PROJECT DOCUMENT FOR STOCK PRICE PREDICTION

PROBLEM STATEMENT :

The problem at hand is to develop an accurate and robust model for predicting the future price of a specific stock or stock market index. The goal is to provide investors, traders, and financial analysts with reliable predictions that can inform their decision-making and potentially yield profitable investment strategies**.** The challenge is to develop a predictive model capable of accurately forecasting stock prices. This model should provide actionable insights to assist investors and traders in making informed decisions, thus enhancing their ability to manage risk and identify profitable investment opportunities.

Why it's Important: Accurate stock price predictions are valuable for investors, traders, and financial analysts to make informed decisions.

DESIGN THINKING PROCESS:

EMPATHIZE:

Begin by empathizing with the end users of stock price prediction. who are they? What are their goals and pain points? Conduct user interviews, surveys and feedback sessions to gain insights into information needs and decision-making processes.

DEFINE:

Define the specific problem you aim to solve. Is it short-term or long-term price prediction, portfolio optimization or risk management. Create a clear problem statement that aligns with the user’s needs and objectives.

IDEATE:

Potential Data sources, features, and machine learning , LSTM for the Stock price prediction encourage creative thinking and consider the alternative approaches to solve the defined problem.

PROTOTYPE:

Build a prototype or minimum viable product(MVP) of your stock price prediction model. This could involve selecting a dataset, features and implementing a basic model.

TEST AND REFINE:

Test the prototype of the dataset and refine the models as more data becomes available.

PHASES OF DEVELOPMENT:

Phase 1: Data collection

- **Data sources:** Describe where you obtained the historical stock price data. common source include financial API'S ,website or databases**.**
- **Data Frequency:** Specify the data frequency and time period covered by the dataset.

Phase 2: Data preprocessing

- **Data cleansing:** How the missing data are handled and outliers and any inconsistent in the dataset.
- **Feature Engineering:** Describe any new features created from the raw data.
- **Scaling and normalization:** for suitable modeling the data to make it scaled or normalized.

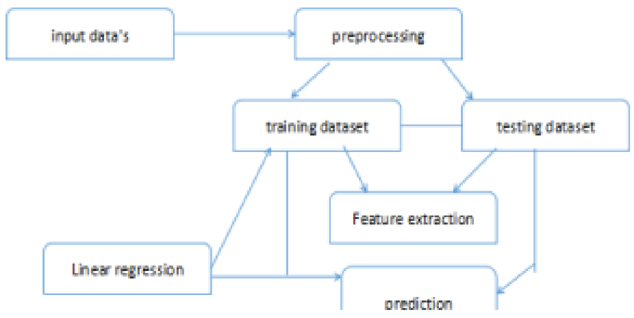
Phase 3: Model Training

- **Model selection:** the choice of machine learning models for stock price prediction is LSTM.
- **Training procedure:** in this I am going to explain the data splitting , training-validation sets and the training process.

Phase 4: Evaluation

- **Evaluation metrics:** Define the metrics used for evaluating model performance. That is mean absolute error , mean squared error , r squared etc,…
- **Visualization:** use visual aids such as charts and plot to help to interpret the model's performance.

ARCHITECTURE DESIGN:



DATASET:

- **Data description:** It provide information about the dataset, including the source, size and features included (e.g., open, high, low, close Prices, trading volume) the common features for stock price prediction may included:
- Date: Date associated with the data point.
- Open Price: The opening price of the stock on that date.
- High Price: The highest price the stock reached during the trading day.
- Low Price: The lowest price the stock reached during the trading day.
- Close Price: The closing price of the stock on that date.
- Volume: The trading volume on that date.

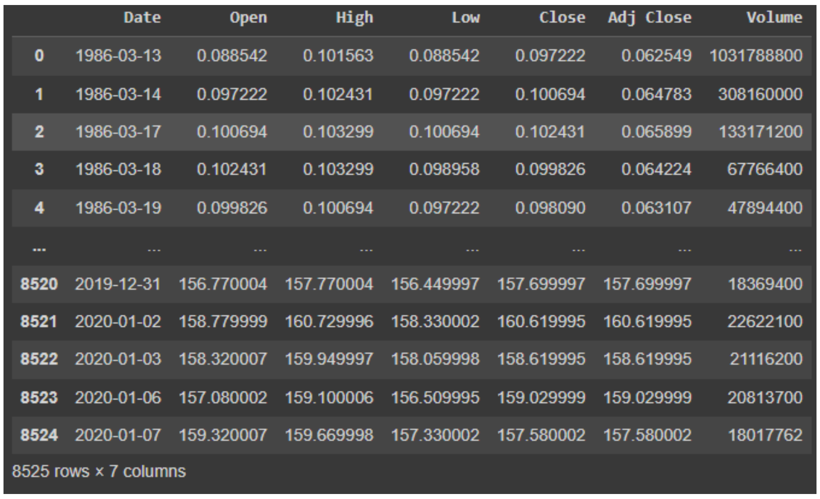
Code :

# Importing the data set:

```
import math

import pandas_datareader as web

import numpy as np

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

import matplotlib.pyplot as plt

plt.style.use('fivethirtyeight')

import pandas as pd
```

```
df = pd.read_csv('MSFT.csv')
df
```

output:



```
            Date        Open        High         Low       Close   Adj Close      Volume
0     1986-03-13    0.088542    0.101563    0.088542    0.097222    0.062549  1031788800
1     1986-03-14    0.097222    0.102431    0.097222    0.100694    0.064783   308160000
2     1986-03-17    0.100694    0.103299    0.100694    0.102431    0.065899   133171200
3     1986-03-18    0.102431    0.103299    0.098958    0.099826    0.064224    67766400
4     1986-03-19    0.099826    0.100694    0.097222    0.098090    0.063107    47894400
...          ...         ...         ...         ...         ...         ...         ...
8520  2019-12-31  156.770004  157.770004  156.449997  157.699997  157.699997    18369400
8521  2020-01-02  158.779999  160.729996  158.330002  160.619995  160.619995    22622100
8522  2020-01-03  158.320007  159.949997  158.059998  158.619995  158.619995    21116200
8523  2020-01-06  157.080002  159.100006  156.509995  159.029999  159.029999    20813700
8524  2020-01-07  159.320007  159.669998  157.330002  157.580002  157.580002    18017762
8525 rows × 7 columns
```

DATA PREPROCESSING STEPS:

- **Data cleansing:** how the data are handled missing data, outliers or inconsistencies.

Code:

```
#Drop missing values and the last row (since there's no target value)
import pandas as pd
df = pd.read_csv('MSFT.csv')
df.dropna(inplace=True)
```

output:

```
True
```

```
# reset the drop missing value
import pandas as pd
df = pd.read_csv('MSFT.csv')
df.reset_index(drop=True, inplace=True)
```

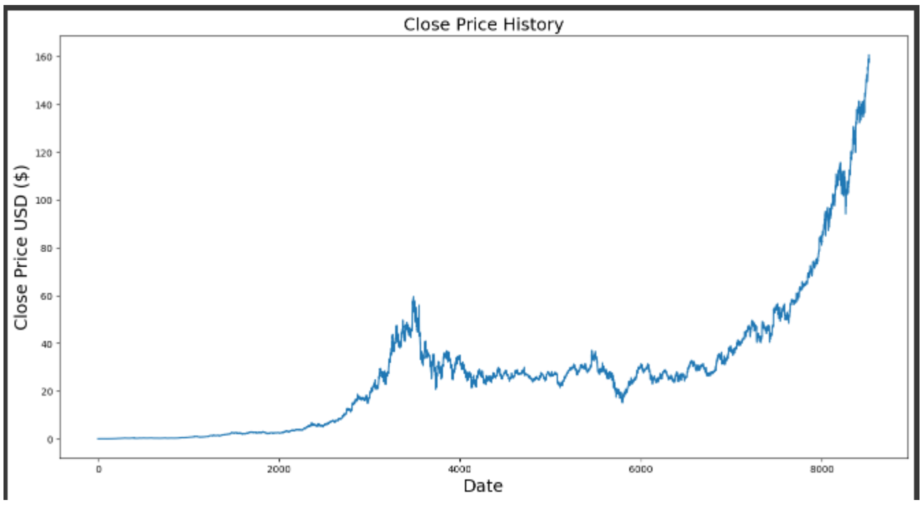
- **Feature Engineering:** the features created or modified for modeling. extract and create relevant features , such as the historical stock prices, trading volumes and technical indicator. here is the list of common features you might consider:

1.Historical prices : Daily or minute-level historical prices, including open, high, low , close and adjusted close.

Code:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(16,8))
plt.title('Close Price History', fontsize=18)
plt.plot(df['Close'])
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```

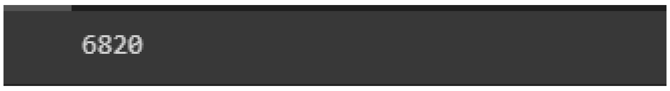
output:



```
# filtering the OHLC Prices
import math
import numpy as np
close_data = df.filter(['Close'])
dataset = close_data.values
training_data_len = math.ceil(len(dataset) * .8)
```

```
training_data_len
```

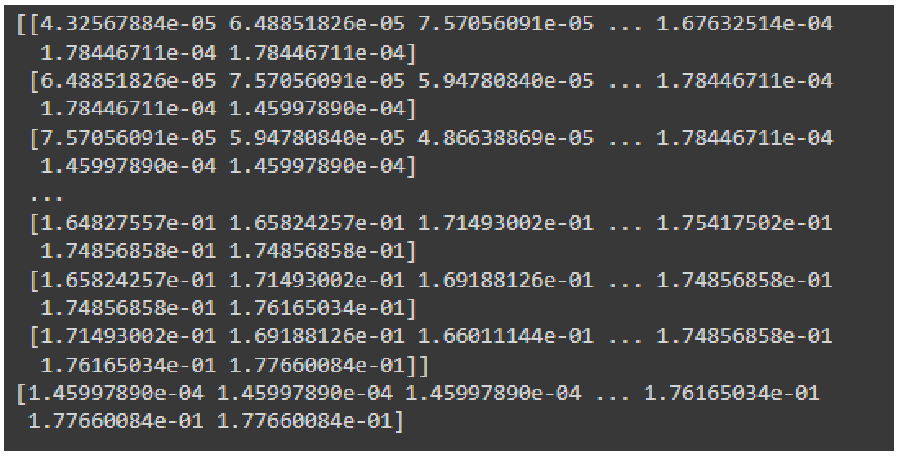
Output:



```
6820
```

```
# Split the data into training and testing sets
train_data = scaled_data[0:training_data_len, :]
x_train = []
y_train = []
for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])
x_train, y_train = np.array(x_train), np.array(y_train)


print(x_train)
print(y_train)
```

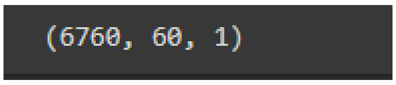
Output:



```
[[4.32567884e-05 6.48851826e-05 7.57056091e-05 ... 1.67632514e-04
  1.78446711e-04 1.78446711e-04]
 [6.48851826e-05 7.57056091e-05 5.94780840e-05 ... 1.78446711e-04
  1.78446711e-04 1.45997890e-04]
 [7.57056091e-05 5.94780840e-05 4.86638869e-05 ... 1.78446711e-04
  1.45997890e-04 1.45997890e-04]
 ...
 [1.64827557e-01 1.65824257e-01 1.71493002e-01 ... 1.75417502e-01
  1.74856858e-01 1.74856858e-01]
 [1.65824257e-01 1.71493002e-01 1.69188126e-01 ... 1.74856858e-01
  1.74856858e-01 1.76165034e-01]
 [1.71493002e-01 1.69188126e-01 1.66011144e-01 ... 1.74856858e-01
  1.76165034e-01 1.77660084e-01]]
[1.45997890e-04 1.45997890e-04 1.45997890e-04 ... 1.76165034e-01
 1.77660084e-01 1.77660084e-01]
```

```
#spliting the data in training
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))
print(x_train.shape)
```

output:



```
(6760, 60, 1)
```

2.Volume: It help to identify liquidity patterns that help the trading volumes over time.

3. Moving Averages: Simple moving averages(SMA) and Exponential Moving Averages(EMA) can help identify trends.

4.Relative Strength Index(RSI): A momentum oscillator that measures the speed and change of price movements.

5. Moving Average Convergence Divergence(MACD): Another momentum indicator that can signal changes in trend direction.

MODEL TRAINING PROCESS:

Once you have prepared a features, we should proceed to model training. Commonly used models for stock price prediction include:

1.Linear Regression: A simple model that assumes a linear relationship between features and stock prices.

2. Time Series Models: ARIMA (Auto Regressive Integrated Moving Average), GARCH (Generalized Autoregressive Conditional Heteroskedasticity), or LSTM (Long Short-Term Memory) networks designed to handle sequential data.

3.Machine Learning Models: Decision Trees, Random Forests, Support Vector Machines, or Gradient Boosting techniques like XGBoost.

4. Deep Learning Models: Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN) for more complex modeling.

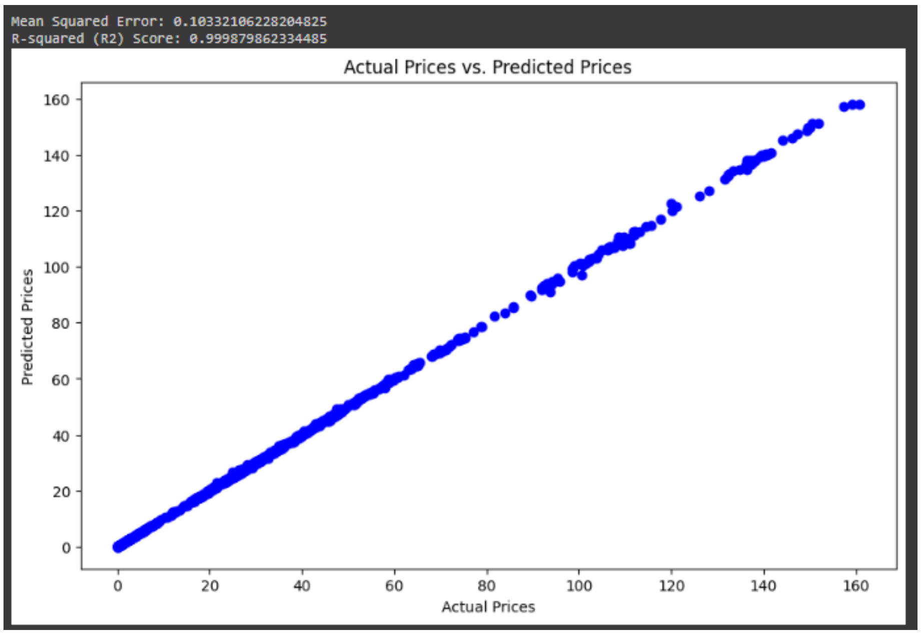
5. Ensemble Methods: Combining multiple models for better predictive performance.

Code :

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean_squared_error, r2_score
```

```
import matplotlib.pyplot as plt

# Load the dataset

df = pd.read_csv('MSFT.csv')

 #Extract the 'Close' prices as the target variable

target = df['Close']

# Extract relevant features from the dataset (you may need to customize this)

features = df[['Open', 'High', 'Low', 'Volume']]

# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=42)

# Initialize and train the Random Forest Regressor

rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

rf_regressor.fit(X_train, y_train)

# Make predictions on the test set

y_pred = rf_regressor.predict(X_test)

# Calculate the model's performance

mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")

print(f"R-squared (R2) Score: {r2}")

# Visualize the actual vs. predicted prices

plt.figure(figsize=(10, 6))

plt.scatter(y_test, y_pred, color='blue')

plt.xlabel("Actual Prices")

plt.ylabel("Predicted Prices")

plt.title("Actual Prices vs. Predicted Prices")

plt.show()
```

output:



```
Mean Squared Error: 0.10332106228204825
R-squared (R2) Score: 0.999879862334485
```

KEY FINDINGS AND INSIGHTS:

Model performance :

When evaluating the performance of a stock price prediction model, several metrics and techniques are commonly used to assess its accuracy and reliability. The choice of specific metrics can depend on the nature of the problem (short-term or long-term prediction) and the characteristics of the dataset. Here are some key metrics and considerations for evaluating model performance in stock price prediction:

Mean Absolute Error (MAE):MAE measures the average absolute difference between the predicted stock prices and the actual prices. It provides a straightforward way to understand the magnitude of prediction errors.

Root Mean Squared Error (RMSE):RMSE is similar to MAE but penalizes larger errors more heavily. It gives more weight to outliers in the data.

Mean Absolute Percentage Error (MAPE):MAPE calculates the percentage difference between predicted and actual prices, making it useful for understanding prediction errors in terms of percentage.

Profit and Loss (P&L) Analysis: Conduct a P&L analysis to determine the actual returns or losses that would have been generated by trading based on the model's predictions. This can include transaction costs, slippage, and other factors.

Visualizations: Create visualizations, such as price charts, prediction vs. actual comparisons, and rolling performance metrics, to gain insights into the model's behavior and identify any patterns or issues.

RECOMMENDATIONS:

Offer recommendations based on the analysis, such as potential investment strategies or insights derived from the model's predictions.

CONCLUSION:

By measuring the accuracy of the stock price prediction, we found that the most suitable for predicting the market price of a stock based on various data points from the historical data.in this historical prices, including open, high, low and adjusted close. Model training in that used models like linear regressions , time series models, machine learning models, deep learning models and ensemble methods. After training models its essential to evaluate its performance. Common evaluation metrics for regression tasks, such as stock price prediction that include mean absolute error , mean squared error, root mean squared error, r-squared(R^2), mean absolute percentage error. Then, enhance a stock price prediction model through data preprocessing, cleansing, model training process. These crucial steps will refine the accuracy and effectiveness of your model, bringing you closer to achieving your predictive goals.