MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

STOCK ANALYSIS-ANALYSIS OF STOCK MARKET

A project report submitted to MALLA REDDY UNIVERSITY in partial fulfilment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

Submitted by

K. NEHADEEP:2111CS020307

V.NEHA:2111CS020308

B. NIHARIKA:2111CS020309

S. NIDHIGNA:2111CS020310

SK. NIKHAT FATHIMA:2111CS020311

CH. NIKHIL:2111CS020312

Under the guidance of

Prof: Kalyani

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

This is to certify that this is the bonafied record of the application development entited, "**STOCK ANALYSIS-ANALYSIS OF STOCK MARKET"**submitted byK.Nehadeep(2111CS020307),V.Neha(2111CS020308),B.Niharika(2111CS020309),S.Nidhigna(2111CS020310), SK.Nikhat Fathima(2111CS020311), CH.Nikhil(2111CS020312). B. Tech II year II semester, Department of CSE (AI&ML) during the year 2022-23. The results embodied in the report have not been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE	**HEAD OF TH DEPARTMENT**
Prof Kalyani	**Dr. Thayyaba Khatoon**
	CSE(AI&ML)

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who have contributed to the completion of our project, "STOCK ANALYSIS-ANALYSIS OF STOCK MARKET"

First and foremost, we would like to thank our project mentor, Prof. Kalyani, for her invaluable guidance and support throughout the duration of the project. We would also like to thank the head of our department, Dr. Thayyaba Khatoon, for providing us with the resources and support necessary to conduct this research. We are grateful for her leadership and her commitment to promoting excellence in research and education.

We would like to extend our appreciation to the rest of the faculty and staff at our institution, who have provided us with assistance and support during the course of this project. We would also like to thank our friends and colleagues who have offered us encouragement and support during the course of this project. We hope that this project will make a meaningful contribution to the field of Computer Vision and will serve as a foundation for future research in this area.

K. NEHADEEP

V.NEHA

B. NIHARIKA

S. NIDHIGNA

SK. NIKHAT FATHIMA

CH. NIKHIL

CONTENTS

CHAPTER-1

1.INTRODUCTION

This project aims to build a application that provides traders to trade well. This application will use the machine learning algorithm called Support vector machine (svm) where users can easily trade and perform trading activities. One of the key features of this Application is trades get notifications updates. In this notifications play a much more crucial role since they help users keep track of their orders. Overall this project will provide a overall valuable tool for anyone who needs to trade. Mostly for beginners to trade safely and for experienced people too it makes more comfortable trading.

1.1 ABSTRACT

Our project is the recent trend in stock market predictions technologies is the use of machine learning which Makes predictions based on the values of current stock Market indices by training on their previous values . Machine learning itself employs different models to Make prediction easier and authentic . 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. Each machine learning algorithm is tested against the National Association of Securities Dealers Automated Quotations System (NASDAQ), New York Stock Exchange (NYSE) , Nikkei, and Financial Times Stock Exchange (FTSE). Furthermore, several machine learning algorithms are compared with a normal and a leaked data set. In this project we are using Machine learning algorithm called Support Vector Machine(SVM) algorithm, which give a prediction of various aspects of a particular stock or an index, such as future values of the opening price, closing price, index value etc. This will help investors and traders make better and faster decisions.

1.2 LIMITATIONS OF “STOCK ANALYSIS-ANALYSIS OF STOCK MARKET”

While the project of building a application for trading stock market features with smart notifications is useful and beneficial , potential limitations to consider. These may include:

1. Small data sets: SVMs work well with large data sets, but may not be effective with small data sets.

2. Complexity: SVMs can be computationally intensive, requiring significant processing power and time.

3. Difficulty with multi-class classification: While SVMs are effective with binary classification, they can be challenging to use in multi-class classification problems.

4.Selection of hyperparameters: SVMs require careful selection of hyperparameters, such as the kernel function and regularization parameter, which can be difficult to tune without domain expertise.

5.Bias: SVMs can be prone to overfitting, especially when the data is imbalanced or when the sample size is small.

CHAPTER-2

2 .Analysis

2.1 Software Requirement specification

The software and hardware requirements for the project of building a application for trading with smart notifications

2.1.1 Software requirement

1.Python or R programming language: SVM algorithms can be implemented in both Python and R programming language, so either of these languages are required.

2.SVM libraries: SVM requires specialized libraries that can be accessed in both Python and R programming language. Some popular SVM libraries include Scikit-learn and LIBSVM.

3.Data Analysis and Visualization libraries: For data analysis and visualization, libraries such as Pandas, NumPy, and Matplotlib are commonly used in Python. In R, packages like dplyr and ggplot2 can be utilized.

4.Real-time market data sources: Real-time market data sources such as APIs (Application Programming Interfaces) or web scraping tools can be used to retrieve stock market data.

5.Stock market dataset: Historical or real-time stock market datasets are required for analysis.

6.Machine Learning platforms: Platforms such as Jupyter Notebook or RStudio can be used to execute and run the SVM models.

7.High-performance computing systems: Due to the large amounts of data involved in stock market analysis using SVM, high-performance computing systems that can handle intensive calculations are needed.2.1.2 Hardware requirement

2.1.2 Hardware requirement

Processor: A multi-core processor with a clock speed of at least 2 GHz is recommended for optimal performance.

RAM: At least 4 GB of RAM is recommended to handle multiple users and concurrent requests.

Storage: You will need sufficient storage space for your code files and database. A solid-state drive (SSD) is recommended for faster read/write speeds.

Network: A reliable and fast internet connection is important, particularly if the website is intended for use by multiple users.

Server: You will need a server to host your website. Depending on your requirements, you may choose to use a cloud-based server, a dedicated physical server, or a virtual private server (VPS).

2.2 Existing System

There are many existing system for trading stock market.

1.Groww : stock market and Mutual funds :- The Groww App has a user-friendly interface which makes it easier for its millennial customer base to Groww started out as a platform for buying direct mutual funds and later ventured into stocks through their trading app.

2.Upstox : Upstox is one of the best discount brokers in India, especially for traders. On one hand Upstox provides the fastest trading platforms which is the main need to book high profits in day trading.

3.stock Edge : StockEdge is an online analysis tool that focuses on not just equities but also mutual funds. The platform's unique selling point is its ability to serve as an investor's "diary". The platform caters to the needs of both entry-level investors and seasoned investors.

Olymp Trade : Olymp Trade is a world renown online trading platform. Providing Forex, Fixed Time, and Stocks on a world-class platform to traders around the globe. Traders are given the tools they need to be successful: analytics, webinars, strategies, and more

2.3 Proposed System:

In our project we are proposing some new features to the existing model. These include smart notifications which is one of the strategy to control profit and loss. Notifications help traders to stay informed and up-to-date about market conditions, market news, price changes, and any other important events or announcements that may impact their trading decisions. Notifications can be sent via email, text message, or directly to the trader's trading platform, allowing traders to quickly react to opportunities and manage their trades in real-time. This can help traders make better informed decisions, minimize losses, and capitalize on profitable opportunities. Notifications can also provide traders with important reminders, such as order expiration, margin calls, and trading schedule updates, ensuring that they stay organized and avoid costly mistakes.

2.4 Modules

To build a application where traders can easily trade with key features as smart notifications , you will need to use several modules and libraries. Here are some of the key modules and libraries you may consider using:

1.Data Processing and Feature Extraction Module: This module is responsible for analyzing the historical stock data and extracting key features that can be used for prediction purposes. It involves cleaning and pre-processing the data, identifying relevant features, and generating the input matrix for the SVM.

2.SVM Model Building Module: This module is responsible for building and training the SVM model. It involves selecting hyperparameters such as kernel type, parameter C and gamma to optimize the performance of the model. The trained model is then validated using test data to evaluate its accuracy.

3.Prediction Module: Once the SVM model is trained and validated, the prediction module is used to predict future stock prices. This module takes the input data and uses the trained model to make predictions on the future values of the stock.

4.Evaluation Module: This module assesses the accuracy of the SVM model by comparing predicted stock prices to actual stock prices. It provides performance metrics such as accuracy, precision, recall, F1 score, and confusion matrix to measure the performance of the model.

5.Visualization Module: This module is responsible for visualizing the predicted and actual stock prices using graphs and charts to help users understand trends and patterns in the stock market. It also provides insights into the model's performance and helps users to understand the limitations of the model.

2.5 Architecture

`

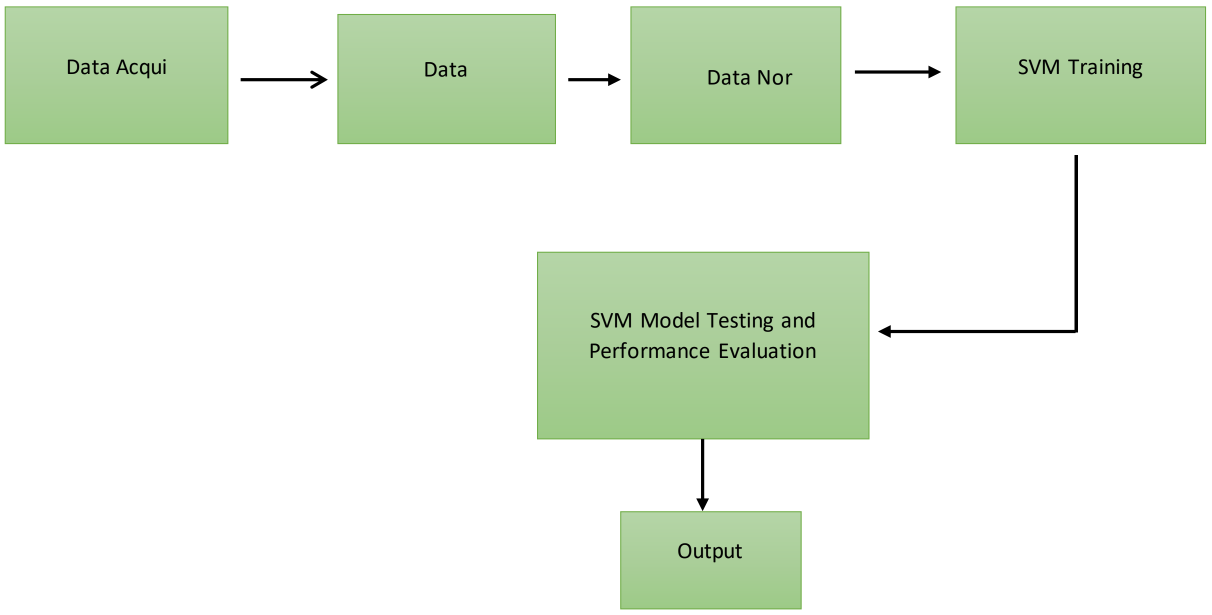


Figure 2.5:Architecture

CHAPTER-3

3.DESIGN

3.1 Data Flow Diagram

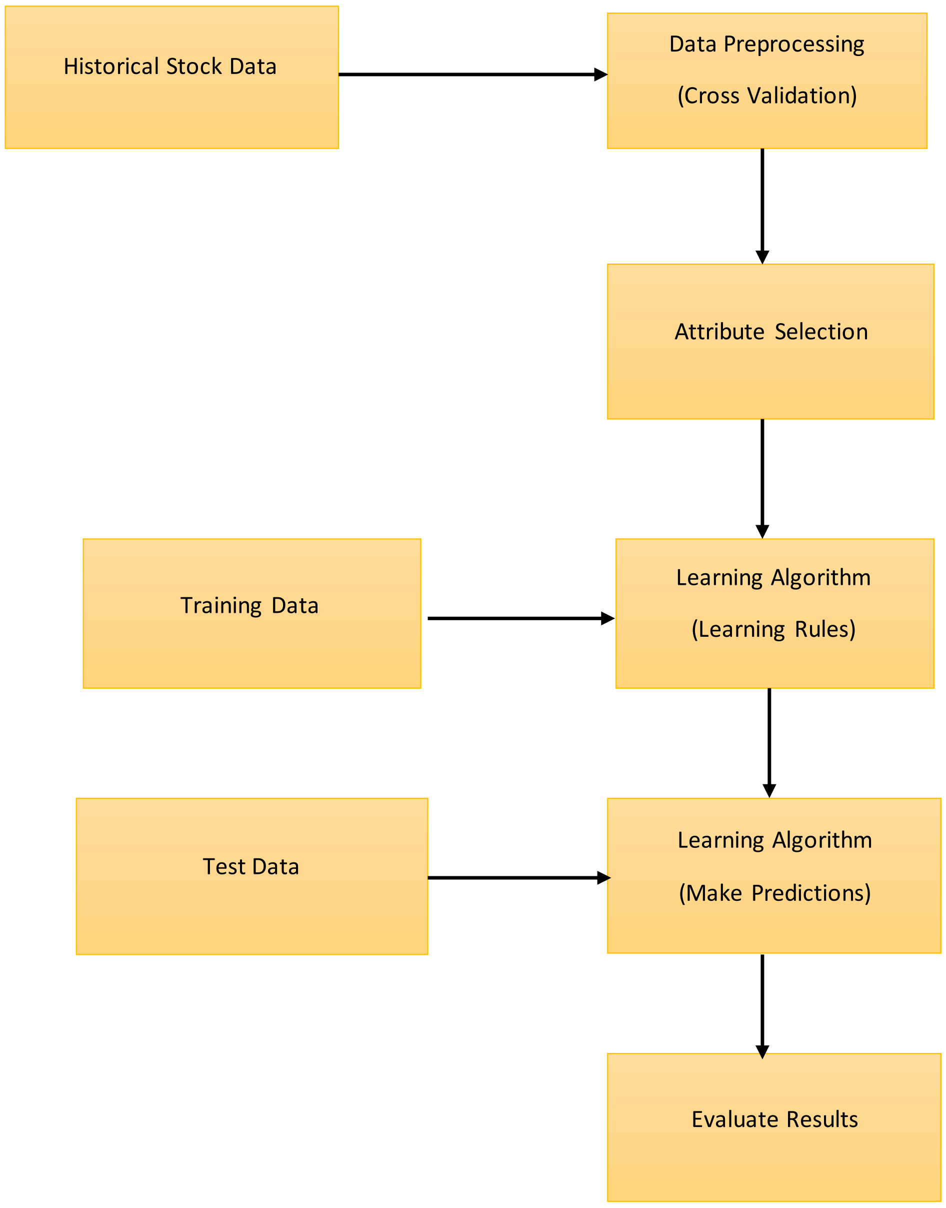


Figure:3.1 Data Flow Diagram

3.2 Data Set Descriptions

Dataset description for stock market analysis using support vector machine (SVM) should include the following:

Data Source: The data source should be mentioned, whether it is a public or private dataset, or data collected from a particular stock market.

Variables: The variables used in the dataset should be listed and their definitions and source should be provided. For example, variables like open price, close price, high price, low price, trading volume, and market capitalization would be relevant.

Time Period: The time period for which the data is collected should be mentioned. This will be useful in understanding the trading patterns and trends of the market.

Target Variable: The target variable that the SVM is trained to predict should be clearly defined. For example, the target variable could be the stock price trend (upward, downward, or neutral), volatility, or the return on investment.

Performance Measures: The performance measures used to evaluate the effectiveness of the SVM should be described, such as accuracy, precision, recall, F1-score, and ROC curves.

3.3Methods and Algorithm

Methods:

Support Vector Machines (SVMs) are a popular tool for stock market analysis due to their ability to effectively handle high-dimensional data and identify patterns within it. Here are some methods for stock market analysis using SVMs:

Feature Selection: The first step in using SVMs for stock market analysis is to identify relevant features. This involves selecting a subset of features that are likely to be predictive of future stock prices. Some common features include technical indicators such as moving averages, volatility measures, and trading volume.

Training the SVM Model: Once the relevant features have been selected, the SVM model can be trained using historical data. This involves feeding the SVM model with a set of labeled data points that represent past stock prices and the corresponding feature values.

Evaluating Model Performance: After the SVM model is trained, its performance can be evaluated using a variety of metrics such as accuracy, precision, and recall. The model can be further optimized by adjusting hyperparameters, such as the kernel function, regularization parameter, and gamma value.

Predicting Future Prices: Once the model is trained and optimized, it can be used to predict future stock prices based on new feature values. The predicted prices can be compared to actual prices to evaluate the accuracy of the model.

Overall, SVMs provide a powerful tool for stock market analysis by enabling the identification of relevant features and patterns within large datasets. However, it is important to note that no model can accurately predict stock prices with 100% accuracy, and the use of SVMs should be complemented with other analytical techniques to gain a more comprehensive understanding of the stock market.

Algorithm

Support Vector Machines (SVM) is a popular machine learning algorithm that can be used for stock market prediction. SVM is a supervised learning method that can be used for both classification and regression tasks.

In the context of stock market prediction, SVM can be used to classify stocks as either "buy," "sell," or "hold" based on historical data. SVM can also be used for regression analysis to predict future stock prices.

One of the main advantages of SVM is its ability to handle large and complex datasets with a high degree of accuracy. SVM also has the ability to deal with non-linear and high-dimensional data, making it an ideal algorithm for analyzing stock market data.

3.4 Building a Model

Building a model for predicting stock market using SVM involves the following steps:

Data Collection: Collect the historical stock price data of the particular company or market you want to analyze. This data should include the opening price, closing price, high price, low price, and volume of shares traded for each trading day.

Data Preprocessing: Clean the data by removing missing or incomplete data, removing outliers, and transforming the data to a suitable format for the SVM algorithm.

Feature Selection: Select the features that are most relevant to predicting the future price of the stock. This may include technical indicators such as moving averages, RSI, or MACD.

Splitting the Data: Divide the data into training and testing datasets. The training dataset will be used to train the SVM model, while the testing dataset will be used to evaluate the model's performance.

SVM Model Training: Train an SVM model using the training dataset. Choose the kernel function that best suits the data and adjust the hyperparameters to optimize the model's performance.

Model Evaluation: Evaluate the performance of the SVM model using the testing dataset. Use appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.

Model Optimization: Refine the SVM model by adjusting the hyperparameters and/or changing the kernel function to improve the model's performance.

Prediction: Use the trained SVM model to predict the future price of the stock based on the test data.

Monitor and Update: Monitor the performance of the model over time and update the model as needed to maintain its accuracy and relevance.

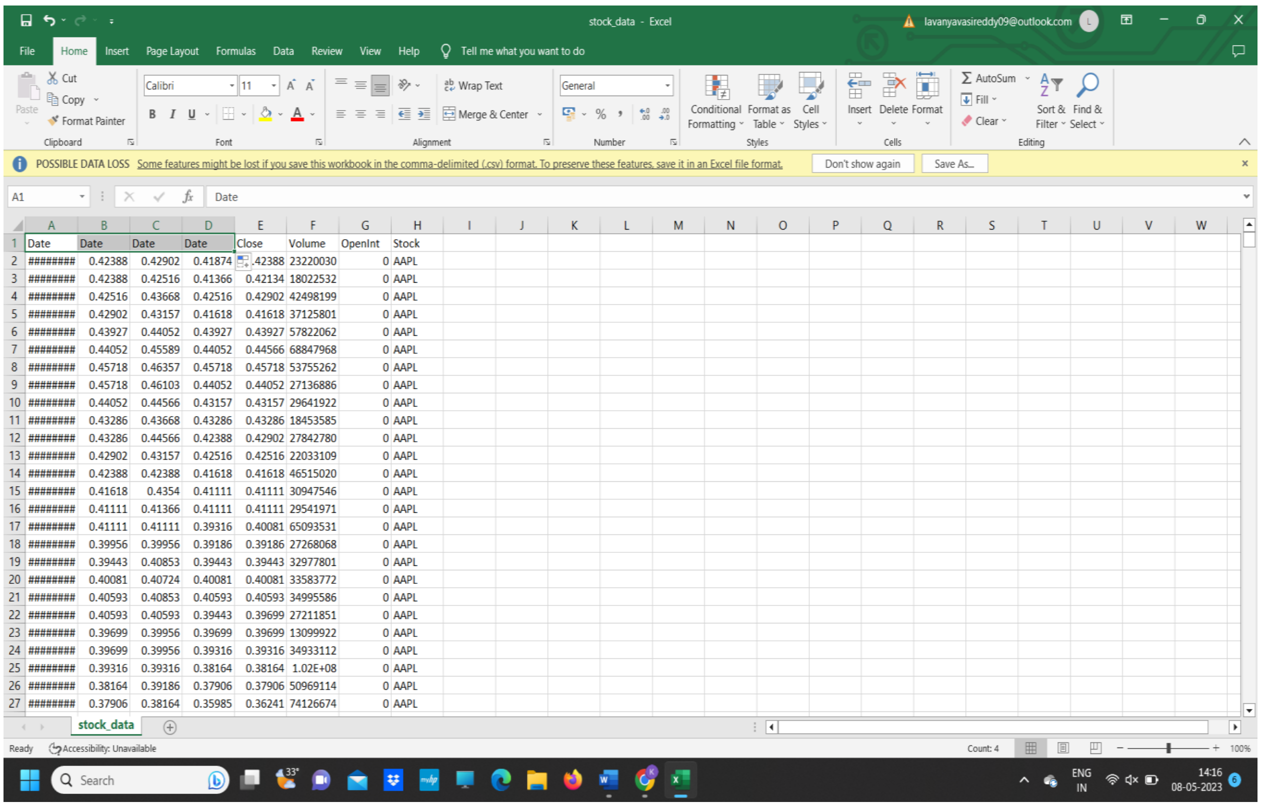
Overall, building an SVM model for predicting stock market trends requires a good understanding of the SVM algorithm, data preprocessing techniques, and evaluation metrics. It also requires a continuous process of monitoring, updating, and refining the model to ensure its effectiveness in predicting stock prices.

CHAPTER-4

4.DEPLOYMENT AND RESULTS

4.1 Source code

DATA SETS



	A	B	C	D	E	F	G	H
1	Date	Date	Date	Date	Close	Volume	OpenInt	Stock
2	########	0.42388	0.42902	0.41874	.42388	23220030	0	AAPL
3	########	0.42388	0.42516	0.41366	0.42134	18022532	0	AAPL
4	########	0.42516	0.43668	0.42516	0.42902	42498199	0	AAPL
5	########	0.42902	0.43157	0.41618	0.41618	37125801	0	AAPL
6	########	0.43927	0.44052	0.43927	0.43927	57822062	0	AAPL
7	########	0.44052	0.45589	0.44052	0.44566	68847968	0	AAPL
8	########	0.45718	0.46357	0.45718	0.45718	53755262	0	AAPL
9	########	0.45718	0.46103	0.44052	0.44052	27136886	0	AAPL
10	########	0.44052	0.44566	0.43157	0.43157	29641922	0	AAPL
11	########	0.43286	0.43668	0.43286	0.43286	18453585	0	AAPL
12	########	0.43286	0.44566	0.42388	0.42902	27842780	0	AAPL
13	########	0.42902	0.43157	0.42516	0.42516	22033109	0	AAPL
14	########	0.42388	0.42388	0.41618	0.41618	46515020	0	AAPL
15	########	0.41618	0.4354	0.41111	0.41111	30947546	0	AAPL
16	########	0.41111	0.41366	0.41111	0.41111	29541971	0	AAPL
17	########	0.41111	0.41111	0.39316	0.40081	65093531	0	AAPL
18	########	0.39956	0.39956	0.39186	0.39186	27268068	0	AAPL
19	########	0.39443	0.40853	0.39443	0.39443	32977801	0	AAPL
20	########	0.40081	0.40724	0.40081	0.40081	33583772	0	AAPL
21	########	0.40593	0.40853	0.40593	0.40593	34995586	0	AAPL
22	########	0.40593	0.40593	0.39443	0.39699	27211851	0	AAPL
23	########	0.39699	0.39956	0.39699	0.39699	13099922	0	AAPL
24	########	0.39699	0.39956	0.39316	0.39316	34933112	0	AAPL
25	########	0.39316	0.39316	0.38164	0.38164	1.02E+08	0	AAPL
26	########	0.38164	0.39186	0.37906	0.37906	50969114	0	AAPL
27	########	0.37906	0.38164	0.35985	0.36241	74126674	0	AAPL

stock_data - Excel
lavanyavasireddy09@outlook.com
File
Home
Insert
Page Layout
Formulas
Data
Review
View
Help
Tell me what you want to do
POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.
Don't show again
Save As...
A1
Date
stock_data
Ready
Accessibility: Unavailable
Count: 4
100%
14:16
08-05-2023

Source code:

```
import dash
from dash import dcc
from dash import html
import pandas as pd
import plotly.graph_objs as go
from dash.dependencies import Input, Output
from keras.models import load_model
from sklearn.preprocessing import MinMaxScaler
import numpy as np
app = dash.Dash()
server = app.server
scaler=MinMaxScaler(feature_range=(0,1))
df_nse = pd.read_csv("./NSE-TATA.csv")
df_nse["Date"]=pd.to_datetime(df_nse.Date,format="%Y-%m-%d")
df_nse.index=df_nse['Date']
data=df_nse.sort_index(ascending=True,axis=0)
new_data=pd.DataFrame(index=range(0,len(df_nse)),columns=['Date','Close'])
for i in range(0,len(data)):
    new_data["Date"][i]=data['Date'][i]
    new_data["Close"][i]=data["Close"][i]
new_data.index=new_data.Date
new_data.drop("Date",axis=1,inplace=True)
dataset=new_data.values
train=dataset[0:987,:]
valid=dataset[987:,:]
scaler=MinMaxScaler(feature_range=(0,1))
scaled_data=scaler.fit_transform(dataset)
x_train,y_train=[],[]
for i in range(60,len(train)):
    x_train.append(scaled_data[i-60:i,0])
    y_train.append(scaled_data[i,0])
    x_train,y_train=np.array(x_train),np.array(y_train)
x_train=np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))
model=load_model("saved_model.h5")
inputs=new_data[len(new_data)-len(valid)-60:].values
inputs=inputs.reshape(-1,1)
inputs=scaler.transform(inputs)
X_test=[]
for i in range(60,inputs.shape[0]):
    X_test.append(inputs[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
closing_price=model.predict(X_test)
closing_price=scaler.inverse_transform(closing_price)
train=new_data[:987]
valid=new_data[987:]
valid['Predictions']=closing_price
df= pd.read_csv("./stock_data.csv")
app.layout = html.Div([
html.H1("Stock Price Analysis Dashboard", style={"textAlign": "center"}),
dcc.Tabs(id="tabs", children=[
dcc.Tab(label='NSE-TATAGLOBAL Stock Data',children=[
                        html.Div([
                                html.H2("Actual closing price",style={"textAlign": "center"}),
                                dcc.Graph(
                                        id="Actual Data",
                                        figure={
                                                "data":[
                                                        go.Scatter(
                                                                x=train.index,
                                                                y=valid["Close"],
                                                                mode='markers'
                                                        )
                                                        ],
                                                        "layout":go.Layout(
                                                                title='scatter plot',
                                                                xaxis={'title':'Date'},
                                                                yaxis={'title':'Closing Rate'}
                                                        )
                                                }
                                        ),
                                        html.H2("LSTM Predicted closing price",style={"textAlign": "center"}),
                                        dcc.Graph(
                                                id="Predicted Data",
                                                figure={
                                                        "data":[
                                                                go.Scatter(
                                                                        x=valid.index,
                                                                        y=valid["Predictions"],
                                                                        mode='markers'
                                                                )
                                                        ],
                                                        "layout":go.Layout(
                                                                title='scatter plot',
                                                                xaxis={'title':'Date'},
                                                                yaxis={'title':'Closing Rat)
                                                }
                        )
                ])

        ]),
        dcc.Tab(label='Facebook Stock Data', children=[
            html.Div([
                html.H1("Stocks High vs Lows",
                        style={'textAlign': 'center'}),
                dcc.Dropdown(id='my-dropdown',
                             options=[{'label': 'Tesla', 'value': 'TSLA'},
                                      {'label': 'Apple','value': 'AAPL'},
                                      {'label': 'Facebook', 'value': 'FB'},
                                      {'label': 'Microsoft','value': 'MSFT'}],
                             multi=True,value=['FB'],
                             style={"display": "block", "margin-left": "auto",
                                    "margin-right": "auto", "width": "60%"}),
                dcc.Graph(id='highlow'),
                html.H1("Stocks Market Volume", style={'textAlign': 'center'}),
                dcc.Dropdown(id='my-dropdown2',
                             options=[{'label': 'Tesla', 'value': 'TSLA'},
                                      {'label': 'Apple','value': 'AAPL'},
                                      {'label': 'Facebook', 'value': 'FB'},
                                      {'label': 'Microsoft','value': 'MSFT'}],
                             multi=True,value=['FB'],
                             style={"display": "block", "margin-left": "auto",
                                    "margin-right": "auto", "width": "60%"}),
                dcc.Graph(id='volume')
            ], className="container"),
        ])
 ])
])
```

```
@app.callback(Output('highlow', 'figure'),
              [Input('my-dropdown', 'value')])
def update_graph(selected_dropdown):
    dropdown = {"TSLA": "Tesla","AAPL": "Apple","FB": "Facebook","MSFT": "Microsoft",}
    trace1 = []
    trace2 = []
    for stock in selected_dropdown:
        trace1.append(
          go.Scatter(x=df[df["Stock"] == stock]["Date"],
                     y=df[df["Stock"] == stock]["High"],
                     mode='lines', opacity=0.7,
                     name=f'High {dropdown[stock]}',textposition='bottom center'))
        trace2.append(
          go.Scatter(x=df[df["Stock"] == stock]["Date"],
                     y=df[df["Stock"] == stock]["Low"],
                     mode='lines', opacity=0.6,
                     name=f'Low {dropdown[stock]}',textposition='bottom center'))
    traces = [trace1, trace2]
    data = [val for sublist in traces for val in sublist]
    figure = {'data': data,
              'layout': go.Layout(colorway=["#5E0DAC", '#FF4F00', '#375CB1',
                                            '#FF7400', '#FFF400', '#FF0056'],
            height=600,
            title=f"High and Low Prices for {', '.join(str(dropdown[i]) for i in selected_dropdown)} Over Time",
            xaxis={"title":"Date",
                   'rangeselector': {'buttons': list([{'count': 1, 'label': '1M',
                                                       'step': 'month',
                                                       'stepmode': 'backward'},
                                                      {'count': 6, 'label': '6M',
                                                       'step': 'month',
                                                       'stepmode': 'backward'},
                                                      {'step': 'all'}])},
                   'rangeslider': {'visible': True}, 'type': 'date'},
             yaxis={"title":"Price (USD)"})}
    return figure


@app.callback(Output('volume', 'figure'),
              [Input('my-dropdown2', 'value')])
def update_graph(selected_dropdown_value):
    dropdown = {"TSLA": "Tesla","AAPL": "Apple","FB": "Facebook","MSFT": "Microsoft",}
    trace1 = []
    for stock in selected_dropdown_value:
        trace1.append(
          go.Scatter(x=df[df["Stock"] == stock]["Date"],
                y=df[df["Stock"] == stock]["Volume"],
                mode='lines', opacity=0.7,
                name=f'Volume {dropdown[stock]}', textposition='bottom center'))
    traces = [trace1]
    data = [val for sublist in traces for val in sublist]
    figure = {'data': data,
            'layout': go.Layout(colorway=["#5E0DAC", '#FF4F00', '#375CB1',
                                        '#FF7400', '#FFF400', '#FF0056'],
            height=600,
            title=f"Market Volume for {', '.join(str(dropdown[i]) for i in selected_dropdown_value)} Over Time",
            xaxis={"title":"Date",
                'rangeselector': {'buttons': list([{'count': 1, 'label': '
                                            'step': 'month',
                                            'stepmode': 'backward'},
                                            {'count': 6, 'label': '6M',
                                            'step': 'month',
                                            'stepmode': 'backward'},
                                            {'step': 'all'}])},
                'rangeslider': {'visible': True}, 'type': 'date'},
             yaxis={"title":"Transactions Volume"})}
    return figure
if __name__=='__main__':
        app.run_server(debug=True)
```

Stock Prediction

```
import sys
import time
import numpy as np
import itertools as it
import multiprocessing
from sklearn import svm
from sklearn.model_selection import KFold, GridSearchCV
class StockSVM:
def __init__(self, values):
    self.values = values
    self.predict_next_k_days = dict()
    self.clf = None
def __repr__(self):
    return self.values.__repr__()
def __str__(self):
    return self.values.__str__()
def addPredictNext_K_Days(self, k_days, predict_next_k_day):
    '''
Add list of prediction according day key
    '''
    self.predict_next_k_days[k_days] = predict_next_k_day
def getValidFitParam(self, predictNext_k_day):
    '''
    Return parameters what is not NaN
    ''' vals, preds = [], []
```

```
        for val, pred in zip(self.values.values, self.predict_next_k_days[predictNext_k_day]):
            if not np.isnan(pred):
vals.append(val)
                preds.append(int(pred))
        return vals, preds
def fit(self, predictNext_k_day, C=1.0, gamma='auto'):
        '''
        Ordinary SVC fitting
        '''


if predictNext_k_day not in self.predict_next_k_days.keys()
 print('{0} day(s) for predictNext not defined! Please, add a vector for that days first!'.format(
                predictNext_k_day))
            sys.exit()
vals, preds = self.getValidFitParam(predictNext_k_day)
svc = svm.SVC(C=C, gamma=gamma)
        self.clf = svc.fit(vals, preds)
def fit_GridSearch(self, predictNext_k_day, parameters, n_jobs=3, k_fold_num=None, verbose=1)
  '''
 Grid Search Cross Validation Fitting
        '''
        if predictNext_k_day not in self.predict_next_k_days.keys():
            print('{0} day(s) for predictNext not defined! Please, add a vector for that days first!'.format(
                predictNext_k_day))
            sys.exit()
 vals, preds = self.getValidFitParam(predictNext_k_day)
svc = svm.SVC()
        self.clf = GridSearchCV(svc, parameters, n_jobs=n_jobs, cv=k_fold_num, verbose=verbose)
 self.clf = self.clf.fit(vals, preds)
def fit_Cross_Validation(self, predictNext_k_day, parameters, k_fold_num=3, maxRunTime=25, verbose=1):
        if predictNext_k_day not in self.predict_next_k_days.keys():
            print('{0} day(s) for predictNext not defined! Please, add a vector for that days first!'.format(
                predictNext_k_day))
            sys.exit()
 vals, preds = self.getValidFitParam(predictNext_k_day)
        keys = parameters.keys()
def __fitSVC__(svc, X, y, queue):
            clf = queue.get()
clf = svc.fit(X, y)
            queue.put(clf)
def __runProcess__(svc, X_train, y_train, queue):
            try:
                p = multiprocessing.Process(target=__fitSVC__,
                                            name="fitSVC",
                                            args=(svc, X_train, y_train, queue))
                t = 0
                interrupted = False
                p.start()  # Start fitting process
                while p.is_alive():
                    print("Time elapsed: {0:.2f}".format(t), end='\r')
                    # Reach maximum time, terminate process
                    if t >= maxRunTime:
                        p.terminate()
                        p.join()
                        interrupted = True
                        break
                    time.sleep(0.01)
                    t += 0.01
except Exception:
                p.terminate()
                p.join()
          return interrupted
        best_score = 0
        best_svc = None
        best_estimators = None
        queue = multiprocessing.Queue()
        for params in it.product(*parameters.values()):
```

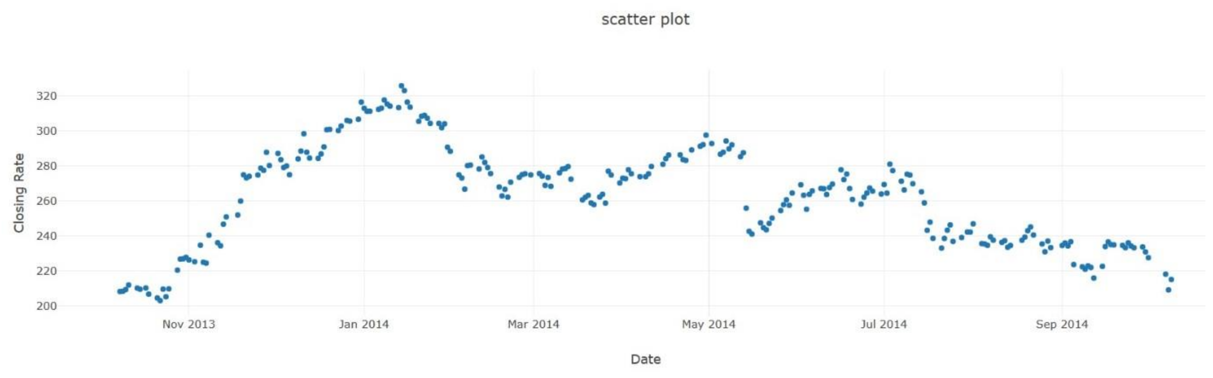
```
 if verbose != 0:
               print("Params: ", params)
        # Create model with current params
           dict_params = dict(zip(keys, params))
           svc = svm.SVC(**dict_params)
           score_sum = 0
           num_scores = 0
 kf = KFold(n_splits=k_fold_num)
           # Split vals, preds in k_fold_num folds
           for train_ids, test_ids in kf.split(vals, preds):
# Get vals, preds train data according current fold
               X_train = [vals[k] for k in train_ids]
y_train = [preds[k] for k in train_ids]
last_clf = None
               queue.put(last_clf)  # Share last_clf variable
# Init fitting parallel process
               print(" Init process")
               interrupted = __runProcess__(svc, X_train, y_train, queue)

               # If process have finished, not interrupted by max run time
               if not interrupted:
                   last_clf = queue.get()  # Get shared last_clf variable
 # Get vals, preds test data according current fold
                   X_test = [vals[k] for k in test_ids]
                   y_test = [preds[k] for k in test_ids]
# Add score for current fold
                   score = last_clf.score(X_test, y_test)
                   score_sum += score
                   num_scores += 1
print(" Last score: ", score)
                   print(" Score sum: ", score_sum)
                   print()
               else:
                   print(" Process interrupted!")
                   print()
              if num_scores != 0:
               avarege_score = score_sum / num_scores
               print(" Average score: ", avarege_score)
               print()
               if avarege_score > best_score:
                   best_score = avarege_score
                   best_svc = svc
                   best_estimators = params
        print()
        print("Best Score: ", best_score)
        print("Best SVC: ", best_svc)
        print("Best Estimators: ", best_estimators)
        print()
    if best_svc is not None:
           self.clf = best_svc.fit(vals, preds)
# * Prediction function
    def predict(self, X):
 return self.clf.predict(X)
```

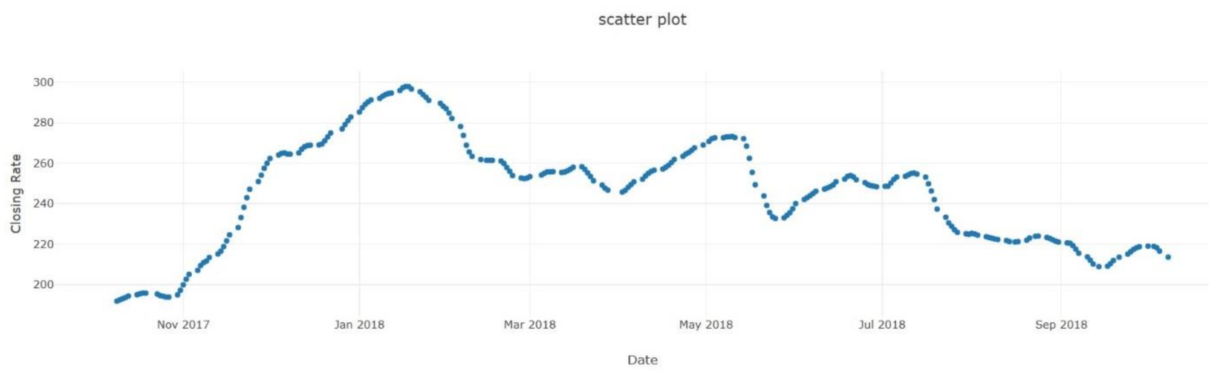
Stock Price Analysis Dashboard

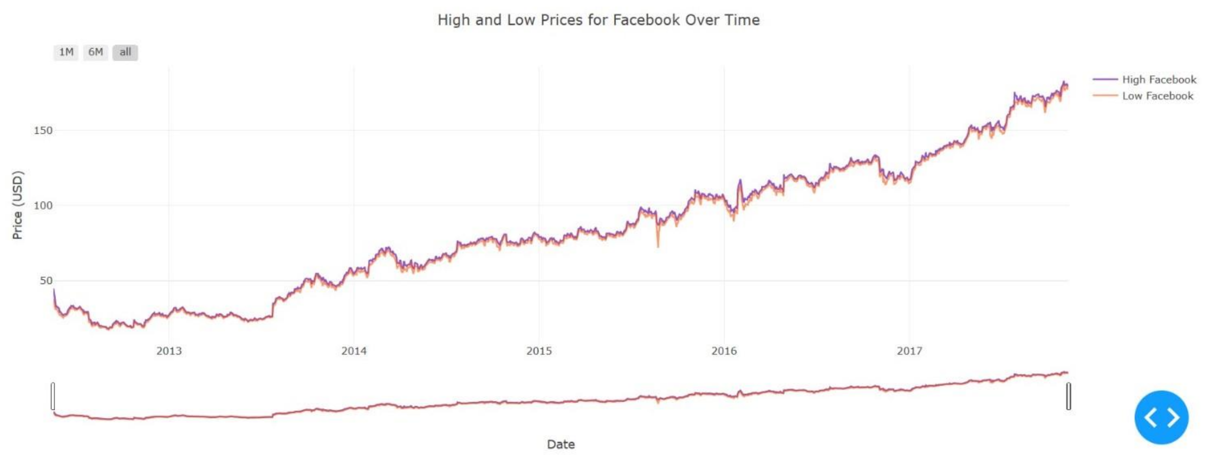
NSE-TATAGLOBAL Stock Data | Facebook Stock Data

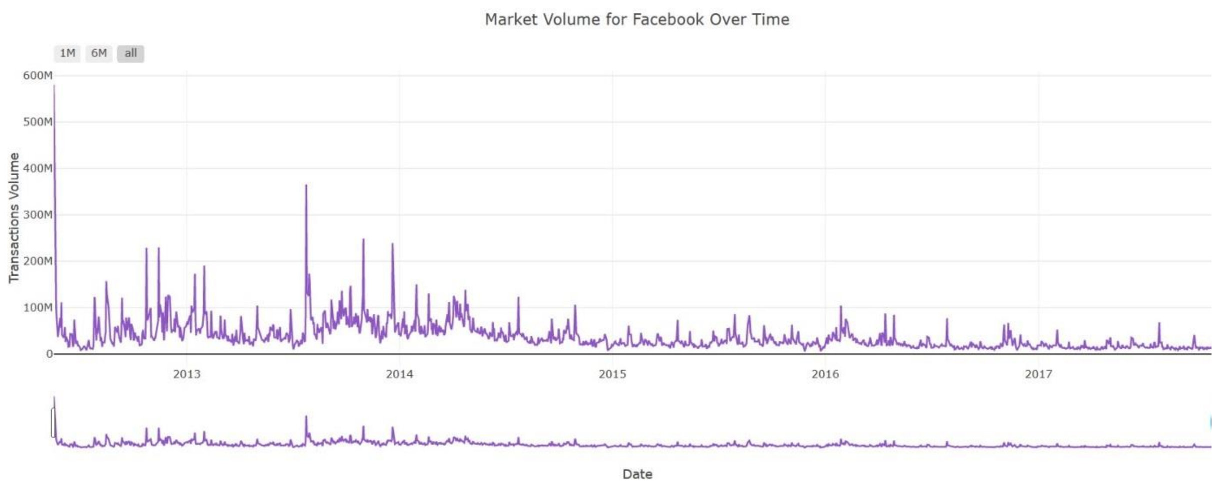
Actual closing price



LSTM Predicted closing price






Market Volume for Facebook Over Time
1M
6M
all
600M
500M
400M
300M
200M
100M
0
Transactions Volume
2013
2014
2015
2016
2017
Date

CHAPTER-5

5.1 Project Conclusion

Conclusion

In conclusion, the stock analysis with smart notifications is and user-friendly tool that enhance user interest to trade and to stay informed up-to-date about market conditions, market news ,price charges, and any other announcements that may impact their trading decisions. With it advanced algorithm and machine learning techniques, the application provides a easy to use platform for traders . Whether you are a professional or beginner you can easily trade and minimize your losses , and capitalize on profitable opportunities. Overall, SVMs can be a valuable tool in stock market analysis, but their effectiveness will depend on the specific problem and the quality of the data used. It is important to carefully evaluate the performance of an SVM model before making investment decisions based on its predictions.

5.2 Future Enhancement

Incorporation of more data sources: SVM models can be improved by incorporating additional data sources into the analysis. For example, social media sentiment analysis or news articles related to the stock market could be included to provide additional insight into the market movements.

Integration of feature selection techniques: SVM models can be improved by incorporating feature selection techniques, such as principal component analysis (PCA) or feature importance ranking. This would help to identify the most important features that impact the stock market and improve the accuracy of the model.

Implementation of ensemble learning: SVM models can be improved by implementing ensemble learning techniques, such as bagging or boosting. This involves training multiple SVM models and combining their outputs to generate a more accurate prediction.

Development of a real-time trading system: SVM models can be implemented in a real-time trading system to provide traders with insights on potential trades. This system could beintegrated with a trading platform to execute trades automatically based on the SVM model's predictions.