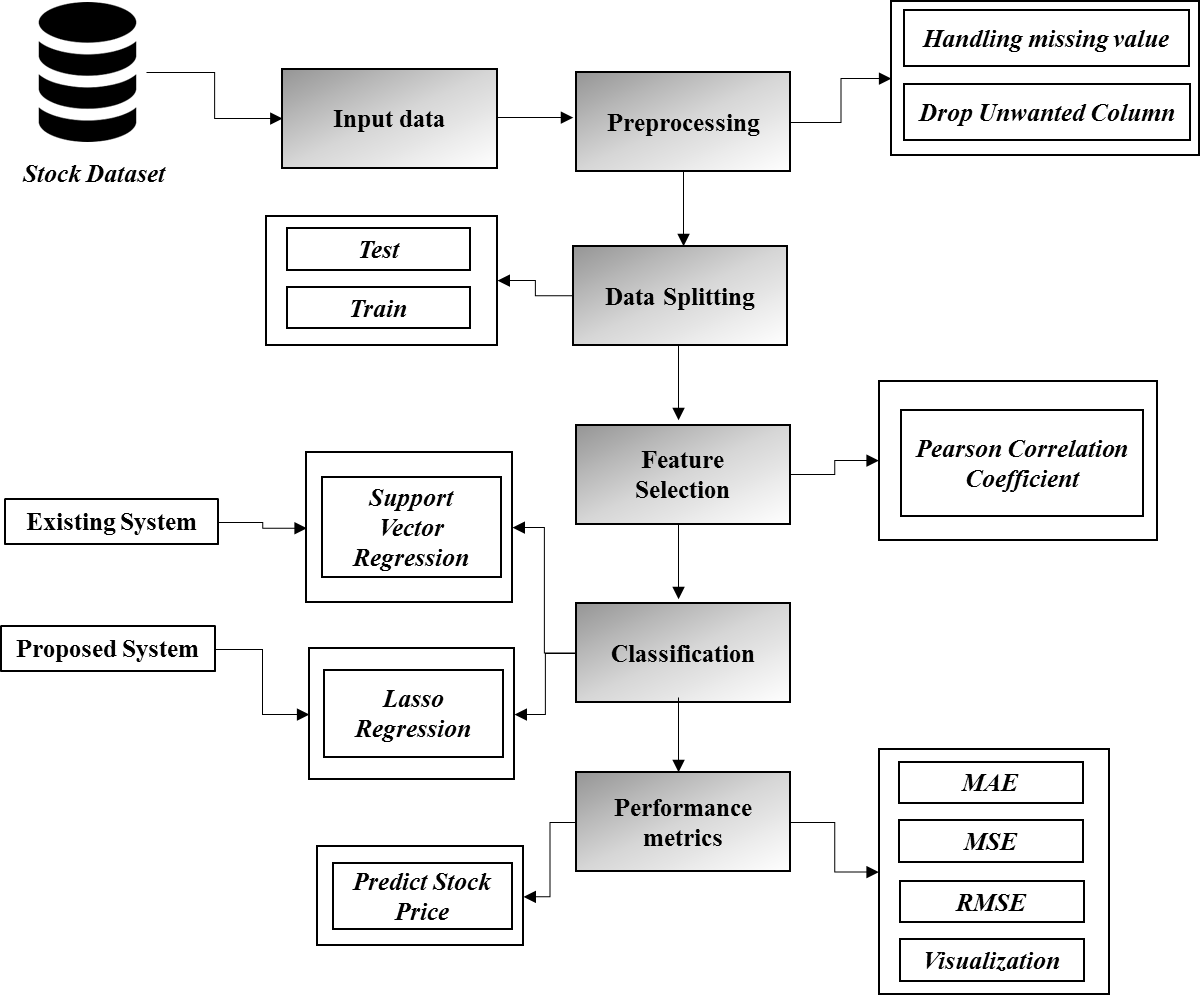
**Disadvantages:**

* To easily reproduce the previous studies as baselines.
* It didn’t implement the more than one algorithm.

**CHAPTER 3**

**SYSTEM DIAGRAMS**

**3.1 SYSTEM ARCHITECTURE:**

****

***FIGURE 3.1: SYSTEM ARCHITECTURE***

**3.2 FLOW DIAGRAM**

Input Data

Preprocessing

Data splitting

Classification

Performance analysis

***FIGURE 3.2: FLOW DIAGRAM***

**3.3 UML DIAGRAMS:**

**3.3.1 USE CASE DIAGRAM:**

System

User

***FIGURE 3.3.1: USE CASE DIAGRAM***

**3.3.2 USE CASE DIAGRAM:**

Input Data

Preprocessing

Data Splitting

Performance Analysis

Classification

***FIGURE 3.3.2: ACTIVITY CASE DIAGRAM***

**3.3.3 SEQUENCE DIAGRAM:**

Input Data

Preprocessing

Classification

Performance Analysis

Select data

Missing value

SVR

Load data

Data splitting

Lasso regression

***FIGURE 3.3.3: SEQUENCE DIAGRAM***

**3.3.4 ER DIAGRAM:**

Data selection

Preprocessing

Data Splitting

Classification

***FIGURE 3.3.4: ER DIAGRAM***

**3.3.6 CLASS DIAGRAM:**

Data Splitting

Select data ()

Load data ()

View data ()

INPUT

Test ()

SVR ()

Train ()

Classification

Prediction ()

Lasso ()

Performance analysis

Preprocessing

Missing values ()

Drop columns ()

MAE ()

***FIGURE 3.3.5: CLASS DIAGRAM***

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 MODULES:**

* Data selection
* Data preprocessing
* Feature Selection
* Data splitting
* Classification
* Result Generation

**4.2 MODULES DESCRIPTION:**

**4.2.1: DATA SELECTION:**

* The data selection is the process of selecting the data for predicting the stock.
* The dataset was collected from dataset repository like UCI.
* The dataset is in the format like ‘.csv’
* In this system, the time series dataset is used for predicting the stock.
* The dataset which contains the information about the high, low, open and close price.
* With the help of panda’s package, we can read or load our input dataset.

**4.2.2: DATA PREPROCESSING:**

* Data pre-processing is the process of removing the unwanted data from the dataset.
* Pre-processing data transformation operations are used to transform the dataset into a structure suitable for machine learning.
* Missing data removal: In this process, the null values such as missing values and Nan values are replaced by 0.
* Missing and duplicate values were removed and data was cleaned of any abnormalities.
* If you want to drop unwanted or unnecessary columns from our dataset, we can drop it in priorly.

**4.2.3: FEATURE SELECTION:**

* In this step, we can select the features from preprocessed data by using Pearson’s correlation.
* One of the measures used for feature selection is dependency measures. Many dependency based methods have been proposed.
* The main measure is Correlation based method. Pearson's Correlation method is used for finding the association between the continuous features and the class feature.
* Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable.
* So, when two features have high correlation, we can drop one of the two features.

**4.2.4: DATA SPLITTING:**

* During the machine learning process, data are needed so that learning can take place.
* In addition to the data required for training, test data are needed to evaluate the performance of the algorithm in order to see how well it works.
* In our process, we considered 70% of our input dataset to be the training data and the remaining 30% to be the testing data.
* Data splitting is the act of partitioning available data into two portions, usually for cross-validator purposes.
* One Portion of the data is used to develop a predictive model and the other to evaluate the model's performance.
* Separating data into training and testing sets is an important part of evaluating data mining models.
* Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing.

**4.2.5: CLASSIFICATION:**

* In our process, we have to implement the two machine learning algorithm such as lasso regression and support vector regression.
* **Support Vector Regression** is a supervised learning algorithm that is used to predict discrete values. Support Vector Regression uses the same principle as the SVMs.
* The basic idea behind SVR is to find the best fit line. In SVR, the best fit line is the hyper plane that has the maximum number of points.
* **Lasso regression** is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean.
* The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters).

**4.2.5: RESULT GENERATION:**

The Final Result will get generated based on the overall classification and prediction. The performance of this proposed approach is evaluated using some measures like,

* **MAE:** In statistics, the **mean absolute error** (MAE) is a way to measure the accuracy of a given model. It is calculated as:

**MAE = (1/n) \* Σ|yi – xi|**

Where:

* **Σ:** A Greek symbol that means “sum”
* **yi:** The observed value for the ith observation
* **xi:** The predicted value for the ith observation
* **n:** The total number of observations
* **MSE:** The mean squared error (MSE) is a common way to measure the prediction accuracy of a model. It is calculated as:

**MSE**= (1/n) \* Σ (actual – prediction) 2

Where:

* **Σ** – a fancy symbol that means “sum”
* **n** – sample size
* **actual** – the actual data value
* **forecast** – the predicted data value

**CHAPTER 5**

**SYSTEM REQUIREMENTS**

**5.1 HARDWARE REQUIREMENTS:**

* System : Pentium IV 2.4 GHz
* Hard Disk : 200 GB
* Mouse : Logitech.
* Keyboard : 110 keys enhanced
* Ram : 4GB

**5.2 SOFTWARE REQUIREMENTS:**

* O/S : Windows 7.
* Language : Python
* Front End : Anaconda Navigator – Spyder

**5.3 SOFTWARE DESCRIPTION:**

**5.3.1 Python**

Python is one of those rare languages which can claim to be both *simple* and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in.

The official introduction to Python is Python is an easy to learn, powerful programming language.

Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

Python stands out as a beacon of accessibility and versatility in the realm of programming languages. Its simplicity doesn't compromise on capability; rather, it amplifies productivity by allowing developers to express complex ideas with remarkable clarity.

This attribute makes Python an attractive choice for both beginners taking their first steps into coding and seasoned professionals tackling intricate challenges. Moreover, Python's expansive standard library provides a wealth of pre-built modules and functions, empowering developers to swiftly craft solutions without reinventing the wheel.

Furthermore, Python's community-driven ethos fosters a culture of collaboration and innovation. A vast ecosystem of third-party libraries and frameworks extends Python's capabilities to practically every imaginable domain, from web development and data analysis to artificial intelligence and scientific computing.

This vibrant ecosystem not only enriches Python's utility but also ensures its relevance in an ever-evolving technological landscape.

In addition to its technical merits, Python champions readability and maintainability, making codebases comprehensible and adaptable even as projects scale in complexity.

Its adherence to clean, consistent syntax encourages best practices and facilitates code review and collaboration among team members. This emphasis on readability not only enhances developer satisfaction but also contributes to the longevity and sustainability of software projects.

In essence, Python's combination of simplicity, power, and community support positions it as a cornerstone of modern software development.

Whether you're embarking on a personal project, contributing to open-source endeavors, or spearheading enterprise solutions, Python remains an invaluable ally, empowering you to turn ideas into reality with elegance and efficiency.

Machine learning, a subfield of artificial intelligence, empowers computers to learn from data and improve their performance over time without explicit programming.

Python has emerged as the de facto language for machine learning due to its extensive libraries, ease of use, and vibrant community support.

One of the most popular libraries for machine learning in Python is TensorFlow, developed by Google, which provides a comprehensive ecosystem for building and deploying machine learning models at scale.

In addition to TensorFlow, another prominent library is scikit-learn, which offers a user-friendly interface for implementing various machine learning algorithms, including classification, regression, clustering, and dimensionality reduction. Both TensorFlow and scikit-learn are complemented by libraries like Keras, PyTorch, and MXNet, which facilitate deep learning, a subset of machine learning focused on neural networks.

Machine learning workflows typically involve data preprocessing, model training, evaluation, and deployment.

Python excels at each stage of this process, offering tools like pandas and NumPy for data manipulation and exploration, matplotlib and seaborn for data visualization, and Jupyter notebooks for interactive experimentation and documentation.

Moreover, Python's versatility extends to specialized domains within machine learning, such as natural language processing (NLP), computer vision, and reinforcement learning.

Libraries like NLTK, spaCy, OpenCV, and Gym provide robust solutions for tackling tasks like text analysis, image recognition, and sequential decision making.

In recent years, Python's dominance in the field of machine learning has only strengthened, with a burgeoning ecosystem of tools, frameworks, and resources continually expanding its capabilities.

Whether you're a novice exploring the fundamentals of machine learning or a seasoned practitioner pushing the boundaries of AI, Python remains the language of choice for unlocking the potential of data and transforming it into actionable insights.

## **5.3.2 Features of Python**

### **Simple**

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

### **Easy to Learn**

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

### **Free and Open Source**

Python is an example of a FLOSS (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

### **High-level Language**

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

### **Portable**

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

You can even use a platform like [Kivy](http://kivy.org) to create games for your computer and for iPhone, iPad, and Android.

Its scalability is evident in its ability to power anything from small scripts to large-scale enterprise systems, with support for multiple programming paradigms ensuring flexibility in design and implementation.

As Python continues to enjoy strong industry adoption, fueled by major companies leveraging its power for critical infrastructure and services, its open-source ethos and collaborative ecosystem ensure that it remains at the forefront of innovation in the world of programming.

Embraced by tech giants and startups alike, Python's influence pervades industries ranging from finance to entertainment, cementing its status as a linchpin of modern software engineering.

In the ever-evolving landscape of programming, Python's collaborative ethos and expansive ecosystem continue to propel innovation forward, inspiring developers to push the boundaries of what's possible.

### **Interpreted**

This requires a bit of explanation.

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options.

When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just run the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it.

All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc.

This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

However, interpreted languages may sacrifice some performance compared to compiled counterparts due to the overhead of interpretation.

Overall, the choice between compiled and interpreted languages depends on factors such as performance requirements, platform considerations, and developer preferences.

On the other hand, interpreted languages like Python offer portability and ease of use. Python code is executed line by line by the interpreter, which translates it into machine code on the fly. This makes Python programs highly portable and eliminates the need for compilation, simplifying the development process.

### **Object Oriented**

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs.

In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

Objects are instances of classes, which define the blueprint for creating objects with specific attributes and methods. Python's approach to OOP is powerful yet straightforward, especially in comparison to languages like C++ or Java.

It emphasizes simplicity and readability, allowing developers to create classes, define inheritance hierarchies, and implement encapsulation and polymorphism with ease. This flexibility makes Python well-suited for a wide range of programming tasks, from small scripts to large-scale applications.

These functions encapsulate specific tasks and can be called multiple times throughout the program. On the other hand, object-oriented programming (OOP) in Python revolves around the concept of objects, which encapsulate both data and functionality.

This flexibility makes Python well-suited for a wide range of programming tasks, from small scripts to large-scale applications.

The differences between concurrency and parallelism, and choosing the appropriate approach based on the requirements of the application, is crucial for building performant and responsive software systems.

### **Extensible**

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

Additionally, frameworks like Django and Flask empower developers to build robust web applications with ease, thanks to their clean and pragmatic design.

Moreover, Python's popularity in the field of machine learning and artificial intelligence is undeniable, with libraries such as TensorFlow, PyTorch, and scikit-learn driving innovation in these areas.

Furthermore, Python's simple syntax and readability make it an ideal choice for teaching programming to beginners, contributing to its widespread adoption in educational settings. Overall, Python's rich ecosystem and user-friendly design make it a go-to language for a wide range of applications, from web development to scientific computing and beyond.

The language itself offers a plethora of libraries and frameworks that streamline development across various domains. For instance, NumPy and SciPy provide powerful tools for numerical computing and scientific computing, making Python a favorite among researchers and data scientists.

Python's role in game development is gaining traction, with frameworks like Pygame providing a platform for creating interactive and immersive gaming experiences. Overall, Python's versatility, combined with its rich ecosystem of libraries and frameworks, continues to make it a top choice for developers across a wide range of industries and applications.

### **Embeddable**

You can embed Python within your C/C++ programs to give scripting capabilities for your program's users.

### **Extensive Libraries**

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the Batteries Included philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the Python Package Index.

**5.4 TESTING PRODUCTS:**

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error.

A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. . A series of tests are performed before the system is ready for the user acceptance testing.

Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational.

Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

Testing products is a critical phase in the development lifecycle, ensuring that the final system operates flawlessly and meets user expectations. System testing, occurring during implementation, focuses on verifying the accuracy and efficiency of the system before it goes live. At its core, testing involves executing the program to identify any errors present.

The effectiveness of a test case is measured by its likelihood of uncovering errors, with a successful test being one that reveals previously undiscovered issues.

The significance of testing cannot be overstated, as it serves as a linchpin for the success of the system.

System testing operates under the logical assumption that if all system components function correctly, the overarching goal will be achieved seamlessly. Before reaching the user acceptance testing phase, a series of tests are conducted to ensure the system's readiness.

Various testing approaches can be employed, depending on the product's specifications and internal workings.

One approach involves testing the specified functions of the product to demonstrate their full operational capacity. This method scrutinizes each function individually to ascertain its functionality and adherence to specifications.

Alternatively, testing can delve into the internal mechanisms of the product, ensuring that all internal components interact harmoniously and perform according to specifications. This includes verifying that intricate internal operations, such as gear meshing in mechanical systems, function smoothly and as intended.

In essence, rigorous testing is indispensable for validating the reliability, functionality, and performance of engineered products. By meticulously examining every aspect of the system, from its individual functions to its internal workings, testing ensures that the final product meets quality standards and fulfills user requirements with precision and efficiency.

**Importance of Testing**

Testing is a crucial stage in product development aimed at ensuring accuracy and efficiency before live operation commences. It operates under the logical assumption that if all system parts function correctly, the desired goal will be achieved successfully.

**System Testing**

This stage involves comprehensive testing to ensure the entire system operates as expected before user acceptance testing.

**Functionality Testing**

Tests are conducted to demonstrate that each function a product has been designed for is fully operational.

**Internal Working Testing**

This involves testing to ensure all internal components of a product are adequately exercised and that the internal operation aligns with specifications.

**5.4.1 UNIT TESTING:**

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’.

The primary purpose of unit testing is to ensure that each unit of the software performs as expected and functions correctly in isolation. By isolating each module and testing it independently, developers can identify and fix errors early in the development process, thereby improving the overall quality and reliability of the software.

During unit testing, developers write test cases specifically designed to exercise the functionalities of individual modules. These test cases validate the inputs, outputs, and behaviors of each module, ensuring that they conform to the specified requirements and design.

Unit tests are typically written using testing frameworks such as JUnit for Java, NUnit for .NET, or pytest for Python. These frameworks provide utilities for writing, executing, and analyzing test cases efficiently.

Unit testing plays a crucial role in ensuring the reliability, maintainability, and quality of software systems. By rigorously testing individual modules, developers can identify and rectify defects early, leading to a more robust and stable software product.

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

The modules of the system are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

**5.4.2 INTEGRATION TESTING:**

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

i) Top-down integration testing.

ii) Bottom-up integration testing.

**Top-down Integration testing**

Top-down integration testing is a systematic approach used to evaluate the functionality and interaction of modules within a software system.

This method starts with testing the top-level modules, typically those closest to the user interface or main control flow, before progressively integrating and testing lower-level modules.

During testing, stubs or simulated versions of lower-level modules are employed to mimic their behavior until they are fully developed or available.

This approach allows for the early detection of interface issues and major defects in critical functionalities, enabling developers to address them promptly. However, it may pose challenges such as dependency on accurate stubs and limited visibility of lower-level issues until integration occurs.

**Bottom-up Integration testing**

Bottom-up integration testing is an incremental testing approach employed to assess the functionality and integration of modules within a software system. Unlike top-down integration testing, this method begins with testing the lowest-level modules first, focusing on the foundational components of the system.

These modules are then gradually integrated, moving upwards towards higher-level modules until the entire system is tested as a whole.

During this process, drivers are utilized to simulate the behavior of higher-level modules, allowing for the testing of lower-level modules in isolation.

Bottom-up integration testing enables early detection of defects within individual modules and provides a comprehensive assessment of their interactions as integration progresses.

While it offers benefits such as early identification of module-level issues and simplified testing of independent components, it may present challenges related to the integration of complex modules and the need for comprehensive test coverage.

Nevertheless, bottom-up integration testing plays a crucial role in ensuring the robustness and reliability of software systems by systematically verifying the integration of modules from the ground up.

**5.4.3 TESTING TECHNIQUES/STRATEGIES:**

* **WHITE BOX TESTING:**

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we

derived test cases that guarantee that all independent paths within a module have been exercised at least once.

Black box testing is another test case design method that focuses on testing the functionality of a software system without considering its internal structure.

This approach involves creating test cases based on the requirements and specifications of the system, rather than its implementation details. With black box testing, testers do not have access to the source code or knowledge of the internal workings of the system, allowing them to evaluate its behavior from an external perspective.

Equivalence partitioning is a testing technique that involves dividing the input data of a software system into equivalence classes based on certain criteria, and then selecting representative values from each class as test cases.

This method helps in reducing the number of test cases while ensuring that the system is thoroughly tested by covering different input scenarios.

Boundary value analysis is another testing technique that complements equivalence partitioning.

It focuses on selecting test cases at the boundaries of equivalence classes, as these are more likely to uncover errors or bugs in the software.

By testing boundary values, testers can identify potential issues related to edge cases and ensure the robustness of the system.

Mutation testing is a method for evaluating the quality of test cases by introducing small changes, known as mutations, into the source code of a software system.

The goal is to determine whether the existing test cases are able to detect these mutations, thereby assessing their effectiveness in identifying faults or errors in the code.

Mutation testing helps in improving the reliability and effectiveness of test suites by identifying weaknesses and areas for improvement.

Pairwise testing, also known as all-pairs testing, is a combinatorial testing technique that aims to reduce the number of test cases while ensuring adequate coverage of input combinations.

It involves selecting a subset of test cases that covers all possible pairs of input parameters or factors, thereby minimizing the total number of tests required. Pairwise testing is particularly useful in situations where exhaustive testing of all combinations is not feasible due to time or resource constraints.

Static testing is a type of software testing that does not involve the execution of code. Instead, it focuses on reviewing and analyzing the software artifacts, such as requirements, design documents, and source code, to identify defects or issues early in the development process.

Static testing techniques include code reviews, inspections, and walkthroughs, which help in improving the quality and reliability of the software by detecting errors before they manifest as problems during runtime.

Dynamic testing, on the other hand, involves the execution of code to evaluate its behavior and performance. This type of testing includes various techniques such as unit testing, integration testing, system testing, and acceptance testing, which aim to validate the functionality, correctness, and reliability of the software under test.

Dynamic testing is essential for identifying defects and ensuring that the software meets the specified requirements and quality standards.

white box testing is another test case design method that focuses on testing the functionality of a software system without considering its internal structure.

This approach involves creating test cases based on the requirements and specifications of the system, rather than its implementation details. With black box testing, testers do not have access to the source code or knowledge of the internal workings of the system, allowing them to evaluate its behavior from an external perspective

* **BLACK BOX TESTING:**

1. Black box testing is done to find incorrect or missing function
2. Interface error
3. Errors in external database access
4. Performance errors.
5. Initialization and termination errors

In ‘functional testing’, is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’. It tests the external behaviour of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

**5.4.4 SOFTWARE TESTING STRATEGIES**

**VALIDATION TESTING:**

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as man

**USER ACCEPTANCE TESTING:**

User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

**OUTPUT TESTING**:

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs. For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

**CHAPTER 6**

**CONCLUSION**

We conclude that, the stock price dataset was taken as input. The input dataset was mentioned in our research paper. We are implemented the different machine algorithm such as support vector regression and lasso regression. Then, we are predicted the house price and performance metrics such as MAE, MSE and RMSE.

**CHAPTER 7**

**FUTURE ENHANCEMENT**

In the future, we should like to hybrid the two different machine learning. In future, it is possible to provide extensions or modifications to the proposed clustering and classification algorithms to achieve further increased performance. Apart from the experimented combination of data mining techniques, further combinations and other clustering algorithms can be used to improve the detection accuracy.

**CHAPTER 8**

**SAMPLE CODING**

#==================== IMPORT PACKAGES================

import warnings

warnings.filterwarnings("ignore")

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

import matplotlib.pyplot as plt

from sklearn.svm import SVR

from sklearn import metrics

from sklearn import linear\_model

from scipy.stats import pearsonr

#==================== READ INPUT DATA ==============

print("--------------------------------------")

print(" Input data ")

print("--------------------------------------")

print()

dataframe=pd.read\_csv('Stock Dataset.csv')

print(dataframe.head(20))

#==================== PREPROCESSING ==================

#==== checking missing values =====

print("----------------------------------------------")

print(" Data Preprocessing ")

print("----------------------------------------------")

print()

print (dataframe.isnull().sum())

#==== cdrop unwanted columns =====

columns = ['Date']

dataframe.drop(columns, inplace=True, axis=1)

#=================== FEATURE SELECTION ================

# === Correlation ===

print("-------------------------------------------------------")

print("Correlation")

print("-------------------------------------------------------")

print()

val1 = dataframe['Close']

val2 = dataframe['High']

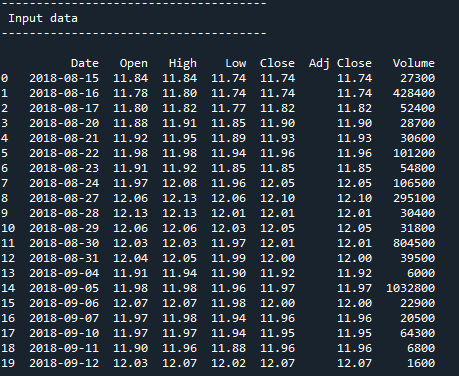
corr, \_ = pearsonr(val1, val2)

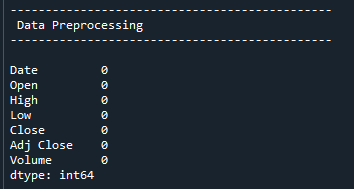
print('Pearsons correlation :' % corr)

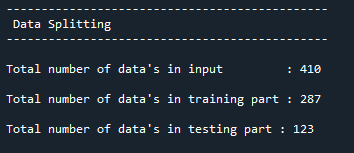
print()

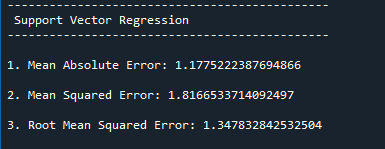
**CHAPTER 9**

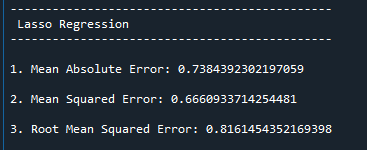
**SAMPLE SCREENSHOTS**

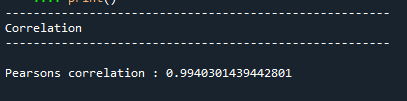


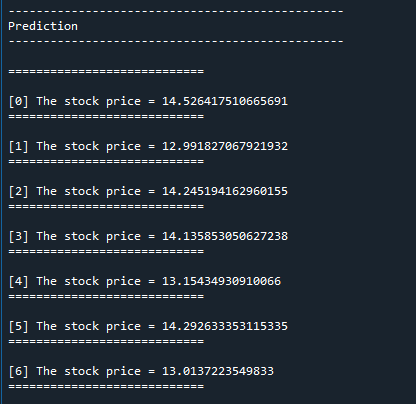


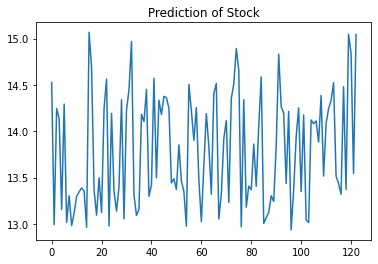


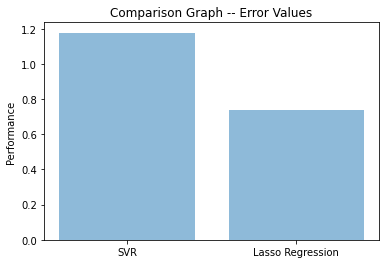












**REFERENCES**

* Wei Huang, Yoshiteru Nakamori, Shou-Yang Wang, “Forecasting stock market movement direction with support vector machine”, Computers & Operations Research, Volume 32, Issue 10, October 2005, Pages 2513–2522.
* C.J.C. Burges, “A Tutorial on Support Vector Machines for Pattern Recognition”, Data Mining and Knowledge Discovery, Volume 2, pp. 1-43, Kluwer Academic Publishers, Boston, 1998.
* C. Cortes and V. Vapnik, “Support Vector Networks”, Machine Learning, 20, 273-297, 1995.
* M.Pontil and A. Verri, “Properties of Support Vector Machines”, Technical Report, Massachusetts Institute of Technology, 2016
* E.E. Osuna, R. Freund and F. Girosi, “Support Vector Machines: Training and Applications”, Technical Report, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, AI Memo No. 1602, , 2014.
* Masoud, Najeb MH. (2017) “The impact of stock market performance upon economic growth.” International Journal of Economics and Financial Issues 3 (4) : 788–798.
* Murkute, Amod, and Tanuja Sarode. (2015) “Forecasting market price of stock using artificial neural network.” International Journal of Computer Applications 124 (12) : 11-15.
* Hur, Jung, Manoj Raj, and Yohanes E. Riyanto. (2006) “Finance and trade: A cross-country empirical analysis on the impact of financial development and asset tangibility on international trade.” World Development 34 (10) : 1728-1741.
* Li, Lei, Yabin Wu, Yihang Ou, Qi Li, Yanquan Zhou, and Daoxin Chen. (2017) “Research on machine learning algorithms and feature extraction for time series.” IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC): 1-5.
* Seber, George AF and Lee, Alan J. (2012) “Linear regression analysis.” John Wiley & Sons 329
* Zhanpeng Zhang, Ping Luo, Chen Change Loy (May 2016), "Learning deep representation for face alignment with auxiliary attribute", IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 38, No. 3, pp. 67-73.
* Reichek, Nathaniel, and Richard B. Devereux. (1982) “Reliable estimation of peak left ventricular systolic pressure by M-mode echographicdetermined end-diastolic relative wall thickness: identification of severe valvular aortic stenosis in adult patients.” American heart journal 103 (2) : 202-209.