EXPENSE TRACKER

Submitted by

ANUSHA PATRA[RA2111027010022]
ABHIGNYA PRIYADARSHINI[RA2111027010067]

Under the Guidance of

Dr. E SASIKALA

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

with specialization in Big Data Analytics

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

OCTOBER 2023

COLLEGE OF ENGINEERING & TECHNOLOGY

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

S.R.M. NAGAR, KATTANKULATHUE – 603 203

Chengalpattu District

BONAFIDE CERTIFICATE

Register No. **RA2111027010022** Certified to be the bonafide work done by **ANUSHA PATRA** of II Year/IV Sem B.Tech Degree Course in the **Practical Machine Learning 18CSE392T** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,** Kattankulathur during the academic year 2023 – 2024.

LAB INCHARGE

E Sasikala

Associate Professor

Department of DSBS

SRMIST – KTR.

Head of the Department

TABLE OF CONTENTS

ABSTRACT

In this machine learning project, we investigate stock market prediction focusing on the stock prices of Google and Tesla, two prominent technology companies. Through the utilization of historical stock price data, financial indicators, and sentiment analysis from news and social media sources, our primary goal is to develop and evaluate predictive models that can offer insights into the future trends of Google and Tesla stocks.

The project encompasses comprehensive data collection, including historical stock prices and key financial metrics, along with the analysis of market sentiment from news articles and social media content. Features are thoughtfully engineered from this data, encompassing technical indicators, fundamental metrics, and sentiment scores. Multiple machine learning models, including regression and time series forecasting, are evaluated for their accuracy in predicting stock prices. Evaluation metrics like R2 and MSE are used to gauge model performance, and trading simulations are conducted to assess the practical profitability of the predictions.

Ultimately, this project seeks to reveal the potential of machine learning in stock market prediction and offer insights into the specific dynamics of Google and Tesla stock prices.

PROBLEM STATEMENT

Stock market prediction has long been a challenging and financially significant endeavor, with potential applications spanning from investment decisions to risk management. In this context, we address the problem of accurately forecasting the stock prices of two major technology companies, Google and Tesla, using machine learning techniques.

This problem statement entails the following key challenges:

Data Complexity: Historical stock price data, financial indicators, and sentiment analysis from diverse sources introduce complexity. The challenge is to handle, preprocess, and extract meaningful information from this diverse data while dealing with inconsistencies and outliers.

Model Selection: Given the multitude of available machine learning algorithms, the problem is to determine the most suitable models for predicting Google and Tesla stock prices accurately. These models should account for the unique dynamics of each company's stock.

Evaluation and Real-World Applicability: Developing predictive models is not sufficient; the challenge lies in evaluating their performance rigorously using appropriate metrics. Additionally, it is crucial to assess the practical applicability of these models in real-world investment scenarios.

Market Dynamics and Sentiment Analysis: Capturing the dynamic and ever-changing nature of stock markets, as well as the influence of market sentiment from news and social media, is a substantial challenge. Developing models that adapt to these dynamics is essential.

This project aims to address these challenges, providing valuable insights into the potential of machine learning in stock market prediction, ultimately assisting stakeholders in making more informed financial decisions regarding Google and Tesla stocks.

IDENTIFYING REQUIREMENTS

To successfully implement a machine learning project for stock market prediction, particularly for Google and Tesla, a set of clear requirements must be established. These requirements encompass data, technology, and project management aspects:

Data Requirements:
a. Historical Stock Price Data: Access to historical daily, weekly, or intra-day stock price data for Google and Tesla, covering a significant time period.
b. Financial Indicators: Comprehensive financial metrics, including trading volume, moving averages, price-to-earnings ratios, and dividend yields.
c. News and Social Media Data: A substantial corpus of news articles and social media content, with relevant timestamps, to conduct sentiment analysis.
d. Feature Engineering Data: Access to additional datasets for feature engineering, such as economic indicators, market indices, and other related financial data.

Data Preprocessing Requirements:
a. Data Cleaning: Robust data cleaning processes to handle missing values, outliers, and data inconsistencies.
b. Feature Engineering: Implementation of feature engineering techniques to create relevant features for the machine learning models.

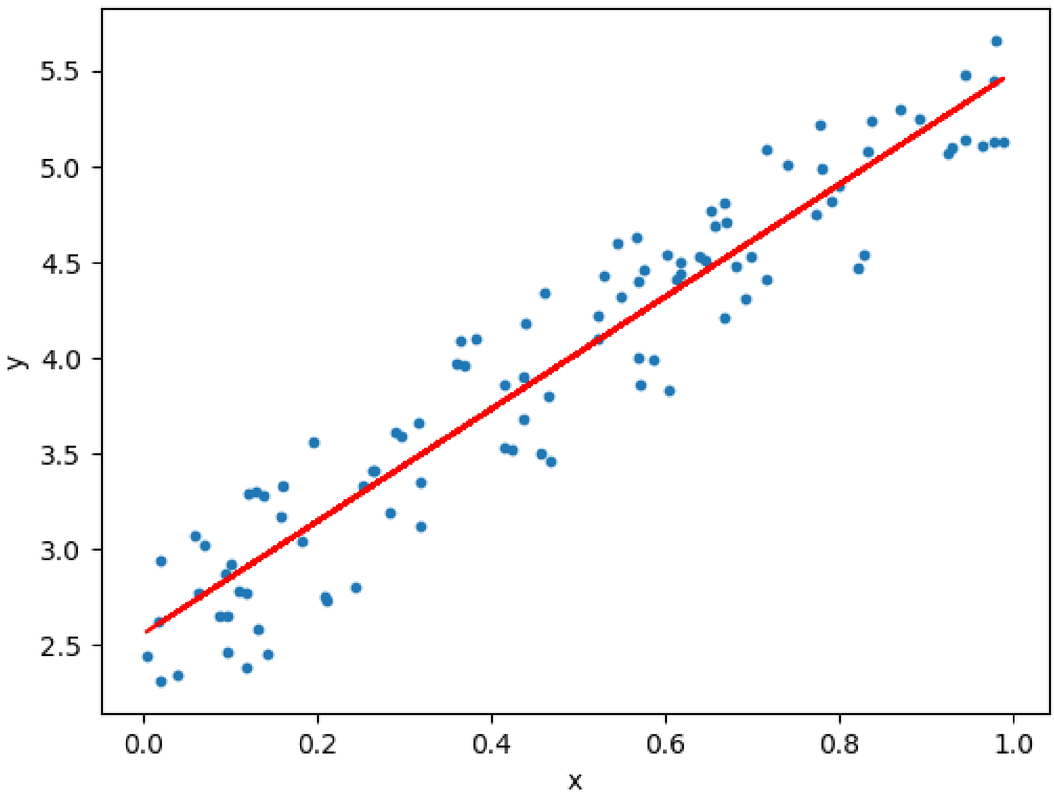
Machine Learning Models:
a. Algorithm Selection: Evaluation and selection of machine learning algorithms, including regression models, time series forecasting methods, and deep learning approaches.
b. Model Training: Development and training of the chosen models using the historical data.
c. Hyperparameter Tuning: Fine-tuning model hyperparameters to optimize their performance.

Evaluation Metrics:
a. Metric Selection: Define appropriate evaluation metrics, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and others, to assess model performance.
b. Cross-Validation: Implement cross-validation techniques to ensure robust model evaluation.

ALGORITHM USED

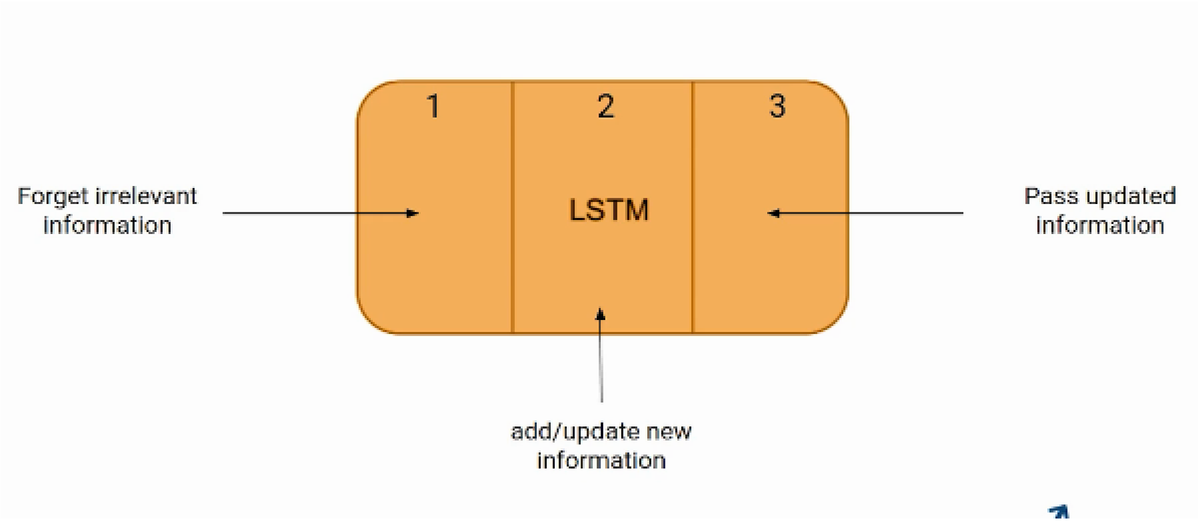
Linear Regression

Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between a dependent variable and one or more independent features. When the number of the independent feature, is 1 then it is known as Univariate Linear regression, and in the case of more than one feature, it is known as multivariate linear regression. The goal of the algorithm is to find the best linear equation that can predict the value of the dependent variable based on the independent variables. The equation provides a straight line that represents the relationship between the dependent and independent variables. The slope of the line indicates how much the dependent variable changes for a unit change in the independent variable



LSTM: LSTM (Long Short-Term Memory) is a recurrent neural network (RNN) architecture widely used in Deep Learning. It excels at capturing long-term dependencies, making it ideal for sequence prediction tasks.

Unlike traditional neural networks, LSTM incorporates feedback connections, allowing it to process entire sequences of data, not just individual data points. This makes it highly effective in understanding and predicting patterns in sequential data like time series, text, and speech.



PERFORMANCE METRICS USED

Mean Square Error:

In statistics, the mean squared error (MSE) is defined as the mean or average of the squared differences between the actual and estimated values. Mean Squared Error (MSE) measures the amount of error in a statistical model. Evaluate the mean squared difference between observed and predicted values. If the model has no errors, the MSE is zero. Its value increases as the model error increases. The mean squared error is also known as the mean squared deviation (MSD). For example, in regression, the mean squared error represents the mean squared residual.

R2 Error:

R squared (R2) is a regression error metric that justifies the performance of the model. It represents the value of how much the independent variables are able to describe the value for the response/target variable.

Thus, an R-squared model describes how well the target variable is explained by the combination of the independent variables as a single unit.

The R squared value ranges between 0 to 1 and is represented by the below formula:

R2= 1- SSres / SStot

Here,

SSres: The sum of squares of the residual errors.
SStot: It represents the total sum of the errors.

CODE

STOCK MARKET PREDICTION OF TESLA:

In [1]:
```
#RA2111027010022(Anusha Patra)
#RA2111027010067(Abhignya Priyadarshini)

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import chart_studio.plotly as py
import plotly.graph_objs as go
from plotly.offline import plot

#for offline plotting
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
```

In [2]:
```
tesla = pd.read_csv(r"C:\Users\anush\OneDrive\Desktop\ML Project\tesla.csv")
tesla.head()
```

Out[2]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	29-06-2010	19.000000	25.00	17.540001	23.889999	23.889999	18766300
1	30-06-2010	25.790001	30.42	23.299999	23.830000	23.830000	17187100
2	01-07-2010	25.000000	25.92	20.270000	21.959999	21.959999	8218800
3	02-07-2010	23.000000	23.10	18.709999	19.200001	19.200001	5139800
4	06-07-2010	20.000000	20.00	15.830000	16.110001	16.110001	6866900

In [3]:
```
tesla.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2193 entries, 0 to 2192
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       2193 non-null   object 
 1   Open       2193 non-null   float64
 2   High       2193 non-null   float64
 3   Low        2193 non-null   float64
 4   Close      2193 non-null   float64
 5   Adj Close  2193 non-null   float64
 6   Volume     2193 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 120.1+ KB
```

In [4]:
```
tesla['Date'] = pd.to_datetime(tesla['Date'], format='%d-%m-%Y')

# Now you can calculate the minimum and maximum dates and the total number of days
min_date = tesla['Date'].min()
max_date = tesla['Date'].max()
total_days = (max_date - min_date).days

print(f'Dataframe contains stock prices between {min_date} and {max_date}')
print(f'Total days = {total_days} days')
```

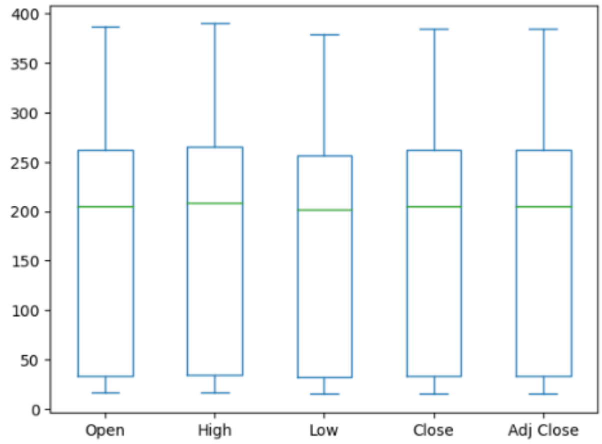
```
Dataframe contains stock prices between 2010-06-29 00:00:00 and 2019-03-15 00:00:00
Total days = 3181 days
```

In [5]:
```
tesla.describe()
```

Out[5]:

	Open	High	Low	Close	Adj Close	Volume
count	2193.000000	2193.000000	2193.000000	2193.000000	2193.000000	2.193000e+03
mean	175.652882	178.710262	172.412075	175.648555	175.648555	5.077449e+06
std	115.580903	117.370092	113.654794	115.580771	115.580771	4.545398e+06
min	16.139999	16.629999	14.980000	15.800000	15.800000	1.185000e+05
25%	33.110001	33.910000	32.459999	33.160000	33.160000	1.577800e+06
50%	204.990005	208.160004	201.669998	204.990005	204.990005	4.171700e+06
75%	262.000000	265.329987	256.209991	261.739990	261.739990	6.885600e+06
max	386.690002	389.609985	379.350006	385.000000	385.000000	3.716390e+07

In [6]:
```
tesla[['Open','High','Low','Close','Adj Close']].plot(kind='box')
```
Out[6]: <AxesSubplot:>



In [7]:
```
# Setting the layout for our plot
layout = go.Layout(
    title='Stock Prices of Tesla',
    xaxis=dict(
        title='Date',
        titlefont=dict(
            family='Courier New, monospace',
            size=18,
            color='#7f7f7f'
        )
    ),
    yaxis=dict(
        title='Price',
        titlefont=dict(
            family='Courier New, monospace',
            size=18,
            color='#7f7f7f'
        )
    )
)

tesla_data = [{'x':tesla['Date'], 'y':tesla['Close']}]
plot = go.Figure(data=tesla_data, layout=layout)
```

```
In [8]: #plot(plot) #plotting offline
        iplot(plot)
```


```
In [17]: # Building the regression model
         from sklearn.model_selection import train_test_split

         #For preprocessing
         from sklearn.preprocessing import MinMaxScaler
         from sklearn.preprocessing import StandardScaler

         #For model evaluation
         from sklearn.metrics import mean_squared_error as mse
         from sklearn.metrics import r2_score
```

```
In [18]: #Split the data into train and test sets
         X = np.array(tesla.index).reshape(-1,1)
         Y = tesla['Close']
         X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=101)
```

```
In [19]: # Feature scaling
         scaler = StandardScaler().fit(X_train)
```

```
In [20]: from sklearn.linear_model import LinearRegression
```

```
In [21]: #Creating a linear model
         lm = LinearRegression()
         lm.fit(X_train, Y_train)
```

```
Out[21]: ▾ LinearRegression
         LinearRegression()
```

```
In [22]: #Plot actual and predicted values for train dataset
         trace0 = go.Scatter(
             x = X_train.T[0],
             y = Y_train,
             mode = 'markers',
             name = 'Actual'
         )
         trace1 = go.Scatter(
             x = X_train.T[0],
             y = lm.predict(X_train).T,
             mode = 'lines',
             name = 'Predicted'
         )
         tesla_data = [trace0,trace1]
         layout.xaxis.title.text = 'Day'
         plot2 = go.Figure(data=tesla_data, layout=layout)
```

In [23]:

```
iplot(plot2)
```

In [24]:

```
#Calculate scores for model evaluation
scores = f'''
{'Metric'.ljust(10)}{'Train'.center(20)}{'Test'.center(20)}
{'r2_score'.ljust(10)}{r2_score(Y_train, lm.predict(X_train))}\t{r2_score(Y_test, lm.predict(X_test))}
{'MSE'.ljust(10)}{mse(Y_train, lm.predict(X_train))}\t{mse(Y_test, lm.predict(X_test))}
'''
print(scores)
```

```
Metric            Train               Test
r2_score  0.8658871776828707    0.8610649253244574
MSE       1821.3833862936174    1780.987539418845
```

STOCK MARKET PREDICTION OF GOOGLE:

In [19]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense,LSTM,Dropout
```

In [21]:
```
data = pd.read_csv(r"C:\Users\anush\OneDrive\Desktop\ML Project\Google_train_data.csv")
data.head()
```

Out[21]:

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800

In [23]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1258 entries, 0 to 1257
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    1258 non-null   object
 1   Open    1258 non-null   float64
 2   High    1258 non-null   float64
 3   Low     1258 non-null   float64
 4   Close   1258 non-null   object
 5   Volume  1258 non-null   object
dtypes: float64(3), object(3)
memory usage: 59.1+ KB
```

In [25]:
```
data["Close"]=pd.to_numeric(data.Close,errors='coerce')
data = data.dropna()
trainData = data.iloc[:,4:5].values
```

In [27]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1149 entries, 0 to 1257
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    1149 non-null   object
 1   Open    1149 non-null   float64
 2   High    1149 non-null   float64
 3   Low     1149 non-null   float64
 4   Close   1149 non-null   float64
 5   Volume  1149 non-null   object
dtypes: float64(4), object(2)
memory usage: 62.8+ KB
```

In [29]:
```
sc = MinMaxScaler(feature_range=(0,1))
trainData = sc.fit_transform(trainData)
trainData.shape
```

Out[29]: (1149, 1)

In [31]:
```
X_train = []
y_train = []

for i in range (60,1149): #60 : timestep // 1149 : length of the data
    X_train.append(trainData[i-60:i,0])
    y_train.append(trainData[i,0])

X_train,y_train = np.array(X_train),np.array(y_train)
```

In [33]:
```
X_train = np.reshape(X_train,(X_train.shape[0],X_train.shape[1],1)) #adding the batch_size axis
X_train.shape
```

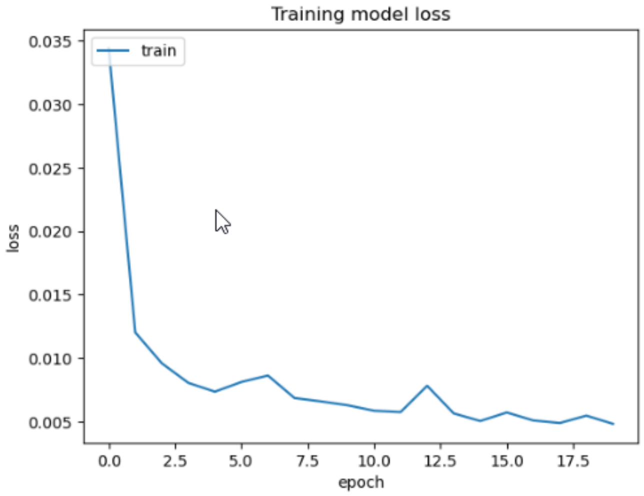
Out[33]: (1089, 60, 1)

In [35]:
```
model = Sequential()

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

model.add(LSTM(units=100, return_sequences = True))
model.add(Dropout(0.2))

model.add(LSTM(units=100, return_sequences = True))
model.add(Dropout(0.2))

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

model.add(Dense(units =1))
model.compile(optimizer='adam',loss="mean_squared_error")
```

In [37]:
```
hist = model.fit(X_train, y_train, epochs = 20, batch_size = 32, verbose=2)
```

```
Epoch 1/20
35/35 - 13s - loss: 0.0344 - 13s/epoch - 359ms/step
Epoch 2/20
35/35 - 8s - loss: 0.0120 - 8s/epoch - 215ms/step
Epoch 3/20
35/35 - 8s - loss: 0.0096 - 8s/epoch - 217ms/step
Epoch 4/20
35/35 - 8s - loss: 0.0081 - 8s/epoch - 216ms/step
Epoch 5/20
35/35 - 8s - loss: 0.0074 - 8s/epoch - 216ms/step
Epoch 6/20
35/35 - 8s - loss: 0.0081 - 8s/epoch - 216ms/step
Epoch 7/20
35/35 - 8s - loss: 0.0086 - 8s/epoch - 216ms/step
Epoch 8/20
35/35 - 8s - loss: 0.0069 - 8s/epoch - 216ms/step
Epoch 9/20
35/35 - 8s - loss: 0.0066 - 8s/epoch - 215ms/step
Epoch 10/20
35/35 - 8s - loss: 0.0063 - 8s/epoch - 217ms/step
Epoch 11/20
35/35 - 8s - loss: 0.0059 - 8s/epoch - 217ms/step
Epoch 12/20
35/35 - 8s - loss: 0.0058 - 8s/epoch - 217ms/step
Epoch 13/20
35/35 - 8s - loss: 0.0078 - 8s/epoch - 215ms/step
Epoch 14/20
35/35 - 8s - loss: 0.0056 - 8s/epoch - 217ms/step
Epoch 15/20
35/35 - 8s - loss: 0.0050 - 8s/epoch - 215ms/step
Epoch 16/20
35/35 - 8s - loss: 0.0057 - 8s/epoch - 216ms/step
Epoch 17/20
35/35 - 8s - loss: 0.0051 - 8s/epoch - 215ms/step
Epoch 18/20
35/35 - 8s - loss: 0.0049 - 8s/epoch - 217ms/step
Epoch 19/20
35/35 - 8s - loss: 0.0055 - 8s/epoch - 216ms/step
Epoch 20/20
35/35 - 8s - loss: 0.0048 - 8s/epoch - 216ms/step
```

In [39]:
```
plt.plot(hist.history['loss'])
plt.title('Training model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train'], loc='upper left')
plt.show()
```



In [41]:
```
testData = pd.read_csv(r"C:\Users\anush\OneDrive\Desktop\ML Project\Google_test_data.csv")
testData["Close"]=pd.to_numeric(testData.Close,errors='coerce')
testData = testData.dropna()
testData = testData.iloc[:,4:5]
y_test = testData.iloc[60:,0:].values

#input array for the model
inputClosing = testData.iloc[:,0:].values
inputClosing_scaled = sc.transform(inputClosing)
inputClosing_scaled.shape
X_test = []
length = len(testData)
timestep = 60
for i in range(timestep,length):
    X_test.append(inputClosing_scaled[i-timestep:i,0])
X_test = np.array(X_test)
X_test = np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
X_test.shape
```

Out[41]:
```
(192, 60, 1)
```

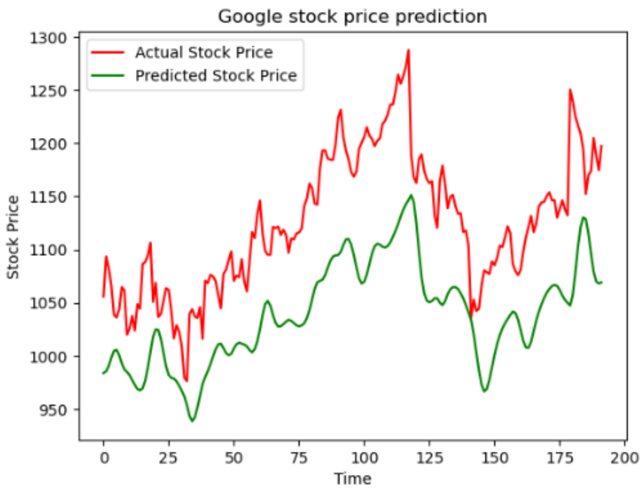
In [61]:
```
y_pred = model.predict(X_test)
y_pred
```

```
6/6 [==============================] - 1s 83ms/step
```

Out[61]:
```
array([[1.1430609],
       [1.1469665],
       [1.1587611],
       [1.176418 ],
       [1.191589 ],
       [1.193861 ],
       [1.1824082],
       [1.1643461],
       [1.1509361],
       [1.1456653],
       [1.138519 ],
       [1.1276344],
       [1.1175207],
       [1.1082188],
       [1.1056073],
       [1.1097211],
       [1.1269869],
```

In [62]:
```
predicted_price = sc.inverse_transform(y_pred)
```

In [63]:
```
plt.plot(y_test, color = 'red', label = 'Actual Stock Price')
plt.plot(predicted_price, color = 'green', label = 'Predicted Stock Price')
plt.title('Google stock price prediction')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```

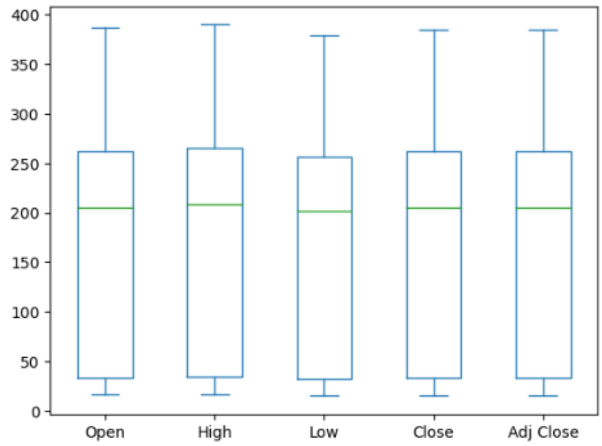


OUTPUT

Stock Market Prediction for Tesla:

In [6]:
```
tesla[['Open','High','Low','Close','Adj Close']].plot(kind='box')
```

Out[6]: <AxesSubplot:>



In [8]:
```
#plot(plot) #plotting offline
iplot(plot)
```

In [23]:
```
iplot(plot2)
```

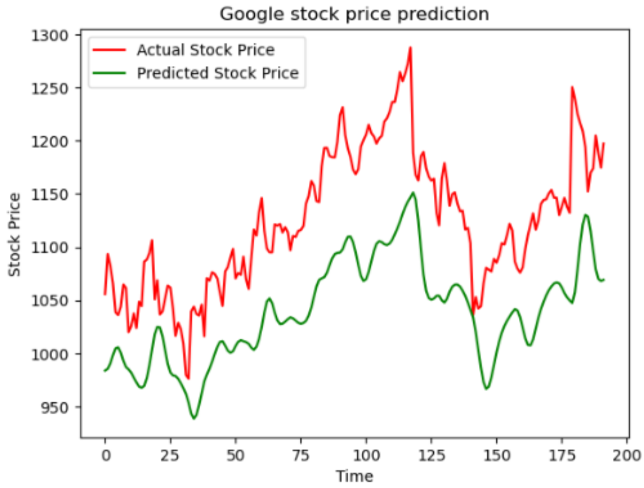
MSE and R2 Scores:

In [24]:
```
#Calculate scores for model evaluation
scores = f'''
{'Metric'.ljust(10)}{'Train'.center(20)}{'Test'.center(20)}
{'r2_score'.ljust(10)}{r2_score(Y_train, lm.predict(X_train))}\t{r2_score(Y_test, lm.predict(X_test))}
{'MSE'.ljust(10)}{mse(Y_train, lm.predict(X_train))}\t{mse(Y_test, lm.predict(X_test))}
'''
print(scores)
```

```
Metric           Train                Test
r2_score  0.8658871776828707    0.8610649253244574
MSE       1821.3833862936174    1780.987539418845
```

Stock Market Prediction for Google:

In [63]:
```
plt.plot(y_test, color = 'red', label = 'Actual Stock Price')
plt.plot(predicted_price, color = 'green', label = 'Predicted Stock Price')
plt.title('Google stock price prediction')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```



FUTURE ENHANCEMENTS

Continuing to improve and expand a machine learning project for stock market prediction is essential to stay relevant and adapt to evolving market dynamics. Here are some potential future enhancements:

Incorporate Alternative Data Sources:

Include alternative data sources such as satellite imagery, weather data, and consumer sentiment indices to provide a more comprehensive view of the factors influencing stock prices.

Ensemble Learning:

Implement ensemble learning techniques that combine predictions from multiple models to enhance accuracy and robustness.

Reinforcement Learning for Trading Strategies:

Develop and integrate reinforcement learning algorithms to automate and optimize trading strategies based on model predictions.

Real-time Data Streaming:

Transition to real-time data streaming and analysis to make predictions and decisions on the most up-to-date information.

Natural Language Processing (NLP) Advancements:

Improve sentiment analysis with advanced NLP techniques, including sentiment-specific word embeddings and context-aware sentiment analysis.

Deep Reinforcement Learning:

Explore deep reinforcement learning for portfolio management and risk assessment, considering multiple assets and their interdependencies.

Interpretability and Explainability:

Enhance model interpretability and explainability to make predictions more transparent and understandable to users.

Automated Hyperparameter Tuning:

Implement automated hyperparameter tuning methods like Bayesian optimization to optimize model performance efficiently.

Market Microstructure Analysis:

Incorporate market microstructure analysis to understand the impact of order book dynamics, liquidity, and trading activity on stock prices.

CONCLUSION

The stock market prediction project focusing on Google and Tesla, empowered by machine learning techniques and data-driven insights, represents a significant step towards understanding and harnessing the dynamic world of financial markets. This project has strived to provide a robust framework for predicting the stock prices of these technology giants, offering valuable contributions to investors, traders, and financial analysts.

This project's findings underscore the importance of adapting to market dynamics and sentiment analysis derived from news and social media content. Such insights have the potential to guide investment decisions and risk management strategies effectively. Moreover, the research has highlighted the role of ethical considerations in machine learning projects involving financial data, ensuring fairness and compliance with industry regulations.

As we conclude this project, it's essential to recognize that the ever-evolving nature of financial markets necessitates continuous improvement and adaptation. Future enhancements, including the incorporation of alternative data sources, advanced NLP techniques, and real-time analysis, will further enhance the accuracy and relevance of our predictions. The project stands as a testament to the power of machine learning in navigating the complexities of stock markets and serves as an open door to further innovations and insights in the field of stock market prediction.

REFERENCES

1. **https://scholarworks.lib.csusb.edu/cgi/viewcontent.cgi?article=1435&context=jitim**
2. **https://www.sciencedirect.com/science/article/pii/S1877050922021937**
3. **https://link.springer.com/chapter/10.1007/978-981-16-3915-9_10**
4. **https://www.ijeast.com/papers/258-262,Tesma508,IJEAST.pdf**