Artificial Intelligence and Machine Learning

Project Report

STOCK MARKET PREDICTION

Submitted by :-

Vanshika Koli (2210990948)

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

ABSTRACT

Unlocking the potential of time series data, this project delves into the realm of stock market analysis, focusing on major technology stocks such as Apple, Amazon, Google, and Microsoft. Leveraging Python libraries like yfinance, pandas, Matplotlib, and Seaborn, this study offers a multifaceted exploration of stock market behavior over a one-year period

.

Beginning with data retrieval from Yahoo Finance, the analysis proceeds to visualize various aspects of stock performance and risk. Descriptive statistics and visualizations elucidate changes in stock prices over time and the volume of stocks traded daily. Moving averages are computed to discern trends amidst market fluctuations, while daily returns unveil the inherent risk associated with each stock.

Furthermore, the study investigates correlations between stock returns, providing insights into the interplay between different stocks. Seaborn's versatile plotting capabilities reveal complex relationships, offering a holistic view of market dynamics.

Notably, the project also delves into quantifying the risk of investing in specific stocks, assessing expected returns against the standard deviation of daily returns. Additionally, it ventures into predictive modeling using Long Short-Term Memory (LSTM) techniques to forecast future stock prices, exemplifying the intersection of data science and finance.

In conclusion, this comprehensive analysis not only equips investors and analysts with valuable insights but also underscores the power of Python in dissecting intricate financial data with precision and sophistication.

Keywords: Stock Market Analysis, Time Series Data, Technology Stocks, Python, yfinance, Pandas, Matplotlib, Seaborn, Risk Analysis, Correlation, Predictive Modeling, LSTM

TABLE OF CONTENTS

INTRODUCTION

Predicting stock market movements is a challenging yet highly valuable task that has drawn significant interest from researchers and investors alike. The ability to forecast stock prices accurately can offer substantial benefits, including informed decision-making for trading and investment strategies. In recent years, machine learning techniques, particularly supervised learning algorithms, have demonstrated promising capabilities in this domain.

This project aims to explore the application of supervised learning models for stock market prediction. By leveraging historical stock market data and associated features, such as company fundamentals, market indicators, and technical analysis metrics, we seek to build predictive models that can forecast future stock prices or directional movements with reasonable accuracy.

Supervised learning involves training a model on labeled data, where the input data (features) and corresponding output (target variable) are known. In the context of stock market prediction, the historical stock data serves as the training dataset, where each data point comprises various features and the corresponding stock price movement (e.g., price increase or decrease) over a specific time horizon.

OBJECTIVES

The objective of this project is to develop and deploy a supervised learning-based stock market prediction model using historical market data. Specifically, the project aims to:

1. **Data Collection and Preparation**: Gather historical stock market data from reliable sources, including price history, trading volumes, company fundamentals, and relevant economic indicators.

1. **Feature Engineering**: Perform feature engineering to extract meaningful features from the raw data, including technical indicators (e.g., moving averages, relative strength index), sentiment analysis of news and social media, and macroeconomic variables that can potentially influence stock prices.

1. **Model Selection and Training**: Evaluate and compare different supervised learning algorithms such as regression (e.g., linear regression, decision trees), classification (e.g., logistic regression, random forests), and ensemble methods to identify the most suitable approach for stock market prediction.

1. **Model Evaluation and Validation**: Assess the performance of the trained models using appropriate evaluation metrics (e.g., mean squared error, accuracy, F1-score) on validation datasets to ensure robustness and generalization capability.

1. **Deployment and Real-Time Prediction**: Implement the best-performing prediction model into a practical application that can provide real-time or near-real-time stock price forecasts, enabling investors and traders to make informed decisions.

SIGNIFICANCE

The development of a supervised learning-based stock market prediction model holds significant importance in the field of finance and investment for several reasons:

1. **Enhanced Decision-Making**: Accurate stock market predictions empower investors and traders to make informed decisions regarding buying, selling, or holding stocks.

1. **Risk Management**: Reliable stock market forecasts assist in risk management by identifying potential market downturns or fluctuations. This enables investors to adjust their portfolios accordingly and mitigate potential losses.

1. **Algorithmic Trading**: Machine learning-based prediction models can be integrated into algorithmic trading systems, automating the execution of trades based on predefined strategies and predicted market conditions. This enhances trading efficiency and responsiveness to market dynamics.

1. **Portfolio Optimization**: Predictive models aid in optimizing investment portfolios by identifying opportunities for diversification and asset allocation based on forecasted market trends and expected returns.

1. **Market Efficiency**: Accurate stock market predictions contribute to overall market efficiency by incorporating data-driven insights into pricing mechanisms and reducing inefficiencies caused by irrational market behaviors.

PROBLEM DEFINATION AND REQUIREMENT

Stock market prediction is a challenging task that requires leveraging historical data and advanced analytics to forecast future price movements accurately. Traditional methods often rely on subjective analysis and may not fully capture the complex dynamics of financial markets. Therefore, there is a need to develop a robust and reliable predictive model using supervised learning techniques to enhance stock market forecasting.

The main challenges addressed by this project include:

1. **Data Complexity**: Stock market data is inherently noisy, high-dimensional, and influenced by various internal and external factors (e.g., market sentiment, economic indicators). Developing a predictive model that can effectively handle and extract relevant signals from this complex data is crucial.

1. **Prediction Accuracy**: Accurately predicting stock prices is essential for informed decision-making in trading and investment. Improving prediction accuracy beyond random chance requires exploring advanced machine learning algorithms capable of capturing subtle patterns and trends in the data.

1. **Feature Engineering**: Identifying and engineering meaningful features from raw data is critical for building effective prediction models. This involves selecting appropriate technical indicators, sentiment analysis of news and social media, and incorporating macroeconomic factors that impact stock market behavior.

REQUIREMENT

1.Pandas library : The Pandas library is a powerful and popular Python library used for data manipulation and analysis. It provides easy-to-use data structures and data analysis tools, making it an essential tool for working with structured data, especially in the context of data science, machine learning, and finance.

1.NumPy library : NumPy (Numerical Python) is a powerful open-source library in Python that provides support for large, multi-dimensional arrays and matrices. It also includes a large collection of high-level mathematical functions to operate on these arrays

1. Matplotlib Library : Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It provides a wide range of tools and functions for generating high-quality plots, graphs, and charts.

1.Seaborn library : Seaborn is a Python data visualization library built on top of Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

1.Datetime : The datetime module in Python provides classes for working with dates, times, and date/time combinations. It is a part of the Python standard library and is a powerful tool for handling temporal data in your applications.

6.yfinance : yfinance is a Python library that provides a simple and efficient way to download historical stock data from Yahoo Finance. It is a popular choice for developers, data analysts, and researchers who need to access and work with financial data.

7. Keras : Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It is designed to enable fast experimentation with deep neural networks and supports both convolutional networks and recurrent networks, as well as a combination of the two.

8. TensorFlow : TensorFlow is a powerful open-source library for numerical computation and large-scale machine learning. It was developed by the Google Brain team and is widely used in the field of deep learning and artificial intelligence.

SUPERVISED LEARNING

Supervised learning is a type of machine learning where the algorithm is trained on a labeled dataset, meaning the input data has known target outputs or labels associated with it. The goal of supervised learning is to learn a function that can accurately map the input data to the corresponding output labels.

There are two main types of supervised learning tasks:

1. **Classification**: In this task, the algorithm learns to predict a discrete, categorical output label based on the input data. For example, classifying an email as "spam" or "not spam", or predicting whether a person will default on a loan or not.

1. **Regressio**n: In this task, the algorithm learns to predict a continuous, numeric output value based on the input data. For example, predicting the price of a house based on its features, or forecasting the stock price of a company.

Some common supervised learning algorithms include:

1. **Linear Regression**: A simple algorithm that models the relationship between input features and a continuous target variable using a linear function.
2. **Logistic Regression**: A classification algorithm that models the probability of a binary target variable as a function of the input features.
3. **Decision Trees**: A tree-based algorithm that recursively partitions the input space to make predictions.

PROPOSED DESIGN / METHODOLOGY

Project Overview

The goal of this project is to build a supervised learning model that can predict the price of a used car based on its features. The model will be trained on a dataset of used car listings, and the performance of the model will be evaluated using appropriate metrics.

Data Exploration and Visualization

Perform exploratory data analysis to understand the characteristics of the dataset. This can include visualizing the distribution of the target variable (sale price), identifying correlations between features, and examining any potential relationships between the features and the target.

This is a high-level overview of the proposed design for a supervised learning project. The specific details and implementation steps may vary depending on the complexity of the problem, the available data, and the chosen machine learning algorithms.

Getting the Data

```
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)
```

Data Fetched

Out[1]:

Date	Open	High	Low	Close	Adj Close	Volume	company_name
2024-04-29	182.750000	183.529999	179.389999	180.960007	180.960007	54063900	AMAZON
2024-04-30	181.089996	182.990005	174.800003	175.000000	175.000000	94639800	AMAZON
2024-05-01	181.639999	185.149994	176.559998	179.000000	179.000000	94645100	AMAZON
2024-05-02	180.850006	185.100006	179.910004	184.720001	184.720001	54303500	AMAZON
2024-05-03	186.990005	187.869995	185.419998	186.210007	186.210007	39172000	AMAZON
2024-05-06	186.279999	188.750000	184.800003	188.699997	188.699997	34725300	AMAZON
2024-05-07	188.919998	189.940002	187.309998	188.759995	188.759995	34048900	AMAZON
2024-05-08	187.440002	188.429993	186.389999	188.000000	188.000000	26136400	AMAZON
2024-05-09	188.880005	191.699997	187.440002	189.500000	189.500000	43368400	AMAZON
2024-05-10	189.160004	189.889999	186.929993	187.479996	187.479996	34121700	AMAZON

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.

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

Out[5]:

	Open	High	Low	Close	Adj Close	Volume
count	251.000000	251.000000	251.000000	251.000000	251.000000	2.510000e+02
mean	181.511992	182.976136	180.124223	181.588925	180.989200	5.830329e+07
std	8.542130	8.377917	8.513766	8.487742	8.407987	1.871880e+07
min	165.350006	166.399994	164.080002	165.000000	164.776505	2.404830e+07
25%	173.989998	175.900002	172.854996	174.099998	173.399620	4.678545e+07
50%	181.270004	182.759995	179.529999	181.179993	180.914627	5.366560e+07
75%	189.294998	189.990005	187.695000	189.334999	188.653915	6.486690e+07
max	198.020004	199.619995	197.000000	198.110001	197.589523	1.632241e+08

Information About the Data

.info() method prints information about a DataFrame including the index dtype and columns, non-null values, and memory usage.

General info

```
In [6]: AAPL.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 251 entries, 2023-05-12 to 2024-05-10
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: 15.7+ 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 is the standard benchmark used by investors to track its performance over time.

Let's see a historical view of the closing price

In [7]:
```
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.

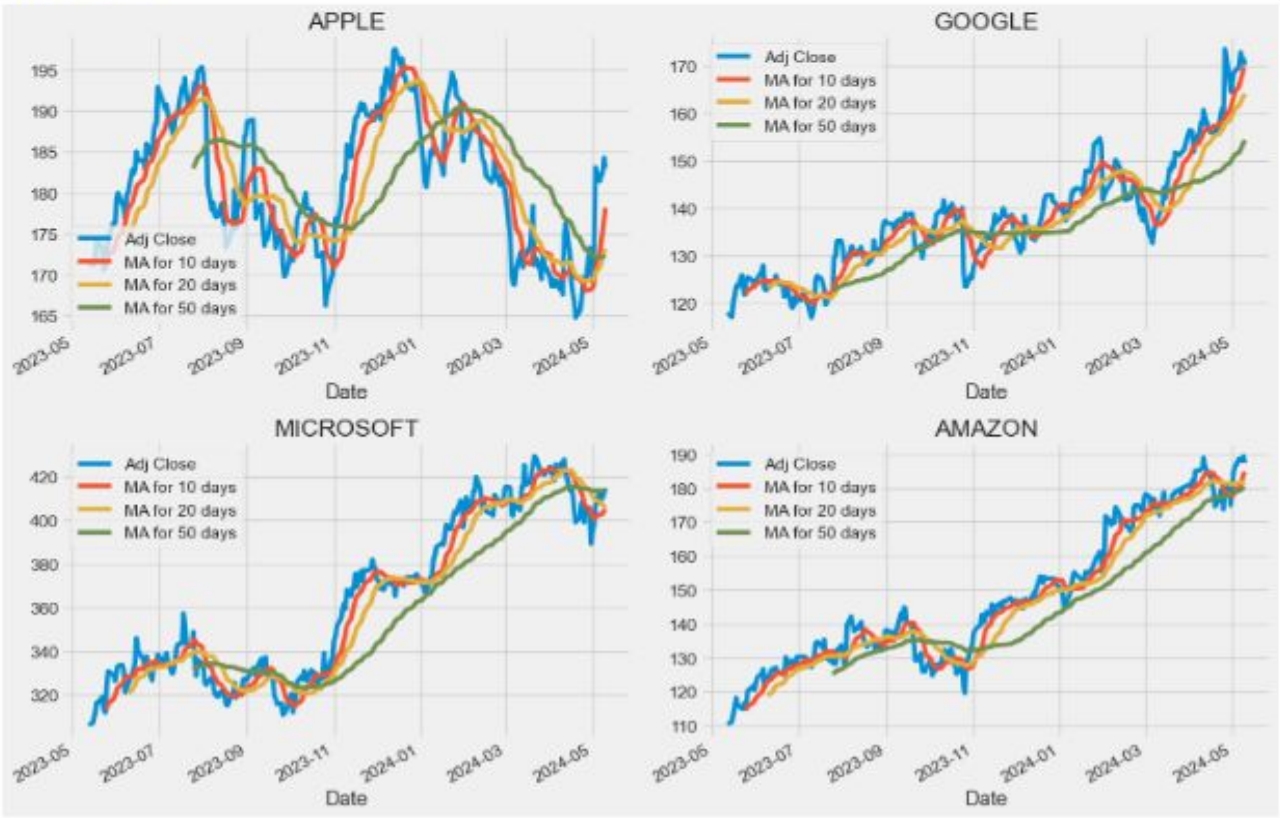
Now let's plot the total volume of stock being traded each day

In [8]:
```
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()
```

What was the moving average of the various stocks?

The moving average (MA) is a simple technical analysis tool that smooths out price data by creating a constantly updated average price. The average is taken over a specific period of time, like 10 days, 20 minutes, 30 weeks, or any time period the trader chooses.

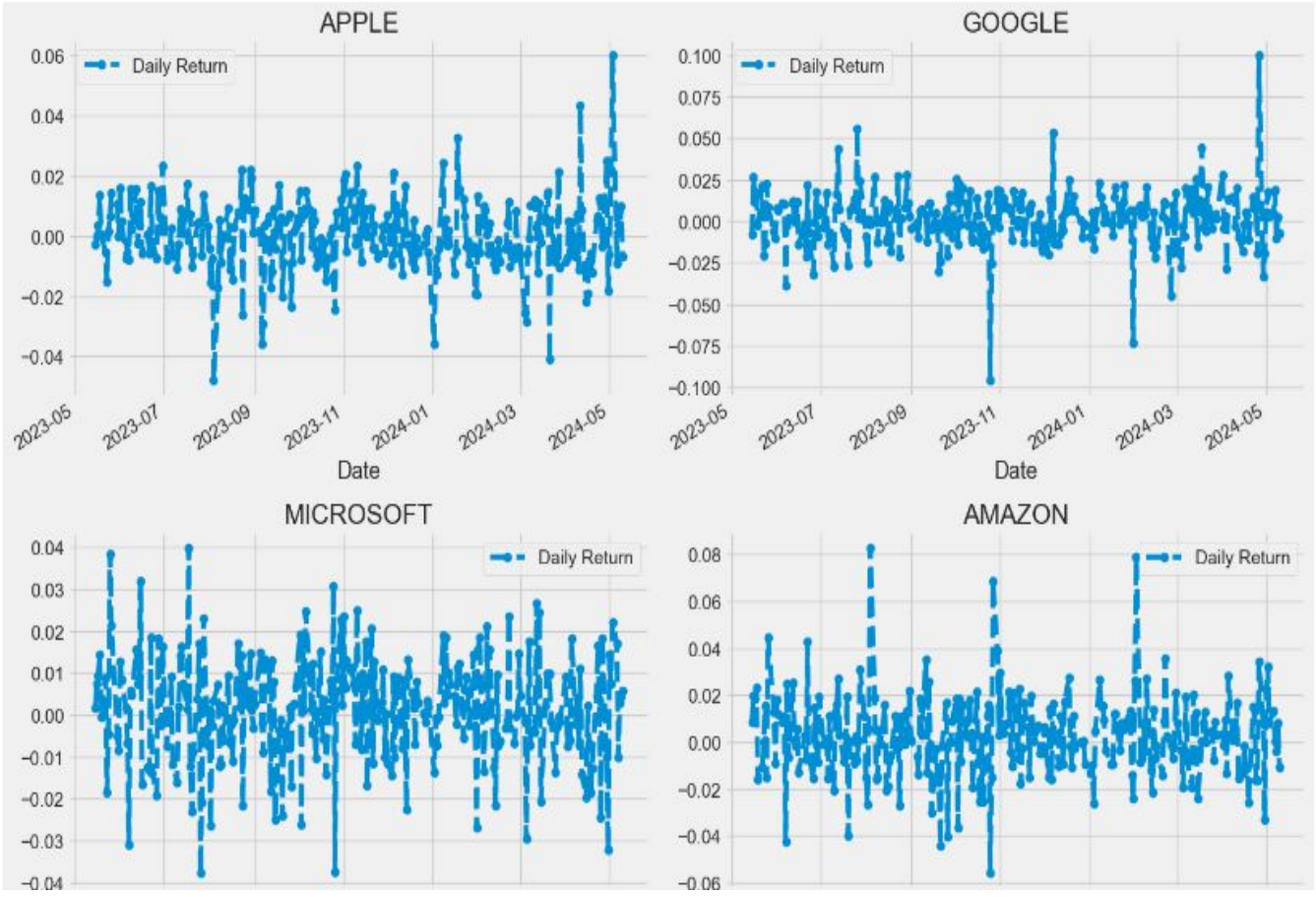
In [9]:
```
ma_day = [10, 20, 50]

for ma in ma_day:
    for company in company_list:
        column_name = f"MA for {ma} days"
        company[column_name] = company['Adj Close'].rolling(ma).mean()


fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_figheight(10)
fig.set_figwidth(15)

AAPL[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,0])
axes[0,0].set_title('APPLE')

GOOG[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,1])
axes[0,1].set_title('GOOGLE')

MSFT[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,0])
axes[1,0].set_title('MICROSOFT')

AMZN[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,1])
axes[1,1].set_title('AMAZON')

fig.tight_layout()
```



What was the daily return of the stock on average?

We're now going to analyze the risk of the stock. In order to do so we'll need to take a closer look at the daily changes of the stock, and not just its absolute value. Let's go ahead and use pandas to retrieve teh daily returns for the Apple stock.

```
In [10]: # We'll use pct_change to find the percent change for each day
for company in company_list:
    company['Daily Return'] = company['Adj Close'].pct_change()

# Then we'll plot the daily return percentage
fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_figheight(10)
fig.set_figwidth(15)

AAPL['Daily Return'].plot(ax=axes[0,0], legend=True, linestyle='--', marker='o')
axes[0,0].set_title('APPLE')

GOOG['Daily Return'].plot(ax=axes[0,1], legend=True, linestyle='--', marker='o')
axes[0,1].set_title('GOOGLE')

MSFT['Daily Return'].plot(ax=axes[1,0], legend=True, linestyle='--', marker='o')
axes[1,0].set_title('MICROSOFT')

AMZN['Daily Return'].plot(ax=axes[1,1], legend=True, linestyle='--', marker='o')
axes[1,1].set_title('AMAZON')

fig.tight_layout()
```



What was the correlation between different stocks closing prices?

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].

In [13]:
```
# 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()
```

```
[**********100%%**********]  4 of 4 completed
```

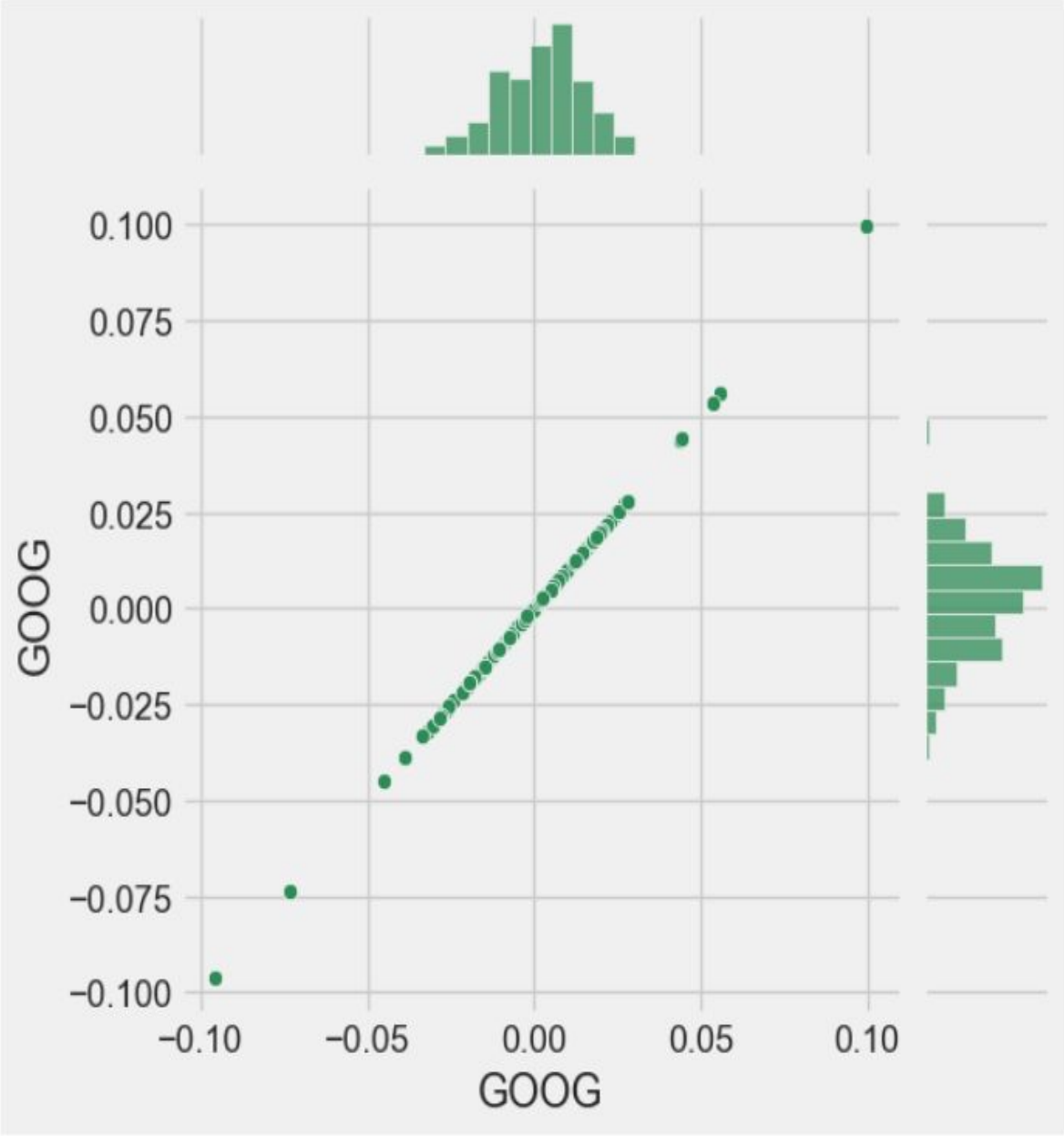
Out[13]:

Ticker	AAPL	AMZN	GOOG	MSFT
Date				
2023-05-15	NaN	NaN	NaN	NaN
2023-05-16	0.000000	0.019784	0.026761	0.007368
2023-05-17	0.003603	0.018519	0.011575	0.009451
2023-05-18	0.013666	0.022944	0.016793	0.014395
2023-05-19	0.000628	-0.016081	-0.002186	-0.000565

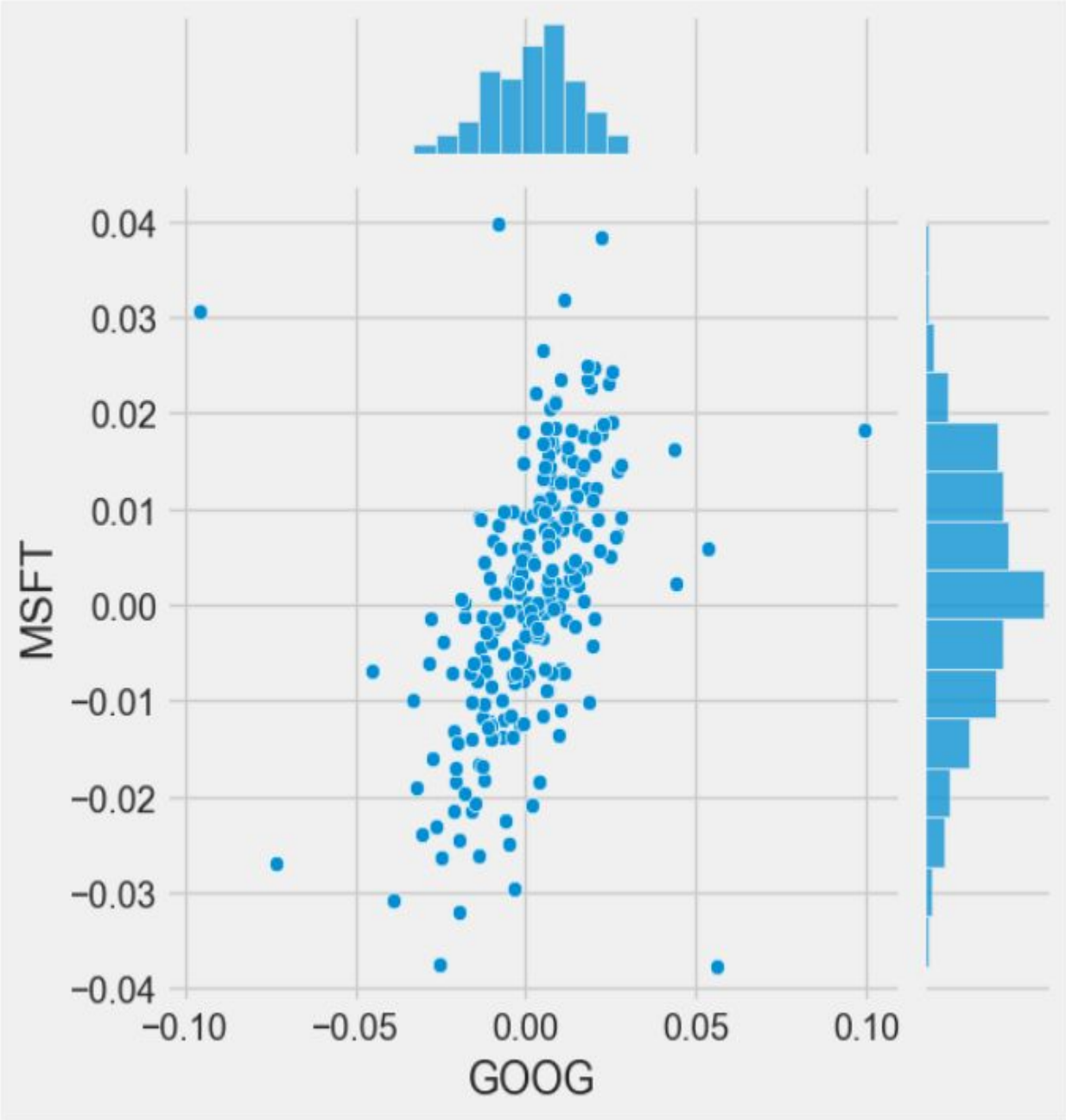
Now we can compare the daily percentage return of two stocks to check how correlated. First let's see a sotck compared to itself.

```
In [14]: # Comparing Google to itself should show a perfectly linear relationship
         sns.jointplot(x='GOOG', y='GOOG', data=tech_rets, kind='scatter', color='seagreen')
```

```
Out[14]: <seaborn.axisgrid.JointGrid at 0x1f58a165730>
```

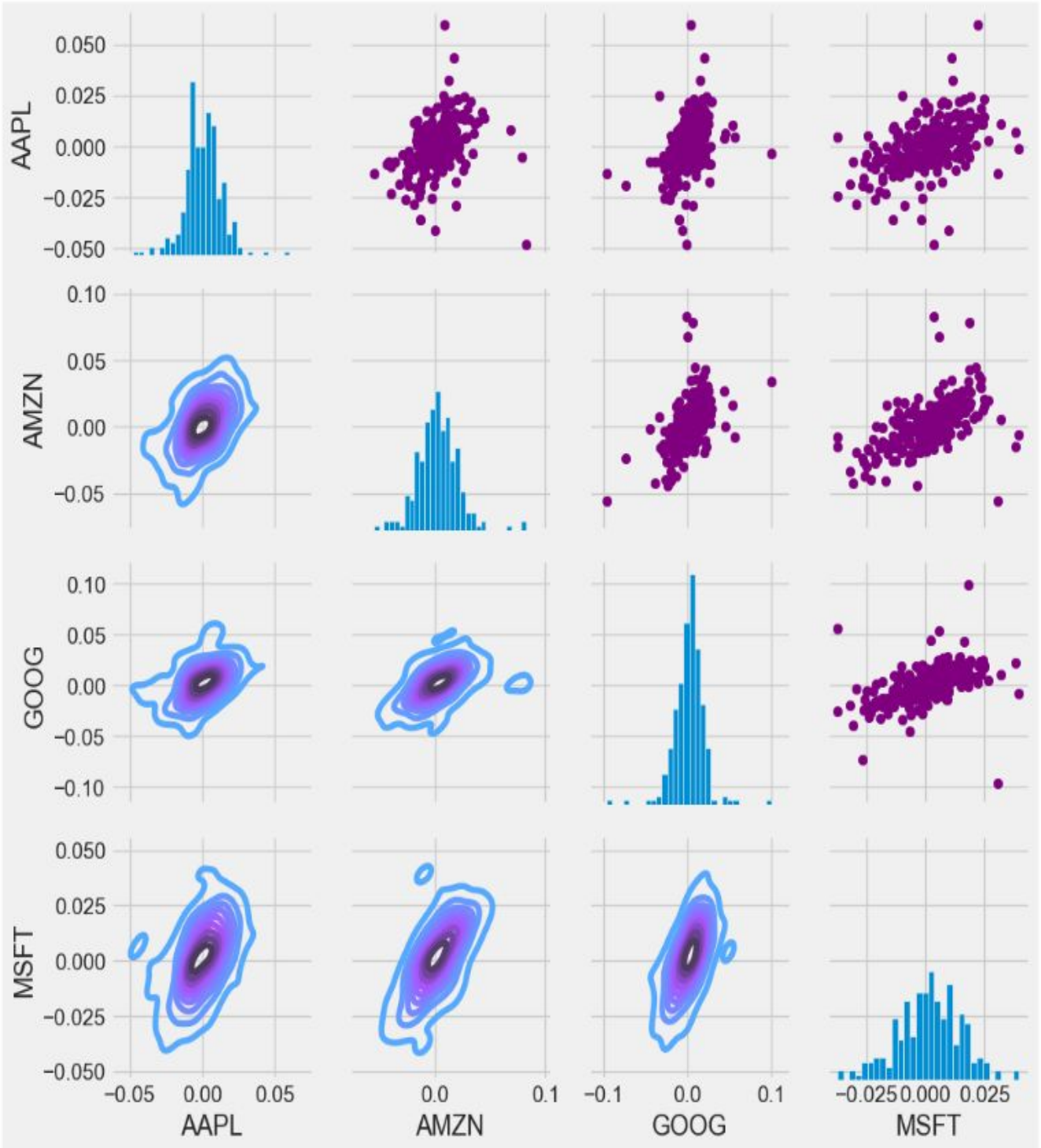


```
In [15]: # We'll use joinplot to compare the daily returns of Google and Microsoft
         sns.jointplot(x='GOOG', y='MSFT', data=tech_rets, kind='scatter')

Out[15]: <seaborn.axisgrid.JointGrid at 0x1f58a0f1e50>
```



Above we can see all the relationships on daily returns between all the stocks. A quick glance shows an interesting correlation between Google and Amazon daily returns. It might be interesting to investigate that individual comaprison.

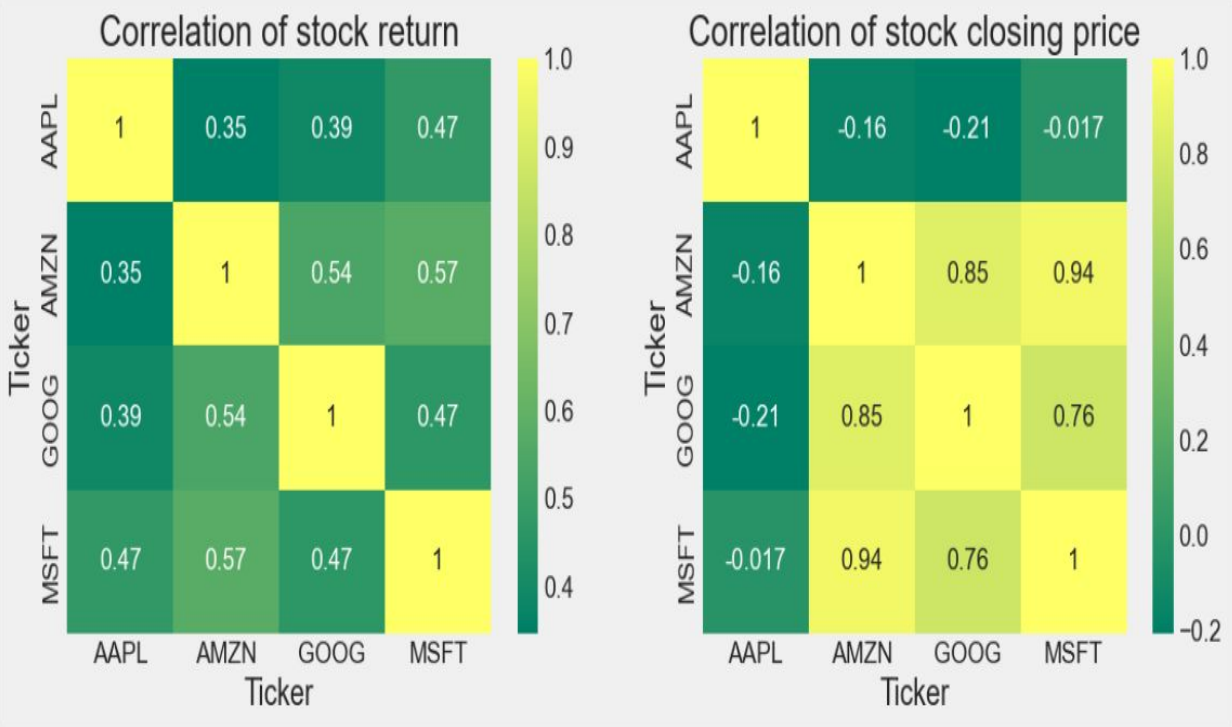
Out[18]: <seaborn.axisgrid.PairGrid at 0x1f58d52ccd0>



Finally, we could also do a correlation plot, to get actual numerical values for the correlation between the stocks' daily return values. By comparing the closing prices, we see an interesting relationship between Microsoft and Apple.

```
In [20]: plt.figure(figsize=(12, 10))

         plt.subplot(2, 2, 1)
         sns.heatmap(tech_rets.corr(), annot=True, cmap='summer')
         plt.title('Correlation of stock return')

         plt.subplot(2, 2, 2)
         sns.heatmap(closing_df.corr(), annot=True, cmap='summer')
         plt.title('Correlation of stock closing price')
```

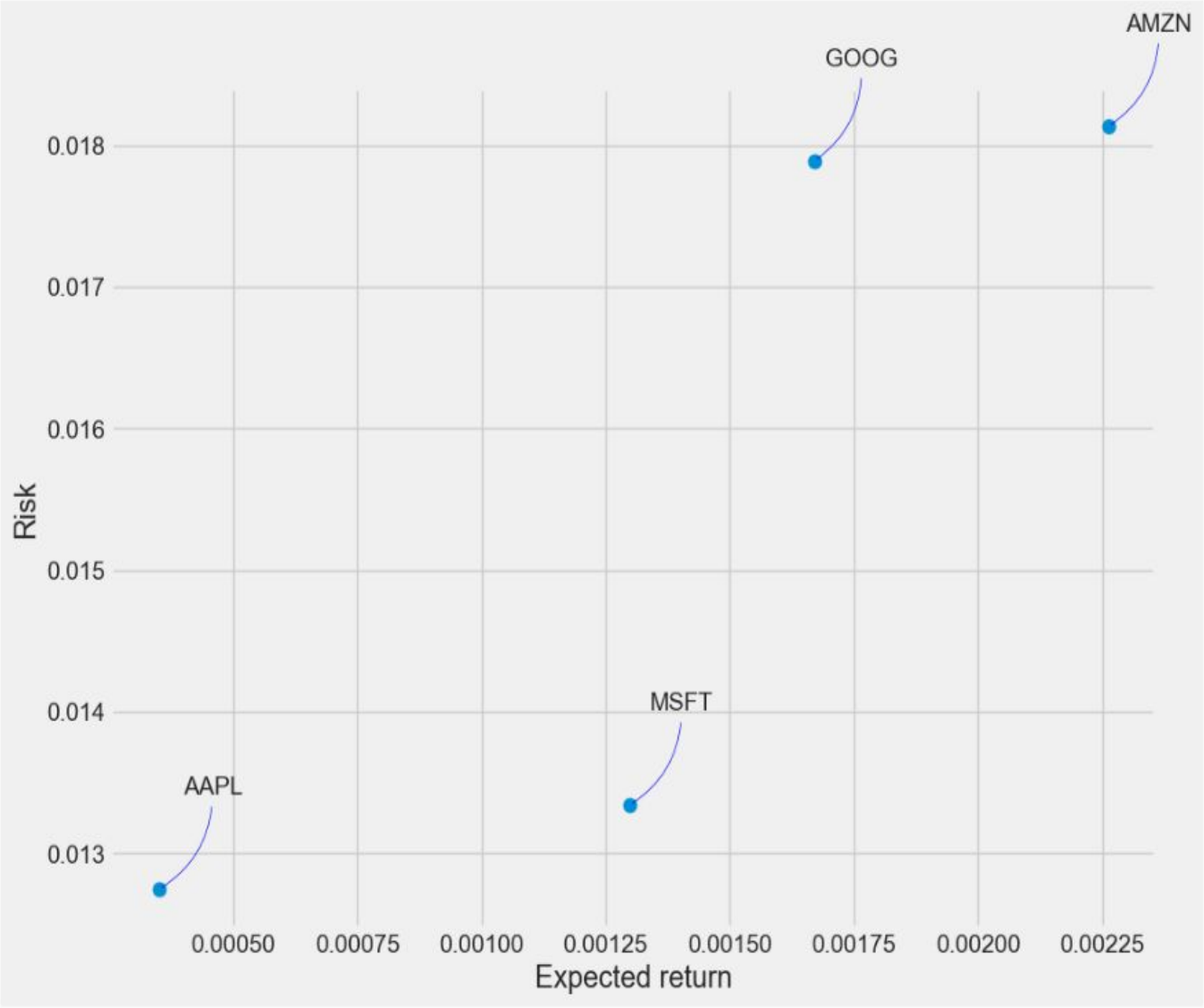
```
Out[20]: Text(0.5, 1.0, 'Correlation of stock closing price')
```



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 .

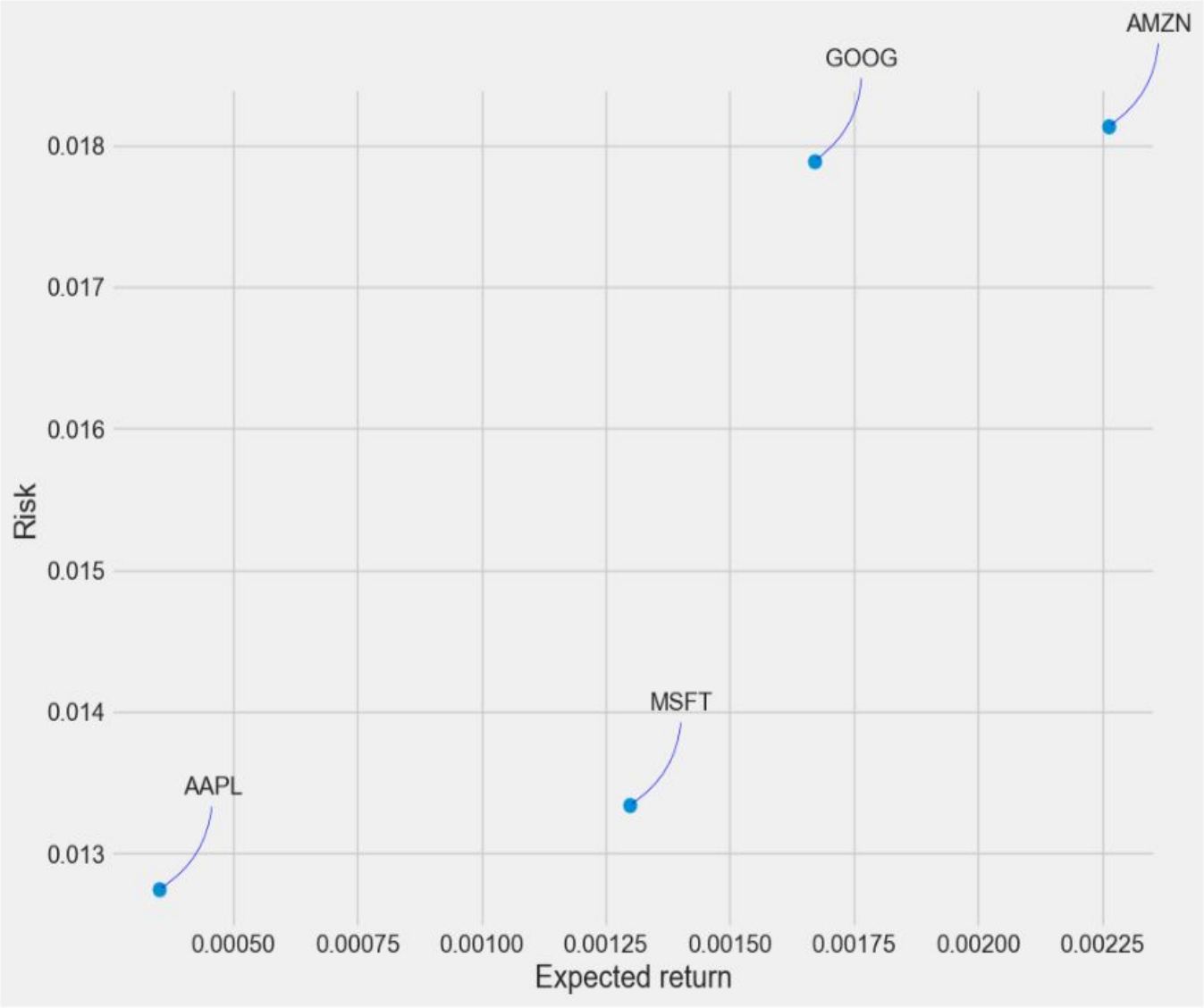
In [21]:
```
rets = tech_rets.dropna()

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='bottom',
                 arrowprops=dict(arrowstyle='-', color='blue', connectionstyle='arc3,rad=-0.3'))
```



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 .

In [21]:
```
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='bottom',
                 arrowprops=dict(arrowstyle='-', color='blue', connectionstyle='arc3,rad=-0.3'))
```



Predicting the closing price stock price of APPLE inc:

In [22]:
```
# Get the stock quote
df = pdr.get_data_yahoo('AAPL', start='2012-01-01', end=datetime.now())
# Show teh data
df
```

```
[**********100%%**********]  1 of 1 completed
```

Out[22]:

Date	Open	High	Low	Close	Adj Close	Volume
2012-01-03	14.621429	14.732143	14.607143	14.686786	12.416985	302220800
2012-01-04	14.642857	14.810000	14.617143	14.765714	12.483712	260022000
2012-01-05	14.819643	14.948214	14.738214	14.929643	12.622309	271269600
2012-01-06	14.991786	15.098214	14.972143	15.085714	12.754256	318292800
2012-01-09	15.196429	15.276786	15.048214	15.061786	12.734026	394024400
...	...	...	...	...	...	...
2024-05-06	182.350006	184.199997	180.419998	181.710007	181.463882	78569700
2024-05-07	183.449997	184.899994	181.320007	182.399994	182.152924	77305800
2024-05-08	182.850006	183.070007	181.449997	182.740005	182.492477	45057100
2024-05-09	182.559998	184.660004	182.110001	184.570007	184.320007	48983000
2024-05-10	184.899994	185.089996	182.130005	183.050003	183.050003	50727400

3109 rows × 6 columns

In [23]:
```
plt.figure(figsize=(16,6))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```

In [25]:
```
# Scale the data
from sklearn.preprocessing import MinMaxScaler

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

scaled_data
```

Out[25]:
```
array([[0.00401431],
       [0.00444289],
       [0.00533302],
       ...,
       [0.91654112],
       [0.92647801],
       [0.91822441]])
```

In [26]:
```
# Create the training data set
# Create the scaled training data set
train_data = scaled_data[0:int(training_data_len), :]
# Split the data into x_train and y_train data sets
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])
    if i<= 61:
        print(x_train)
        print(y_train)
        print()

# Convert the x_train and y_train to numpy arrays
x_train, y_train = np.array(x_train), np.array(y_train)

# Reshape the data
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
# x_train.shape
```

```
[array([0.00401431, 0.00444289, 0.00533302, 0.00618049, 0.00605056,
       0.00634339, 0.00620958, 0.00598462, 0.00567821, 0.00662652,
       0.00748175, 0.007218  , 0.00577323, 0.00715207, 0.00579457,
       0.01088518, 0.01049151, 0.01100542, 0.01211663, 0.01278955,
       0.01273332, 0.01252582, 0.01341013, 0.01424207, 0.01518457,
       0.01670691, 0.01990478, 0.01995326, 0.02173353, 0.02306387,
       0.02077746, 0.02165789, 0.02164044, 0.02410915, 0.02375813,
       0.02440779, 0.02557523, 0.0262249 , 0.02809631, 0.02945961,
       0.02985329, 0.02999098, 0.02765997, 0.02709757, 0.02718096,
       0.02937236, 0.02998905, 0.03131358, 0.03443581, 0.03860139,
       0.0378218 , 0.03782373, 0.04083544, 0.04177794, 0.04110694,
       0.04049413, 0.03985611, 0.04197573, 0.0434302 , 0.04403914])]
[0.042534249860459186]

[array([0.00401431, 0.00444289, 0.00533302, 0.00618049, 0.00605056,
       0.00634339, 0.00620958, 0.00598462, 0.00567821, 0.00662652,
       0.00748175, 0.007218  , 0.00577323, 0.00715207, 0.00579457,
       0.01088518, 0.01049151, 0.01100542, 0.01211663, 0.01278955,
       0.01273332, 0.01252582, 0.01341013, 0.01424207, 0.01518457,
       0.01670691, 0.01990478, 0.01995326, 0.02173353, 0.02306387,
       0.02077746, 0.02165789, 0.02164044, 0.02410915, 0.02375813,
```

In [37]:
```
from sklearn.metrics import r2_score

# Predictions on training data
y_train_pred = model.predict(x_train)
train_r2_score = r2_score(y_train, y_train_pred)

print("R^2 score on training data:", train_r2_score)
```

```
91/91 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step
R^2 score on training data: 0.9949746857947416
```

In [29]:
```
# Create the testing data set
# Create a new array containing scaled values from index 1543 to 2002
test_data = scaled_data[training_data_len - 60: , :]
# Create the data sets x_test and y_test
x_test = []
y_test = dataset[training_data_len:, :]
for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])

# Convert the data to a numpy array
x_test = np.array(x_test)

# Reshape the data
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1 ))

# Get the models predicted price values
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)

# Get the root mean squared error (RMSE)
rmse = np.sqrt(np.mean(((predictions - y_test) ** 2)))
rmse
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 145ms/step
```

Out[29]:
```
6.819433672925514
```

```
: # 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()
```

```
C:\Users\saksh\AppData\Local\Temp\ipykernel_20652\2388977846.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  valid['Predictions'] = predictions
```

WORKFLOW

1. Data Acquisition:

• Utilize the yfinance library to fetch historical stock market data from Yahoo Finance for technology stocks (Apple, Amazon, Google, Microsoft).

2. Data Exploration:

• Load the acquired data into a pandas DataFrame and inspect its structure.

• Visualize the closing prices and trading volume of each stock over time using Matplotlib and Seaborn.

3. Data Preprocessing:

• Compute descriptive statistics to understand the central tendency, dispersion, and shape of the data.

• Scale the data using MinMaxScaler to prepare it for modeling.

4. Moving Average Calculation:

• Calculate moving averages for different time periods (e.g., 10, 20, and 50 days) to identify trends and smooth out noise in the data.

5. Risk Analysis:

• Compute daily returns for each stock and visualize them using histograms and kernel density estimation plots.

• Measure the correlation between different stocks' returns to assess their interdependencies.

6. Predictive Modeling:

- Split the data into training and testing sets.
- Implement a Long Short-Term Memory (LSTM) neural network model using Keras to predict future stock prices.
- Train the model on the training data and evaluate its performance using root mean squared error (RMSE).
- Visualize the predicted prices alongside the actual prices to assess the model's accuracy.

7. Conclusion and Recommendations:

- Summarize the findings of the analysis, including insights into stock trends, risk factors, and predictive modeling results.

- Offer recommendations or insights for investors based on the analysis conducted.

SUMMARY

In this Project , you discovered and explored stock data.

Specifically, you learned:

• How to load stock market data from the YAHOO Finance website using yfinance.

• How to explore and visualize time-series data using Pandas, Matplotlib, and Seaborn.

• How to measure the correlation between stocks.

• How to measure the risk of investing in a particular stock.