Market Price Prediction

Abstract

Time series data is utilized to predict the future trend. Stock price technique use machine learning algorithms to develop a sustainable model and see in the future course of the data. This project aims to develop a stock price prediction machine learning model and then deploy. There are 2 stages for this project. First a machine learning model is created for the time series data extracted from Yahoo Finance. Next, a web app is developed locally using python's Flask library.

Keywords: Time Series, Machine Learning, LSTM, Web App, Python, Flask

Contents

Abstract 1
Introduction 3
    The Challenge 3
    Data 3
    LSTM (Long Short-Term Memory) 3
    Dependencies 4
Data Preparation and Model Building 5
Model Deployment 7
Conclusion 8
References 8
Appendix 9
    Model Code 9
    Flask Code 11
    HTML Form Code 13
    HTML Plot Code 14

Introduction

In this Project, I will be going through a five-step process to predict Market prices:

1. Getting real-time Market data.
2. Prepare data for training and testing.
3. Predict the price of a Market using LSTM neural network.
4. Visualize the prediction results.
5. Predict One Day in future
6. Deploy the model using Flask framework.

The Challenge

To forecast Market prices using all the trading features like price, volume, open, high, low values present in the dataset.

Data

I'm going to use YFinance to import the dataset I need. YFinance contains Real Time Market prices.

The main 5 features that I'm going to use in the prediction process and their details are as follows:

Close Price — It is the market close price for currency for that particular day.

High Price — It is highest price of currency for the day.

Low Price — It is the lowest price for currency for that day.

Open Price — It is market open price for currency for that day.

Volume — The volume of currency that is being in trade for that day.

LSTM (Long Short-Term Memory)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behaviour required in complex problem domains like machine translation, speech recognition, and more. LSTMs are a complex area of deep learning. This is important in my case because the previous price of a stock is crucial in predicting its future price. The main advantage is that since the model uses RNN, LSTM, Machine Learning and Deep Learning models the prediction of stock prices will be more accurate. And also, in the model it can predict the future 30 days Stock Prices and it can show it in a graph.

Dependencies

•Python 3.9

• Visualization libraries: Matplotlib

• Libraries for data and array: pandas and numpy

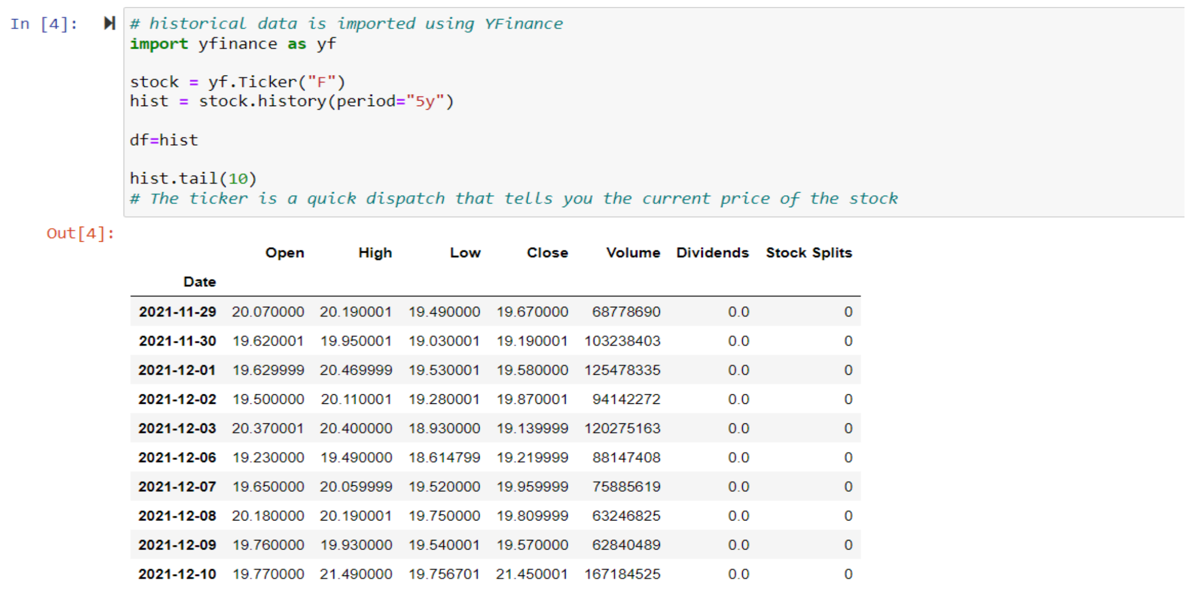
• Machine learning libraries:Tensorflow, Keras, Sciki-Learn

• Web App library: Flask

• Financial data parsing library: Yfinance

Data Preparation and Model Building

First step is to create an ML model for the time series data from historical stock price. Data extracted from Yahoo Finance using YFiance library. I started the project code by importing the necessary libraries. Next, the historical data is imported using YFinance. To parse the data, the stock name needs to be keyed in.



```
In [4]:
# historical data is imported using YFinance
import yfinance as yf

stock = yf.Ticker("F")
hist = stock.history(period="5y")

df=hist

hist.tail(10)
# The ticker is a quick dispatch that tells you the current price of the stock
```

Out[4]:

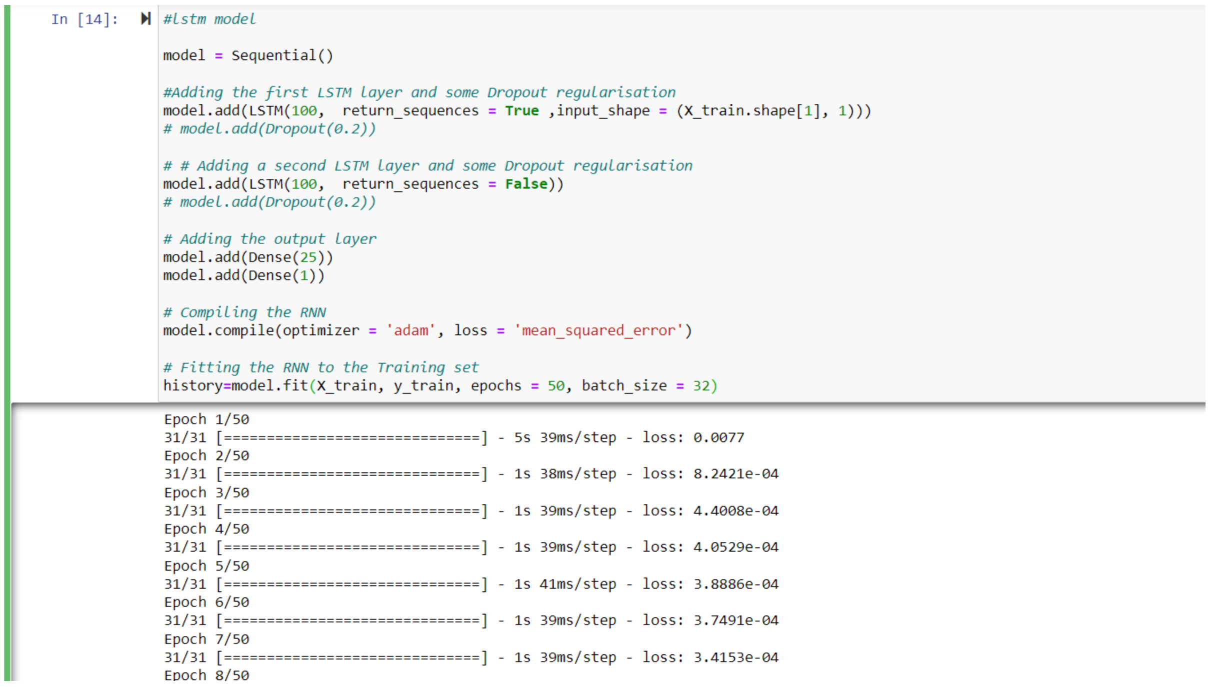
Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2021-11-29	20.070000	20.190001	19.490000	19.670000	68778690	0.0	0
2021-11-30	19.620001	19.950001	19.030001	19.190001	103238403	0.0	0
2021-12-01	19.629999	20.469999	19.530001	19.580000	125478335	0.0	0
2021-12-02	19.500000	20.110001	19.280001	19.870001	94142272	0.0	0
2021-12-03	20.370001	20.400000	18.930000	19.139999	120275163	0.0	0
2021-12-06	19.230000	19.490000	18.614799	19.219999	88147408	0.0	0
2021-12-07	19.650000	20.059999	19.520000	19.959999	75885619	0.0	0
2021-12-08	20.180000	20.190001	19.750000	19.809999	63246825	0.0	0
2021-12-09	19.760000	19.930000	19.540001	19.570000	62840489	0.0	0
2021-12-10	19.770000	21.490000	19.756701	21.450001	167184525	0.0	0

The next phase is the creation of training and test dataset. Training dataset is the first 80 percent of the total data and the remaining 20 percent will be predicted and will be used as test data.

The dataset is scaled and reshaped in the following step, Then the model is defined using several layers of LSTM from TensorFlow. TensorFlow is an open-source library for machine learning. It can handle sequence information using Recurrent Neural Network (RNN). TensorFlow has several recurrent layer types including SimpleRNN, Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM). This project utilizes LSTM feature.

While going through the RNN, some information may be lost due to data transformations. If the initial data is lost, the subsequent result will have no trace of the first inputs. This is true for human too. If we do not memorize a data, it will be lost from out memory after some time. To fix this issue, RNN can be deployed with cells having Long Short-Term Memory (LSTM) that can effectively have information of the previous data. So, while using LSTM, we do not need to bother about long term dependencies of the data. Then the model is defined.

When the model runs, it will provide the loss values after each epoch. As the model have more epochs, the loss value will drop. 2 LSTM layers are added with 100 units of cells. 2 units of dense layer are added at the bottom and the model is then compiled with ’Adam’ optimizer and ’mean_aquared_error’ loss.

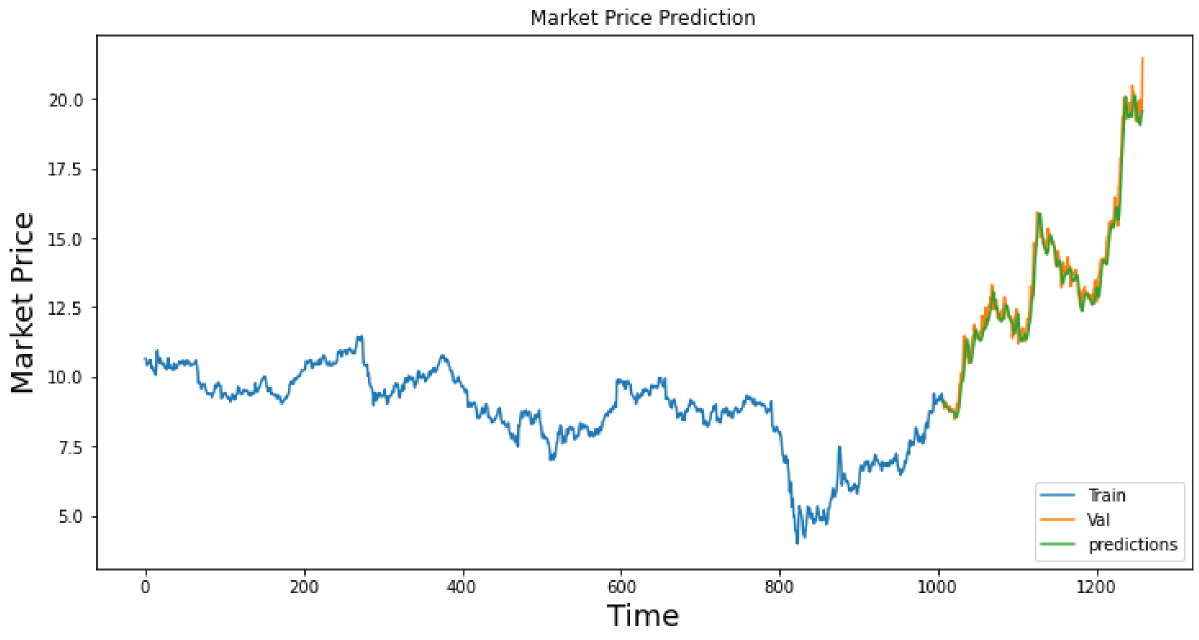


```
In [14]:   #lstm model

           model = Sequential()

           #Adding the first LSTM layer and some Dropout regularisation
           model.add(LSTM(100,  return_sequences = True ,input_shape = (X_train.shape[1], 1)))
           # model.add(Dropout(0.2))

           # # Adding a second LSTM layer and some Dropout regularisation
           model.add(LSTM(100,  return_sequences = False))
           # model.add(Dropout(0.2))

           # Adding the output layer
           model.add(Dense(25))
           model.add(Dense(1))

           # Compiling the RNN
           model.compile(optimizer = 'adam', loss = 'mean_squared_error')

           # Fitting the RNN to the Training set
           history=model.fit(X_train, y_train, epochs = 50, batch_size = 32)
```

```
Epoch 1/50
31/31 [==============================] - 5s 39ms/step - loss: 0.0077
Epoch 2/50
31/31 [==============================] - 1s 38ms/step - loss: 8.2421e-04
Epoch 3/50
31/31 [==============================] - 1s 39ms/step - loss: 4.4008e-04
Epoch 4/50
31/31 [==============================] - 1s 39ms/step - loss: 4.0529e-04
Epoch 5/50
31/31 [==============================] - 1s 41ms/step - loss: 3.8886e-04
Epoch 6/50
31/31 [==============================] - 1s 39ms/step - loss: 3.7491e-04
Epoch 7/50
31/31 [==============================] - 1s 39ms/step - loss: 3.4153e-04
Epoch 8/50
```

Once the model is defined, the test dataset is extracted out from the historical data and from there we need to take the first 30 days as input to the model (of course this number can be defined by the user but here for the sake of understanding, the number 30 is taken to predict the 31st point).The test data is reshaped afterwards.

The final step of part 1 is to predict the test data set and plot along with the actual dataset Predicted stock price plotted with the actual price.



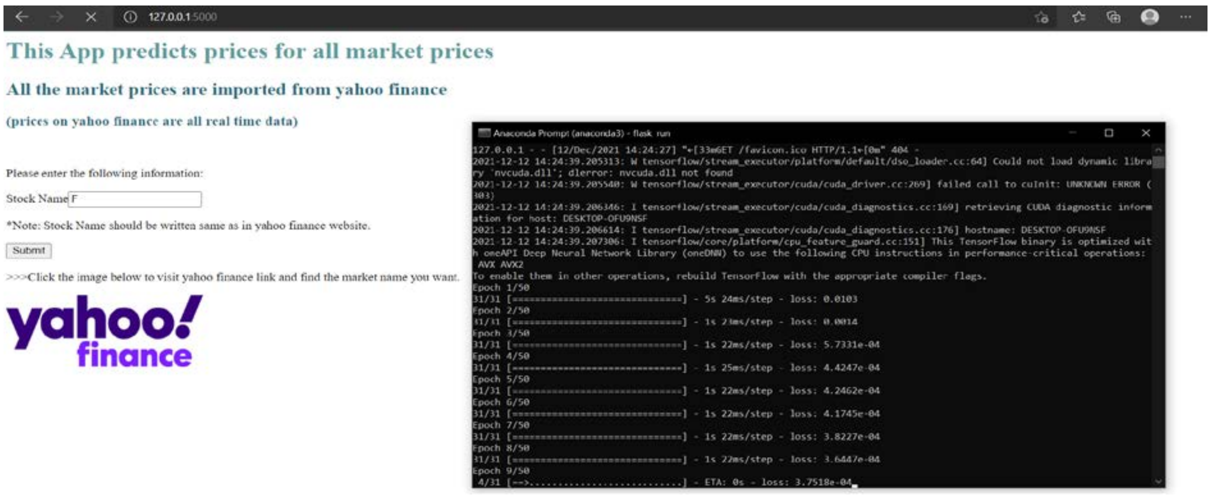
Model Deployment

In Part 2, the local hosting of the web app will be implemented. To use the local server, python's flask library will be utilized in this project because of its ease of deployment. The first step in Part 2 is to consolidate all the project files inside a project folder. The project folder contains the main project file in (.py) and the html files are saved inside 'templates' folder. That's all to run the app in localhost.

Once the project folder is setup, the command line is used to initiate the web app. First a separate python environment is created for flask and later it is activated using 'Conda'. 'Conda' is a python library and environment manager. Then the current directory is changed to the project folder using 'cd'. Afterwards the (.py) file set as the running script for the web app.

Now the .html files need to set for proper user input. I have created two html files. One named 'form.html' which takes the user input for the machine learning model training and the other one is 'plot.html' which delivers the predicted plot. Those two files are placed inside the templates folder. At the final step, 'flask run' command is executed to deploy the app on local server. For this app I am basically trying to predict the future stock trend of any market.

Stock Name is the CAPITALIZED letters for the stock. AAPL for Apple, BTC-USD for Bitcoin, TSLA for Tesla, etc.. While the machine is learning on the training data, the command line will show the number of epochs. Higher number of epochs will take longer time to complete the training. When completed, the app will be redirected to the plot.html and will deliver the predicted price along with the actual price of the stock.



This App predicts prices for all market prices

All the market prices are imported from yahoo finance

(prices on yahoo finance are all real time data)

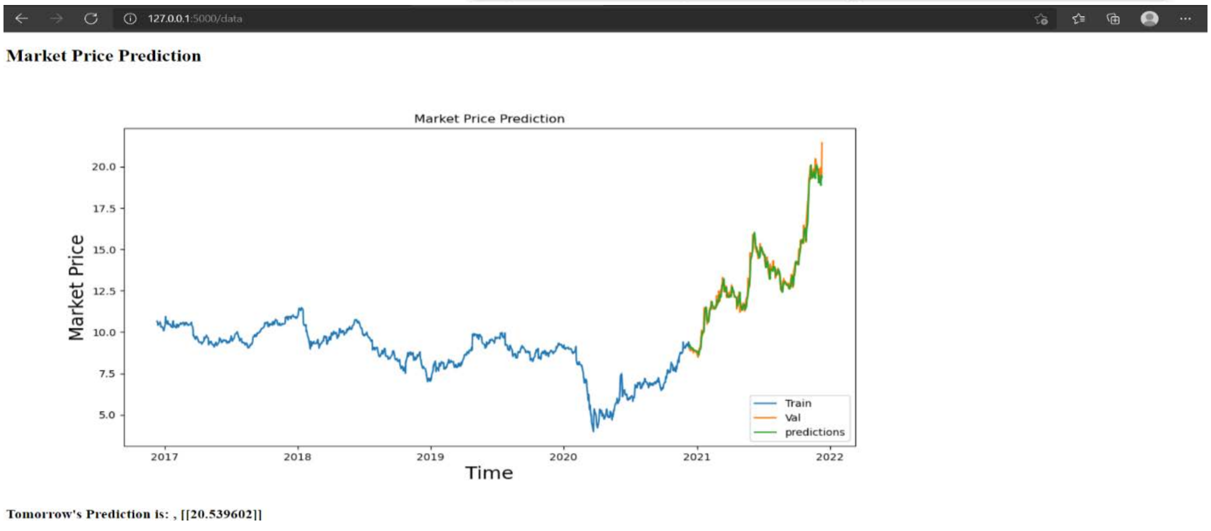
Please enter the following information:

Stock Name F

*Note: Stock Name should be written same as in yahoo finance website.

>>>Click the image below to visit yahoo finance link and find the market name you want.

```
127.0.0.1 - - [12/Dec/2021 14:24:27] "←[33mGET /favicon.ico HTTP/1.1←[0m" 404 -
2021-12-12 14:24:39.205313: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic libra
ry 'nvcuda.dll'; dlerror: nvcuda.dll not found
2021-12-12 14:24:39.205540: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (
303)
2021-12-12 14:24:39.206346: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic inform
ation for host: DESKTOP-OFU9NSF
2021-12-12 14:24:39.206614: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-OFU9NSF
2021-12-12 14:24:39.207306: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized wit
h oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:
 AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 1/50
31/31 [==============================] - 5s 24ms/step - loss: 0.0103
Epoch 2/50
31/31 [==============================] - 1s 23ms/step - loss: 0.0014
Epoch 3/50
31/31 [==============================] - 1s 22ms/step - loss: 5.7331e-04
Epoch 4/50
31/31 [==============================] - 1s 25ms/step - loss: 4.4247e-04
Epoch 5/50
31/31 [==============================] - 1s 22ms/step - loss: 4.2462e-04
Epoch 6/50
31/31 [==============================] - 1s 22ms/step - loss: 4.1745e-04
Epoch 7/50
31/31 [==============================] - 1s 22ms/step - loss: 3.8227e-04
Epoch 8/50
31/31 [==============================] - 1s 22ms/step - loss: 3.6447e-04
Epoch 9/50
 4/31 [==>...........................] - ETA: 0s - loss: 3.7518e-04
```



Market Price Prediction

Tomorrow's Prediction is: , [[20.539602]]

Conclusion

This work demonstrates the implementation of a stock price prediction model using machine learning libraries. The model seems reliably following the actual price of stocks for tomorrow’s price. 2 layers of LSTM has been used in the model. The app is deployed on the local server which provides the flexibility to input any stock’s name. This web app has shown the capability to get trained on any time series data and successfully predicted the stock price.

References

https://www.youtube.com/watch?v=H6du_pfuznE&ab_channel=KrishNaik

https://www.youtube.com/watch?v=QIUxPv5PJOY&ab_channel=ComputerScience

https://medium.com/codex/stock-predication-using-regression-algorithm-in-python-fb8b426453b9

https://www.analyticsvidhya.com/blog/2021/05/stock-price-prediction-and-forecasting-using-stacked-lstm/

https://www.analyticsvidhya.com/blog/2021/10/machine-learning-for-stock-market-prediction-with-step-by-step-implementation/

Appendix

Model Code

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
import progressbar
import matplotlib.pyplot as plt
print(tf.__version__)

import math
import matplotlib.pyplot as plt
import keras
import pandas as pd
import numpy as np
from array import array
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.layers import *
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from keras.callbacks import EarlyStopping

import yfinance as yf

stock = yf.Ticker("F")
hist = stock.history(period="5y")

df=hist

hist.tail(10)

print('Is there any missing value? ', df3.isnull().values.any())
print('How many missing values? ', df3.isnull().values.sum())
df3.dropna(inplace=True)
print('Number of participants after eliminating missing values: ',
len(df3))

plt.figure(figsize=(10,6))
plt.title('close price history',fontsize=24)
plt.plot(df3['Close'])
plt.xlabel('Date',fontsize=18)
plt.ylabel('close price',fontsize=18)
plt.show()

d=30
n=int(hist.shape[0]*0.8)
data = df3.filter(['Close'])
dataset = data.values

sc = MinMaxScaler(feature_range = (0, 1))
scaled_data = sc.fit_transform(dataset)
```

```
scaled_data

train_data = scaled_data[0:n,:]

X_train = []
y_train = []
for i in range(d, len(train_data)):
    X_train.append(train_data[i-d:i, 0])
    y_train.append(train_data[i, 0])
    if i<d:
        print(X_train)
        print(y_train)
        print()

X_train, y_train = np.array(X_train), np.array(y_train)
X_train.shape
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_train.shape

model = Sequential()

model.add(LSTM(100,  return_sequences = True ,input_shape =
(X_train.shape[1], 1)))
# model.add(Dropout(0.2))

model.add(LSTM(100,  return_sequences = False))
# model.add(Dropout(0.2))

model.add(Dense(25))
model.add(Dense(1))

model.compile(optimizer = 'adam', loss = 'mean_squared_error')

history=model.fit(X_train, y_train, epochs = 50, batch_size = 32)

model.summary()

test_data = scaled_data[n-d:,:]

X_test = []
y_test = dataset[n:,:]
for i in range(d, len(test_data)):
    X_test.append(test_data[i-d:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
print(X_test.shape)

predictions = model.predict(X_test)
predictions = sc.inverse_transform(predictions)

rmse=np.sqrt(np.mean(predictions - y_test)**2)
rmse

plt.figure(figsize=(10,6))
plt.title('close price history',fontsize=24)
plt.plot(df3['Close'])
plt.xlabel('Date',fontsize=18)
plt.ylabel('close price',fontsize=18)
plt.show()

train = data[:n]
```

```
valid = data[n:]
valid['predictions'] = predictions
plt.figure(figsize=(12,6))
plt.title('Market Price Prediction')
plt.xlabel('Time', fontsize = 18)
plt.ylabel('Market Price', fontsize = 18)
plt.plot(train['Close'])
plt.plot(valid[['Close','predictions']])
plt.legend(['Train','Val','predictions'],loc = 'lower right')
plt.show()

plt.plot(history.history['loss'])
plt.show()

new_stock = yf.Ticker("F")
new_hist = stock.history(period="5y")

new_df=new_hist
hist.tail(10)

new_df2 = new_df.filter(['Close'])

last_30_days = new_df2[-d:].values

last_30_days_scaled = sc.transform(last_30_days)

x_test = []

x_test.append(last_30_days_scaled)

x_test = np.array(x_test)

x_test = np.reshape(x_test,(x_test.shape[0],x_test.shape[1],1))

predictions_price = model.predict(x_test)

predictions_price = sc.inverse_transform(predictions_price)

print(predictions_price)
```

Flask Code

```
from flask import Flask ,render_template, request      # , send_file
from io import BytesIO
import base64
import tensorflow as tf
from tensorflow import keras
import progressbar
import matplotlib.pyplot as plt
import math
import keras
import pandas as pd
import numpy as np
from array import array
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.layers import *
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from keras.callbacks import EarlyStopping
import yfinance as yf

app = Flask(__name__)


@app.route('/')
def form():
    return render_template('form.html')


@app.route('/data', methods=['POST'])
def hello():
    stock_name = request.form['Name']
    stock = yf.Ticker(stock_name)

    hist = stock.history(period="5y")
    df3 = hist
    d = 30
    n = int(hist.shape[0] * 0.8)
    data = df3.filter(['Close'])
    dataset = data.values

    sc = MinMaxScaler(feature_range=(0, 1))
    scaled_data = sc.fit_transform(dataset)

    train_data = scaled_data[0:n, :]

    X_train = []
    y_train = []
    for i in range(d, len(train_data)):
        X_train.append(train_data[i - d:i, 0])
        y_train.append(train_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 = Sequential()
    model.add(LSTM(50, return_sequences=True,
input_shape=(X_train.shape[1], 1)))
    # model.add(Dropout(0.2))
    # # Adding a second LSTM layer and some Dropout regularisation
    model.add(LSTM(50, return_sequences=False))
    # model.add(Dropout(0.2))
    # Adding the output layer
    model.add(Dense(25))
    model.add(Dense(1))
    # Compiling the RNN
    model.compile(optimizer='adam', loss='mean_squared_error')
    # Fitting the RNN to the Training set
    history = model.fit(X_train, y_train, epochs=50, batch_size=32)

    test_data = scaled_data[n - d:, :]
    X_test = []
    y_test = dataset[n:, :]
    for i in range(d, len(test_data)):
        X_test.append(test_data[i - d:i, 0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

    predictions = model.predict(X_test)
    predictions = sc.inverse_transform(predictions)

    train = data[:n]
    valid = data[n:]
    valid['predictions'] = predictions
    plt.figure(figsize=(12, 6))
    plt.title('Market Price Prediction')
    plt.xlabel('Time', fontsize=18)
    plt.ylabel('Market Price', fontsize=18)
    plt.plot(train['Close'])
    plt.plot(valid[['Close', 'predictions']])
    plt.legend(['Train', 'Val', 'predictions'], loc='lower right')
    STOCK = BytesIO()
    plt.savefig(STOCK, format="png")

    new_stock = stock
    new_hist = hist

    new_df = hist

    new_df2 = new_df.filter(['Close'])

    last_30_days = new_df2[-d:].values

    last_30_days_scaled = sc.transform(last_30_days)

    x_test = []

    x_test.append(last_30_days_scaled)

    x_test = np.array(x_test)

    x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

    predictions_price = model.predict(x_test)

    predictions_price = sc.inverse_transform(predictions_price)

    STOCK.seek(0)
    plot_url = base64.b64encode(STOCK.getvalue()).decode('utf8')
    return render_template("plot.html", plot_url=plot_url,
key=predictions_price)
```

HTML Form Code

```
<h1 style="color: #5e9ca0;">This App predicts prices for all market
prices</h1>
<h2 style="color: #2e6c80;">All the market prices are imported from yahoo
finance</h2>
<h3 style="color: #2e6c80;">(prices on yahoo finance are all real time
data)</h3>
<ul style="list-style-type: square;">
</ul>
<p>&nbsp;</p>

<form action="/data" method = "POST">
```

```
    <p>Please enter the following information:</p>
    <p>Stock Name<input type = "text" name = "Name" /></p>
    <p>*Note: Stock Name should be written same as in yahoo finance
website.</p>
    <p><input type = "submit" value = "Submit" /></p>
<p>>>>>Click the image below to visit yahoo finance link and find the market
name you want.</p>
<a href="https://finance.yahoo.com/calendar/">
<img
src="https://www.researchamerica.org/sites/default/files/logo_library/media
/yahoofinancelogo.jpg" alt="Yahoo Finance"
style="width:250px;height:100px;">
</a>


</form>
```

HTML Plot Code

```
<!doctype html>
<title>Market Price Prediction</title>
<section>
  <h2>Market Price Prediction</h2>
  <img src="data:image/png;base64, {{ plot_url }}">
  <h3> Tomorrow's Prediction is: , {{ key }}  </h3>
</section>
```