STOCK INDEX FORECASTING USING ML

Mini Project Report

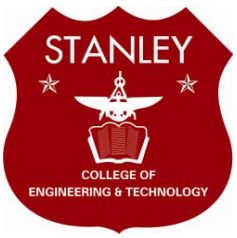
Submitted in partial fulfillment of the III year (Semester-I) BE degree as per the requirements of Osmania University, Hyderabad

By

Dhannarapu Reshmika 160619733011

Yelgoe Vyshnavi 160619733059

Sogala Sreeja 160619733044



Department of Computer Science & Engineering

Stanley College of Engineering and Technology for Women

(Affiliated to OU & Approved by AICTE)

Chapel Road, Abids, Hyderabad - 500001

2021-2022

Stanley College of Engineering & Technology for Women

Chapel Road, Hyderabad

(Affiliated to OU & Approved by AICTE)

Ref No: SCETW/CSE Dept/III Year 2022/MiniProject Date:

CERTIFICATE

This is to certify that the mini project titled "**STOCK INDEX FORECASTING USING ML**" is a bonafied work carried over by Ms. **DHANNARAPU RESHMIKA** (H.T. No. 160619733011), Ms. **YELGOE VYSHNAVI** (H.T. No. 160619733059) and Ms. **SOGALA SREEJA** (H.T. No. 160619733044) in partial fulfillment of the requirements for the award of the degree Bachelor of Engineering in Computer Science and Engineering from Osmania University during the III/IV Semester-I of their B.E. course during the academic year 2021-2022.

Dr.Y V S S Pragathi
Head, Department of CSE

Mrs.T.Monika Singh
Project Guide

TABLE OF CONTENTS

ABSTRACT

In the era of investing, wherein more than 4000 trade companies are enlisted in stock exchange market, making a huge sum of investments. The stock market is known for being volatile, dynamic and nonlinear. which raises questions on the ease of the investment process. The most crucial problem of the investor is to find the right approach in order to attain maximum profit. One right decision in stock market can create a huge difference on an investor's life. Thus, analyzing the impacting factors of stock market before making a decision is challenging. This made us develop a web application to create a good investment decision support system irrespective of the user's existing or lack of prior knowledge of statistical distributions which represent trends of stock prices. From people who just started investing to people who invest daily can use this web application as it provides the assistance of prior knowledge and being able to predict the stock flow for the next required number of days to invest right.

This Stock Index Forecaster uses machine learning to predict the future direction of stock flow using the libraries available such as yfinance, pandas, dash etc. which not only records metadata such as date and time but also takes in user input of stock code and number of days to forecast thereby returning significant results.

LIST OF FIGURES

1. INTRODUCTION

The financial market is a dynamic and composite system where people can buy and sell currencies, stocks, equities and derivatives over virtual platforms supported by brokers. The stock market allows investors to own shares of public companies through trading either by exchange or over the counter markets. This market has given investors the chance of gaining money and having a prosperous life through investing small initial amounts of money, low risk compared to the risk of opening new business or the need of high salary career. Stock markets are affected by many factors causing the uncertainty and high volatility in the market.

The primeval way of predicting stock was to use manual labor for data retrieving, analysis and calculations which not only were economically impractical but also lead to possible human errors and low accuracy. Many researchers conducted by scholars all around the world suggested an Automated Trading System (ATS). Many factors are incorporated and considered when developing an ATS. Consequently, this makes amateurs unable to use these platforms.

The thought behind this project was to provide equal usage to both beginners and experts to predict the prices for a better strategy making and investments. Machine learning eradicates the place for any human errors and ensures high levels of accuracy while prediction making it the better way to help out the users and put them in control of the data provided.

However, prediction was not the only focus, we also concentrated on basic details such as description of the company, indicators (exponential moving avg price plot to determine the general direction of prices) and previous price plots. This helps the users to have an outlook of the investment company they chose.

1.1 PURPOSE

Performing a research before making an investment is a must. It is only after a thorough research that you can make some assumptions into the value and future performance of an investment. Even if you are following stock trading tips, it ideal to do some research, just to ensure that you are making an investment that's expected to get you maximum returns.

When you invest in equity, you purchase some portions of a business expecting to make money upon increase in the value of the business. Before buying anything, be it a car or phone, you do some degree of research about its performance and quality. An investment is no different. It is your hard earned money that you are about to invest, so you must have a fair knowledge of what you are investing in.

1.2 SCOPE

An accurate forecast takes into account changes in the overall market as well as the changing needs and priorities of customers. The other objective of the project is to develop a statistical Model based on the dataset available. We are using the historical market data for the previous years dated back to 1995 to predict the stock prices and visualized it for easy understand.

Analysis of stocks using machine learning will be useful for new investors to invest in stock market based on the various factors considered by the software. Stock market includes daily activities like Sensex calculation, exchange of shares. The exchange provides an efficient and transparent market for trading in equity, debt instruments and derivatives.

The successful prediction of a stock's future price could yield significant profit. The efficient-market hypothesis suggests that stock prices reflect all currently available

information and any price changes that are not based on newly revealed information thus are inherently unpredictable.

1.3 STUDY OF EXISTING SYSTEM

Stock market prediction is an act of trying to determine the future value of a stock other financial instrument traded on a financial exchange. The technical and fundamental or the time series analysis is used by the most of the stockbrokers while making the stock predictions. The programming language is used to predict the stock market using machine learning is Python. In the existing system, it proposes a Machine Learning (ML) approach that will be trained from the available stocks data and gain intelligence and then uses the acquired knowledge for an accurate prediction. In this context the study uses a machine learning technique called K-Nearest Neighbor (KNN) to predict stock prices for the large and small capitalizations and in the three different markets, employing prices with both daily and up-to-the-minute frequencies.

1.4 PROPOSED SYSTEM

The proposed system helps in getting more accurate forecasting. This accuracy helps in increasing the chance of getting high returns to the investors who can rely on this model as an investment decision support system. The system is trained using data from yahoo finance and by using the machine learning algorithm Support Vector Regression (SVR) and when deployed it shows company description, price plot of the designated dates by the user and it also displays a separate plotting of the prices effected by the indicators and lastly the predicted price plot for the next user-prompted number of days.

2. REQUIREMENTS

2.1 SOFTWARE REQUIREMENTS

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development.

Pip packages

Pip stands for preferred installer program which is a recursive program and a package-management system written in Python used to install and manage software packages. In our project we included several pip packages such as NumPy, dash, yfinance, pandas, sklearn, matplotlib etc.

NumPy

NumPy (Numerical Python) is a linear algebra library in Python. It is a very important library on which almost every data science or machine learning Python packages such as SciPy (Scientific Python), Mat−plotlib (plotting library), Scikit-learn, etc. depends on it on a reasonable extent. It is also very useful for performing mathematical and logical operations on array

Dash

Dash is a python framework created by plotly for creating interactive web applications. It is open source and the application build using this framework are viewed on the web browser. It is a substitute of HTML, CSS, JavaScript in order to create interactive dashboards.

yFinance

Yfinance is a python package that enables us to fetch historical market data from Yahoo Finance API in a Pythonic way.It becomes so easy for all the Python developers to get data with the help of yfinance

MatPlotLib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. Matplotlib is one of the most powerful libraries in python for data visualization.

Pandas

Pandas is a Python library for data analysis. Pandas is built on top of two core Python libraries—matplotlib for data visualization and NumPy for mathematical operations. Pandas acts as a wrapper over these libraries, allowing you to access many of matplotlib's and NumPy's methods with less code.

Microsoft visual studio code

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity).

2.2 HARDWARE REQUIREMENTS

Processor

Intel CORE i5 (or above) processor with minimum speed 2.9 GHz speed.

RAM

Minimum 4 GB RAM

3. SYSTEM ARCHITECTURE

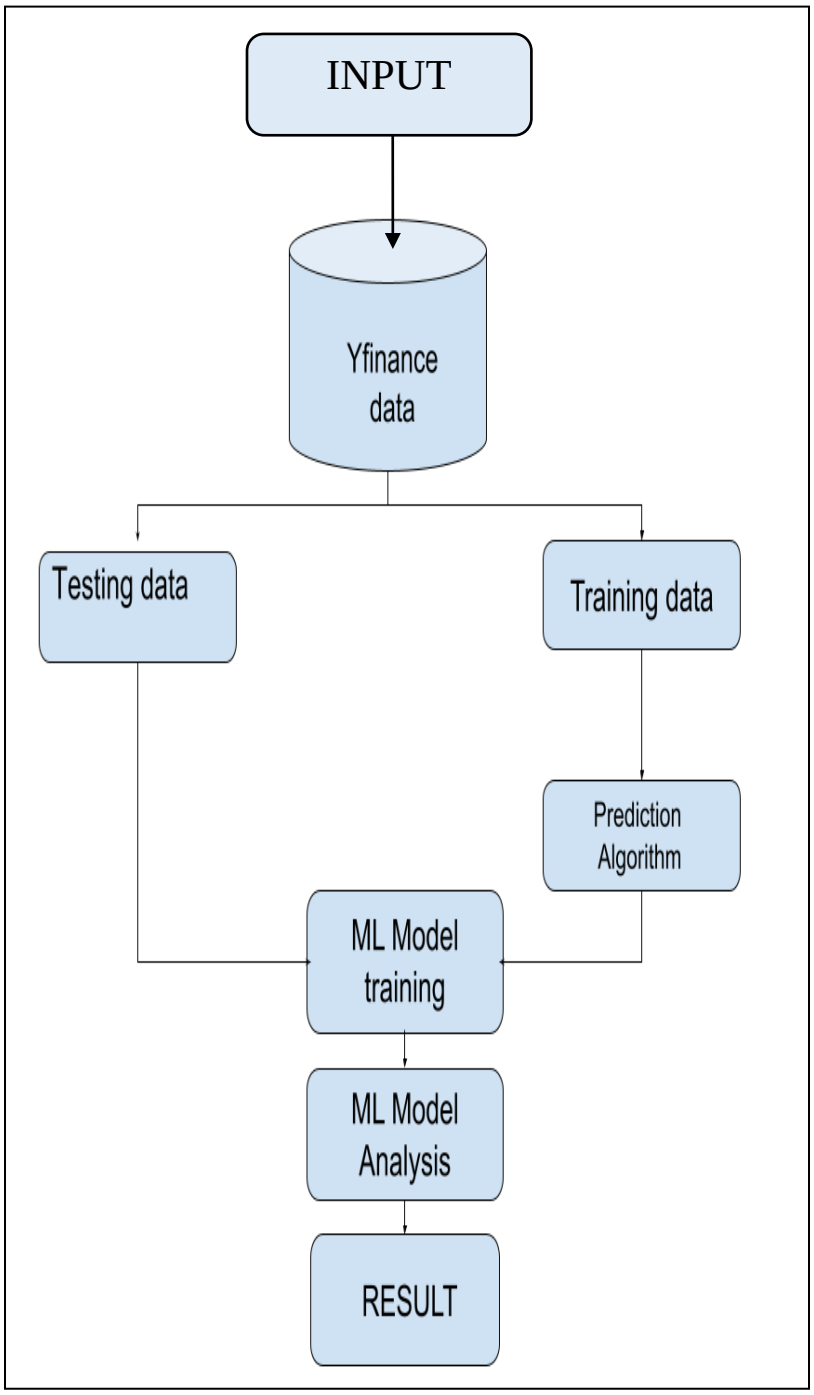


Fig 3.1: ML Architecture

3.1 INPUT DATASET

A dataset is a collection of data or objects. In other words, a data set contains the contents of a single database table, in which every column represents a specific variable and every row responds to a given member of the data set. It could be of various formats for different purposes, for instance if we wish to form a ML model for business purposes, then the dataset will be different from the dataset required for email spam detection. So, every dataset is different from another dataset. Majorly the data is categorized into four basic types from the machine learning aspect: time-series data, text, numerical data, categorical data.

As we all know machine learning using training datasets. It is the specific data used to train the model for performing various operations. Here during this project, the yfinance dataset is used. Yfinance is an open-source library available in python which consists of all the financial data collected by the company and website and is available for everyone on the internet to use and view.

3.2 TRAINING AND TESTING:

In ML projects, Real world values can’t be used to test the model because that is platform dependent so we split the input data set into testing and training data sets.

Training data-the set of data required to train the ML algorithm to learn patterns and characteristics of each entity to be able to predict, this step is the foundation of the model.

Testing is an important aspect of our project, it says how well the ML model is able to predict the output, it measures the efficiency and accuracy.

To train the model, we used 10% of the data available and to test the model we used the remaining of the 90% data of the dataset.

SVR:

In machine learning, Support Vector Machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. The SVM regression algorithm is referred to as **Support Vector Regression** or **SVR**.

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 Support Vector Regression, the straight line that is required to fit the data is referred to as **hyperplane**.

The objective of a support vector machine algorithm is to find a hyperplane in an n-dimensional space that distinctly classifies the data points. The data points on either side of the hyperplane that are closest to the hyperplane are called **Support Vectors**.

Unlike other Regression models that try to minimize the error between the real and predicted value, the SVR tries to fit the best line within a threshold value. The threshold value is the distance between the hyperplane and boundary line.

HYPERPARAMETERS USED IN SVR

The various hyperparameters used in the SVR are:

1. Hyperplane

Hyperplanes are decision boundaries that is used to predict the continuous output. The data points on either side of the hyperplane that are closest to the hyperplane are called

Support Vectors. These are used to plot the required line that shows the predicted output of the algorithm.

2. Kernel

A kernel is a set of mathematical functions that takes data as input and transform it into the required form. These are generally used for finding a hyperplane in the higher dimensional space.

The most widely used kernels include **Linear**, **Non-Linear**, **Polynomial**, **Radial Basis Function (RBF)** and **Sigmoid**. By default, RBF is used as the kernel. Each of these kernels are used depending on the dataset.

3. Boundary Lines:

These are the two lines that are drawn around the hyperplane at a distance of **ε (epsilon)**. It is used to create a margin between the data points.

In SVR, the best fit line is the hyperplane that has the maximum number of points

The SVR algorithm is implemented using the following steps:

- Loading the data
- Exploring the data
- Splitting the data
- Generating the model
- Model evaluation

4. SOFTWARE DESIGN

4.1 PROJECT FLOW

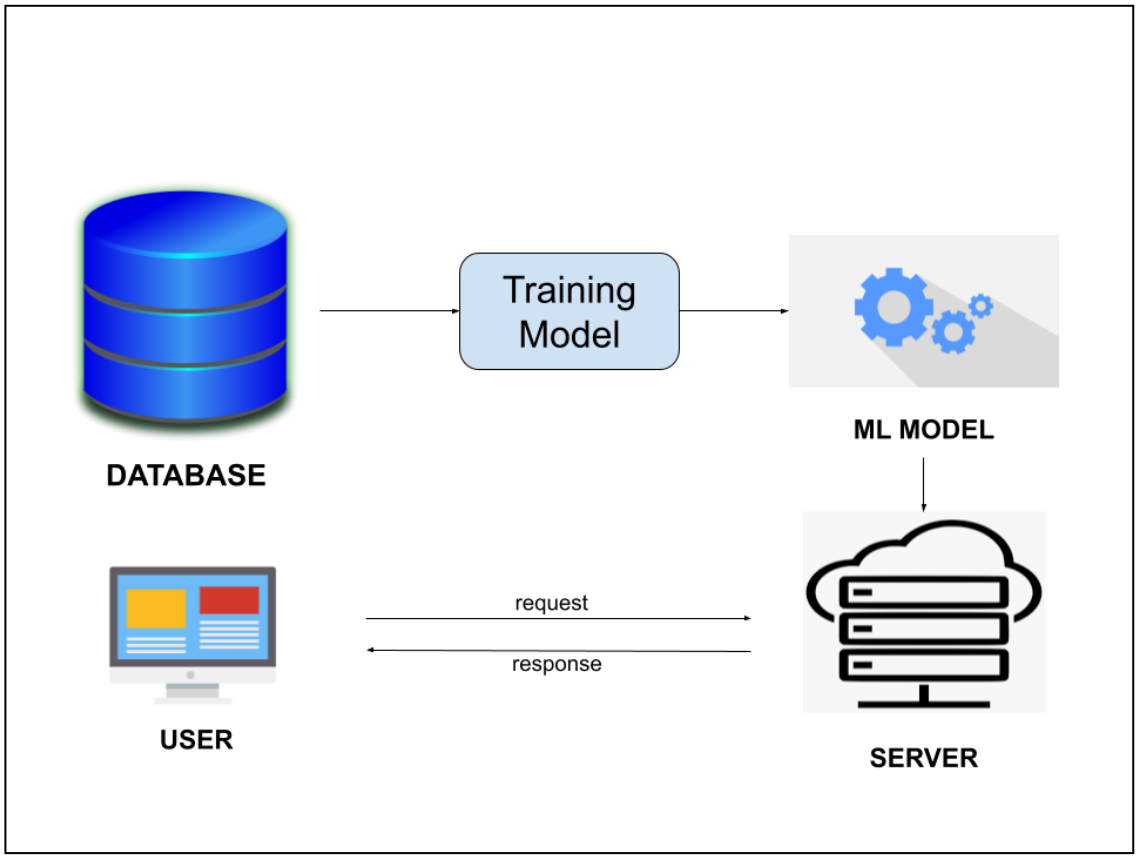


Fig 4.1: Project flow

- Create basic website layout
- Creating a Machine Learning Model
- Generating a company's information and graphs
- Binding elements in a Single Server
- Styling the application's web page

4.2 UML DIAGRAM

A UML diagram is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains graphical elements (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts.

The kind of the diagram is defined by the primary graphical symbols shown on the diagram. For example, a diagram where the primary symbols in the contents area are classes is class diagram. A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines.

UML specification does not preclude mixing of different kinds of diagrams, e.g. to combine structural and behavioral elements to show a state machine nested inside a use case. Consequently, the boundaries between the various kinds of diagrams are not strictly enforced. At the same time, some UML Tools do restrict set of available graphical elements which could be used when working on specific type of diagram.

UML specification defines two major kinds of UML diagram: structure diagrams and behavior diagrams.

Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other.
The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

4.2.1 USE CASE DIAGRAM

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

• Scenarios in which your system or application interacts with people, organizations, or external systems.

• Goals that your system or application helps those entities (known as actors) achieve.

• The scope of your system.

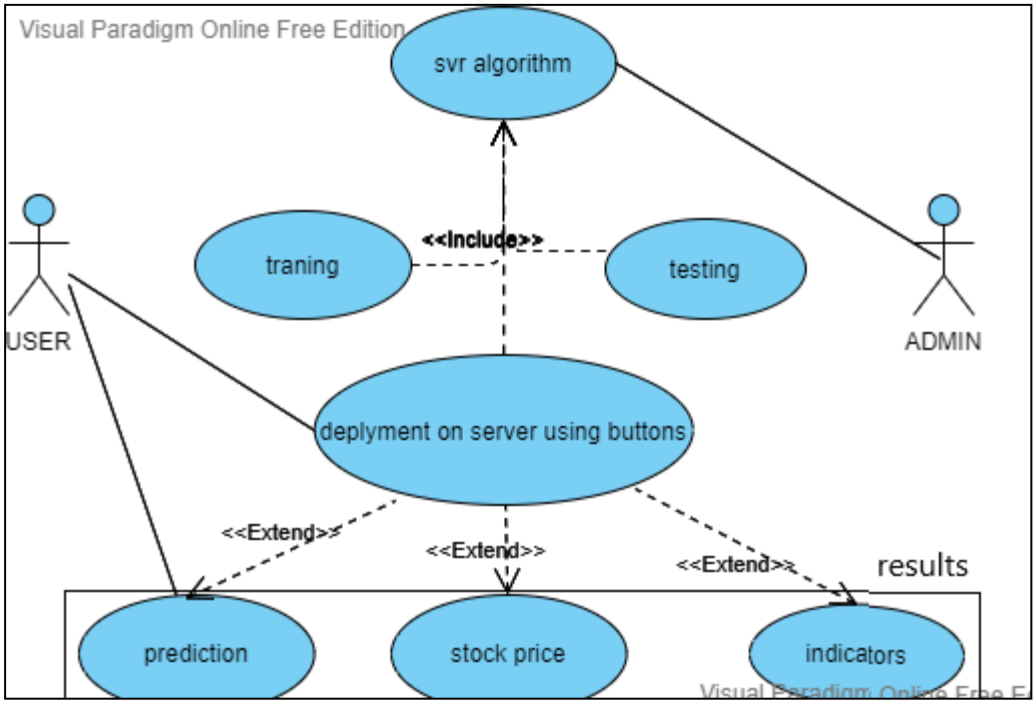


Fig 4.2.1: Use-Case Diagram

4.3 E-R DIAGRAM

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.

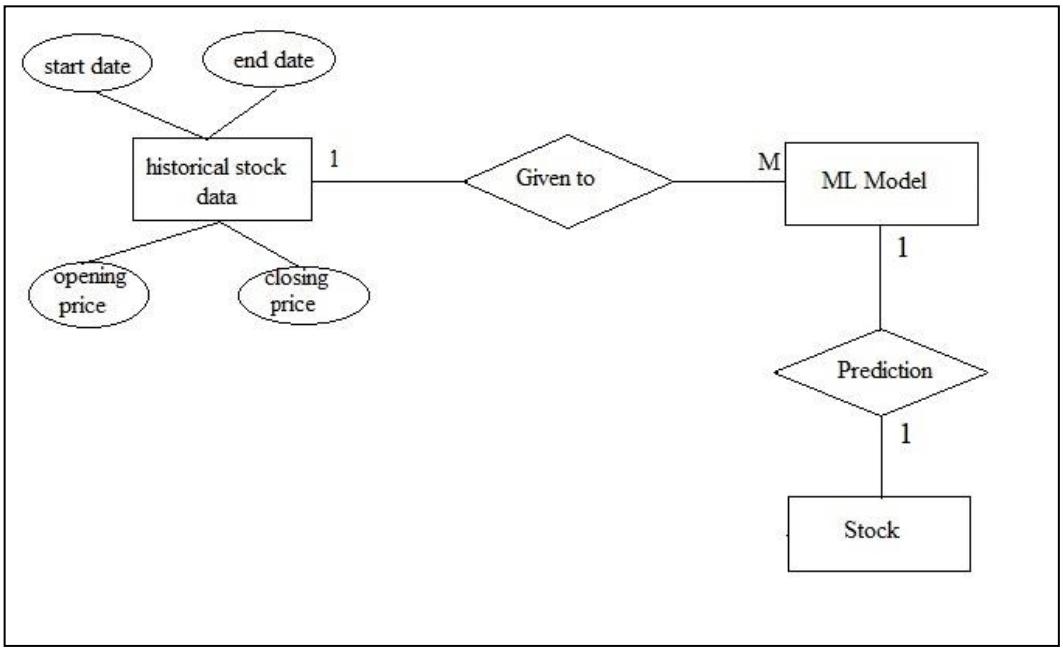


Fig 4.3: E-R Diagram

5. CODE TEMPLATE

app.py

C: > Users > dhann > OneDrive > Desktop > stocks > app.py > ...

```python
import dash
from dash import dcc
from dash import html
from datetime import datetime as dt
import yfinance as yf
from dash.dependencies import Input, Output, State
from dash.exceptions import PreventUpdate
import pandas as pd
import plotly.graph_objs as go
import plotly.express as px
# model
from model import prediction
from sklearn.svm import SVR


def get_stock_price_fig(df):

    fig = px.line(df,
                  x="Date",
                  y=["Close", "Open"],
                  title="Closing and Openning Price vs Date")

    return fig


def get_more(df):
    df['EWA_20'] = df['Close'].ewm(span=20, adjust=False).mean()
    fig = px.scatter(df,
                     x="Date",
                     y="EWA_20",
                     title="Exponential Moving Average vs Date")
    fig.update_traces(mode='lines+markers')
    return fig


app = dash.Dash(
    __name__,
```

Fig 5.1: App code 1

app.py

C: > Users > dhann > OneDrive > Desktop > stocks > app.py > ...

```python
    external_stylesheets=[
        "https://fonts.googleapis.com/css2?family=Roboto&display=swap"
    ])
server = app.server
# html layout of site
app.layout = html.Div(
    [

        # content
        html.Div(
            [
                html.Div(
                    [  # header
                        html.Img(id="logo"),
                        html.P(id="ticker")
                    ],
                    className="header"),
                html.Div(id="description", className="decription_ticker"),
                html.Div([], id="graphs-content"),
                html.Div([], id="main-content"),
                html.Div([], id="forecast-content")
            ],
            className="content"),

         # nav
        html.Div(
            [
                # Navigation
                html.P("Welcome to the Stock Dash App!", className="start"),
                html.Div([
                    html.P("Input stock code: "),
                    html.Div([
                        dcc.Input(id="dropdown_tickers", type="text"),
                        html.Button("Submit", id='submit'),
                    ],
                             className="form")
                ],
```

Fig 5.2: App code 2

app.py

C: > Users > dhann > OneDrive > Desktop > stocks > app.py > ...

```
                    className="input-place"),
                html.Div([
                    dcc.DatePickerRange(id='my-date-picker-range',
                                        min_date_allowed=dt(1995, 8, 5),
                                        max_date_allowed=dt.now(),
                                        initial_visible_month=dt.now(),
                                        end_date=dt.now().date()),
                ],
                    className="date"),
                html.Div([
                    html.Button(
                        "Stock Price", className="stock-btn", id="stock"),
                    html.Button("Indicators",
                                className="indicators-btn",
                                id="indicators"),
                    dcc.Input(id="n_days",
                              type="text",
                              placeholder="number of days"),
                    html.Button(
                        "Forecast", className="forecast-btn", id="forecast"
                ],className="buttons"),
                # here
            ],
            className="nav"),
    ],
    className="container")


# callback for company info
@app.callback([
    Output("description", "children"),
    Output("logo", "src"),
    Output("ticker", "children"),
    Output("stock", "n_clicks"),
    Output("indicators", "n_clicks"),
    Output("forecast", "n_clicks")
], [Input("submit", "n_clicks")], [State("dropdown_tickers", "value")])
def update data(n, val):  # input parameter(s)
```

Fig 5.3: App Code 3

app.py

C: > Users > dhann > OneDrive > Desktop > stocks > app.py > stock_price

```
    if n == None:
        return " Please enter a legitimate stock code to get details.",
        "https://cdn-icons-png.flaticon.com/512/1042/1042526.png", "Stock
        Forecaster", None, None, None
        # raise PreventUpdate
    else:
        if val == None:
            raise PreventUpdate
        else:
            ticker = yf.Ticker(val)
            inf = ticker.info
            df = pd.DataFrame().from_dict(inf, orient="index").T
            df[['logo_url', 'shortName', 'longBusinessSummary']]
            return df['longBusinessSummary'].values[0], df['logo_url'].
            values[
                0], df['shortName'].values[0], None, None, None
# callback for stocks graphs
@app.callback([
    Output("graphs-content", "children"),
], [
    Input("stock", "n_clicks"),
    Input('my-date-picker-range', 'start_date'),
    Input('my-date-picker-range', 'end_date')
], [State("dropdown_tickers", "value")])
def stock_price(n, start_date, end_date, val):
    if n == None:
        return [""]
        #raise PreventUpdate
    if val == None:
        raise PreventUpdate
    else:
        if start_date != None:
            df = yf.download(val, str(start_date), str(end_date))
        else:
            df = yf.download(val)
```

Fig 5.4: App Code 4

app.py

C: > Users > dhann > OneDrive > Desktop > stocks > app.py > stock_price

```python
# callback for indicators
@app.callback([Output("main-content", "children")], [
    Input("indicators", "n_clicks"),
    Input('my-date-picker-range', 'start_date'),
    Input('my-date-picker-range', 'end_date')
], [State("dropdown_tickers", "value")])
def indicators(n, start_date, end_date, val):
    if n == None:
        return [""]
    if val == None:
        return [""]

    if start_date == None:
        df_more = yf.download(val)
    else:
        df_more = yf.download(val, str(start_date), str(end_date))

    df_more.reset_index(inplace=True)
    fig = get_more(df_more)
    return [dcc.Graph(figure=fig)]
# callback for forecast
@app.callback([Output("forecast-content", "children")],
              [Input("forecast", "n_clicks")],
              [State("n_days", "value"),
               State("dropdown_tickers", "value")])
def forecast(n, n_days, val):
    if n == None:
        return [""]
    if val == None:
        raise PreventUpdate
    fig = prediction(val, int(n_days) + 1)
    return [dcc.Graph(figure=fig)]


if __name__ == '__main__':
    app.run_server(debug=True)
```

Fig 5.5: App Code 5

model.py > prediction

```python
def prediction(stock, n_days):
    import dash
    import dash_core_components as dcc
    import dash_html_components as html
    from datetime import datetime as dt
    import yfinance as yf
    from dash.dependencies import Input, Output, State
    from dash.exceptions import PreventUpdate
    import pandas as pd
    import plotly.graph_objs as go
    import plotly.express as px
    # model
    from model import prediction
    from sklearn.model_selection import train_test_split
    from sklearn.model_selection import GridSearchCV
    import numpy as np
    from sklearn.svm import SVR
    from datetime import date, timedelta

    # load the data
    df = yf.download(stock, period='60d')
    df.reset_index(inplace=True)
    df['Day'] = df.index

    days = list()
    for i in range(len(df.Day)):
        days.append([i])

    # Splitting the dataset
    X = days
    Y = df[['Close']]

    x_train, x_test, y_train, y_test = train_test_split(X,
                                                        Y,
                                                        test_size=0.1,
                                                        shuffle=False)

    gsc = GridSearchCV(
```

Fig 5.6: Model Code 1

```
    estimator=SVR(kernel='rbf'),
    param_grid={
        'C': [0.001, 0.01, 0.1, 1, 100, 1000],
        'epsilon': [
            0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10,
            50, 100, 150, 1000
        ],
        'gamma': [0.0001, 0.001, 0.005, 0.1, 1, 3, 5, 8, 40, 100, 1000]
    },
    cv=5,
    scoring='neg_mean_absolute_error',
    verbose=0,
    n_jobs=-1)

y_train = y_train.values.ravel()
y_train
grid_result = gsc.fit(x_train, y_train)
best_params = grid_result.best_params_
best_svr = SVR(kernel='rbf',
               C=best_params["C"],
               epsilon=best_params["epsilon"],
               gamma=best_params["gamma"],
               max_iter=-1)

# Support Vector Regression Models

# RBF model
#rbf_svr = SVR(kernel='rbf', C=1000.0, gamma=4.0)
rbf_svr = best_svr

rbf_svr.fit(x_train, y_train)

output_days = list()
for i in range(1, n_days):
    output_days.append([i + x_test[-1][0]])

dates = []
```

Fig 5.7 Model Code 2

```

dates = []
current = date.today()
for i in range(n_days):
    current += timedelta(days=1)
    dates.append(current)

fig = go.Figure()
fig.add_trace(
    go.Scatter(
        x=dates,  # np.array(ten_days).flatten(),
        y=rbf_svr.predict(output_days),
        mode='lines+markers',
        name='data'))
fig.update_layout(
    title="Predicted Close Price of next " + str(n_days - 1) + " days",
    xaxis_title="Date",
    yaxis_title="Closed Price",
    # legend_title="Legend Title",
)

return fig
```

Fig 5.8 Model Code 3

```
# styles.css
assets > # styles.css > .nav
  margin: 0;
  padding: 0;
  box-sizing: 0;
  font-family: "Helvetica", Helvetica;;
}

body {
  overflow-x: hidden;
  background:url("https://cdn.pixabay.com/photo/2020/08/09/14/25/
  business-5475664_960_720.jpg");
  background-size: cover;
  position: absolute;
}

.container {
  display: flex;
}

.nav {
  /*position: fixed;*/
  width: 25vw;
  /* height: 100%; */
  align-items: center;
  display: flex;
  justify-content: flex-start;
  flex-direction: column;
  /* background-color: rgb(5, 107, 107); */
}

.content {
  width: 65vw;
  padding: 1rem 5rem;
}

.start {
  margin-top: 2rem;
  margin-bottom: 5rem;
  text-align: center;
```

Fig 5.9: CSS Code 1

```
# styles.css
assets > # styles.css > .nav
  font-size: larger;
  color: rgb(166, 243, 248);
}

.input-place p {
  color: antiquewhite;
  font-size: larger;
}
.form {
  display: flex;
  margin-top: 1rem;
  margin-bottom: 2rem;
  height: 1.5rem;
}
.form button {
  width: 5rem;
  background-color: yellow;
  color: black;
  border: none;
}
.buttons {
  margin: 2rem;
  width: 100%;
  display: flex;
  flex-wrap: wrap;
  position: relative;
  justify-content: space-around;
}
#forecast {
  margin-top: 2rem;
  height: 2.2rem;
}
#n_days {
  height: 2rem;
  margin-top: 2rem;
}
.buttons button {
  width: 7.5rem;
```

Fig 5.10: CSS Code 2

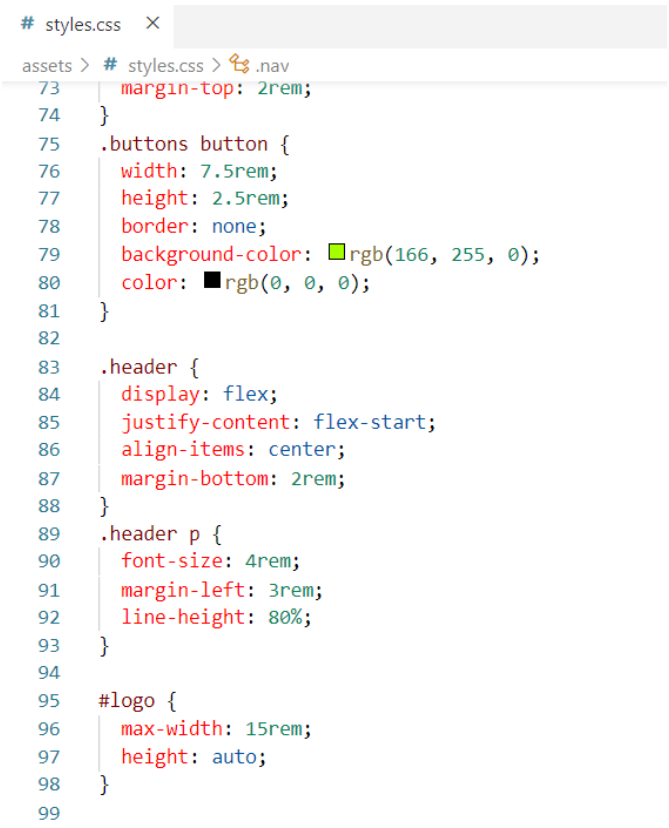


Fig 5.11: CSS Code 3

6. TESTING

Testing of software can be done in both manual and automatic testing method but it's totally depending on the project requirement and budget associated with the project, and which Testing method will be beneficial to the project. It provides basic information about manual and automation testing.

Manual testing

Manual testing is a method used by software developers to run tests manually. There are many manual testing types which are carried out manually as well as automatically. These are,

Black box testing

It's is a testing method to test functionalities and require the system. It does not test the internal part of the system.

White box testing

It is a testing method based on the information of the internal logic of a application's code and also known as Glass box testing. It works internal working code of the system. Tests are based on coverage of code statements, branches, paths, conditions.

Integration testing

It is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated. It is applicable especially to client/server and distributed systems.

System testing

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.

Unit testing

Unit testing involves the testing of each unit or an individual component of the software application. A unit is a single testable part of a software system and tested during the development phase of the application software. The purpose of unit testing is to test the correctness of isolated code. A unit component is an individual function or code of the application. It is specially done by programmers and but not by testers.

Acceptance testing

This type of testing verifies that the system meets the customer specified requirements or not. User or a customer does this testing to decide whether to accept application

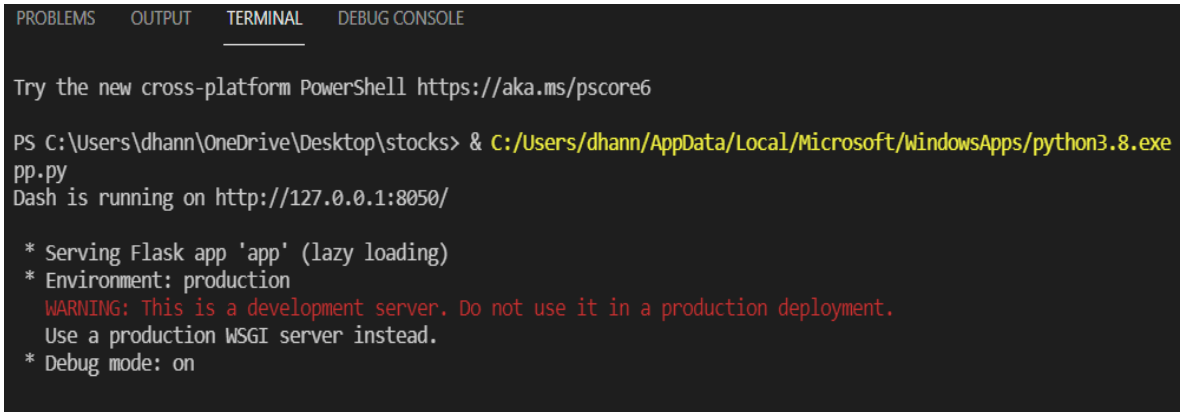


Fig 6: Testing in local host

7. OUTPUT SCREEN

Fig 7.1 Output 1

Fig 7.2 Output 2

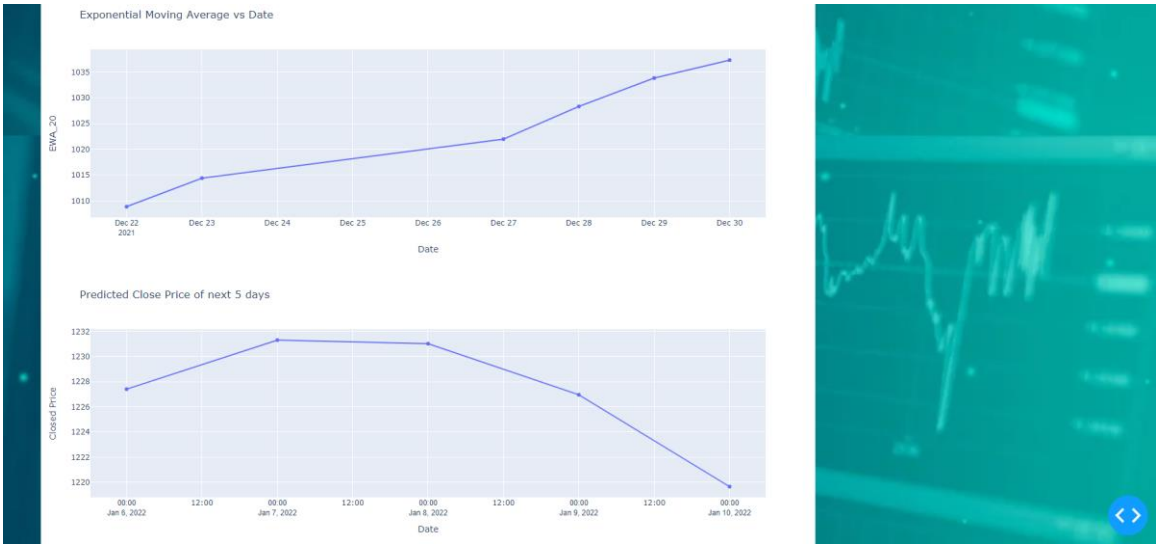


Fig 7.3 Output 3

8. REFERENCES

1. https://dash.plotly.com/layout
2. https://dash.plotly.com/dash-html-components
3. https://dash.plotly.com/dash-core-components
4. https://www.w3schools.com/css/default.asp
5. https://css-tricks.com/snippets/css/a-guide-to-flexbox/
6. https://plotly.com/python/plotly-express/
7. https://dash.plotly.com/basic-callbacks