***STOCK PRICE PREDICTION***

**INTRODUCTION:**

**Stock market prediction** is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit. The efficient - market hypothesis suggests that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable. Others disagree and those with this viewpoint possess myriad methods and technologies which purportedly allow them to gain future price information. A market where shares are publicly issued and traded is known as a share market. Implementing the concept of algorithmic trading, which uses automated, pre-programmed trading strategies to predict stock prices. Time series forecasting (predicting future values based on historical values) applies well to stock forecasting. Developed a User Interface. Stock price prediction is the process of using various techniques and data analysis to forecast the future prices of individual stocks or the overall stock market. It involves the examination of historical stock price data, as well as the consideration of various factors such as market trends, economic indicators, and company-specific information, to make informed estimates about the direction in which a stock price may move. This field is of significant interest to investors, traders, and financial analysts seeking to make informed decisions in the stock market.

# Clearly outline the problem statement, design thinking process, and the phases of development

* Load the CSV file as a DataFrame using Pandas. Since the data is indexed by date (each row
* represents data from a different date), we can also index our DataFrame by the date column
* Plotting the High and Low points of Netflix stock
* It will be challenging for a model in the stock prediction using machine learning project to
* correctly estimate the rapid changes that we can see.
* Similarly, plotting the Open and Close value of the stock for each day gives equivalent
* observations.
* We use matplotlib to plot the DataFrame columns directly against the Date index column. To
* make things flexible while plotting against dates, lines 6-8 convert our date strings into
* datetime format and plot them cleanly and legibly. The interval parameter in line 7 defines
* the interval in days between each tick on the date axis.
* Importing the Libraries for Stock Price Prediction Project
* We will be building our LSTM models using Tensorflow Keras and preprocessing our stock
* prediction machine learning data using scikit-learn. These imports are used in different steps
* of the entire process, but it is good to club these statements together.
* Data Preprocessing for Stock Market Prediction using Machine Learning
* As with any other machine learning model, it is always good to normalize or rescale the data
* within a fixed range when dealing with real data. This will avoid features with larger numeric
* values to unjustly interfere and bias the model and help achieve rapid convergence in the
* machine learning stock prediction project.
* First, we define the features and the target
* Next, we use a StandardScaler to rescale our values between -1 and 1.
* Scikit-learn also provides a popular MinMaxScaler preprocessing module. However,
* considering the context, stock prices might max out or minimise on different days, and using
* those values to influence others might not be great. The change in values from using either
* of these methods would not be much, so we stick to StandardScaler.
* So, the next step would be to split it into training and testing sets. As explained above, the
* training of an LSTM model requires a window or a timestep of data in each training step. For
* instance, the LSTM will take 10 data samples to predict the 10th one by weighing the first
* nine input samples in one step. So, we need a different approach than the train\_test\_split
* provided by scikit-learn.
* Let’s define a splitting function called lstm\_split() which will make windows of size “n\_steps”
* starting from the first sample of data and ending at n\_steps’th sample (if n\_steps=10, then
* the 10th sample) from the end. We understand the latter part because, for each time step,
* LSTM will take n\_steps-1 samples for training and predict the last sample. Loss calculation is
* done based on the error in this prediction. So if n\_steps=10, you cannot use the last 9
* samples to predict anything because the “10th” data point for the current step does not
* exist in the dataset.
* The function below takes the entity.
* Given the simplicity of the model and the data, we note that the loss reduction stagnates
* after only 20 epochs. You can observe this by plotting the training loss against the number of
* epochs, and LSTM does not learn much after 10-20 epochs.

# Describe the dataset used, data preprocessing steps, and model training process.

**FUTURE WORK:**

Machine learning and data science is a game changer in this domain so there is a lot of data to find patterns in for predicting with high degree of accuracy. In future we’ll try to predict the values based on multiple factors such as politics, global economic conditions, unexpected

events like covid, companies financial performance, and so on. We are going to implement multiple types of algorithms because different types of data requires different types of techniques. Decided to implement simple User Interface to operate this whole process for users so to make people engage in Stock Market.

**STEPS PERFORMED:**

1.Importing and Cleaning data

2. Split the Data into training/test sets

3. Creating and Training the Model

4. Making Predictions

5. Evaluating and Improving Predictions

**NEED OF PROJECT:**

* + The stock market is known for being volatile, dynamic,&amp; nonlinear
  + Accurate stock price prediction is extremely challengingbecause of multiple factors.
  + But, all of this also means that there is a lot of data to findpatterns in.
  + So, we keep exploring analytics techniques to detect stockmarket trends.
  + So, they can be analysed as a sequence of discrete time data
  + Despite the volatility, stock prices are not just randomly generated numbers.

**METHODOLOGY:**

PYTHON Language:

Python is a rich language for Data Science and AI

Libraries:

Pandas, Numpy, Sklearn, Tensorflow, etc

**Algorithm:**

Long Short Term Memory(LSTM)

Streamlit UI:

Provided User Interface using Streamlit

**CODE:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sb

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from xgboost import XGBClassifier

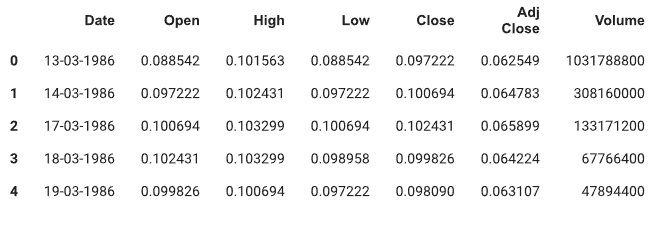
from sklearn import metric

import warnings

warnings.filterwarnings(&#39;ignore &#39;)

df = pd.read\_csv(&#39;/content/MSFT.csv &#39;)

**OUTPUT :**



**CODE:**

df.shape

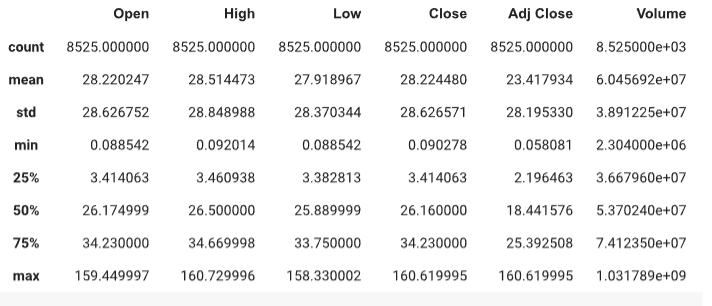
**OUTPUT:**

(8525, 7)

**CODE:**

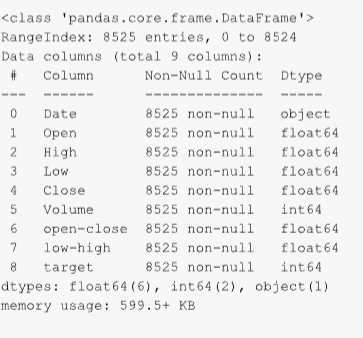
df.describe()

**OUTPUT:**

**CODE:**

df.info()

**OUTPUT:**



**CODE:**

plt.figure(figsize=(15,5))

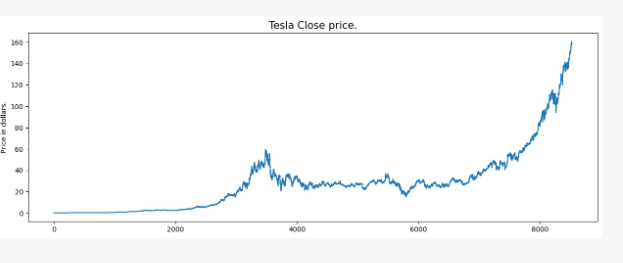
plt.plot(df[&#39;Close&#39;])

plt.title(&#39;Tesla Close price.&#39;, fontsize=15)

plt.ylabel(&#39;Price in dollars.&#39;)

plt.show()

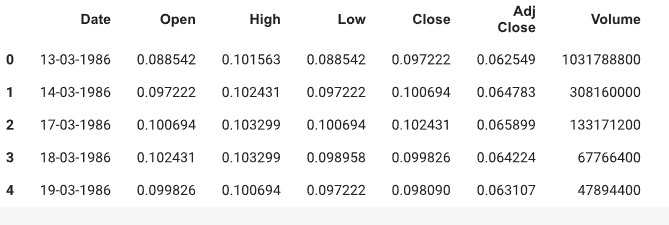
**OUTPUT:**



**CODE:**

df.head()

**OUTPUT:**



**CODE:**

df[df[&#39;Close&#39;] == df[&#39;Adj Close &#39;]].shape

**OUTPUT:**

(32, 7)

**CODE:**

df.isnull().sum()

**OUTPUT:**

Date 0

Open 0

High 0

Low 0

Close 0

Volume 0

dtype: int64

**CODE:**

features = [&#39;Open&#39;, &#39;High&#39;, &#39;Low&#39;, &#39;Close&#39;, &#39;Volume&#39;]

plt.subplots(figsize=(20,10))

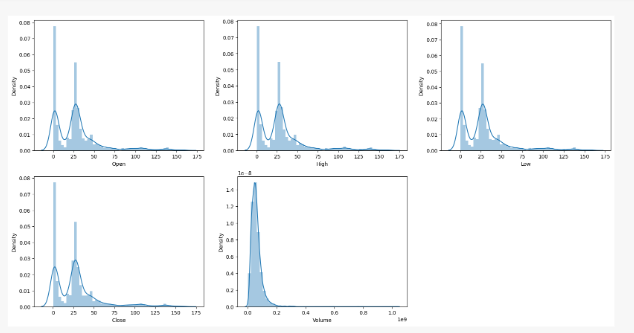
for i, col in enumerate(features):

  plt.subplot(2,3,i+1)

  sb.distplot(df[col])

plt.show()

**OUTPUT:**



**CODE:**

plt.subplots(figsize=(20,10))

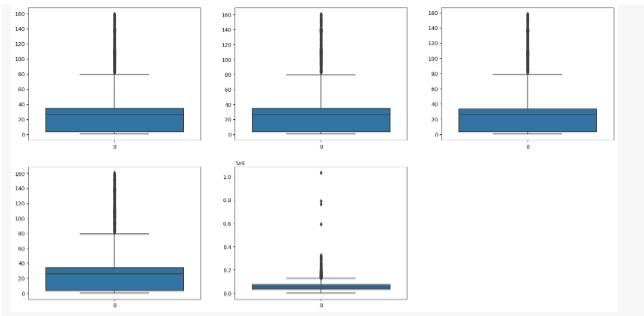
for i, col in enumerate(features):

  plt.subplot(2,3,i+1)

  sb.boxplot(df[col])

plt.show()

**OUTPUT:**



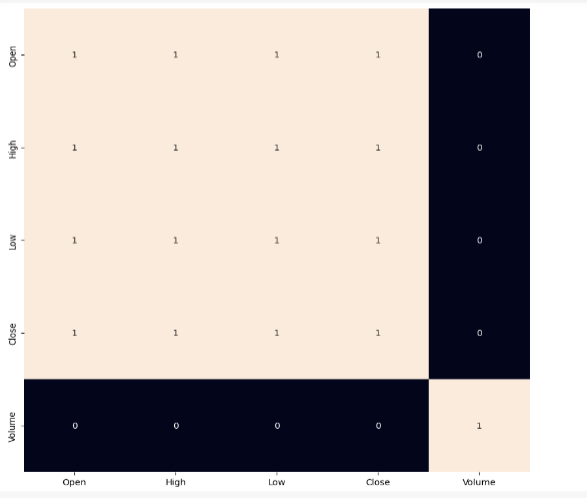
**CODE:**

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

sb.heatmap(df.corr() &gt; 0.9, annot=True, cbar=False)

plt.show()

**OUTPUT:**



**CODE:**

df[&#39;open-close&#39;] = df[&#39;Open&#39;] - df[&#39;Close&#39;]

df[&#39;low-high&#39;] = df[&#39;Low&#39;] - df[&#39;High&#39;]

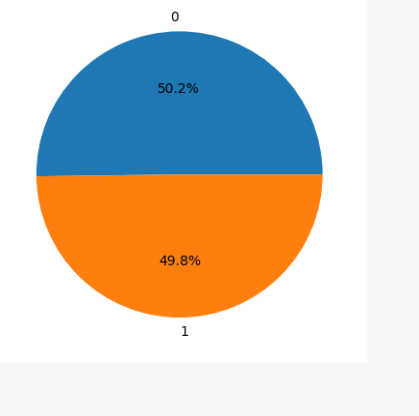
df[&#39;target&#39;] = np.where(df[&#39;Close&#39;].shift(-1) &gt; df[&#39;Close&#39;], 1, 0)

plt.pie(df[&#39;target&#39;].value\_counts().values,

    labels=[0, 1], autopct=&#39;%1.1f%%&#39;)

plt.show()

**OUTPUT:**



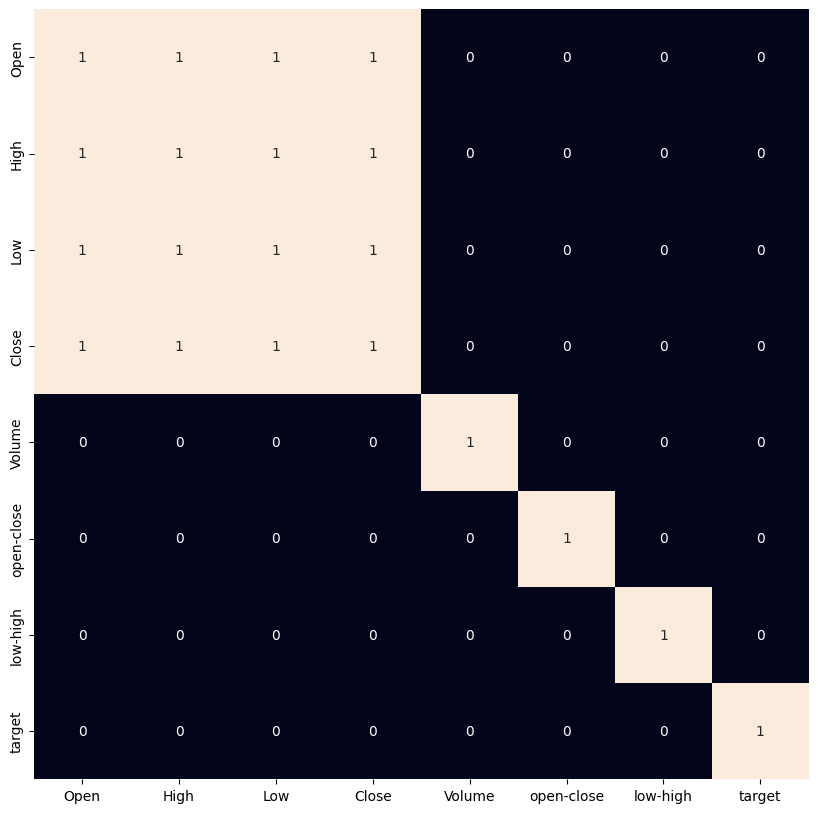
**CODE:**

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

sb.heatmap(df.corr() &gt; 0.9, annot=True, cbar=False)

plt.show()

**OUTPUT:**



# Present key findings, insights, and recommendations based on the analysis.

**CODE:**

import pandas as pd

import numpy as np

​import matplotlib.pyplot as plt

import seaborn as sns

sns.set\_style(&#39;whitegrid&#39;)

plt.style.use(&quot;fivethirtyeight&quot;)

%matplotlib inline

​# For reading stock data from yahoo

from pandas\_datareader.data import DataReader

import y finance as yf

from pandas\_datareader import data as pdr

​yf.pdr\_override()

​# For time stamps

from datetime import datetime

​# The tech stocks we&#39;ll use for this analysis

tech\_list = [&#39;AAPL&#39;, &#39;GOOG&#39;, &#39;MSFT&#39;, &#39;AMZN&#39;]

​# Set up End and Start times for data grab

tech\_list = [&#39;AAPL&#39;, &#39;GOOG&#39;, &#39;MSFT&#39;, &#39;AMZN&#39;]

​end = datetime.now()

start = datetime(end.year - 1, end.month, end.day)

​for stock in tech\_list:

globals()[stock] = yf.download(stock, start, end)

​company\_list = [AAPL, GOOG, MSFT, AMZN]

company\_name = [&quot;APPLE&quot;, &quot;GOOGLE&quot;, &quot;MICROSOFT&quot;, &quot;AMAZON&quot;]

​for company, com\_name in zip(company\_list, company\_name):

company[&quot;company\_name&quot;] = com\_name

df = pd.concat(company\_list, axis=0)

df.tail(10)

**OUTPUT:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Open | High | Low | Close | Adj Close | | Volume | | company\_name | |
| Date |  |  |  |  | |  | |  | |  | |
| 2023-01-17 00:00:00-05:00 | 98.680000 | 98.889999 | 95.730003 | 96.050003 | | 96.050003 | | 72755000 | | AMAZON | |
| 2023-01-18 00:00:00-05:00 | 97.250000 | 99.320000 | 95.379997 | 95.459999 | | 95.459999 | | 79570400 | | AMAZON | |
| 2023-01-19 00:00:00-05:00 | 94.739998 | 95.440002 | 92.860001 | | 93.680000 | | 93.680000 | | 69002700 | | AMAZON | |
| 2023-01-20 00:00:00-05:00 | 93.860001 | 97.349998 | 93.199997 | 97.250000 | | 97.250000 | | 67307100 | | AMAZON | |
| 2023-01-23 00:00:00-05:00 | 97.559998 | 97.779999 | 95.860001 | 97.519997 | | 97.519997 | | 76501100 | | AMAZON | |
| 2023-01-24 00:00:00-05:00 | 96.930000 | 98.089996 | 96.000000 | 96.320000 | | 96.320000 | | 66929500 | | AMAZON | |
| 2023-01-25 00:00:00-05:00 | 92.559998 | 97.239998 | 91.519997 | 97.180000 | | 97.180000 | | 94261600 | | AMAZON | |
| 2023-01-26 00:00:00-05:00 | 98.239998 | 99.489998 | 96.919998 | 99.220001 | | 99.220001 | | 68523600 | | AMAZON | |
| 2023-01-27 00:00:00-05:00 | 99.529999 | 103.489998 | 99.529999 | 102.239998 | | 102.239998 | | 87678100 | | AMAZON | |
| 2023-01-30 00:00:00-05:00 | 101.089996 | 101.739998 | 99.010002 | 100.550003 | | 100.550003 | | 70566100 | | AMAZON | |

Reviewing the content of our data, we can see that the data is numeric and the date is the index of the data. Notice also that

weekends are

missing from the records.

**Descriptive Statistics about the Data:**

.describe() generates descriptive statistics. Descriptive statistics include

those that summarize the central tendency, dispersion, and shape of a

dataset’s distribution, excluding NaN values.

**CODE:**

AAPL.describe()

**OUTPUT**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Open | High | Low | Close | Adj Close | Volume |
| count | 251.000000 | 251.000000 | 251.000000 | 251.000000 | 251.000000 | 2.510000e+02 |
| mean | 152.117251 | 154.227052 | 150.098406 | 152.240797 | 151.861737 | 8.545738e+07 |
| std | 13.239204 | 13.124055 | 13.268053 | 13.255593 | 13.057870 | 2.257398e+07 |
| min | 126.010002 | 127.769997 | 124.169998 | 125.019997 | 125.019997 | 3.519590e+07 |
| 25% | 142.110001 | 143.854996 | 139.949997 | 142.464996 | 142.190201 | 7.027710e+07 |
| 50% | 150.089996 | 151.990005 | 148.199997 | 150.649994 | 150.400497 | 8.100050e+07 |
| 75% | 163.434998 | 165.835007 | 160.879997 | 163.629997 | 163.200417 | 9.374540e+07 |
| max | 178.550003 | 179.610001 | 176.699997 | 178.960007 | 178.154037 | 1.826020e+08 |

**Information About the Data:**

.info() method prints information about a DataFrame including the

index dtype and columns, non-null values, and memory usage.

**CODE:**

AAPL.info()

**OUTPUT:**

<class 'pandas.core.frame.DataFrame'>

DatetimeIndex: 251 entries, 2022-01-31 00:00:00-05:00 to 2023-01-30 00:00:00-05:00

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Open 251 non-null float64

1 High 251 non-null float64

2 Low 251 non-null float64

3 Close 251 non-null float64

4 Adj Close 251 non-null float64

5 Volume 251 non-null int64

6 company\_name 251 non-null object

dtypes: float64(5), int64(1), object(1)

memory usage: 23.8+ KB

**Closing Price:**

The closing price is the last price at which the stock is traded during the

regular trading day. A stock’s closing price used by investors to track its

performance over time.

**CODE:**

# Let&#39;s see a historical view of the closing price

plt.figure(figsize=(15, 10))

plt.subplots\_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(company\_list, 1):

plt.subplot(2, 2, i)

company[&#39;Adj Close&#39;].plot()

plt.ylabel(&#39;Adj Close&#39;)

plt.xlabel(None)

plt.title(f&quot;Closing Price of {tech\_list[i - 1]}&quot;)

plt.tight\_layout()

**Volume of Sales:**

Volume is the amount of an asset or security that changes hands over some

period of time, often over the course of a day. For instance, the stock trading

volume would refer to the number of shares of security traded between its daily

open and close. Trading volume, and changes to volume over the course of

time, are important inputs for technical traders.

**CODE:**

# Now let&#39;s plot the total volume of stock being traded each day

plt.figure(figsize=(15, 10))

plt.subplots\_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(company\_list, 1):

plt.subplot(2, 2, i)

company[&#39;Volume&#39;].plot()

plt.ylabel(&#39;Volume&#39;)

plt.xlabel(None)

plt.title(f&quot;Sales Volume for {tech\_list[i - 1]}&quot;)

plt.tight\_layout()

 How much value do we put at risk by investing in a particular

stock?

There are many ways we can quantify risk, one of the most basic ways using the

information we&#39;ve gathered on daily percentage returns is by comparing the

expected return with the standard deviation of the daily returns.

**CODE:**

rets = tech\_rets.dropna()

area = np.pi \* 20

plt.figure(figsize=(10, 8))

plt.scatter(rets.mean(), rets.std(), s=area)

plt.xlabel(&#39;Expected return&#39;)

plt.ylabel(&#39;Risk&#39;)

for label, x, y in zip(rets.columns, rets.mean(), rets.std()):

plt.annotate(label, xy=(x, y), xytext=(50, 50), textcoords=&#39;offset points&#39;, ha=&#39;right&#39;,

va=&#39;bottom&#39;,

arrowprops=dict(arrowstyle=&#39;-&#39;, color=&#39;blue&#39;, connectionstyle=&#39;arc3,rad=-0.3&#39;))

**CODE:**

Grab all the closing prices for the tech stock list into one DataFrame

closing\_df = pdr.get\_data\_yahoo(tech\_list, start=start, end=end)[&#39;Adj Close&#39;]

# Make a new tech returns DataFrame

tech\_rets = closing\_df.pct\_change()

tech\_rets.head()

**OUTPUT:**

|  |  |  |  |
| --- | --- | --- | --- |
| AAPL | AMZN | GOOG | MSFT |
| Date |  |  |  |  |
| 2022-01-31 00:00:00-05:00 | NaN | NaN | NaN | NaN |
| 2022-02-01 00:00:00-05:00 | -0.000973 | 0.010831 | 0.016065 | -0.007139 |
| 2022-02-02 00:00:00-05:00 | 0.007044 | -0.003843 | 0.073674 | 0.015222 |
| 2022-02-03 00:00:00-05:00 | -0.016720 | -0.078128 | -0.036383 | -0.038952 |
| 2022-02-04 00:00:00-05:00 | -0.001679 | 0.135359 | 0.002562 | 0.015568 |

So now we can see that if two stocks are perfectly (and positivley) correlated with eachother a linear relationship between its daily return values should occur.Seaborn and pandas make it very easy to repeat this comparison analysis for every possible combination of stocks in our technology stock ticker list. We can use sns.pairplot() to automatically create this plot

**CODE:**

# Plot the data

train = data[:training\_data\_len]

valid = data[training\_data\_len:]

valid[&#39;Predictions&#39;] = predictions

# Visualize the data

plt.figure(figsize=(16,6))

plt.title(&#39;Model&#39;)

plt.xlabel(&#39;Date&#39;, fontsize=18)

plt.ylabel(&#39;Close Price USD ($)&#39;, fontsize=18)

plt.plot(train[&#39;Close&#39;])

plt.plot(valid[[&#39;Close&#39;, &#39;Predictions&#39;]])

plt.legend([&#39;Train&#39;, &#39;Val&#39;, &#39;Predictions&#39;], loc=&#39;lower right&#39;)

plt.show()

**OUTPUT:**

|  |  |
| --- | --- |
| Close | Predictions |
| Date |  |  |
| 2022-07-13 00:00:00-04:00 | 145.490005 | 146.457565 |
| 2022-07-14 00:00:00-04:00 | 148.470001 | 146.872879 |
| 2022-07-15 00:00:00-04:00 | 150.169998 | 147.586197 |
| 2022-07-18 00:00:00-04:00 | 147.070007 | 148.572937 |
| 2022-07-19 00:00:00-04:00 | 151.000000 | 148.995255 |
| ... | ... | ... |
| 2023-01-24 00:00:00-05:00 | 142.529999 | 138.565536 |
| 2023-01-25 00:00:00-05:00 | 141.860001 | 140.022110 |
| 2023-01-26 00:00:00-05:00 | 143.960007 | 141.225128 |
| 2023-01-27 00:00:00-05:00 | 145.929993 | 142.469315 |
| 2023-01-30 00:00:00-05:00 | 143.000000 | 143.833130 |

**Continue building the stock price prediction model by  Feature engineering, Model training, Evaluation**.

**Feature Engineering:**

1. Data Collection: Ensure you have the historical stock price and relevant financial data for the stock(s) you want to analyze. You can collect this data from sources like Yahoo Finance, Alpha Vantage, or other financial data providers.

2. Data Preprocessing: Clean and preprocess the data, handling missing values and outliers, and converting date columns to the appropriate datetime format.

3. Feature Selection: Carefully select or engineer features that are likely to impact stock prices. These features can include technical indicators, fundamental data, market sentiment, and economic indicators. Common features might include moving averages, volume, volatility, and company- specific metrics like earnings and dividends.

4. Lag Features: Create lag features to capture historical trends and patterns in the data. For example, you can create lag features for the closing price of the stock over the past few days or weeks.

5. Technical Indicators: Calculate technical indicators like Moving Averages, Relative Strength Index (RSI), Bollinger Bands, and MACD (Moving Average Convergence Divergence).

6. Market Sentiment Analysis: Incorporate market sentiment data, such as news sentiment scores, social media sentiment, or sentiment from financial news articles.

7. Volatility Measures: Compute measures of volatility like historical volatility and Average True Range (ATR).

**Model Training:**

1. Select a Model: Choose a suitable machine learning or statistical model for stock price prediction. Common models include linear regression, time series models (e.g., ARIMA or LSTM), and machine learning models like decision trees, random forests, or gradient boosting.

2. Data Split: Split your data into training and testing sets. You can use historical data for training and reserve a portion for testing to evaluate your model&#39;s performance.

3. Train the Model: Train the chosen model using the training data. If you&#39;re using a machine learning model, you may want to fine-tune hyperparameters for better performance.

4. Make Predictions: Use the trained model to make predictions on the testing data or future data.

**Evaluation:**

1. Evaluation Metrics: Evaluate the performance of your model using appropriate metrics such as Mean Squared Error (MSE), Root Mean

Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2). Additionally, consider financial metrics like Sharpe Ratio or Annualized

Return.

2. Visualization: Visualize the model&#39;s predictions and compare them to the actual stock prices using line plots or candlestick charts. This helps you assess the model&#39;s ability to capture trends and patterns.

3. Backtesting: If your analysis includes trading strategies, consider backtesting the strategies using historical data to assess their performance

over time.

4. Cross-Validation: Perform cross-validation to ensure your model&#39;s robustness and avoid overfitting.

5. Fine-Tuning: If the model&#39;s performance is not satisfactory, consider fine- tuning the model, exploring different features, or trying other models.

**Source Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

df = pd.read\_csv(&#39;stocks.csv&#39;)

df[&#39;Date&#39;] = pd.to\_datetime(df[&#39;Date&#39;])

df = df.sort\_values(&#39;Date&#39;)

df.set\_index(&#39;Date&#39;, inplace=True)

df[&#39;Daily Return&#39;] = df[&#39;Close&#39;].pct\_change()

mean\_daily\_return = df[&#39;Daily Return&#39;].mean()

std\_daily\_return = df[&#39;Daily Return&#39;].std()

annualized\_return = ((1 + mean\_daily\_return) \*\* 252) - 1

plt.figure(figsize=(10, 5))

plt.plot(df.index, df[&#39;Daily Return&#39;])

plt.title(&#39;Daily Returns&#39;)

plt.xlabel(&#39;Date&#39;)

plt.ylabel(&#39;Daily Return&#39;)

plt.grid()

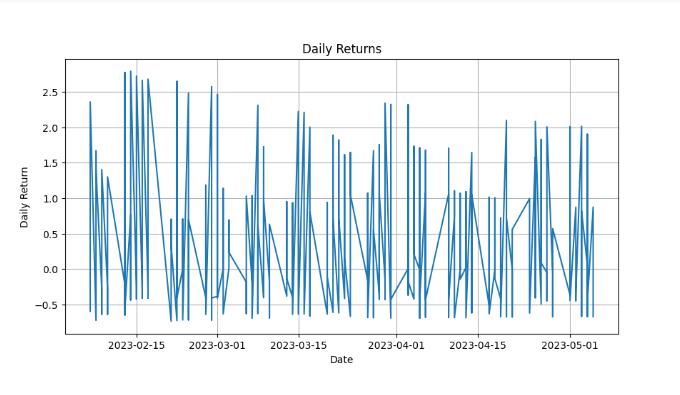
plt.show()

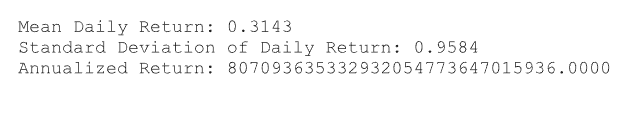
print(f&quot;Mean Daily Return: {mean\_daily\_return:.4f}&quot;)

print(f&quot;Standard Deviation of Daily Return: {std\_daily\_return:.4f}&quot;)

print(f&quot;Annualized Return: {annualized\_return:.4f}&quot;)

**OUTPUT:**





**CONCLUSION:**

In conclusion, stock price prediction is a complex and challenging endeavor that combines historical data analysis, statistical models, and factors affecting financial markets. While advancements in machine learning and artificial intelligence have improved prediction accuracy, its important to remember that stock markets are inherently volatile and unpredictable. Predictions should be used as tools for making informed decisions, but they come with inherent uncertainties and risks. Diversification and a long-term investment perspective remain crucial strategies for managing the inherent unpredictability of stock markets