PREDICT STOCK PRICES

Done By,

SR NO	NAME	REGNO	ROLLNO	GROUP	PROJCT NO
1	MOHAMMED JAKHIR HUSSAN J	11918687	A21	G15	23
2	BHARATH KUMAR R	11812345	A32	G15	23

Course code : INT404

Subject : ARTIFICIAL INTELLIGENCE

Submitted to the Professor : Ashish Shrivastava

UID Number : 25703

OBJECTIVE

For a family of stocks, generally belonging to the same sector or industry, there exists a correlation between prices of each of the stocks. There, though, exist anomalous times when for a small period of time, the correlation is broken. But the market self corrects in some time and the correlation is re-established. During this small window of time when correlation is anomalous, there exists a money-making opportunity for quantitative traders.

The main objective of the project is that to predict the stock market prices. So for doing this project code I have the chosen the stocks market data of some of the companies which are been in for a long time. To write the code I have used some of the libraries of python and some of the concepts of data science taken from the facebook to predict the stock market so according to that anyone can see the output of the code which will be displayed in a graph form.

LIBRARIES USED

Quandl : It is the most important uses the project code to get the data of stock market prices. Quandl offers a simple API for stock market data downloads. It helps in our daily data feeds deliver end-of-day prices,
historical stock fundamental data, harmonized fundamentals, financial ratios, indexes, options and volatility, earnings estimates, analyst ratings, investor sentiment and more.

Numpy : It stands for numerical python. It is used for processing the table formed data like a CSV file. It is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

Pandas : It is for processing and fetch the data from the CSV file to get the inputs to plot a graph. It is used for manipulate and wrangle the data. Pandas is also an elegant solution for time series data. It is very similar and it works same most of the similar operations as numpy does.

Fbprophet : It is facebook data science library for python to predict stock market. Implements 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.

Matplotlib : It is used for plotting for the graphs. matplotlib is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

ALGORITHM

FBPROPHET ALGORITHM

Fbprophet algorithm is used in the project to predict the stock prices based on this the code is written to make it more easy to understand.

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.

WORKING CODE

In [ ]:

```
import quandl  # quandl to get stock market data
import pandas as pd  # pandas for processing csv files
import numpy as np  # numpy for processing table formed data
import fbprophet  # facebook data science library for python to predict stock market
import matplotlib.pyplot as plt  # matplotlib pyplot for plotting

class Stocker():

    def __init__(self, stock_name, exchange='WIKI'):

        self.stock_name = stock_name.upper() # Capitalize stock name

        quandl.ApiConfig.api_key = 'TcKtBtukcckAFsspxbkS' # API Key for quandl

        stock = quandl.get('%s/%s' % (exchange, stock_name)) # Fetch stock data from quandl

        # Set the index to a column called Date
        stock = stock.reset_index(level=0)

        # Columns required for prophet
        stock['ds'] = stock['Date']

        if ('Adj. Close' not in stock.columns):
            stock['Adj. Close'] = stock['Close']
            stock['Adj. Open'] = stock['Open']

        stock['y'] = stock['Adj. Close']
        stock['Daily Change'] = stock['Adj. Close'] - stock['Adj. Open']

        # Data assigned as class attribute
        self.stock = stock.copy()
```

```
    # Prophet parameters
    # Default prior from library
    self.changepoint_prior_scale = 0.05
    self.weekly_seasonality = False
    self.daily_seasonality = False
    self.monthly_seasonality = True
    self.yearly_seasonality = True
    self.changepoints = None

# Create a prophet model without training
def create_model(self):

    # Make the model
    model = fbprophet.Prophet(daily_seasonality=self.daily_seasonality,
                              weekly_seasonality=self.weekly_seasonality,
                              yearly_seasonality=self.yearly_seasonality,
                              changepoint_prior_scale=self.changepoint_prior_scale,
                              changepoints=self.changepoints)

    if self.monthly_seasonality:
        # Add monthly seasonality
        model.add_seasonality(name = 'monthly', period = 30.5, fourier_order = 5)

    return model

def predict_future(self, days=30):

    # Use past self.training_years years for training
    train = self.stock[self.stock['Date'] > (max(self.stock['Date']) - pd.DateOffset(years=3))]

    model = self.create_model()
```

```
    model.fit(train)

    # Future dataframe with specified number of days to predict
    future = model.make_future_dataframe(periods=days, freq='D')
    future = model.predict(future)

    # Only concerned with future dates
    future = future[future['ds'] >= max(self.stock['Date'])]

    # Calculate whether increase or not
    future['diff'] = future['yhat'].diff()

    future = future.dropna() # Drop empty rows

    # Set up plot

    plt.plot(future['ds'],future['yhat'], label='Prediction')
    plt.xticks(rotation = '45') # Rotate dates by 45 degree for visibility
    plt.ylabel('Predicted Stock Price (US $)')
    plt.xlabel('Date')
    plt.title('Predictions for %s' % self.stock_name)
    plt.show()
```

```
def get_input_data():

    stock_name = input(
        "Enter stock name(Eg.AMZN for amazon, MSFT for Microsoft): ")

    number_of_days_to_predict = input(
        "Enter number of days to predict(Number of days is after 2018-04-10):")

    return stock_name, number_of_days_to_predict


def predict_stocks(stock_name, number_of_days_to_predict):

    stock_object = Stocker(stock_name)

    stock_object.predict_future(days=int(number_of_days_to_predict))


stock_name, number_of_days_to_predict = get_input_data()
predict_stocks(stock_name, number_of_days_to_predict)
```

OUTPUT