1. INTRODUCTION

Stock market plays a very important role in fast economic growth of the developing country like India. So our country and other developing nation's growth may depend on performance of stock market. If stock market rises, then countries economic growth would be high. If stock market falls, then countries economic growth would be down. In other words, we can say that stock market and country growth is tightly bounded with the performance of stock market. In any country, only 10% of the people engaging themselves with the stock market investment because of the dynamic nature of the stock market. There is a misconception about the stock market i.e., buying or selling of shares is an act of gambling. Hence, this misconception can be changed by bringing the awareness across the people for this. The prediction techniques in stock market can play a crucial role in bringing more people and existing investors at one place. Among the popular methods that have been employed, Machine Learning techniques are very popular due to the capacity of identifying stock trends from massive amounts of data that capture the underlying stock price dynamics. In this paper, we applied supervised learning methods for stock price trend forecasting

1.1 Problem Statement

Identifying the characteristic properties i.e,dependent and independent variables which can accurately predict the stock price.Rich usage of labels are not there so predictions results a false value.

1.2 Objectives

The comparative study of the supervised machine learning algorithms using the time window of size 1 to 90 has to be implemented.. The algorithms have been compared based upon the parameters: Size of the dataset and Number of technical indicators used. Accuracy and F measure values have to be computed for each algorithm. Long term model have to compute the accuracy and F-measure. The proposed architecture for the implemented work mainly consist of four steps: feature extraction from the given dataset, supervised classification of the training dataset, supervised classification of the test dataset, and result evaluation.

1.3 Features

A perfect feature set contains discriminating information, which can differentiate one object from another. It must be as robust as possible so as to prevent generating various feature codes for the objects in the same class. The selected set of features should be a small whose values effectively differentiate among patterns of various classes, but are same for patterns within the same class. Features can be classified into two categories:

1. Local features
2. Global features

1.4 Feature Extraction

Feature extraction makes the task of classifying pattern easy by formal procedure by describing a relevant shape information that a pattern contains. The main purpose of feature extraction is to get the most relevant information from the original data and represent it in a lesser dimensionality space. Feature extraction follows the pre-processing stage in character recognition system. The primary task of pattern recognition is to take input pattern and assign it correctly as one of the possible output classes. This process can be categorized into two general phases: Feature selection and Classification. Feature selection is important to the whole process as the classifier won't be able to detect from poorly selected features. Feature extraction is a crucial step in the development of any pattern classification and aims at extracting the relevant information that characterizes all the classes. Feature vectors are formed from relevant features extracted from alphabets/objects. These feature vectors are then used by classifiers to detect the input unit with target unit. It becomes easier for the classifier to classify between distinct classes by looking at the features as is fairly easy to distinguish. Feature extraction is the process to retrieve the most crucial data from the raw data. Feature extraction is searching the set of parameters that define the shape of a character correctly and uniquely. In feature extraction stage, each character is represented by a feature vector, which becomes its identity. The major goal of feature extraction is to retrieve a set of features, which increases the recognition rate with the least possible amount of elements and to generate same feature set for variety of instance of the similar symbol.

1.5 Importance of Feature Extraction:

When the pre-processing and the segmentation up to desired level has been achieved, some feature extraction techniques are applied to the segments to obtain features, which is followed by application of classification and post processing techniques. It is important to focus on the feature extraction phase as it has a great impact on the efficiency of the recognition system. Feature selection of a feature extraction method is the most important factor in achieving high performance on recognition. Feature extraction has been given as “extracting from the raw information that is best suitable for classification, while minimizing 2 the inner class pattern variability and enhancing the between class pattern variability”. Taking into consideration all these factors, it becomes important to look at various available techniques for feature extraction in a given domain, covering vast possibilities of cases.

2. LITERATURE REVIEW

2.1 Study of the System

1. Numpy
2. Pandas
3. Matplotlib
4. Scikit –learn

1 . Numpy:

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

2. Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

3.Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

4.Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- NumPy: Base n-dimensional array package
- SciPy: Fundamental library for scientific computing
- Matplotlib: Comprehensive 2D/3D plotting
- IPython: Enhanced interactive console
- Sympy: Symbolic mathematics
- Pandas: Data structures and analysis
- Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

2.2 Input and Output

Inputs:

- Importing the all required packages like numpy, pandas, matplotlib, scikit – learn and required machine learning algorithms packages .
- Setting the dimensions of visualization graph.
- Downloading and importing the dataset and convert to data frame.

Outputs:

- Preprocessing the importing data frame for imputing nulls with the related information.
- All are displaying cleaned outputs.
- After applying machine learning algorithms it will give good results and visualization plots.

Too many software development efforts go awry when development team and customer personnel get caught up in the possibilities of automation. Instead of focusing on high priority features, the team can become mired in a sea of nice to have features that are not essential to solve the problem, but in themselves are highly attractive. This is the root cause of large percentage of failed and or abandoned development efforts and is the primary reason the development team utilizes the iterative model.

2.3.2 Roles and Responsibilities of PDR AND PER

The iterative lifecycle specifies two critical roles that act together to clearly communicate project issues and concepts between the end-user community and the development team.

2.3.3 Primary End-user Representative (PER)

The PER is a person who acts as the primary point of contact and principal approver for the end-user community. The PER is also responsible for ensuring that appropriate subject matter experts conduct end-user reviews in a timely manner.

2.3.4 PER-PDR Relationship

The PER and PDR are the brain trust for the development effort. The PER has the skills and domain knowledge necessary to understand the issues associated with the business processes to the supported by the application and has a close working relationship with the other members of the end-user community. The PDR has the same advantages regarding the application development process and the other members of the development team together, they act as the concentration points for knowledge about the application to be developed.

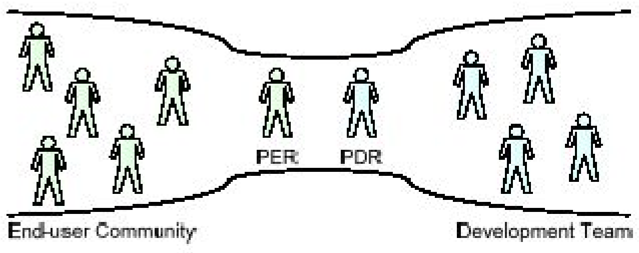


Fig: 2.2 PER-PDR Relationship

The objective of this approach is to create the close relationship that is characteristic of a software project with one developer and one end-user in essence, this approach the "pair programming" concept from Agile methodologies and extends it to the end-user community. While it is difficult to create close relationships between the diverse members of an end-user community and a software development team, it is much simpler to create a close relationship between the lead representatives for each group.

When multiple end-users are placed into relationship with multiple members of a development team, communication between the two groups degrades as the number of participants grows. In this model, members of end-user community may communicate with members of the development team as needed, but it is the responsibility of all participants to keep the PER and PDR apprised of the communications for example, this allows the PER and PDR to resolve conflicts that arise when two different end-users communicate different requirements for the same application feature to different members of the development team.

2.3.5 Input Design

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

2.3.6 Input Stages

The main input stages before the information gets stored in the database media:

Ex: In this project voter either existing or new user data will be stored in database as the inputs given by users….

- Data recording ,Data transcription, Data conversion, Data verification
- Data control, Data transmission, Data validation, Data correction

3. TECHNICAL REQUIREMENTS

3.1 Technologies

Python:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Anaconda:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. Package versions are managed by the package management system anaconda

Jupyter:

Project Jupyter is a non-profit, open-source project, born out of the IPython Project in 2014 as it evolved to support interactive data science and scientific computing across all programming languages. Jupyter will always be 100% open-source software, free for all to use and released under the liberal terms of the modified BSD license.The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

3.2 Hardware Requirements:

- RAM: 4GB and Higher
- Processor: Intel i3 and above
- Hard Disk: 500GB: Minimum

3.3 Software Requirements:

- OS: Windows or Linux
- Python IDE : python 2.7.x and above
- Pycharm IDE Required
- Setup tools and pip to be installed for 3.6 and above
- Language   : Python Scripting

4. SYSTEM ARCHITECTURE AND DESIGN

4.1. Introduction

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

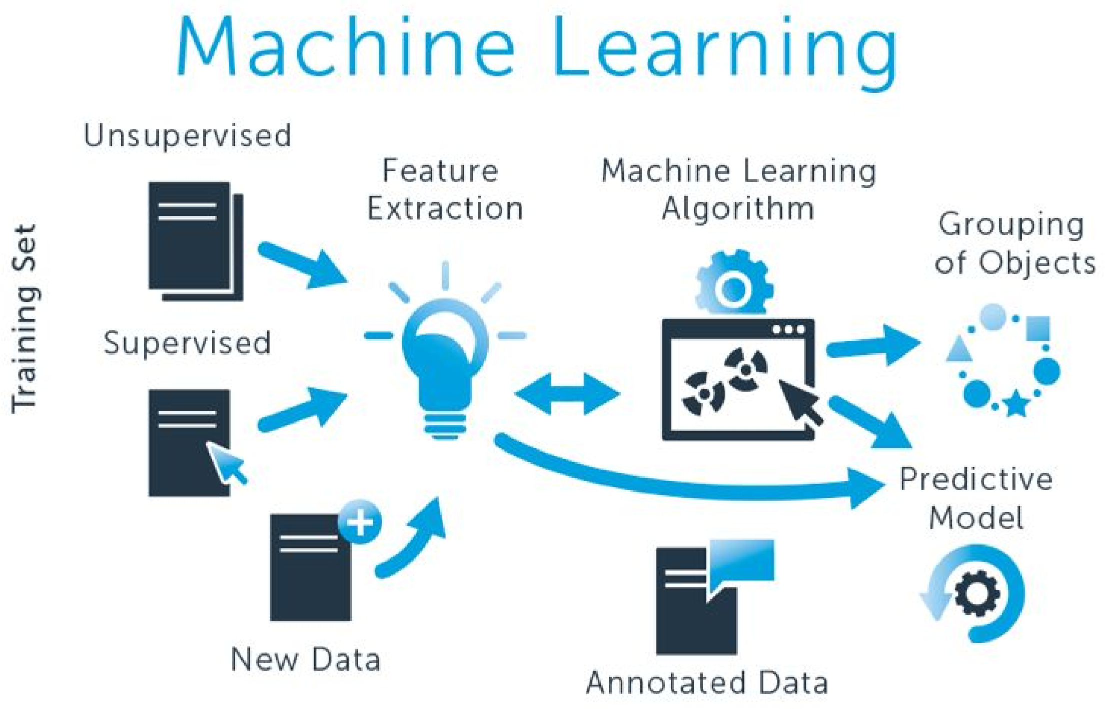


Fig:4.1 System Architecture

4.2 NORMALIZATION

It is a process of converting a relation to a standard form. The process is used to handle the problems that can arise due to data redundancy i.e. repetition of data in the database, maintain data integrity as well as handling problems that can arise due to insertion, updation, deletion anomalies.

Decomposing is the process of splitting relations into multiple relations to eliminate anomalies and maintain anomalies and maintain data integrity. To do this we use normal forms or rules for structuring relation.

Insertion anomaly: Inability to add data to the database due to absence of other data.

Deletion anomaly: Unintended loss of data due to deletion of other data.

Update anomaly: Data inconsistency resulting from data redundancy and partial update

Normal Forms: These are the rules for structuring relations that eliminate anomalies.

4.3 Support Vector Machine:

SVM is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyperplane that differentiate the two classes very well (look at the below snapshot).

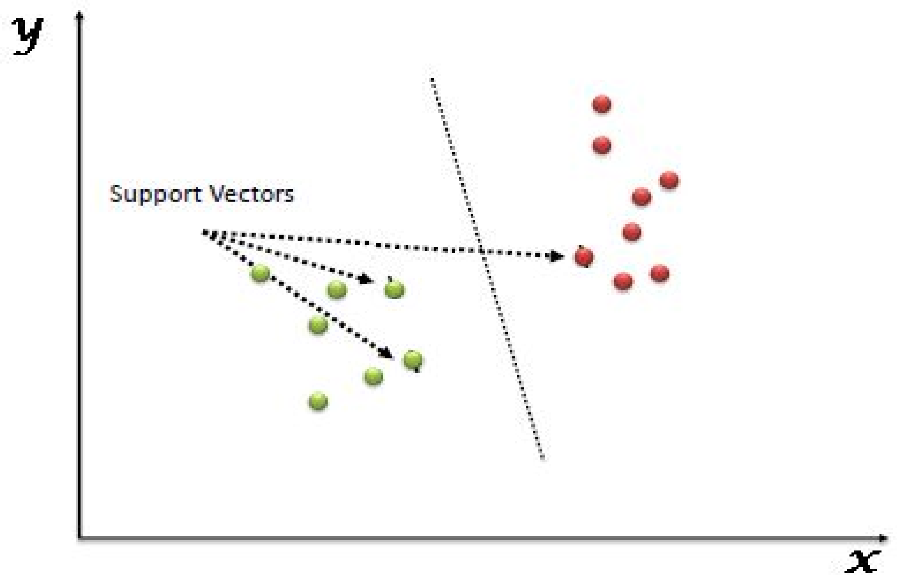


Fig: 4.2 Support Vector Machine

Support Vectors are simply the coordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

4.4 Random Forest:

Random forest is like bootstrapping algorithm with Decision tree (CART) model. Say, we have 1000 observation in the complete population with 10 variables. Random forest tries to build multiple CART model with different sample and different initial variables. For instance, it will take a random sample of 100 observation and 5 randomly chosen initial variables to build a CART model. It will repeat the process (say) 10 times and then make a final prediction on each observation. Final prediction is a function of each prediction. This final prediction can simply be the mean of each prediction.

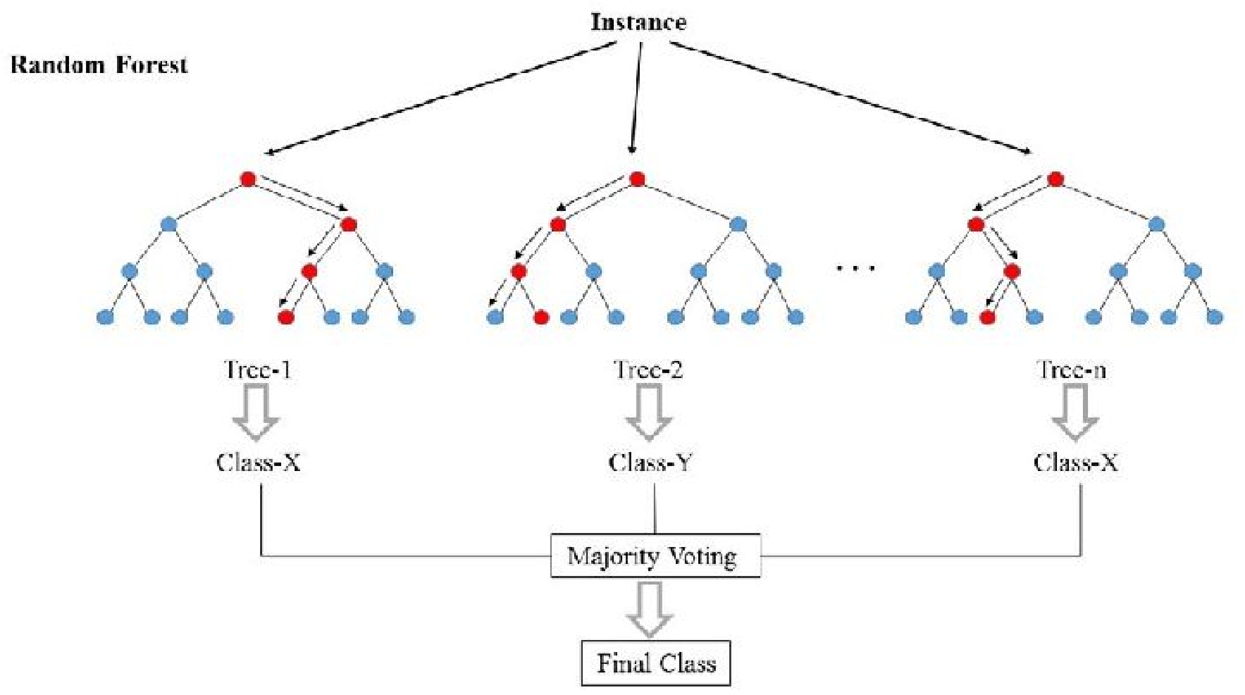


Fig: 4.3 Random Forest

Random forest gives much more accurate predictions when compared to simple CART/CHAID or regression models in many scenarios. These cases generally have high number of predictive variables and huge sample size. This is because it captures the variance of several input variables at the same time and enables high number of observations to participate in the prediction.

4.5 K Nearest Neighbour(KNN)

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

1. Ease to interpret output

2. Calculation time

3. Predictive Power

KNN algorithm fairs across all parameters of considerations. It is commonly used for its easy of interpretation and low calculation time.

KNN can be implemented by following steps:

- Load the data
- Initialise the value of k
- For getting the predicted class, iterate from 1 to total number of training data points

1. Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
2. Sort the calculated distances in ascending order based on distance values
3. Get top k rows from the sorted array
4. Get the most frequent class of these rows
5. Return the predicted class

To find out the class of the blue star (BS) . BS can either be Red Circles or Green Squares and nothing else. The "K" is KNN algorithm is the nearest neighbors we wish to take vote from. Let's say K = 3. Hence, we will now make a circle with BS as center just as big as to enclose only three data points on the plane.

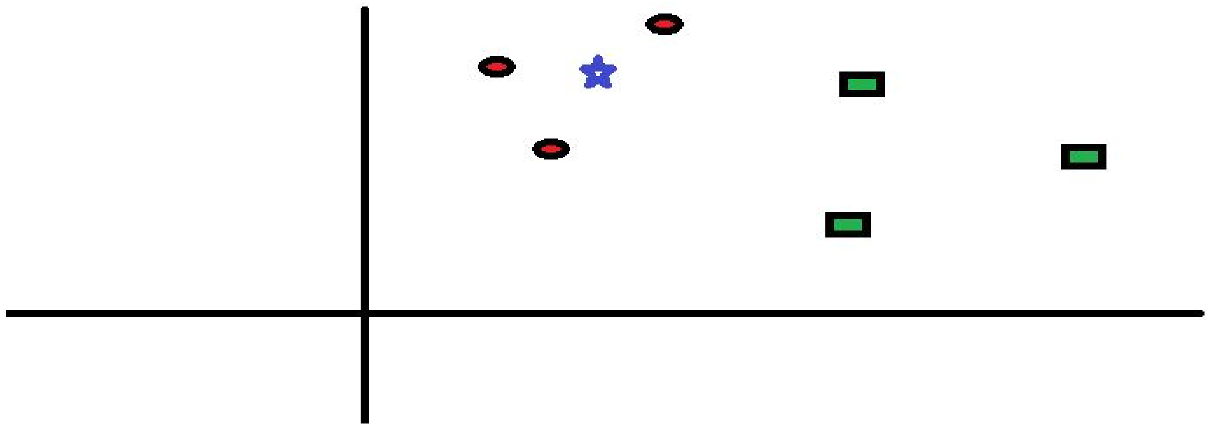


Fig: 4.4 KNN demonstration 1

The three closest points to BS is all RC. Hence, with good confidence level we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from the closest neighbor went to RC. The choice of the parameter K is very crucial in this algorithm.

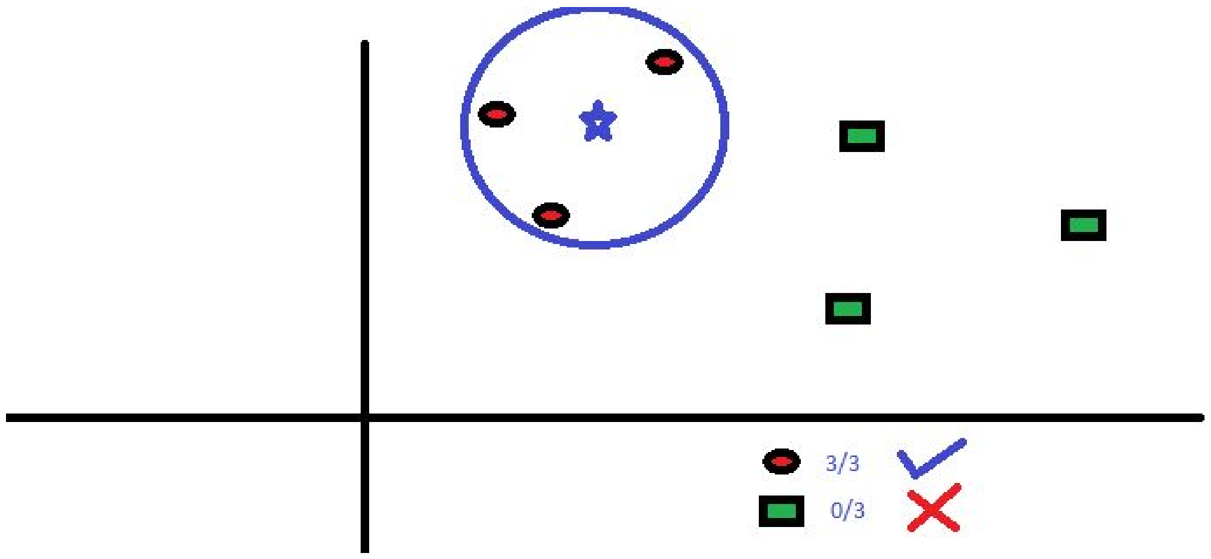


Fig: 4.5 KNN demonstration 2

4.6 Naive Bayes

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:

$$\underset{\text{Posterior Probability}}{P(c \mid x)} = \frac{\overset{\text{Likelihood}}{P(x \mid c)}\,\overset{\text{Class Prior Probability}}{P(c)}}{\underset{\text{Predictor Prior Probability}}{P(x)}}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Fig:4.6 Naive Bayes Formula

- *P*(*c*|*x*) is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).
- *P*(*c*) is the prior probability of *class*.
- *P*(*x*|*c*) is the likelihood which is the probability of *predictor* given *class*.
- *P*(*x*) is the prior probability of *predictor*.

Applications of Naive Bayes

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam email) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)

- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

4.7 SoftMax

The Softmax regression is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1. The output values are between the range [0,1] which is nice because we are able to avoid binary classification and accommodate as many classes or dimensions in our model.

$$P(y=j \mid \Theta^{(i)}) = \frac{e^{\Theta^{(i)}}}{\sum_{j=0}^{k} e^{\Theta_k^{(i)}}} \xleftarrow{\text{Softmax function}}$$

$$\text{where } \Theta = w_0x_0 + w_1x_1 + \ldots + w_kx_k = \sum_{i=0}^{k} w_ix_i = w^Tx$$

Fig:4.7 SoftMax Formula

The function is usually used to compute losses that can be expected when training a data set. Known use-cases of softmax regression are in discriminative models such as Cross-Entropy and Noise Contrastive Estimation. These are only two among various techniques that attempt to optimize the current training set to increase the likelihood of predicting the correct word or sentence.

4.8 System Design

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system.

4.8.1 Use Case Diagram

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined. The use case should contain all system activities that have significance to the users.Use Case Diagram:

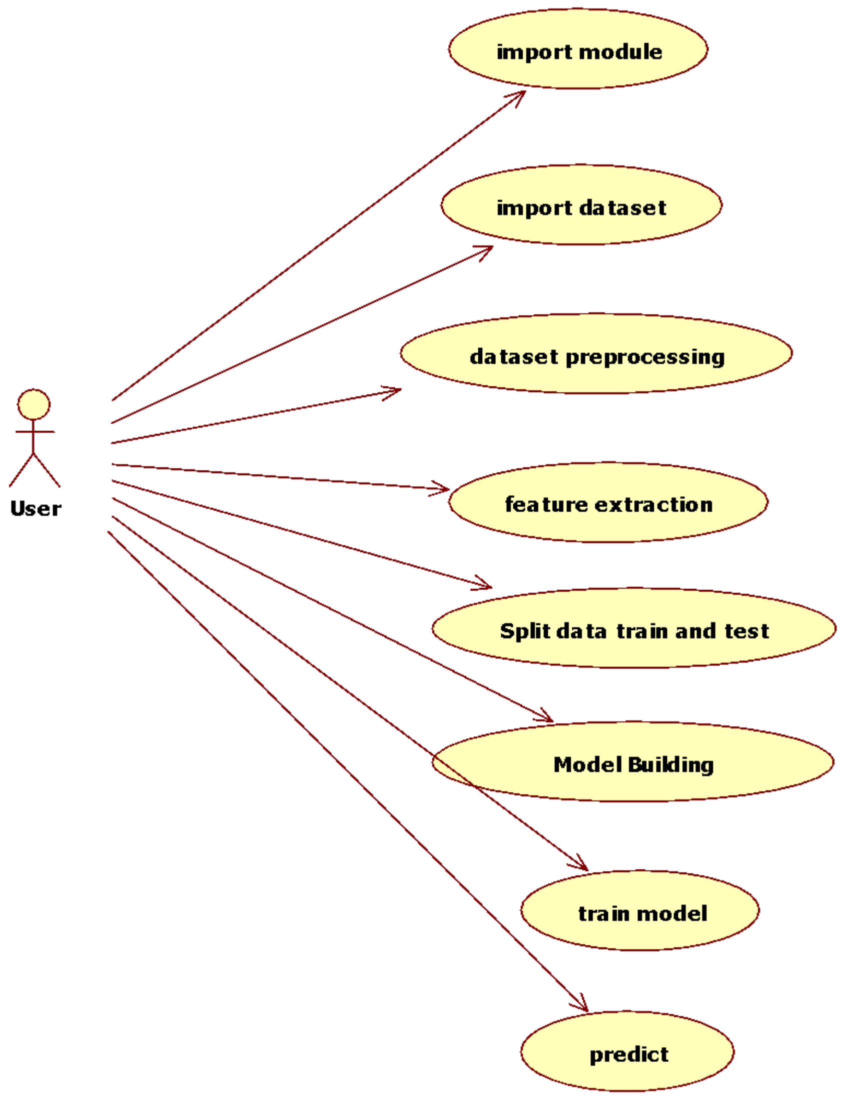


Fig: 4.8 Use Case Diagram

4.8.2 Sequence Diagram

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

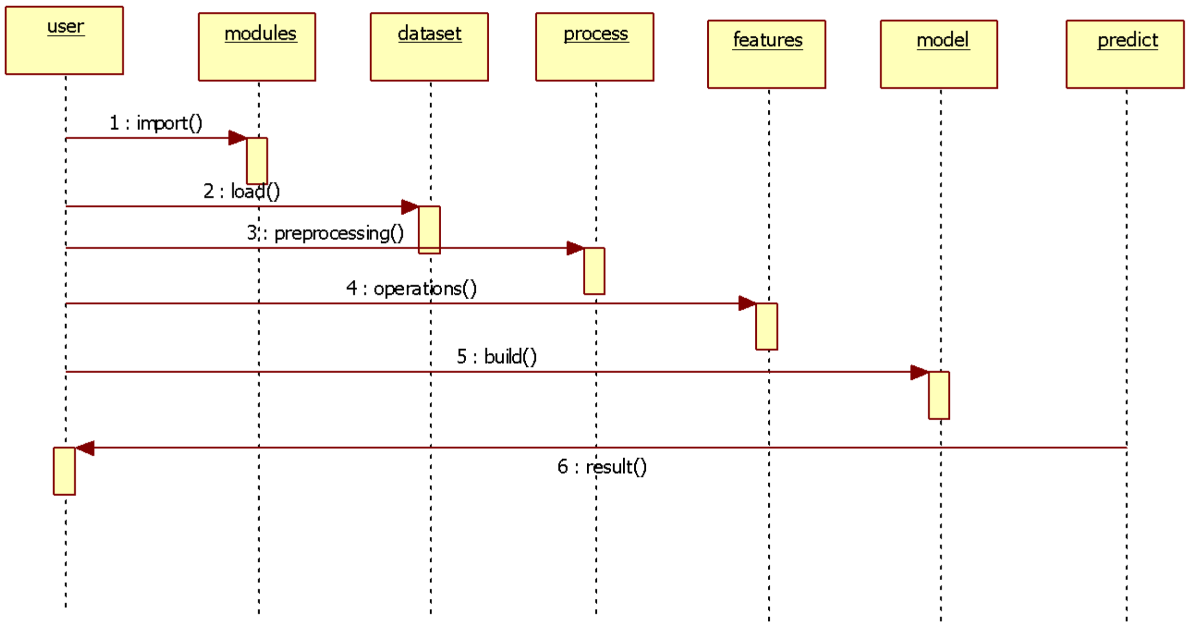


Fig:4.9 Sequence Diagram

4.8.3 Activity Diagram

An activity diagram is used to model a large activity's sequential workflow by focusing on action sequences and respective action initiating conditions. The state of an activity relates to the performance of each workflow step.

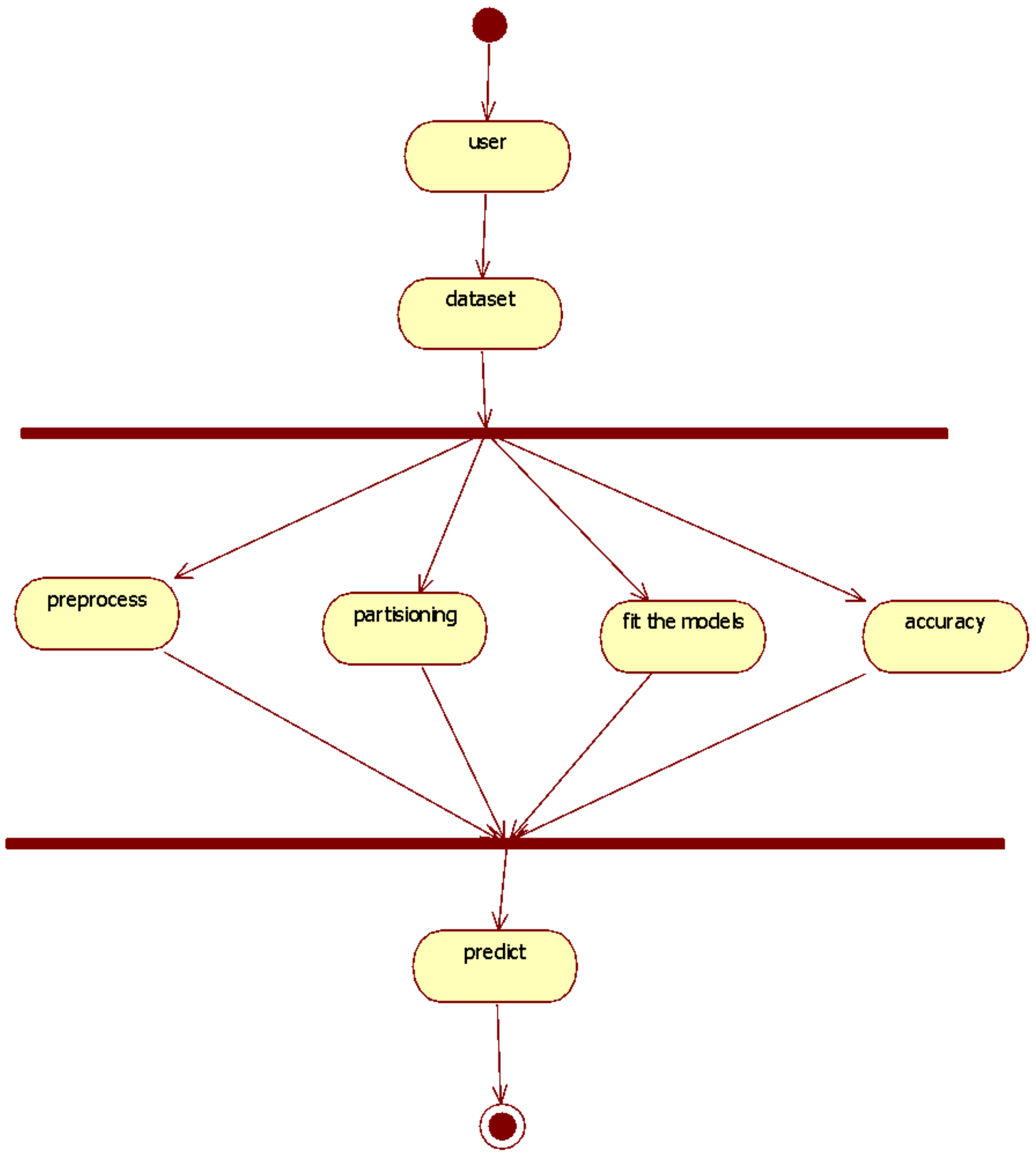


Fig: 4.10 Activity Diagram

4.8.4 Collaboration Diagram

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles.

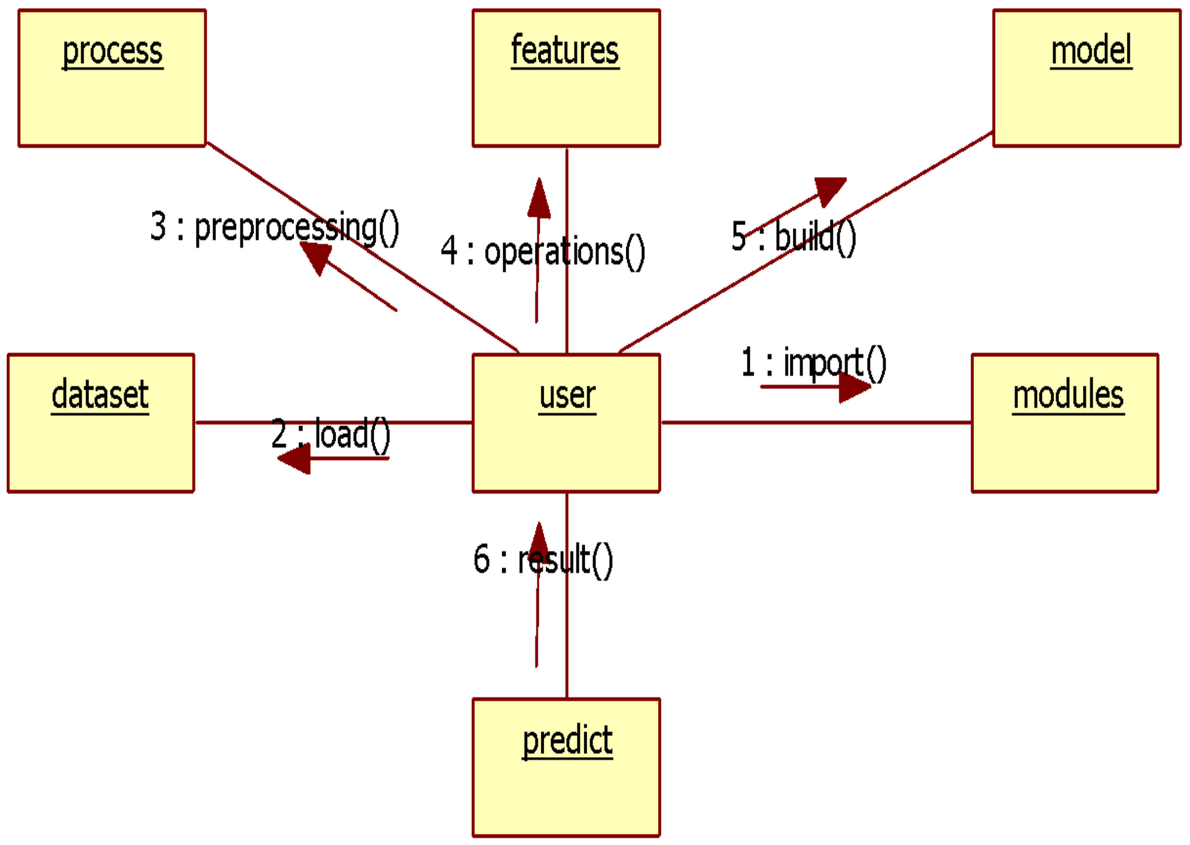


Fig 4.11Collaboration Diagram

4.8.5 Class Diagram

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

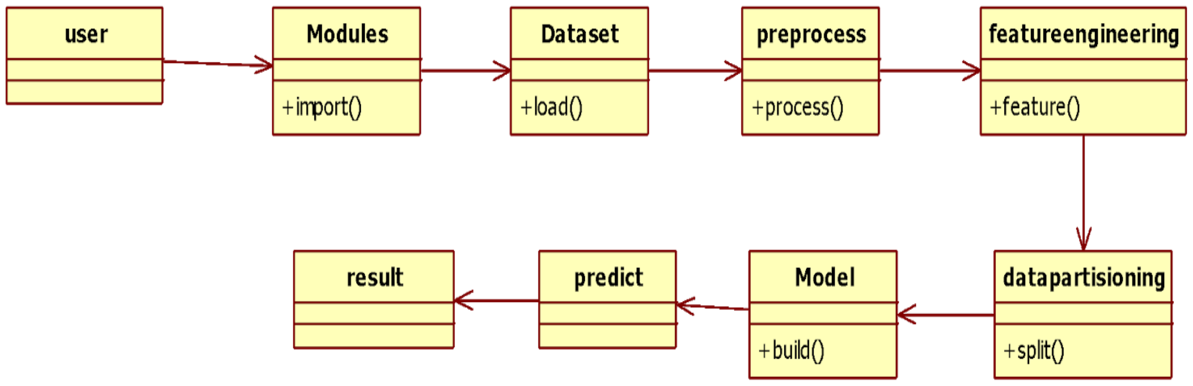


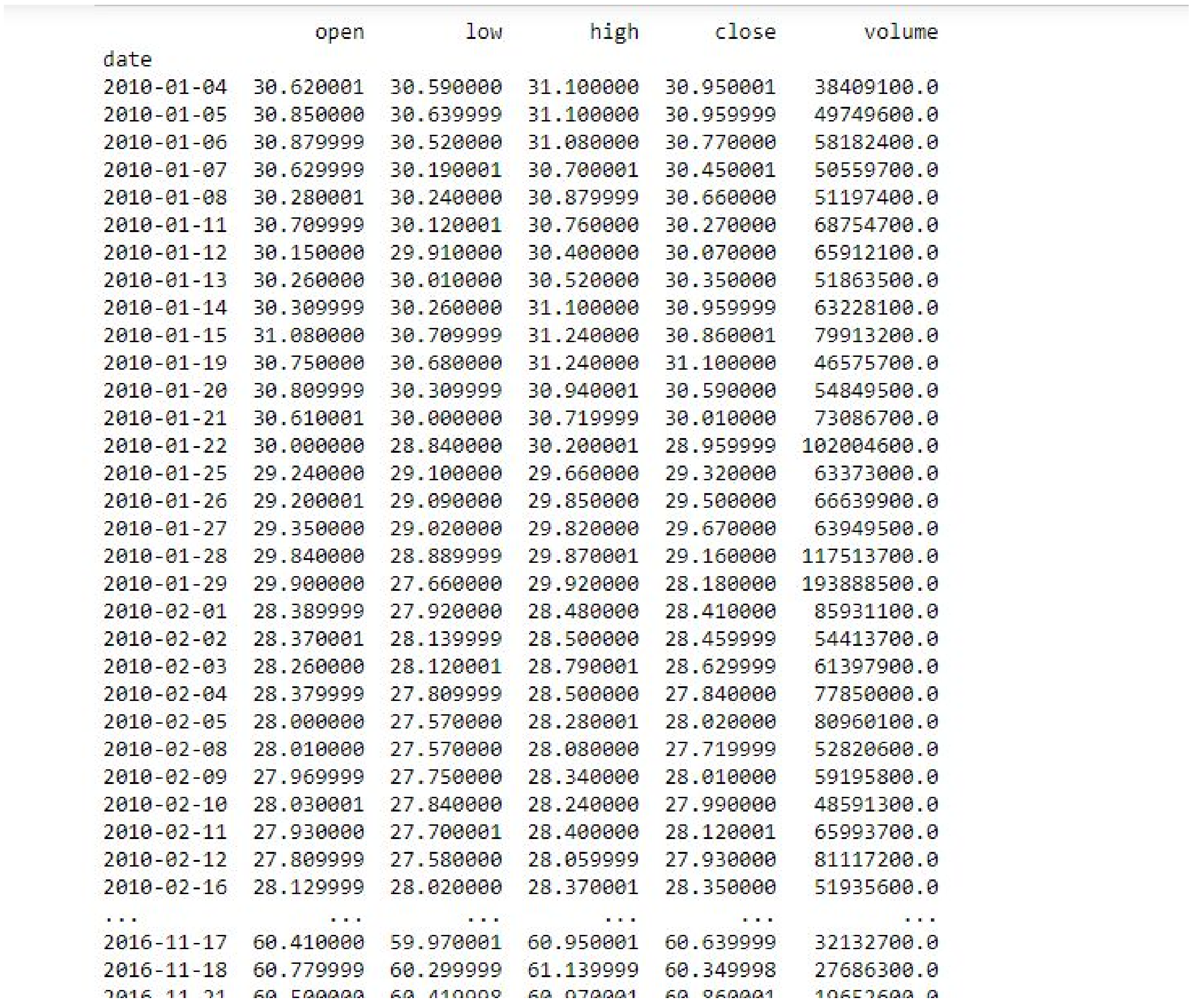
Fig:4.12 Class Diagram

5. Implementation

5.1 Implementing Classification Algorithms

5.1.1 Dataset

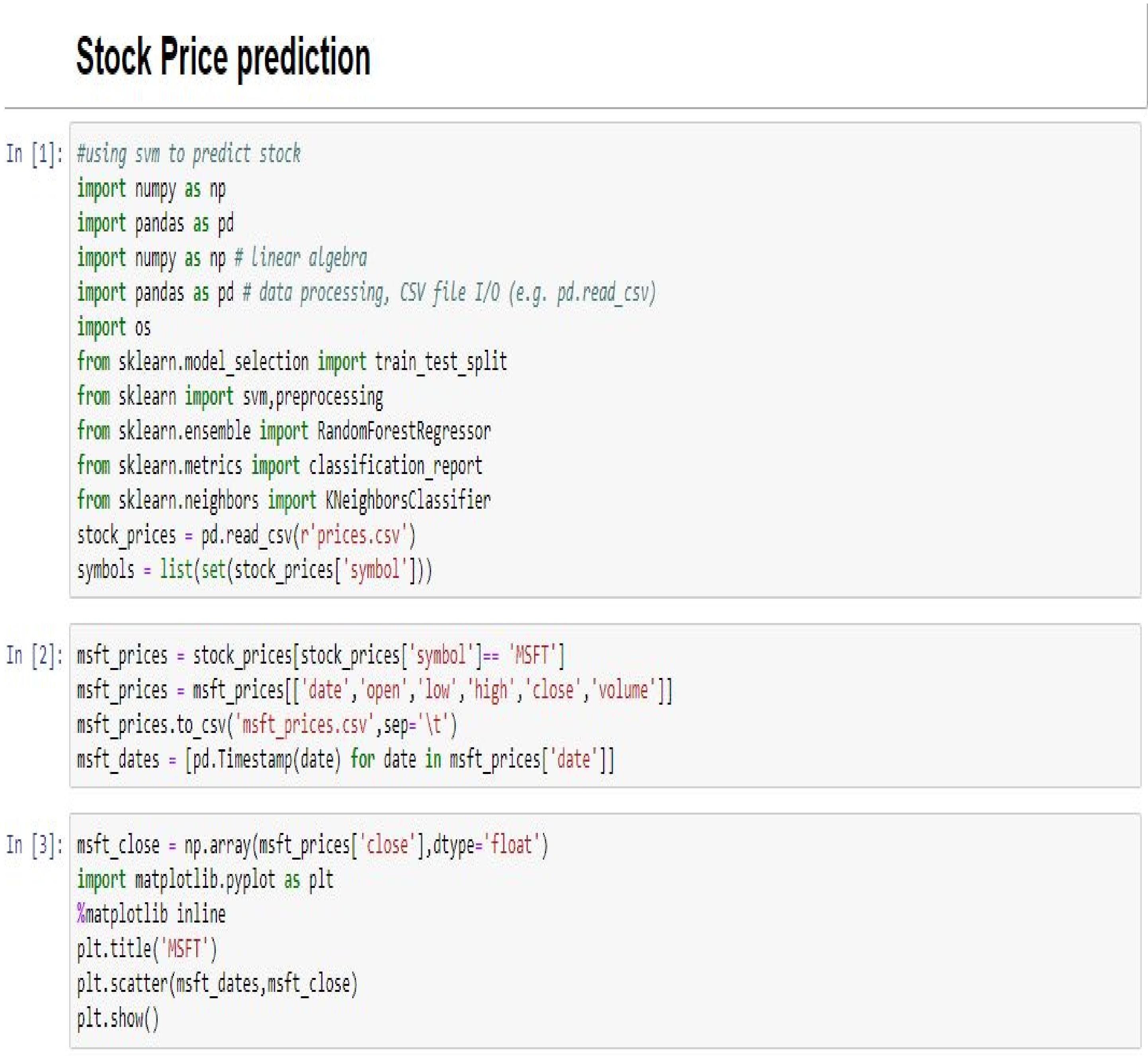
Data set for implementing the algorithm is given below:



```
                 open        low       high      close       volume
date
2010-01-04  30.620001  30.590000  31.100000  30.950001   38409100.0
2010-01-05  30.850000  30.639999  31.100000  30.959999   49749600.0
2010-01-06  30.879999  30.520000  31.080000  30.770000   58182400.0
2010-01-07  30.629999  30.190001  30.700001  30.450001   50559700.0
2010-01-08  30.280001  30.240000  30.879999  30.660000   51197400.0
2010-01-11  30.709999  30.120001  30.760000  30.270000   68754700.0
2010-01-12  30.150000  29.910000  30.400000  30.070000   65912100.0
2010-01-13  30.260000  30.010000  30.520000  30.350000   51863500.0
2010-01-14  30.309999  30.260000  31.100000  30.959999   63228100.0
2010-01-15  31.080000  30.709999  31.240000  30.860001   79913200.0
2010-01-19  30.750000  30.680000  31.240000  31.100000   46575700.0
2010-01-20  30.809999  30.309999  30.940001  30.590000   54849500.0
2010-01-21  30.610001  30.000000  30.719999  30.010000   73086700.0
2010-01-22  30.000000  28.840000  30.200001  28.959999  102004600.0
2010-01-25  29.240000  29.100000  29.660000  29.320000   63373000.0
2010-01-26  29.200001  29.090000  29.850000  29.500000   66639900.0
2010-01-27  29.350000  29.020000  29.820000  29.670000   63949500.0
2010-01-28  29.840000  28.889999  29.870001  29.160000  117513700.0
2010-01-29  29.900000  27.660000  29.920000  28.180000  193888500.0
2010-02-01  28.389999  27.920000  28.480000  28.410000   85931100.0
2010-02-02  28.370001  28.139999  28.500000  28.459999   54413700.0
2010-02-03  28.260000  28.120001  28.790001  28.629999   61397900.0
2010-02-04  28.379999  27.809999  28.500000  27.840000   77850000.0
2010-02-05  28.000000  27.570000  28.280001  28.020000   80960100.0
2010-02-08  28.010000  27.570000  28.080000  27.719999   52820600.0
2010-02-09  27.969999  27.750000  28.340000  28.010000   59195800.0
2010-02-10  28.030001  27.840000  28.240000  27.990000   48591300.0
2010-02-11  27.930000  27.700001  28.400000  28.120001   65993700.0
2010-02-12  27.809999  27.580000  28.059999  27.930000   81117200.0
2010-02-16  28.129999  28.020000  28.370001  28.350000   51935600.0
...               ...        ...        ...        ...          ...
2016-11-17  60.410000  59.970001  60.950001  60.639999   32132700.0
2016-11-18  60.779999  60.299999  61.139999  60.349998   27686300.0
2016-11-21  60.500000  60.419998  60.970001  60.860001   19652600.0
```

Fig: 5.1 Filtered Dataset

5.1.2 Importing Libraries



Stock Price prediction

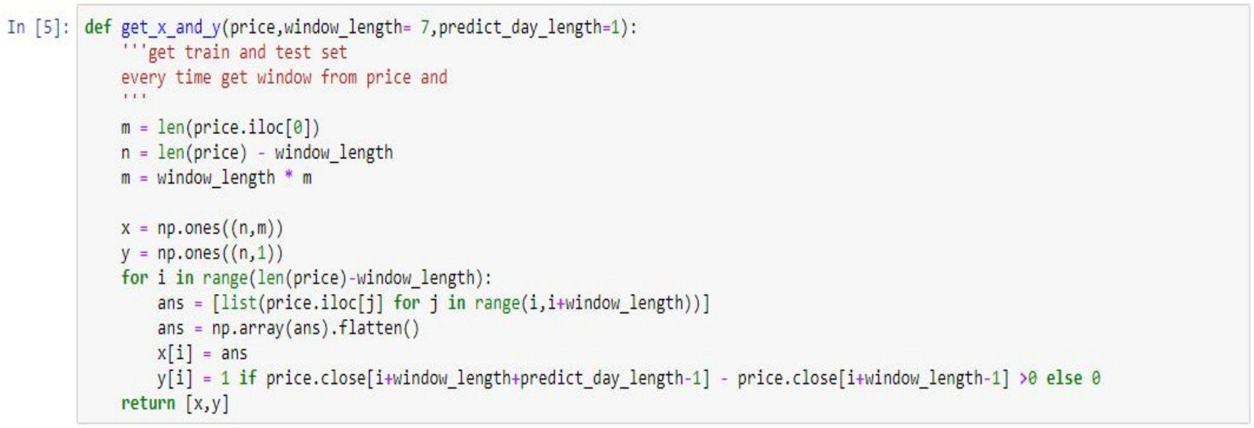
```
In [1]: #using svm to predict stock
        import numpy as np
        import pandas as pd
        import numpy as np # linear algebra
        import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
        import os
        from sklearn.model_selection import train_test_split
        from sklearn import svm,preprocessing
        from sklearn.ensemble import RandomForestRegressor
        from sklearn.metrics import classification_report
        from sklearn.neighbors import KNeighborsClassifier
        stock_prices = pd.read_csv(r'prices.csv')
        symbols = list(set(stock_prices['symbol']))
```

```
In [2]: msft_prices = stock_prices[stock_prices['symbol']== 'MSFT']
        msft_prices = msft_prices[['date','open','low','high','close','volume']]
        msft_prices.to_csv('msft_prices.csv',sep='\t')
        msft_dates = [pd.Timestamp(date) for date in msft_prices['date']]
```

```
In [3]: msft_close = np.array(msft_prices['close'],dtype='float')
        import matplotlib.pyplot as plt
        %matplotlib inline
        plt.title('MSFT')
        plt.scatter(msft_dates,msft_close)
        plt.show()
```

Fig: 5.2 Importing Libraries And Modules

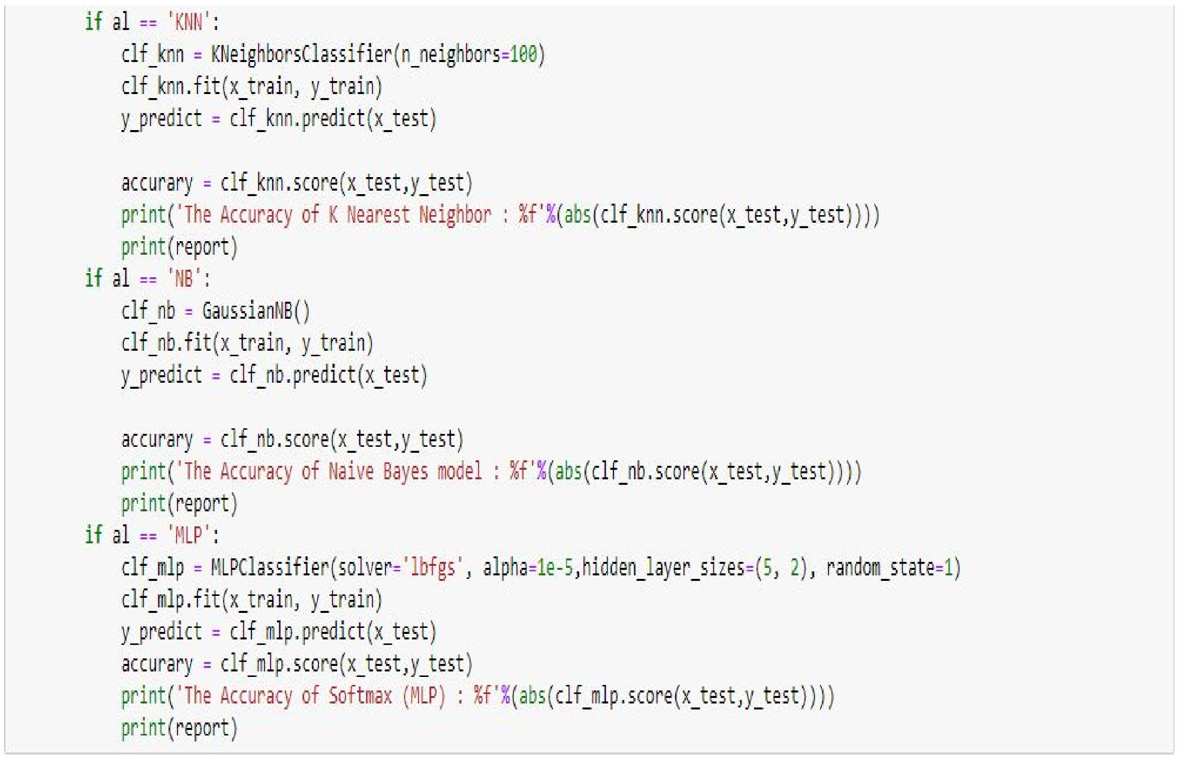
5.1.3 Setting Window Length



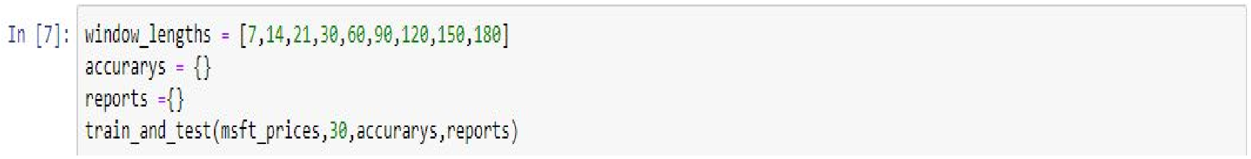
```
In [5]: def get_x_and_y(price,window_length= 7,predict_day_length=1):
            '''get train and test set
            every time get window from price and
            '''
            m = len(price.iloc[0])
            n = len(price) - window_length
            m = window_length * m

            x = np.ones((n,m))
            y = np.ones((n,1))
            for i in range(len(price)-window_length):
                ans = [list(price.iloc[j] for j in range(i,i+window_length))]
                ans = np.array(ans).flatten()
                x[i] = ans
                y[i] = 1 if price.close[i+window_length+predict_day_length-1] - price.close[i+window_length-1] >0 else 0
            return [x,y]
```

Fig:5.3 Window Length

5.1.4 Training Modules



```
        if al == 'KNN':
            clf_knn = KNeighborsClassifier(n_neighbors=100)
            clf_knn.fit(x_train, y_train)
            y_predict = clf_knn.predict(x_test)

            accurary = clf_knn.score(x_test,y_test)
            print('The Accuracy of K Nearest Neighbor : %f'%(abs(clf_knn.score(x_test,y_test))))
            print(report)
        if al == 'NB':
            clf_nb = GaussianNB()
            clf_nb.fit(x_train, y_train)
            y_predict = clf_nb.predict(x_test)

            accurary = clf_nb.score(x_test,y_test)
            print('The Accuracy of Naive Bayes model : %f'%(abs(clf_nb.score(x_test,y_test))))
            print(report)
        if al == 'MLP':
            clf_mlp = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2), random_state=1)
            clf_mlp.fit(x_train, y_train)
            y_predict = clf_mlp.predict(x_test)
            accurary = clf_mlp.score(x_test,y_test)
            print('The Accuracy of Softmax (MLP) : %f'%(abs(clf_mlp.score(x_test,y_test))))
            print(report)
```

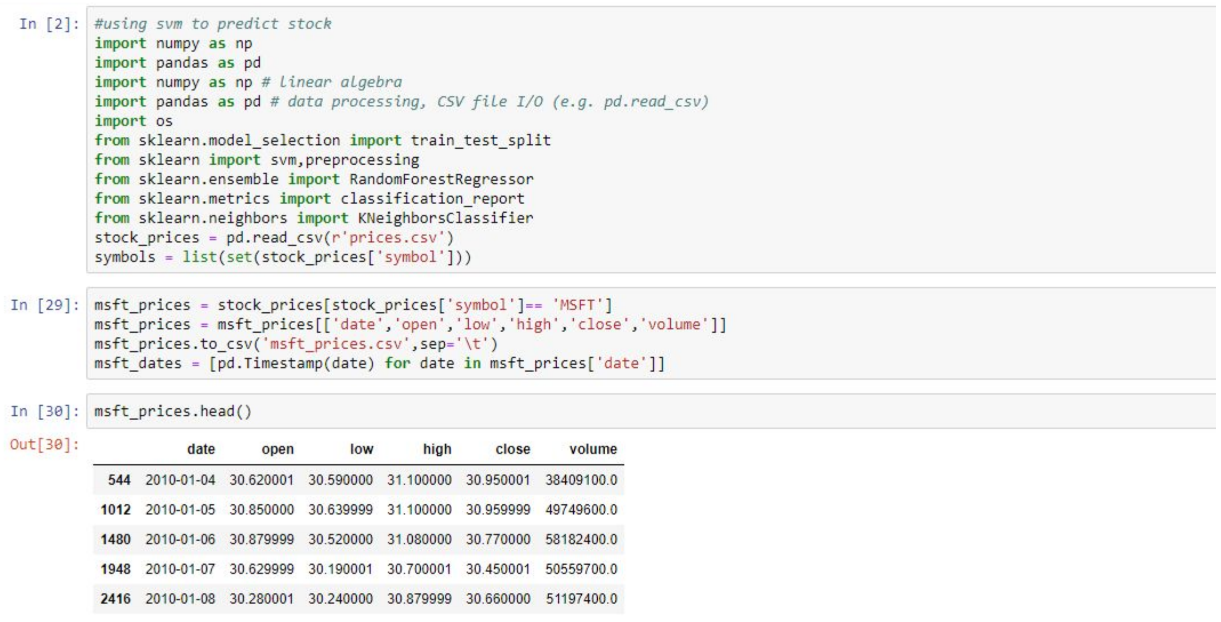


```
In [7]: window_lengths = [7,14,21,30,60,90,120,150,180]
        accurarys = {}
        reports ={}
        train_and_test(msft_prices,30,accurarys,reports)
```

Fig: 5.4 Training Modules

5.2 Implementing Regression Algorithms

5.2.1 Importing libraries and Regression models



```
In [2]: #using svm to predict stock
        import numpy as np
        import pandas as pd
        import numpy as np # linear algebra
        import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
        import os
        from sklearn.model_selection import train_test_split
        from sklearn import svm,preprocessing
        from sklearn.ensemble import RandomForestRegressor
        from sklearn.metrics import classification_report
        from sklearn.neighbors import KNeighborsClassifier
        stock_prices = pd.read_csv(r'prices.csv')
        symbols = list(set(stock_prices['symbol']))
```

```
In [29]: msft_prices = stock_prices[stock_prices['symbol']== 'MSFT']
         msft_prices = msft_prices[['date','open','low','high','close','volume']]
         msft_prices.to_csv('msft_prices.csv',sep='\t')
         msft_dates = [pd.Timestamp(date) for date in msft_prices['date']]
```

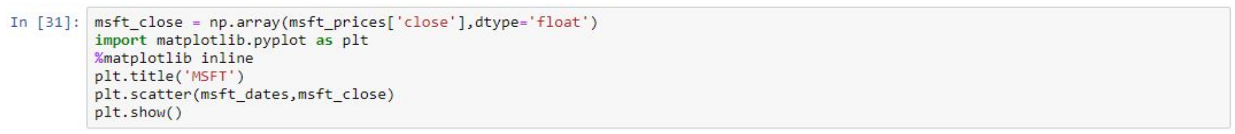
```
In [30]: msft_prices.head()
```

Out[30]:

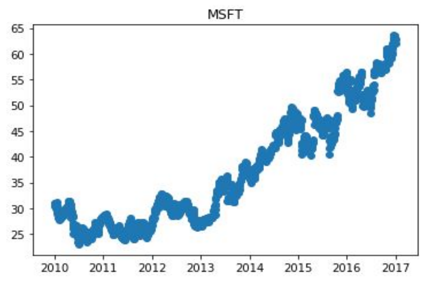
	date	open	low	high	close	volume
544	2010-01-04	30.620001	30.590000	31.100000	30.950001	38409100.0
1012	2010-01-05	30.850000	30.639999	31.100000	30.959999	49749600.0
1480	2010-01-06	30.879999	30.520000	31.080000	30.770000	58182400.0
1948	2010-01-07	30.629999	30.190001	30.700001	30.450001	50559700.0
2416	2010-01-08	30.280001	30.240000	30.879999	30.660000	51197400.0

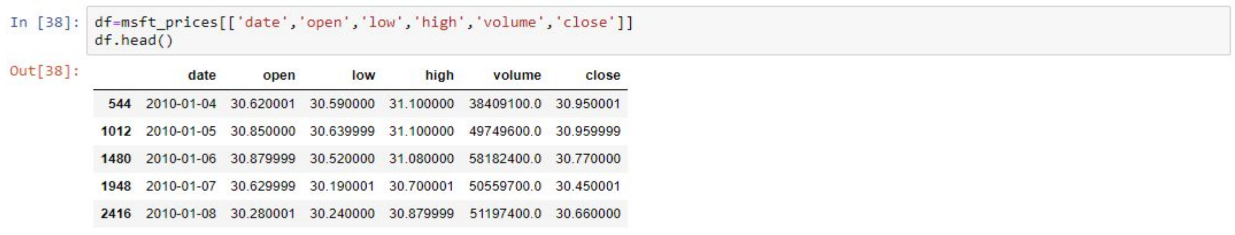
Fig: 5.5 Libraries, Regression Models and Dataset

5.2.2 Graph plot of Dataset



```
In [31]: msft_close = np.array(msft_prices['close'],dtype='float')
         import matplotlib.pyplot as plt
         %matplotlib inline
         plt.title('MSFT')
         plt.scatter(msft_dates,msft_close)
         plt.show()
```





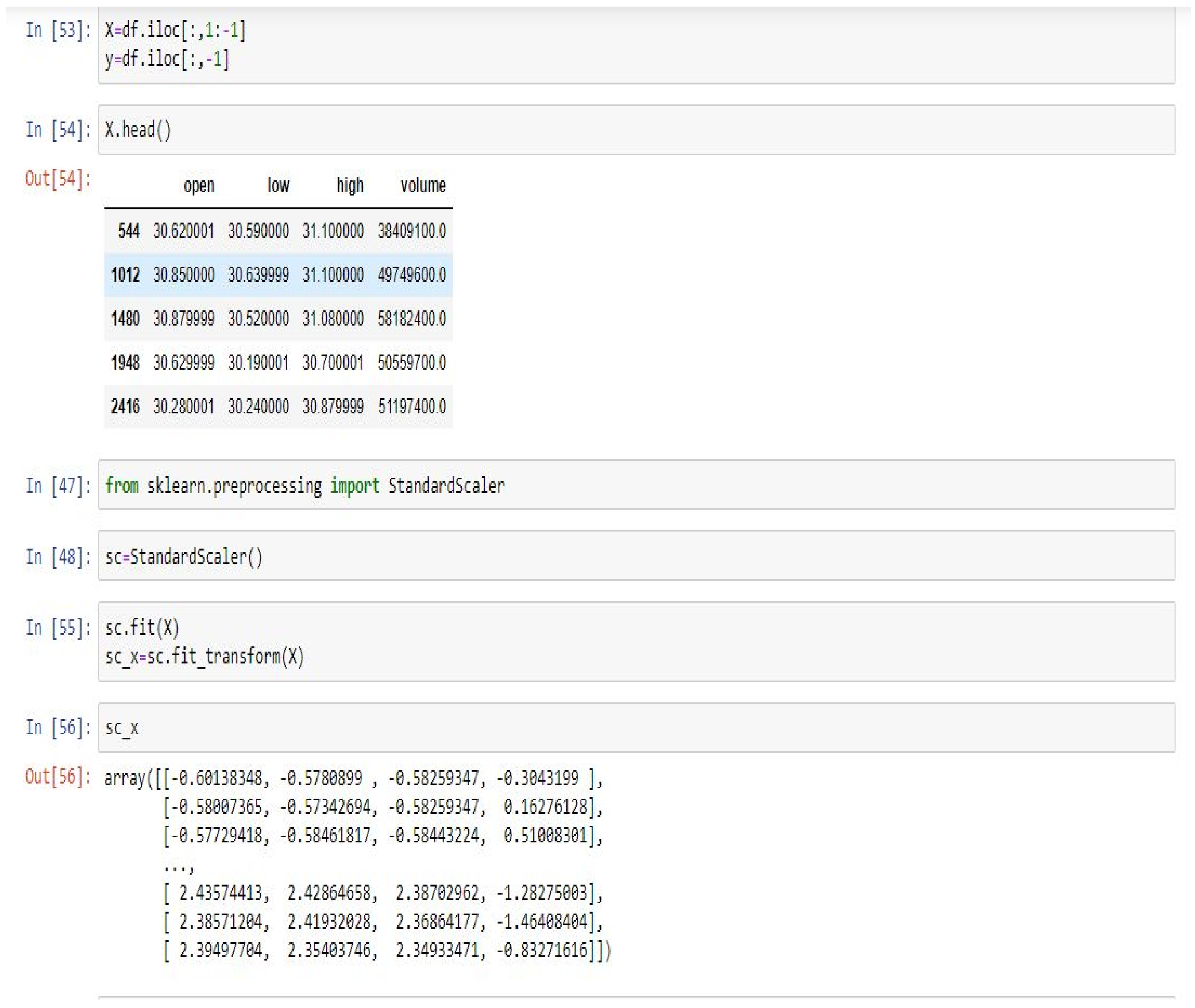
```
In [38]: df=msft_prices[['date','open','low','high','volume','close']]
         df.head()
```

Out[38]:

	date	open	low	high	volume	close
544	2010-01-04	30.620001	30.590000	31.100000	38409100.0	30.950001
1012	2010-01-05	30.850000	30.639999	31.100000	49749600.0	30.959999
1480	2010-01-06	30.879999	30.520000	31.080000	58182400.0	30.770000
1948	2010-01-07	30.629999	30.190001	30.700001	50559700.0	30.450001
2416	2010-01-08	30.280001	30.240000	30.879999	51197400.0	30.660000

Fig: 5.6 Graph of Dataset

5.2.3 Applying Standard Scaler



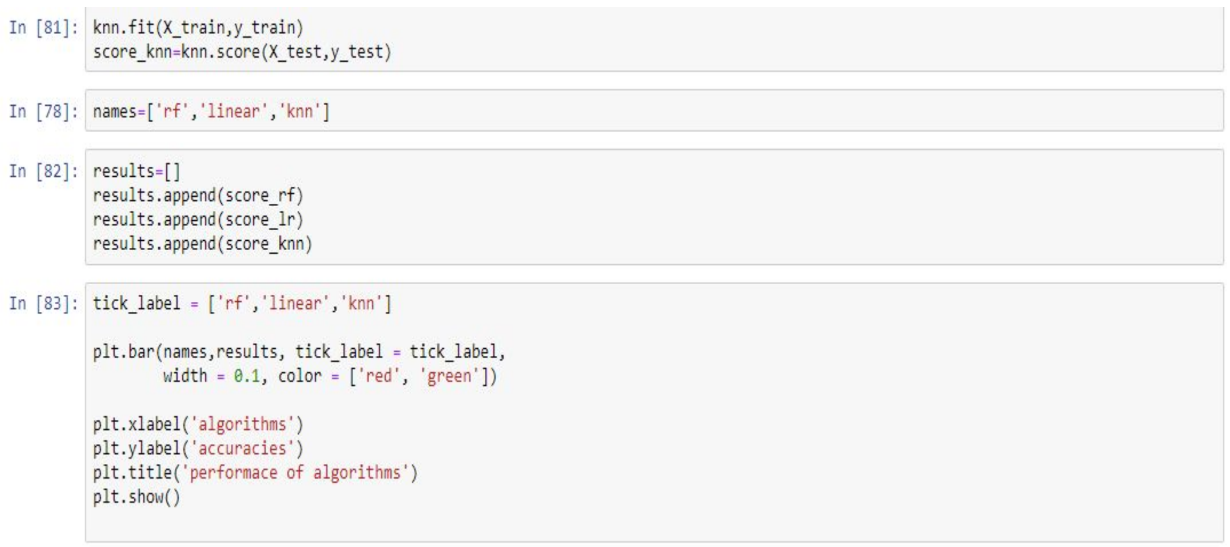
```
In [53]: X=df.iloc[:,1:-1]
         y=df.iloc[:,-1]

In [54]: X.head()

Out[54]:
```

	open	low	high	volume
544	30.620001	30.590000	31.100000	38409100.0
1012	30.850000	30.639999	31.100000	49749600.0
1480	30.879999	30.520000	31.080000	58182400.0
1948	30.629999	30.190001	30.700001	50559700.0
2416	30.280001	30.240000	30.879999	51197400.0

```
In [47]: from sklearn.preprocessing import StandardScaler

In [48]: sc=StandardScaler()

In [55]: sc.fit(X)
         sc_x=sc.fit_transform(X)

In [56]: sc_x

Out[56]: array([[-0.60138348, -0.5780899 , -0.58259347, -0.3043199 ],
                [-0.58007365, -0.57342694, -0.58259347,  0.16276128],
                [-0.57729418, -0.58461817, -0.58443224,  0.51008301],
                ...,
                [ 2.43574413,  2.42864658,  2.38702962, -1.28275003],
                [ 2.38571204,  2.41932028,  2.36864177, -1.46408404],
                [ 2.39497704,  2.35403746,  2.34933471, -0.83271616]])
```

Fig:5.7 Applying Standard Scaler

5.2.4 Graph Plotting Performance Analysis



```
In [81]: knn.fit(X_train,y_train)
         score_knn=knn.score(X_test,y_test)

In [78]: names=['rf','linear','knn']

In [82]: results=[]
         results.append(score_rf)
         results.append(score_lr)
         results.append(score_knn)

In [83]: tick_label = ['rf','linear','knn']

         plt.bar(names,results, tick_label = tick_label,
                 width = 0.1, color = ['red', 'green'])

         plt.xlabel('algorithms')
         plt.ylabel('accuracies')
         plt.title('performace of algorithms')
         plt.show()
```

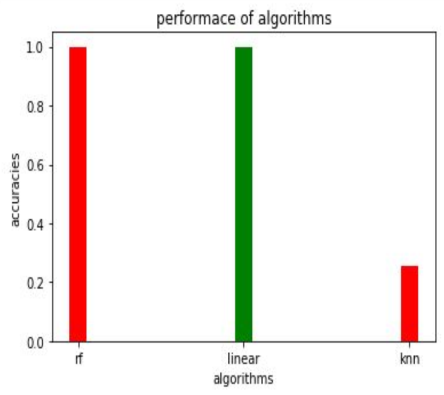
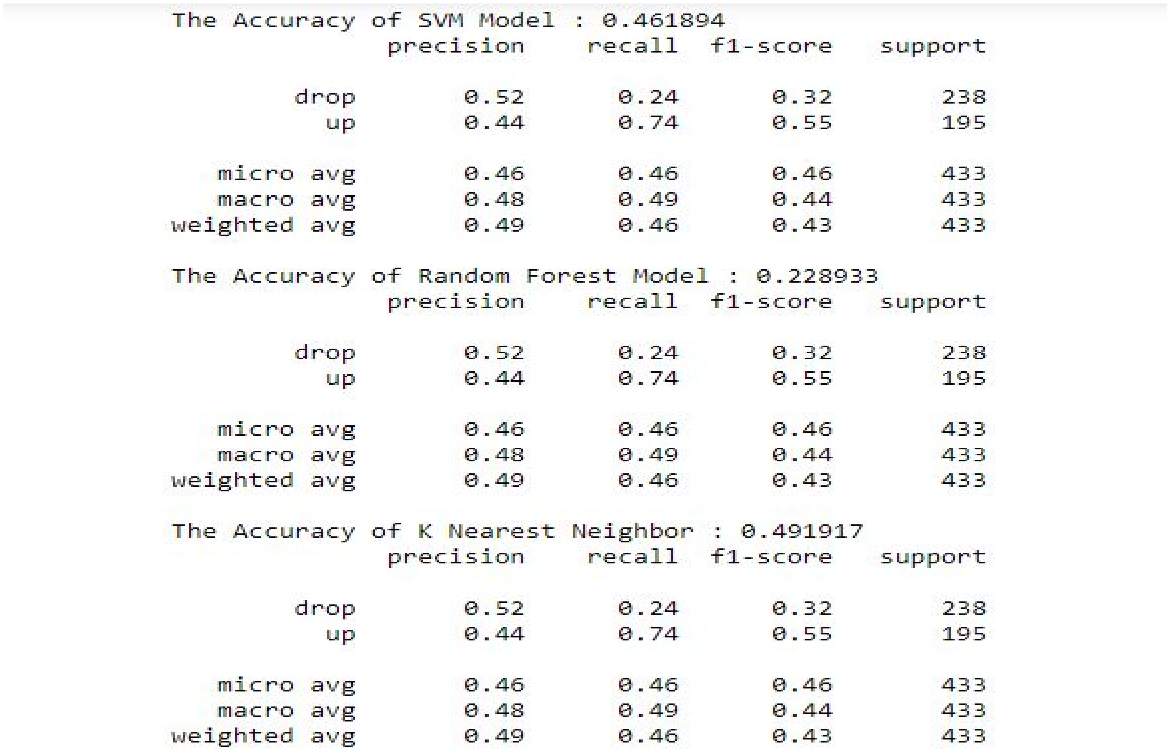


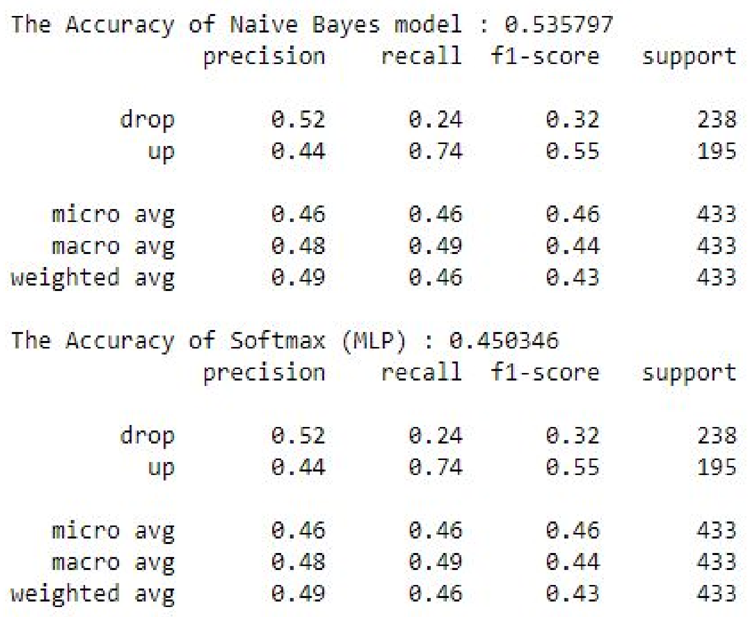
Fig:5.8 Performance of Algorithms

6. EVALUATION AND RESULTS

6.1 Performance of different Classification Models



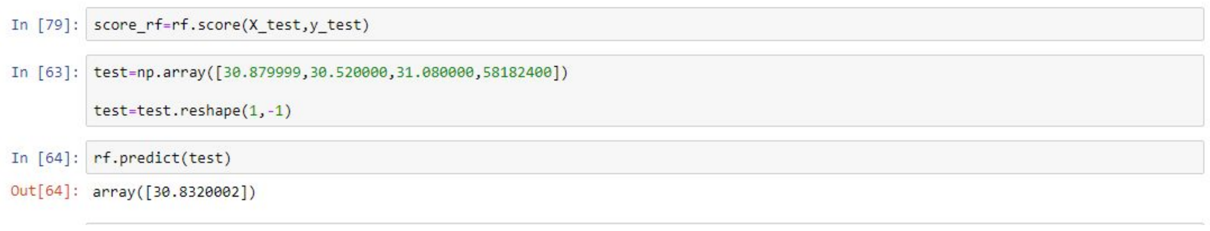
```
The Accuracy of SVM Model : 0.461894
              precision    recall  f1-score   support

        drop       0.52      0.24      0.32       238
          up       0.44      0.74      0.55       195

   micro avg       0.46      0.46      0.46       433
   macro avg       0.48      0.49      0.44       433
weighted avg       0.49      0.46      0.43       433

The Accuracy of Random Forest Model : 0.228933
              precision    recall  f1-score   support

        drop       0.52      0.24      0.32       238
          up       0.44      0.74      0.55       195

   micro avg       0.46      0.46      0.46       433
   macro avg       0.48      0.49      0.44       433
weighted avg       0.49      0.46      0.43       433

The Accuracy of K Nearest Neighbor : 0.491917
              precision    recall  f1-score   support

        drop       0.52      0.24      0.32       238
          up       0.44      0.74      0.55       195

   micro avg       0.46      0.46      0.46       433
   macro avg       0.48      0.49      0.44       433
weighted avg       0.49      0.46      0.43       433
```



```
The Accuracy of Naive Bayes model : 0.535797
              precision    recall  f1-score   support

        drop       0.52      0.24      0.32       238
          up       0.44      0.74      0.55       195

   micro avg       0.46      0.46      0.46       433
   macro avg       0.48      0.49      0.44       433
weighted avg       0.49      0.46      0.43       433

The Accuracy of Softmax (MLP) : 0.450346
              precision    recall  f1-score   support

        drop       0.52      0.24      0.32       238
          up       0.44      0.74      0.55       195

   micro avg       0.46      0.46      0.46       433
   macro avg       0.48      0.49      0.44       433
weighted avg       0.49      0.46      0.43       433
```

Fig:6.1 Accuracies of different Classification models

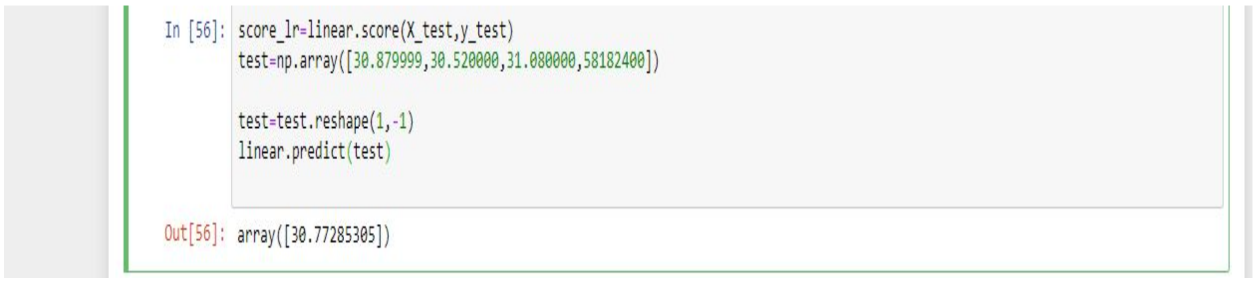
6.2 Price Prediction by Random Forest



```
In [79]: score_rf=rf.score(X_test,y_test)

In [63]: test=np.array([30.879999,30.520000,31.080000,58182400])

         test=test.reshape(1,-1)

In [64]: rf.predict(test)

Out[64]: array([30.8320002])
```

Fig:6.2 Random Forest Output

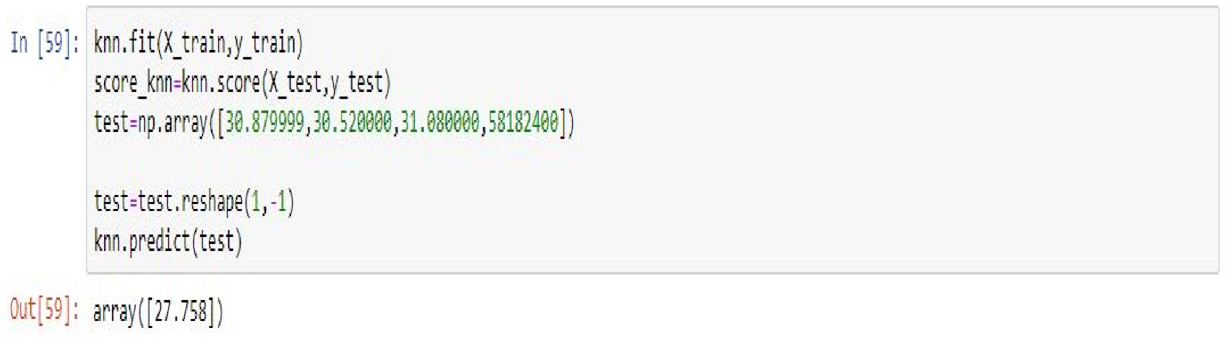
6.3 Price Prediction By Linear Regression



```
In [56]: score_lr=linear.score(X_test,y_test)
         test=np.array([30.879999,30.520000,31.080000,58182400])

         test=test.reshape(1,-1)
         linear.predict(test)

Out[56]: array([30.77285305])
```

Fig:6.3 Linear Regression Output

6.4 Price Prediction by KNN



```
In [59]: knn.fit(X_train,y_train)
         score_knn=knn.score(X_test,y_test)
         test=np.array([30.879999,30.520000,31.080000,58182400])

         test=test.reshape(1,-1)
         knn.predict(test)

Out[59]: array([27.758])
```

Fig:6.4 K Nearest Neighbors Output

7. CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

Supervised machine learning algorithms SVM, Random Forest, KNN, Naive Bayes Algorithm, and Softmax Algorithm have been applied for the stock price prediction. The results reveal that for large dataset, Random Forest Algorithm outperforms all the other algorithms in terms of accuracy and when the size of the dataset is reduced to almost half of the original, then Naïve Bayes Algorithm shows the best results in terms of accuracy. Also, reduction in the number of technical indicators reduces the accuracy of each algorithm in predicting the stock market trends.Different regression models have been used in predicting the future value of a stock when some technical indicators are given.

7.2 Future Enhancements

The outcomes of this project is calculated only on the existing Dataset of the organization It is necessary that additional datasets should be considered for the evaluation of different classification problems as the information growth in the recent technology is extending to heights beyond assumptions. Recent field of technology is growing and data are by nature dynamic. Hence, further classification of the entire system needs to be implemented right from the scratch since the results from the old process have become obsolete. The scope of future work can deal with Incremental learning, which stores the existing model and processes the new incoming data more efficiently. More specifically, the models with incremental learning can be used in categorization process to improve the following aspects in each type of problems.

REFERENCES

[1] G. Zhang, B.E. Patuwo, M.Y. Hu, Forecasting with arti-cial neural networks: the state of the , Int. J. Forecasting 14 (1998) 35–62.

[2] K. Kim, Financial time series forecasting using support vector machines. Neurocomputing, 55, pp. 307–319, 2003.

[3] T. Manojlović* and I. Štajduhar*, Predicting Stock Market Trends Using Random Forest: A Sample of the Zagreb Stock Exchange, IEEE International Convention, pp. 1189- 1193, 2015.

[4] Yuqing Dai, Yuning Zhang, Machine Learning in Stock Price Trend Forecasting, 2013.

[5] Steven B. Achelis, "Technical Analysis from A to Z", 2nd ed., McGraw-Hill Education, 2000.

[6] Koosha Golmohammadi, Osmar R. Zaiane and David Díaz, Detecting Stock Market Manipulation using Supervised Learning Algorithms, IEEE International Conference on Data Science and Advanced Analytics, pp. 435-441, 2014.

[7] P. Hajek, Forecasting Stock Market Trend using Prototype Generation Classifiers, WSEAS Transactions on Systems, Vol.11, No. 12, pp. 671-80, 2012.

[8] A. Kar, "Stock prediction using artificial neural networks," Dept. of Computer Science and Engineering, IIT Kanpur

[9] D. d. l. F. Rafael Rosillo, Javier Giner, "Stock market simulation using support vector machines," Journal of Forecasting, vol. 33, pp. 488–500, July 2014.

[10] K. jae Kim, "Financial time series forecasting using support vector machines," Neurocomputing, vol. 55, 2003.

[11] M. W. W. James H. Stock, Introduction to Econometrics. Addison-Wesley, 2015.

[12] The original paper: Corinna Cortes and V. Vapnik, "Support-Vector Networks", *Machine Learning*, **20**, 1995

[13] The standard reference: Scholkopf and Smola, *Learning with Kernels*, 2002

[14]Algorithms for solving the SVM are discussed E. Osuna, R. Freund, and F. Girosi. "Improved training algorithm for support vector machines." NNSP'97, 1997.

[15] http://citeseer.ist.psu.edu/osuna97improved.html, and in J. Platt, *Fast Training of Support Vector Machines using Sequential Minimal Optimization*, in Advances in Kernel Methods - Support Vector Learning, B. Sch?kopf, C. Burges, and A. Smola, eds., MIT Press, 1999.