STOCK PRICE PREDICTION

PHASE IV REPORT

Submitted by

Supervisor:- Ms. Harsh Sharma(E13523)
Co Supervisor:- Mr. Siroj Kumar Singh(E13617)

NAME OF THE CANDIDATE(S)

NAME	-	UID
ANMOL SHAKYA	-	20BCS5146
BUNTY PRASAD NAYAK	-	20BCS1160
DEVANSH TIWARI	-	20BCS1235

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Chandigarh University

OCTOBER & 2022

CHAPTER 4.

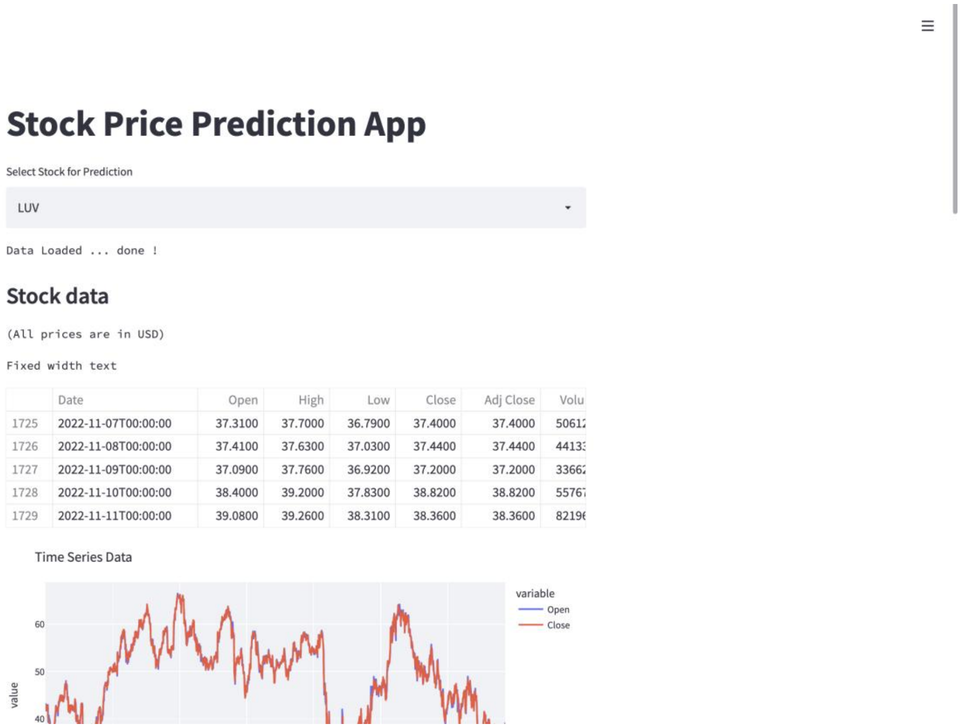
RESULTS ANALYSIS AND VALIDATION

1.1. Implementation of solution -

```
import streamlit as st
# Streamlit is an open-source python framework for building web app. we are using in the
program to build web app and show the result of ML algorithms.
# We  can rapidly build the tools you need. Build apps in a dozen lines of Python with a simple
API. Streamlit is a tool in the Machine Learning Tools category of a tech stack.
# Using it we can also works with TensorFlow, Keras, PyTorch, Pandas, Numpy, Matplotlib,
Seaborn, Altair, Plotly, Bokeh, Vega-Lite, and more.

from datetime import date
# Python Datetime module  , Using it we can get current local  date and time .

import pandas as pd
# Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation
tool, built on top of the Python programming language.
# Pandas is the best tool for handling this real-world messy data. And pandas is one of the open-
source python packages built on top of NumPy.

import yfinance as yf
# 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.
#  We can easily download historical stock data from yfinance.

import plotly.express as px
#The plotly.express module (usually imported as px ) contains functions that can create entire
figures at once, and is referred to as Plotly Express or PX.
#Plotly Express is the easy-to-use, high-level interface to Plotly, which operates on a variety of
```

```
types of data and produces easy-to-style figures.
#Plotly Express provides functions to visualize a variety of types of data. Most functions such as px. bar or px.

from prophet import Prophet
#Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.
#It works best with time series that have strong seasonal effects and several seasons of historical data.

# from plotly import graph_objects as go.
from prophet.plot import plot_plotly, plot_components_plotly

# Get the current local date.
START = "2016-01-01"
# TODAY = "2017-01-01"
TODAY= date.today().strftime("%Y-%m-%d")

# Main title in web app.
st.title("Stock Price Prediction App")

# displaying all the possible stock
stocks =pd.read_csv('https://raw.githubusercontent.com/kaushikjadhav01/Stock-Market-Prediction-Web-App-using-Machine-Learning-And-Sentiment-Analysis/master/Yahoo-Finance-Ticker-Symbols.csv')

# creating a select box in web app.
selected_stocks = st.selectbox("Select Stock for Prediction",stocks)

# Loading data .
@st.cache #Cache the data so we don’t have to download the data again and again.
def load_data(ticker):
  data = yf.download(ticker,START,TODAY)
  data.reset_index(inplace=True)
  return data

data_load_state = st.text("Loading data.........")
data = load_data(selected_stocks)
data_load_state.text("Data Loaded.....done ! ")

st.subheader('Stock data')
st.text("(All prices are in USD)")
st.text('Fixed width text')
# last 5 column will be printed of the selected stock
st.write(data.tail())

# Plot the data on graph
# #df = px.data.stocks()
fig = px.line(data, x='Date', y=['Open','Close'])
fig.layout.update(title_text = "Time Series Data" , xaxis_rangeslider_visible = True)
fig.update_layout(
  margin=dict(l=10, r=20, t=50, b=40),
  )
st.plotly_chart(fig)

# Sliding bar
n_years = st.slider("Years of Prediction: ", 1,7)
period = n_years*365

#Forecasting by train the dataset
df_train = data[['Date','Close']]
df_train  = df_train.rename(columns={"Date": "ds","Close": "y"})
m = Prophet()
m.fit(df_train)
```

```
future = m.make_future_dataframe(periods=period)
forecast = m.predict(future)

# forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
# st.write(forecast)
st.subheader('Predict Stock')
st.write(forecast.tail())
st.write('Predict Stock')
# fig1 = go(m,forecast)
fig1 = plot_plotly(m, forecast )
st.plotly_chart(fig1)
st.write('forecast Components')
fig2 = plot_components_plotly(m, forecast)
st.write(fig2)
```



Stock Price Prediction App

Select Stock for Prediction

LUV

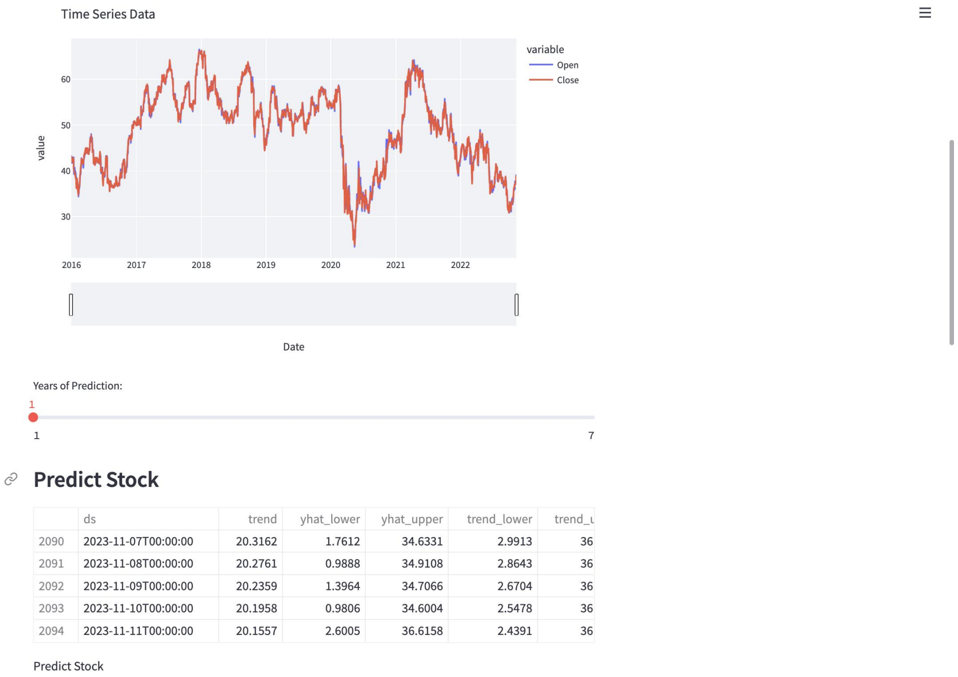
```
Data Loaded ... done !
```

Stock data

```
(All prices are in USD)

Fixed width text
```

	Date	Open	High	Low	Close	Adj Close	Volu
1725	2022-11-07T00:00:00	37.3100	37.7000	36.7900	37.4000	37.4000	50612
1726	2022-11-08T00:00:00	37.4100	37.6300	37.0300	37.4400	37.4400	44133
1727	2022-11-09T00:00:00	37.0900	37.7600	36.9200	37.2000	37.2000	33662
1728	2022-11-10T00:00:00	38.4000	39.2000	37.8300	38.8200	38.8200	55767
1729	2022-11-11T00:00:00	39.0800	39.2600	38.3100	38.3600	38.3600	82196

Fig.1 Dashboard Web App to predict Stock Price



Years of Prediction:

1

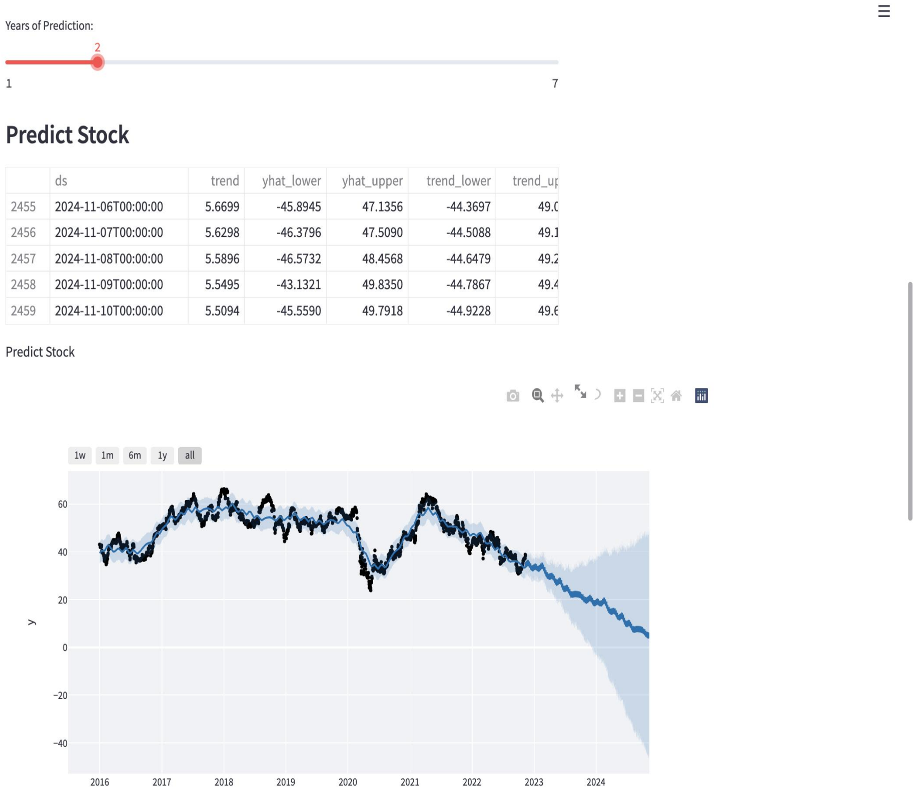
1 7

Predict Stock

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_u
2090	2023-11-07T00:00:00	20.3162	1.7612	34.6331	2.9913	36
2091	2023-11-08T00:00:00	20.2761	0.9888	34.9108	2.8643	36
2092	2023-11-09T00:00:00	20.2359	1.3964	34.7066	2.6704	36
2093	2023-11-10T00:00:00	20.1958	0.9806	34.6004	2.5478	36
2094	2023-11-11T00:00:00	20.1557	2.6005	36.6158	2.4391	36

Predict Stock

Fig.2 Slidebar to predict according to year



Years of Prediction:

2

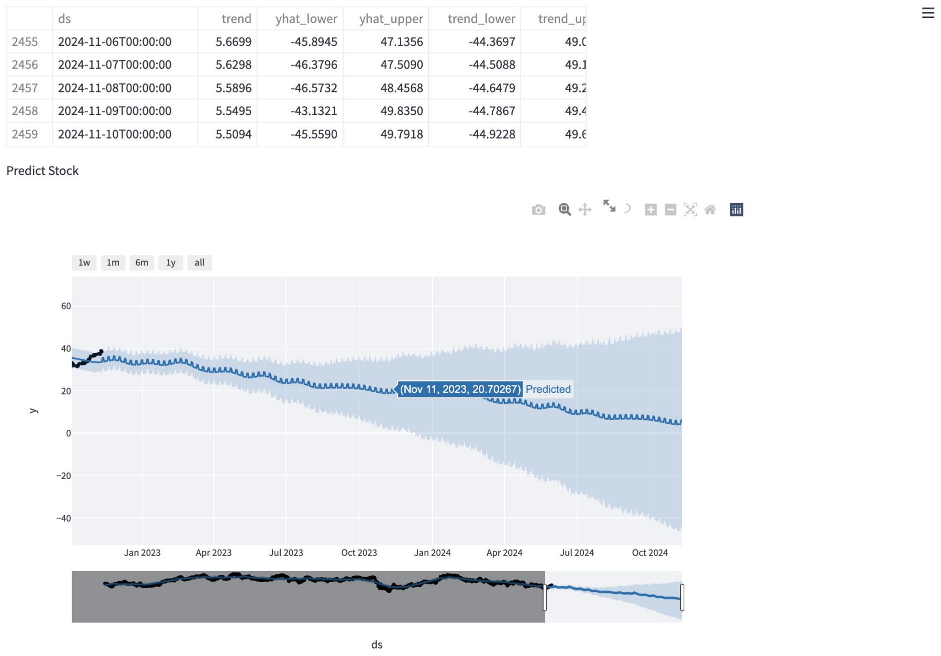
1 7

Predict Stock

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_up
2455	2024-11-06T00:00:00	5.6699	-45.8945	47.1356	-44.3697	49.0
2456	2024-11-07T00:00:00	5.6298	-46.3796	47.5090	-44.5088	49.1
2457	2024-11-08T00:00:00	5.5896	-46.5732	48.4568	-44.6479	49.2
2458	2024-11-09T00:00:00	5.5495	-43.1321	49.8350	-44.7867	49.4
2459	2024-11-10T00:00:00	5.5094	-45.5590	49.7918	-44.9228	49.6

Predict Stock

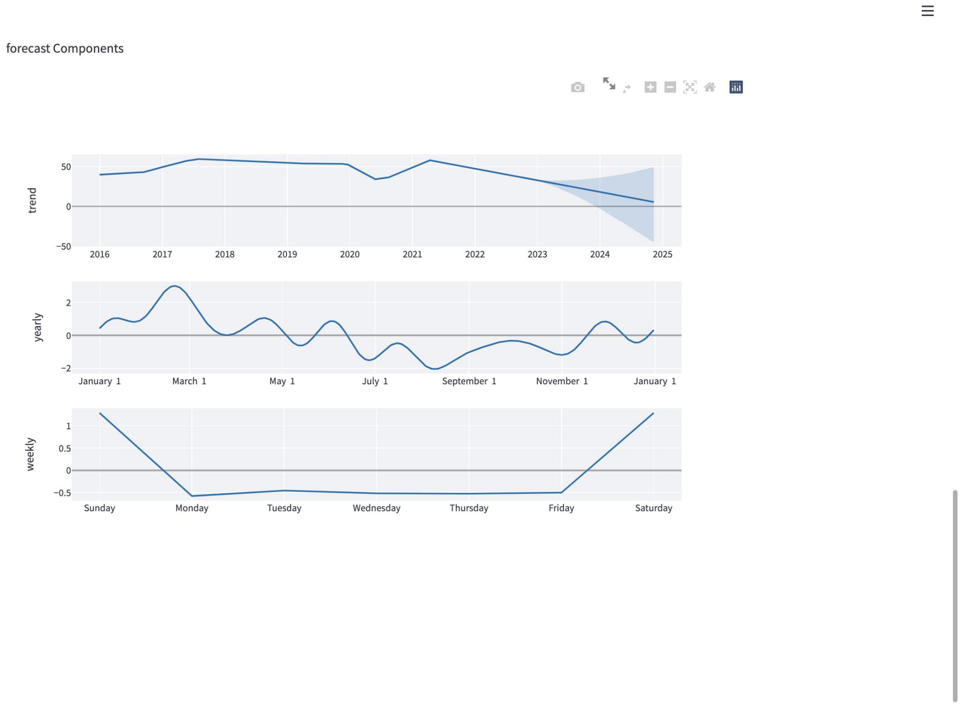
Fig.3 Plotting graph for prediction



	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_up
2455	2024-11-06T00:00:00	5.6699	-45.8945	47.1356	-44.3697	49.0
2456	2024-11-07T00:00:00	5.6298	-46.3796	47.5090	-44.5088	49.1
2457	2024-11-08T00:00:00	5.5896	-46.5732	48.4568	-44.6479	49.2
2458	2024-11-09T00:00:00	5.5495	-43.1321	49.8350	-44.7867	49.4
2459	2024-11-10T00:00:00	5.5094	-45.5590	49.7918	-44.9228	49.6

Predict Stock

Fig.4. Prediction of stock in the months.



forecast Components

Fig.5 Forecast components on trend, yearly, and weekly basis.

REFERENCES

1. A. Ben–Hur and J. Weston. A user's guide to support vector machines. In O. Carugo and F. Eisenhaber (eds.), *Data Mining Techniques for the Life Sciences*, pp. 223––239. Springer, 2010.

2. Bing Search API. Windows Azure Marketplace.

3. N. K. Chowdhury and C. K.–S. Leung. Improved travel time prediction algorithms for intelligent transportation systems. In *Proc. KES 2011, Part II*, pp. 355––365.

4. A. Cuzzocrea, C. K.–S. Leung, and R. K. MacKinnon. Mining constrained frequent itemsets from distributed uncertain data. *FGCS*, 37: 117––126, July 2014.

5. T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *Proc. ICML 2008*, ACM, pp. 304––311. ACM.

6. D. Hodges. Is your fund manager beating the index? *MoneySense*, 15(7):10, Dec. 2013/Jan. 2014.

7. T. Joachims, T. Finley, and C.–N. J. Yu. Cutting–plane training of structural SVMs. *Machine Learning*, 77(1): 27––59, Oct. 2009.

8. D. E. King. Dlib–ml: a machine learning toolkit. *JMLR*, 10: 1755––1758, July 2009.

9. V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts–a review. *IEEE TPAMI*, 29(7): 1274––1279, July 2007.

10. . K.–S. Leung, F. Jiang, and Y. Hayduk. A landmark–model based system for mining frequent patterns from uncertain data streams. In *Proc. IDEAS 2011*, pp. 249––250. ACM.