Institute for Advanced Computing and Software Development (IACSD)
Akurdi, Pune – 411044

Documentation On

"Portfolio Management"

Submitted By:

Group No: 14

Siddhi Bhosale 220341225050

Kalpana Vyavahare 220341225022

Mr. Prashant Karhale
Centre Coordinator

Mr. Akshay Tilekar
Project Guide

1. INTRODUCTION

1.1 Overview

Portfolio management is the art and science of making optimal investment decisions to minimise the risks and maximizing the returns, to meet the investor's financial objectives and risk tolerance. Active portfolio management means strategically buying and selling securities (stocks, bonds, options, commodity, property, cash, etc) in an effort to beat the market. Whereas passive portfolio management means matching the returns of the market by replicating some indexes.

Artificial intelligence-based machine learning (ML) models are the latest technological inovation to hit the portfolio management industry. And while, at least so far, ML models have not been a panacea for investment managers, they've proven themselves a valuable tool by augmenting human decision-making around important activities such as asset allocation, risk management, and portfolio construction.

Machine learning can transform the way investment strategies are administered by all types of managers. Even the most fundamental, non-quantitative managers will be generating ideas from data that originally was sourced and synthesized via ML. For example, deep learning's ability to create structured data could be used to extract topic and sentiment from text sources such as earnings calls, SEC filings, and social media; or for the analysis of satellite imagery for parking lot or crop data; or to evaluate location data from mobile phones.

Machine learning will become increasingly important for asset management and most firms will be utilizing either machine learning tools or data within the next few years. Human involvement will still be critical for risk management and framework selection, but increasingly the strategy innovation process will be automated.

The prospect of better outcomes from deeper analysis of vast amounts of newly available data is enticing. Portfolio managers can turn to artificial intelligence and its subset, machine learning, to fine-tune their investment processes. Recent advances in the investment arena, such as robo-advisers provide asset allocation advice to individual investors are based on machine-learning processes.

1.2 Motivation

Last decades, a considerable progress has been made in the financial mathematics field. Many subjects, such as stochastic modelling, PDEs resolutions, exotic derivatives pricing and trends predictions have been of great interest both within the academics and practitioners. The use of the increasingly powerful computational abilities helped addressing those issues in a new way, and develop model and predict in an almost-automatized way.

The study examines how ML can be employed portfolio management process. It observes the advantages ML has over alternative methods, in terms of its speed and efficiency in handling large amounts of data. The study also identifies risk factors such as poor quality data, and the sheer complexity of the information that is produced, and looks ahead to a time when both the use and study of ML may have been re-shaped by regulation.

The study concludes that ML techniques show great promise for portfolio management but investors should be aware that potential pitfalls exist.

2. PROBLEM STATEMENT

- **Portfolio Management using machine learning**

2.1 Abstract:

Portfolio Management is a concept of selecting the proportions of various assets that is to be held in a portfolio to have a good return without a significant risk exposure. Portfolio Management is one important building block in financial management and investment banking. One possible strategy for minimisation of risk is by enlarging or varying its field of operation for the portfolio. Constructing an optimal portfolio by judging and selecting the best possible combinations of different portfolio is a computationally challenging problem since it comes up with an exponential complexity. Here, we have proposed a Regression model and simple k-means, KNN based clustering strategy for an optimal portfolio. S&P500 stocks are represented by their fundamental financial ratios. Clustering is performed, and a prototype stock is selected from each of the clusters. An equal investment strategy demonstrates superior return as compared to the indices as well as top mutual funds. The classification is done by considering a host of investment parameters. We compare the rate of return of these stocks to the benchmark of S&P500.

2.2 Aim and Objectives:

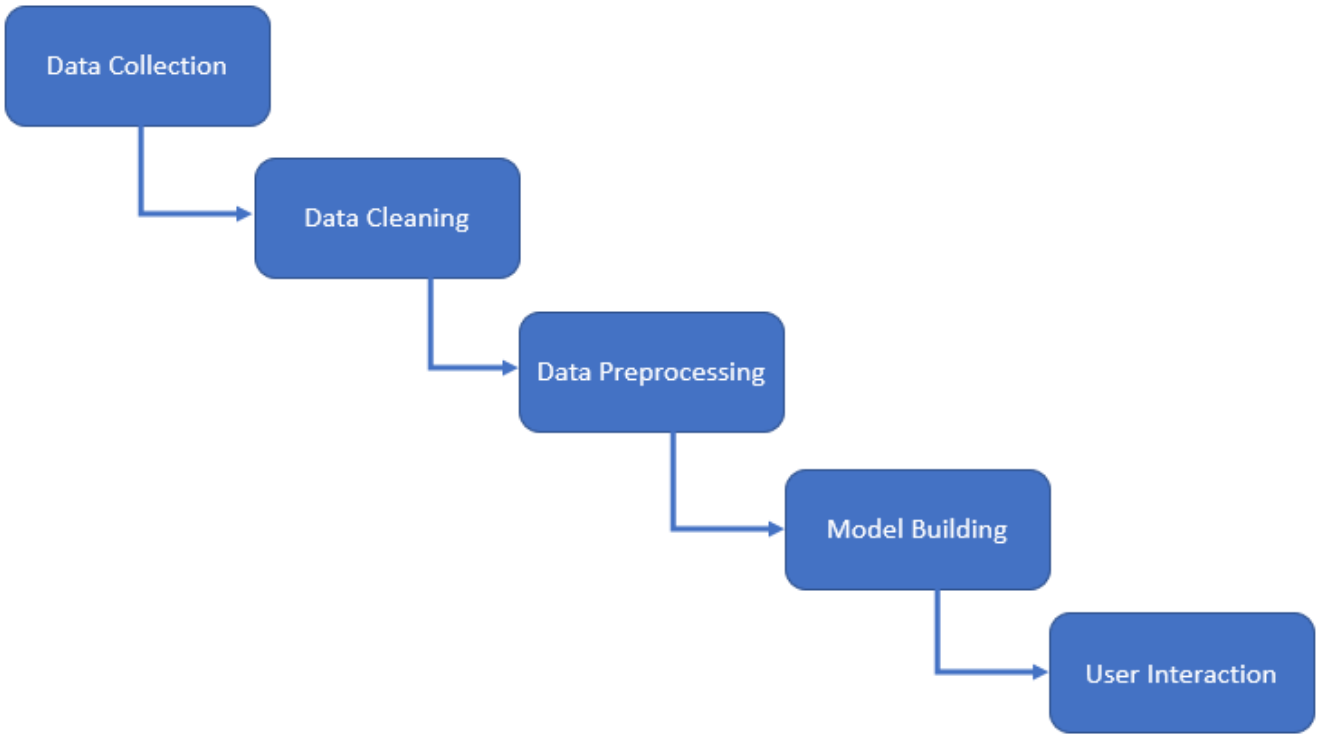
The aim of the project is portfolio management in order to help the invest in different/correct type of stock using ML. We select the stock's historical data and fit the Stock for given trade & other historical information and come up with an effective model to represent the stock using different Machine learning model with hyperparameter tuning. The outcome of this regression model is passed to the cluster model that will decide the cluster for stock in investor portfolio.
A good portfolio should have multiple objectives and achieve a sound balance among them. Any one objective should not be given undue importance at the cost of others. Presented below are some important objectives of portfolio management.

- Forecasting stock price which help investor to identified return.
- Cluster the stock base on prediction
- Manage portfolio for investor

3. OVERALL DESCRIPTION

3.1 Workflow of Project:

- The diagram below shows the workflow of this project



3.2 Data collection:

Collecting data allows to capture a record of past events so that we can use data analysis to find recurring patterns. From those patterns you build predictive models using machine learning algorithms that look for trends and predict future changes.

Predictive models are only good as the data from which they are build, so good data collection practices are crucial to developing high performance models. The data need to be error-free and contain relevant information for the task at hand.

So, for over project we collect data from yahoo finance api yfinance. The Yahoo Finance API is a range of libraries/APIs/methods to obtain historical and real time data for a variety of financial markets and products

One good reason for selecting Yahoo Finance API is because it can be completely free. However, there are also third-party APIs with more support that do charge for their higher usage plans, but even they tend to have free tier options.

```
df_1 = pd.DataFrame(columns=['Date','open','high','low','close','volume','TICKER']).T
dfs =[]

for ticker in ticker_list:
  print(ticker)
  if yf.Ticker('ticker'):
    yf.pdr_override()
    df = pd.DataFrame()
    start = datetime.strptime('2011-01-01', '%Y-%m-%d')
    end = datetime.strptime('2020-01-01', '%Y-%m-%d')
    df = pdr.get_data_yahoo(ticker, start, end)
    df.reset_index(inplace=True)
    df = df.drop(['Adj Close'], axis=1)
    df['TICKER'] = ticker
    dfs.append(df)
  else:
    print("Cannot get info of {ticker}, it probably does not exist")

pd.concat(dfs).to_csv('stock.csv',header=['Date','open','high','low','close','volume','TICKER'] ,index=False)
```

```
[**********100%**********]  1 of 1 completed
```

3.3 Data Cleaning

Data cleaning is one of the important part of machine learning. It play a significant part in building a model. It surely isn't the fanciest part of machine learning and at the same time, there aren't any hidden tricks or secrets to uncover. However, the success or failure of a project relies on proper data cleaning.

Real-world raw data and images are often incomplete, inconsistent and lacking in certain behaviours or trends. They are also likely to contain many errors. So, once collected, they are pre-processed into a format the machine learning algorithm can use for the model

We perform several data cleaning step on our date set to enhance the quality of data set.

- Import Dataset.
- Check null/ missing value in dataset, and fill that null and missing value with ffill method.
- Find out the duplicated data if present or not and if present then drop that record form dataset.

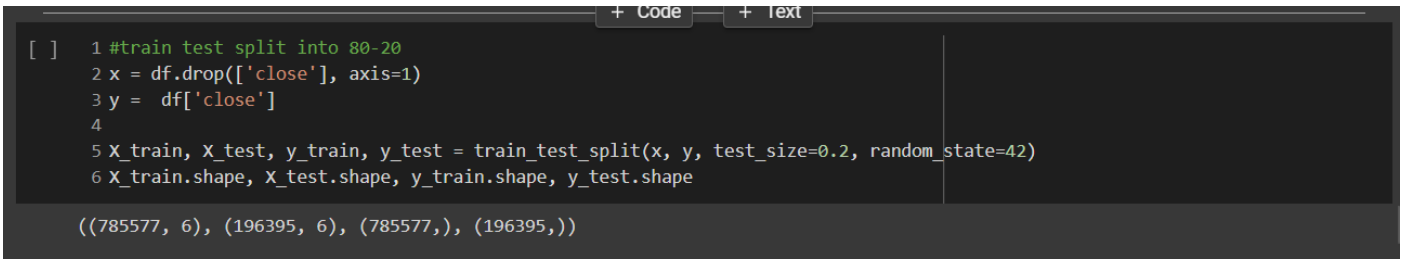
4. DATA PREPROCESSING

➢ Train/Test split:

- The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

- It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

- Our data set contain near about 9 lakh row. We have to perform train test split on it. We split data set into 80:20 ration. 80% data for training and 20% data for testing.



➢ Feature scaling

Standardization is required in some forms of machine learning when the input data points are scaled in different scales. Standardization can be a common scale for these data points.

The basic concept behind the standardization function is to make data points centered about the mean of all the data points presented in a feature with a unit standard deviation. This means the mean of the data point will be zero and the standard deviation will be 1.

This technique also tries to scale the data point between zero to one but in it, we don't use max or minimum. Here we are working with the mean and the standard deviation.

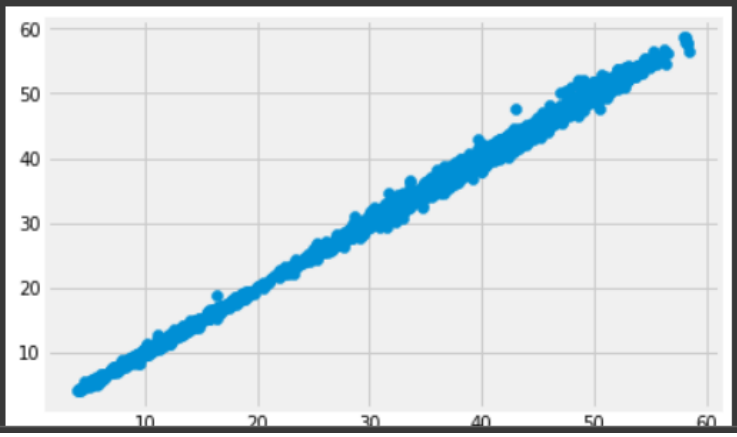
```
from sklearn.preprocessing import StandardScaler# it makes all mean=0 and s.d=1
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

This give as detail information about our dataset. As data are corelated which help to forecast stock price for managing portfolio.

So we use open, high, low, data for predicting stock closing price.

```
[23] corr, _ = pearsonr(df_demo['close'],df_demo['open'])
     print('Pearsons correlation: %.3f' % corr)
     plt.scatter(df_demo['close'],df_demo['open'])#close is the target column ie on y axis
     plt.show()
     #gives corelation between open and close values
```

```
Pearsons correlation: 0.999
```



5. MODEL BUILDING

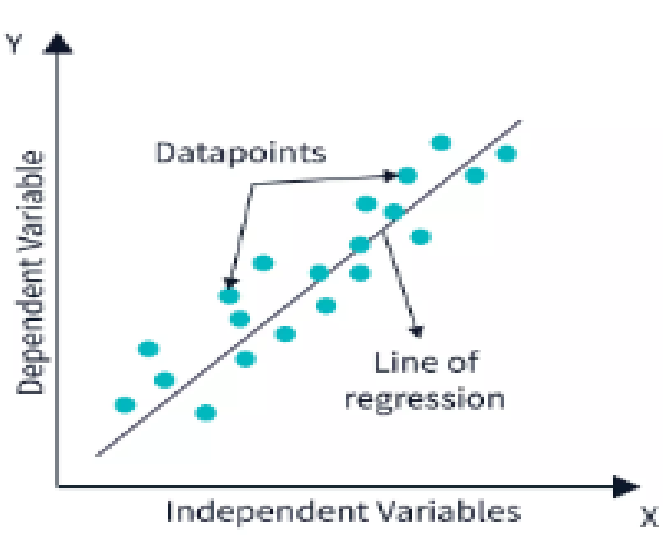
- Appling Different model to find best fit for our task.

5.1 Linear Regression: -

- Linear regression perform the task to predict a dependent variable value base on a given independent variable. So this regression technique find out a linear relationship between x and y.

- Imagine we are predicting weight (y) from height (x). Our linear regression model representation for this problem would be:

$$y = B0 + B1 * x1$$



Appling linear regression here

```
[ ]  1 from sklearn.linear_model import LinearRegression
     2 reg = LinearRegression()
     3 reg.fit(X_train, y_train)
```

```
LinearRegression()
```

```
[ ]  1 Y_pred_reg = reg.predict(X_test)
```

```
[ ]  1 r2, mse, mae = eval_fun(y_test,Y_pred_reg)
     2 print("r2 score = ", r2, "mse = ", mse, " mae =", mae)
```

```
r2 score =  0.9999506899054441 mse =  0.486834909527814  mae = 0.34044377192316344
```

5.2 Decision Tree Regression:-

Decision tree regression observes features of an object and train a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continues output that the output is not discrete.

the Decision Tree used for classification and a solution for regression problems. As it is a predictive model, Decision Tree Analysis is done via an algorithmic approach where a data set is split into subsets as per conditions. The name itself says it is a tree-like model in the form of if-then-else statements. The deeper is the tree and more are the nodes, the better is the model.

```
DTR
```

```
[ ]  1 from sklearn.tree import DecisionTreeRegressor
     2 regressor = DecisionTreeRegressor(random_state = 0)
     3 regressor.fit(X_train, y_train)
```

```
DecisionTreeRegressor(random_state=0)
```

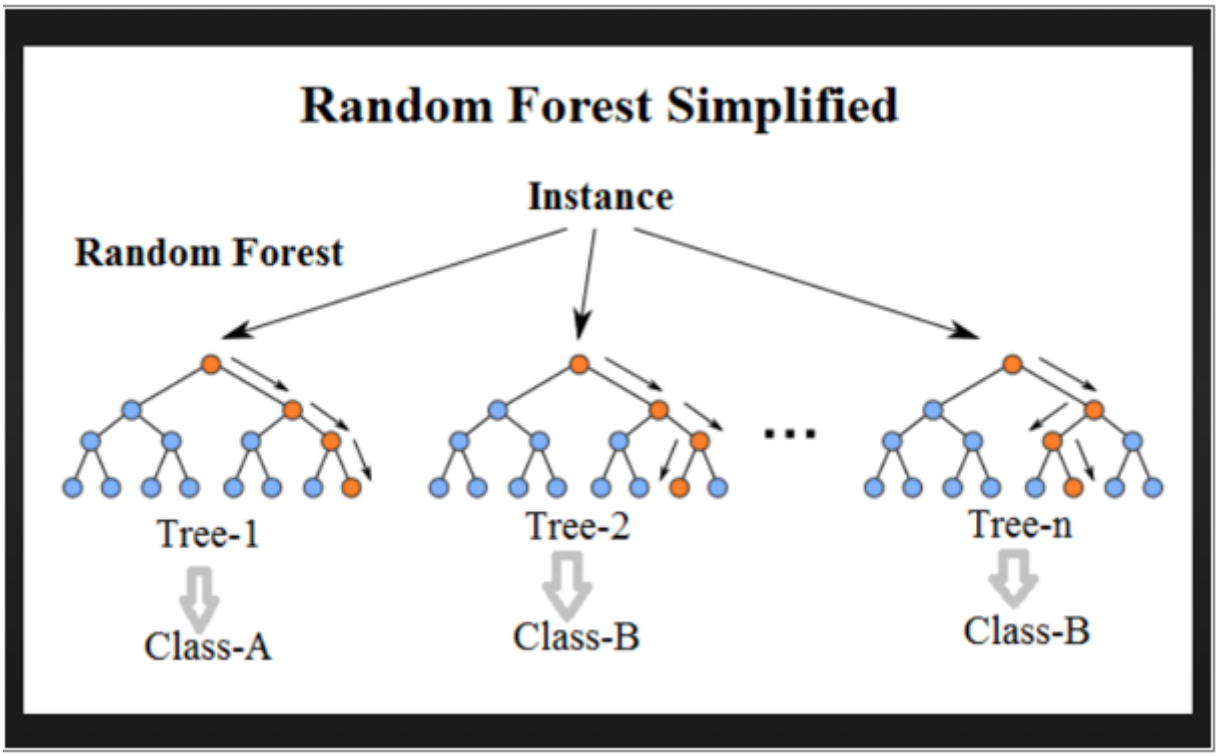
```
[ ]  1 Y_pred_dt = regressor.predict(X_test)
```

```
[ ]  1 r2, mse, mae = eval_fun(y_test,Y_pred_dt)
     2 print("r2 score = ", r2, "mse = ", mse, " mae =", mae)
```

```
r2 score =  0.9998980614950876 mse =  1.0064313050577915  mae = 0.4105517014385292
```

5.3 Random forest

- A random forest is a supervised machine learning algorithm that is constructed from decision tree algorithms. This algorithm is applied in various industries such as banking and e-commerce to predict behaviour and outcomes.



RandomForestRegression

```
[ ]  1 from sklearn.ensemble import RandomForestRegressor
     2 rf = RandomForestRegressor(n_estimators = 100, random_state = 0,oob_score=True)
     3 rf.fit(X_train,y_train)
     4 Y_pred = rf.predict(X_test)
```

Evaluating RF

```
[ ]  1 r2, mse, mae = eval_fun(y_test,Y_pred)
     2 print("r2 score = ", r2, "mse = ", mse, " mae =", mae)

    r2 score =  0.99993012705721 mse =  0.36712921919848046  mae = 0.29243539089930565
```

5.4 Xgboost:-

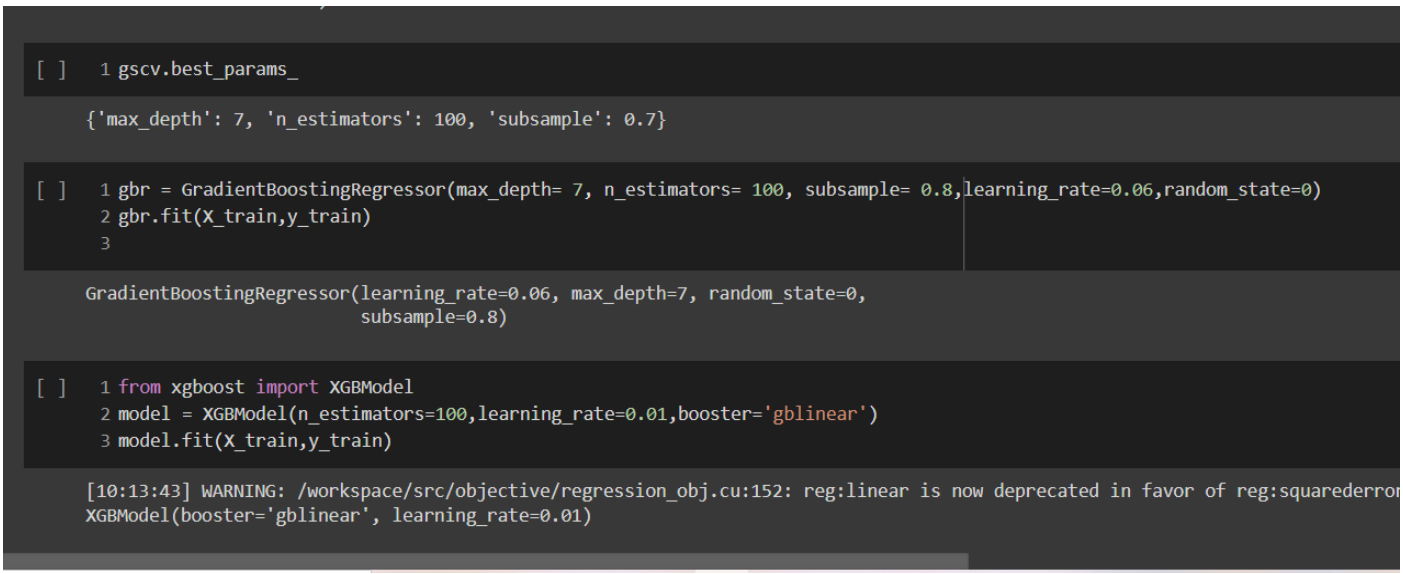
XgBoost stands for Extreme Gradient Boosting, which was proposed by the researchers at the University of Washington. It is a library written in c++ which optimizes the training for Gradient Boosting.

In this algorithm, decision trees are created in sequential form. Weight play an important role in XGBoost. Weight are assigned to all the independent variable which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/ predictions then ensemble to give a strong and more precise model. It can work on regression, classification, ranking and user-define prediction problems.

What Makes XGBoost So Famous?

- **Execution and Speed:** Originally built on C++, it is similarly fast to other gathering classifiers.
- **Centre calculation is parallelizable**: it can outfit the force of multi centre PCs because the centre XGBoost calculation is parallelizable. Moreover, it is parallelizable onto GPUs and across organizations of PCs, making it attainable to prepare on a huge dataset.
- **Reliably outflanks other technique calculations:** It has shown better output on many AI benchmark datasets.
- **Wide assortment of tuning boundaries**: XGBoost inside has boundaries for scikit-learn viable API, missing qualities, regularization, cross-approval, client characterized objective capacities, tree boundaries, etc.

XGBoost (Extreme Gradient Boosting) has a place with a group of helping calculations and utilizations of the slope supporting (GBM) structure at its centre.



5.5 KNN:-

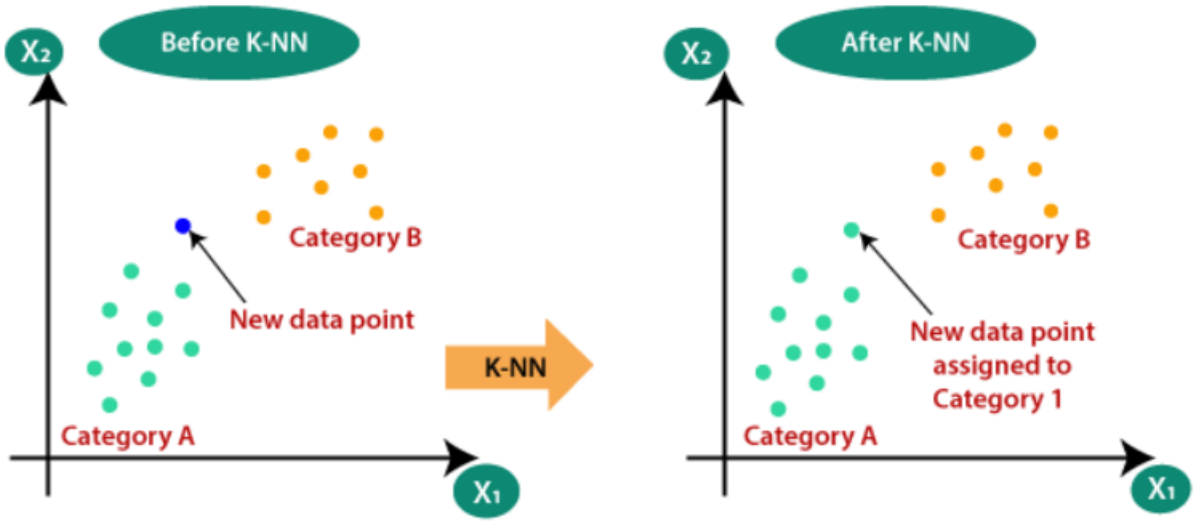
K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

- K-NN algorithm assumes the similarity between the new data and available data and put the new data into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
# instantiate learning model (k = 3)
knn = KNeighborsClassifier(n_neighbors=10)

# fitting the model
knn.fit(X_train_class, y_train_class)
knn.score(X_test_class,y_test_class)
# predict the response
pred = knn.predict(X_test_class)

# evaluate accuracy
print ("Accuracy of KNN ", accuracy_score(y_test_class, pred))
print ("score of KNN ",knn.score(X_test_class,y_test_class) )
```

```
Accuracy of KNN  0.7160493827160493
score of KNN  0.7160493827160493
```

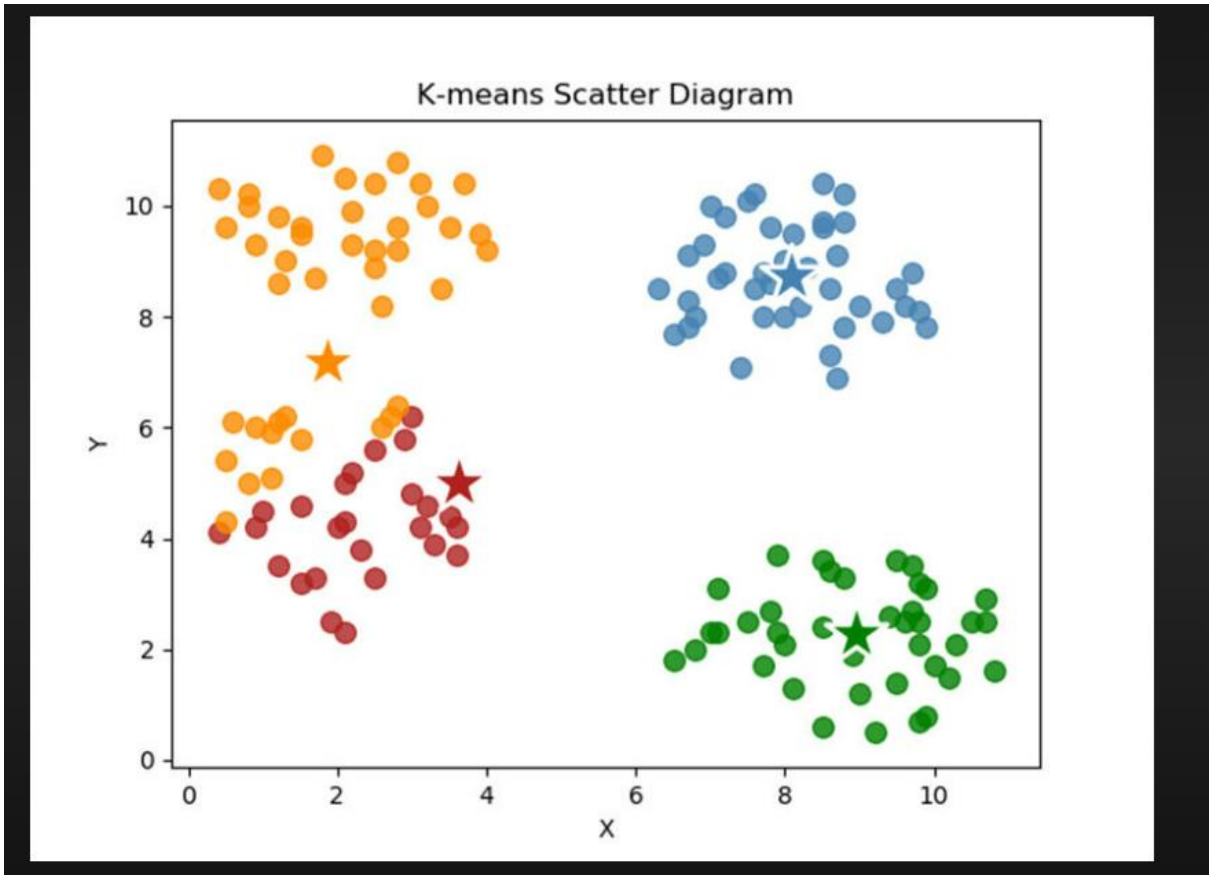
5.6 K MEANS:-

What is K-Means Algorithm?

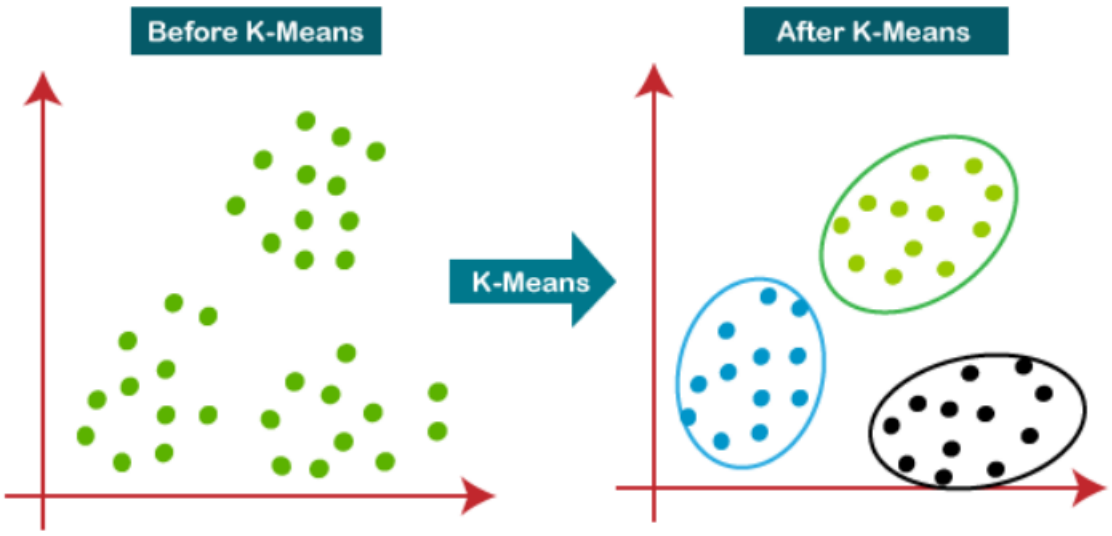
K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.



The below diagram explains the working of the K-means Clustering Algorithm:




```
from sklearn.cluster import KMeans
# since class is target col and ticker is object/chracter col
X_class = df.drop(['CLASS','TICKER','Unnamed: 0'],axis=1)   # end index is exclusive

k_mean = KMeans()

#number of clusters will be decided by K-mean++ , by default its 8
k_mean_model_1 = k_mean.fit(X_class)

print("Number of clusters",k_mean_model_1.n_clusters)
```

```
Number of clusters 8
```

7. REQUIREMENT SPECIFICATIONS

7.1 Hardware Requirement:

- **500 GB hard drive**
- **12 GB RAM**
- **PC x64-bit Intel CPU**

7.2 Software Requirements:-

- **Windows/Mac/Linux**
- **Python – 3.10.x**
- **Pycharm/Anaconda/Spyder**
- **Libraries:**

  - **Numpy**
  - **Pandas**
  - **Scipy**
  - **Scikit-learn**
  - **Matplotlib**

- **Any modern Web Browser like MS-EDGE or Google chrome**

CHALLENGES

- Downloading dataset from yfinance api
- Applying multiple algorithm and large data set time consuming
- Manage data according to model

FUTURE SCOPE

- Use pyspark due to big data set
- Use Deep learning technique for better result
- We can do sentiment analysis.

REFERENCES

https://reader.elsevier.com/reader/sd/pii/S240591882100015 5?token=D34F25FC648B8F1DF95D6F7F52DE193706148A5F646308E9E005BD394D917ADC70B51DCA2587E38CEE3524076EF22670&originRegion=eu-west-1&originCreation=20220925122706

DATA SET LINK:

https://pypi.org/project/yfinance/