PROJECT SUBMISSION PART 3: DEVELOPMENT PART 1
STOCK PRICE PREDICTION

INTRODUCTION:

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's price may move. This field is of significant interest to investors, traders, and financial analysts seeking to make informed decisions in the stock market

1. What was the change in price of the stock overtime?

In this section we'll go over how to handle requesting stock information with pandas, and how to analyze basic attributes of a stock.

CODE:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
plt.style.use("fivethirtyeight")
%matplotlib inline

# For reading stock data from yahoo
from pandas_datareader.data import DataReader
import yfinance as yf
from pandas_datareader import data as pdr

yf.pdr_override()

# For time stamps
from datetime import datetime


# The tech stocks we'll use for this analysis
```

```
tech_list = ['AAPL', 'GOOG', 'MSFT', 'AMZN']

# Set up End and Start times for data grab
tech_list = ['AAPL', 'GOOG', 'MSFT', 'AMZN']

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 = ["APPLE", "GOOGLE", "MICROSOFT", "AMAZON"]

for company, com_name in zip(company_list, company_name):
    company["company_name"] = 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

Open	High	Low	Close	Adj Close	Volume	company_name	
Date							
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

Open	High	Low	Close	Adj Close	Volume	company_name	
Date							
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

	Open	High	Low	Close	Adj Close	Volume
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'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['Adj Close'].plot()
    plt.ylabel('Adj Close')
    plt.xlabel(None)
    plt.title(f"Closing Price of {tech_list[i - 1]}")

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'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['Volume'].plot()
    plt.ylabel('Volume')
    plt.xlabel(None)
    plt.title(f"Sales Volume for {tech_list[i - 1]}")

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'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('Expected return')
plt.ylabel('Risk')

for label, x, y in zip(rets.columns, rets.mean(), rets.std()):
    plt.annotate(label, xy=(x, y), xytext=(50, 50), textcoords='offset points', ha='right', va='bot
tom',
                 arrowprops=dict(arrowstyle='-', color='blue', connectionstyle='arc3,rad=-0.3'))
```

OUTPUT:

What was the correlation between different stocks closing prices?

linkcode

Correlation is a statistic that measures the degree to which two variables move in relation to each other which has a value that must fall between -1.0 and +1.0. Correlation measures association, but doesn't show if x causes y or vice versa — or if the association is caused by a third factor[1].

Now what if we wanted to analyze the returns of all the stocks in our list? Let's go ahead and build a DataFrame with all the ['Close'] columns for each of the stocks dataframes.

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)['Adj Close']

# 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

	AAPL	AMZN	GOOG	MSFT
Date				
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 each other a linear relationship bewteen 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['Predictions'] = predictions
# Visualize the data
plt.figure(figsize=(16,6))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
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

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, it's 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