big-tech-stock-analysis-3

June 28, 2024

1 Big Tech Giants Stock Price Data Analizi

1.1 Veri Setleri Hakkında

Merhaba! Elimizde `big_tech_companies.csv` ve `big_tech_stock_prices.csv` adında iki harika veri seti var. Bu veri setleri, büyük teknoloji şirketlerinin hisse senedi verilerini içeriyor ve bu verilerle çok çeşitli analizler ve modellemeler yapabiliriz.

1.1.1 big_tech_companies.csv

Bu dosya, büyük teknoloji şirketlerinin hisse senedi sembollerini ve şirket isimlerini içeriyor. İşte ilk birkaç satır:

stock_symbol	company
AAPL	Apple Inc.
ADBE	Adobe Inc.
AMZN	Amazon.com, Inc.
CRM	Salesforce, Inc.
CSCO	Cisco Systems, Inc.

1.1.2 big_tech_stock_prices.csv

Bu dosya ise tarihsel hisse senedi fiyat verilerini içeriyor. İlk birkaç satırı aşağıda görebilirsiniz:

stock_symbol	date	open	high	low	close	adj_close	volume
AAPL	2010-01-04	7.622500	7.660714	7.585000	7.643214	6.515213	493729600
AAPL	2010-01-05	7.664286	7.699643	7.616071	7.656429	6.526476	601904800
AAPL	2010-01-06	7.656429	7.686786	7.526786	7.534643	6.422664	552160000
AAPL	2010-01-07	7.562500	7.571429	7.466071	7.520714	6.410790	477131200
AAPL	2010-01-08	7.510714	7.571429	7.466429	7.570714	6.453412	447610800

Bu veri seti sütunları şunları içeriyor: - **stock_symbol:** Hisse senedi sembolü - **date:** Tarih - **open:** Açılış fiyatı - **high:** En yüksek fiyat - **low:** En düşük fiyat - **close:** Kapanış fiyatı - **adj_close:** Düzeltilmiş kapanış fiyatı - **volume:** İşlem hacmi

1.2 Başlangıç Adımları

Bu veri setleri ile neler yapabiliriz? İşte bazı fikirler:

1.2.1 Veri Temizleme ve Hazırlama

- Eksik değerleri kontrol edip, tarih sütunlarını datetime formatına dönüştürebiliriz.

1.2.2 Keşifsel Veri Analizi (EDA)

- Hisse senedi fiyat trendlerini görselleştirip, işlem hacmini analiz edebiliriz.

1.2.3 Hisse Senedi Fiyat Tahmini

- Zaman serisi modelleri kullanarak gelecekteki fiyatları tahmin edebiliriz.

1.2.4 Duygu Analizi Entegrasyonu

- Haber makalelerinden veya sosyal medyadan duygu verilerini toplayıp analiz edebiliriz.

1.2.5 Risk ve Portföy Analizi

- Hisse senetlerinin risk ve getirisini hesaplayıp, portföy optimizasyonu yapabiliriz.

1.3 İleri Düzey Özellikler

Daha ileri düzeyde neler yapabiliriz? İşte bazı yenilikçi fikirler:

1.3.1 Makroekonomik Göstergeler

- Faiz oranları, enflasyon gibi makroekonomik verileri toplayarak hisse senedi fiyatlarına etkilerini modelleyebiliriz.

1.3.2 Sentiment Analizi ve Sosyal Medya Verileri

- Twitter, haber kaynakları ve finansal forumlardan piyasa duyarlılığını analiz edip, fiyat tahmin modellerine entegre edebiliriz.

1.3.3 Anomali Tespiti

- Zaman serisi verilerinde anormallikleri tespit edip, potansiyel nedenlerini belirleyebiliriz.

1.3.4 Volatilite Modellemesi

- GARCH modelleri ile hisse senedi fiyat volatilitesini modelleyebilir ve tahmin edebiliriz.

1.3.5 Otomatik Alım Satım ve Robo-Danışmanlık

- Otomatik alım satım algoritmaları geliştirebilir ve kişiselleştirilmiş yatırım önerileri sunan robo-danışmanlık hizmetleri oluşturabiliriz.

1.3.6 Model Açıklanabilirliği

- SHAP veya LIME gibi tekniklerle makine öğrenimi model kararlarını açıklayıp, modellerdeki önyargıları analiz edebiliriz.

1.3.7 Blok Zinciri ve Kripto Paralar

- Kripto para piyasaları için analiz ve tahmin modelleri geliştirip, finansal analizlerde blok zinciri teknolojisi ve akıllı sözleşmeleri kullanabiliriz.

1.3.8 Görselleştirme ve Panolar

- Etkileşimli panolar oluşturup, fiyat trendlerini ve tahmin analizlerini görselleştirebiliriz.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime

# Zaman serisi analizi için
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
import prophet

# Makine öğrenimi modelleri için
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor

# Duygu analizi ve NLP için
from textblob import TextBlob
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')

# Verileri görselleştirmek için
import plotly.express as px
import plotly.graph_objs as go
import plotly.figure_factory as ff

# Uyarıları kapatmak için
import warnings
warnings.filterwarnings("ignore")

print("Gerekli kütüphaneler başarıyla import edildi!")
```

```
/opt/conda/lib/python3.10/site-packages/nltk/twitter/__init__.py:20:
UserWarning: The twython library has not been installed. Some functionality from
the twitter package will not be available.
  warnings.warn("The twython library has not been installed. "

[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /usr/share/nltk_data…
[nltk_data]   Package vader_lexicon is already up-to-date!
Gerekli kütüphaneler başarıyla import edildi!
```

[2]:
```
# Veri setlerini yükleyelim
big_tech_companies = pd.read_csv('/kaggle/input/
  ↪big-tech-giants-stock-price-data/big_tech_companies.csv')
big_tech_stock_prices = pd.read_csv('/kaggle/input/
  ↪big-tech-giants-stock-price-data/big_tech_stock_prices.csv')
```

[3]:
```
# İlk birkaç satırı görüntüleyelim
print("Big Tech Companies Dataset:")
print(big_tech_companies.head())

print("\nBig Tech Stock Prices Dataset:")
print(big_tech_stock_prices.head())
```

```
Big Tech Companies Dataset:
  stock_symbol              company
0         AAPL           Apple Inc.
1         ADBE           Adobe Inc.
2         AMZN     Amazon.com, Inc.
3          CRM     Salesforce, Inc.
4         CSCO  Cisco Systems, Inc.

Big Tech Stock Prices Dataset:
  stock_symbol        date      open      high       low     close  adj_close  \
0         AAPL  2010-01-04  7.622500  7.660714  7.585000  7.643214   6.515213
1         AAPL  2010-01-05  7.664286  7.699643  7.616071  7.656429   6.526476
2         AAPL  2010-01-06  7.656429  7.686786  7.526786  7.534643   6.422664
3         AAPL  2010-01-07  7.562500  7.571429  7.466071  7.520714   6.410790
4         AAPL  2010-01-08  7.510714  7.571429  7.466429  7.570714   6.453412

      volume
0  493729600
1  601904800
2  552160000
3  477131200
4  447610800
```

[4]:
```
# Veri setlerinin genel bilgilerini görüntüleyelim
print("\nBig Tech Companies Dataset Info:")
```

```
print(big_tech_companies.info())

print("\nBig Tech Stock Prices Dataset Info:")
print(big_tech_stock_prices.info())
```

```
Big Tech Companies Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   stock_symbol  14 non-null     object
 1   company       14 non-null     object
dtypes: object(2)
memory usage: 352.0+ bytes
None

Big Tech Stock Prices Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45088 entries, 0 to 45087
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   stock_symbol  45088 non-null  object 
 1   date          45088 non-null  object 
 2   open          45088 non-null  float64
 3   high          45088 non-null  float64
 4   low           45088 non-null  float64
 5   close         45088 non-null  float64
 6   adj_close     45088 non-null  float64
 7   volume        45088 non-null  int64  
dtypes: float64(5), int64(1), object(2)
memory usage: 2.8+ MB
None
```

[5]:
```
# Veri setlerinin özet istatistiklerini görüntüleyelim
print("\nBig Tech Companies Dataset Description:")
print(big_tech_companies.describe())

print("\nBig Tech Stock Prices Dataset Description:")
print(big_tech_stock_prices.describe())
```

```
Big Tech Companies Dataset Description:
       stock_symbol     company
count            14          14
unique           14          14
```

```
top             AAPL  Apple Inc.
freq               1           1

Big Tech Stock Prices Dataset Description:
                open          high           low         close     adj_close  \
count  45088.000000  45088.000000  45088.000000  45088.000000  45088.000000
mean      89.266584     90.369825     88.111930     89.271306     85.209631
std      101.626955    103.001073    100.124399    101.592916    100.995967
min        1.076000      1.108667      0.998667      1.053333      1.053333
25%       25.670000     25.930135     25.360001     25.660000     22.076433
50%       47.930000     48.459999     47.465000     47.970001     45.377333
75%      128.662502    129.848900    127.253945    128.640609    113.672460
max      696.280029    700.989990    686.090027    691.690002    691.690002

             volume
count  4.508800e+04
mean   5.297813e+07
std    9.324730e+07
min    5.892000e+05
25%    9.629425e+06
50%    2.646315e+07
75%    5.839768e+07
max    1.880998e+09
```

[6]:
```
# Veri setlerindeki eşsiz şirket ve sembol sayısını görüntüleyelim
print("\nUnique Companies in Big Tech Companies Dataset:")
print(big_tech_companies['company'].nunique())

print("\nUnique Stock Symbols in Big Tech Stock Prices Dataset:")
print(big_tech_stock_prices['stock_symbol'].nunique())
```

```
Unique Companies in Big Tech Companies Dataset:
14

Unique Stock Symbols in Big Tech Stock Prices Dataset:
14
```

[7]:
```
# Eksik değerleri kontrol edelim
print("\nMissing Values in Big Tech Companies Dataset:")
print(big_tech_companies.isnull().sum())

print("\nMissing Values in Big Tech Stock Prices Dataset:")
print(big_tech_stock_prices.isnull().sum())
```

```
Missing Values in Big Tech Companies Dataset:
stock_symbol    0
```

```
company         0
dtype: int64

Missing Values in Big Tech Stock Prices Dataset:
stock_symbol    0
date            0
open            0
high            0
low             0
close           0
adj_close       0
volume          0
dtype: int64
```

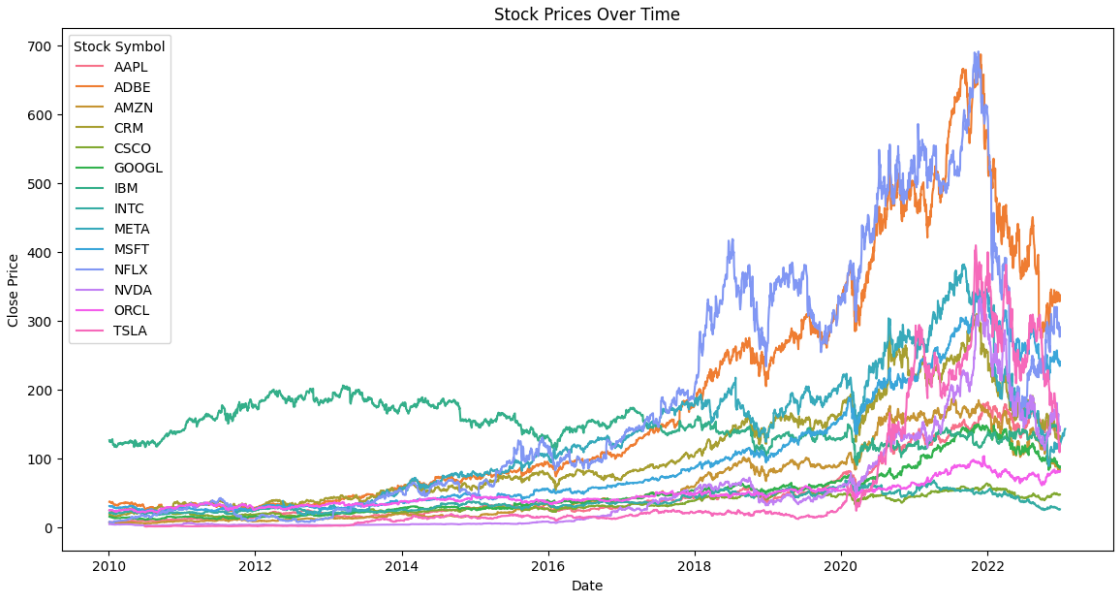
[8]:
```
# Hisse senedi sembollerinin sayısını görüntüleyelim
print("\nStock Symbol Counts in Big Tech Stock Prices Dataset:")
print(big_tech_stock_prices['stock_symbol'].value_counts())
```

```
Stock Symbol Counts in Big Tech Stock Prices Dataset:
stock_symbol
AAPL     3271
ADBE     3271
AMZN     3271
CRM      3271
CSCO     3271
GOOGL    3271
IBM      3271
INTC     3271
MSFT     3271
NFLX     3271
NVDA     3271
ORCL     3271
TSLA     3148
META     2688
Name: count, dtype: int64
```

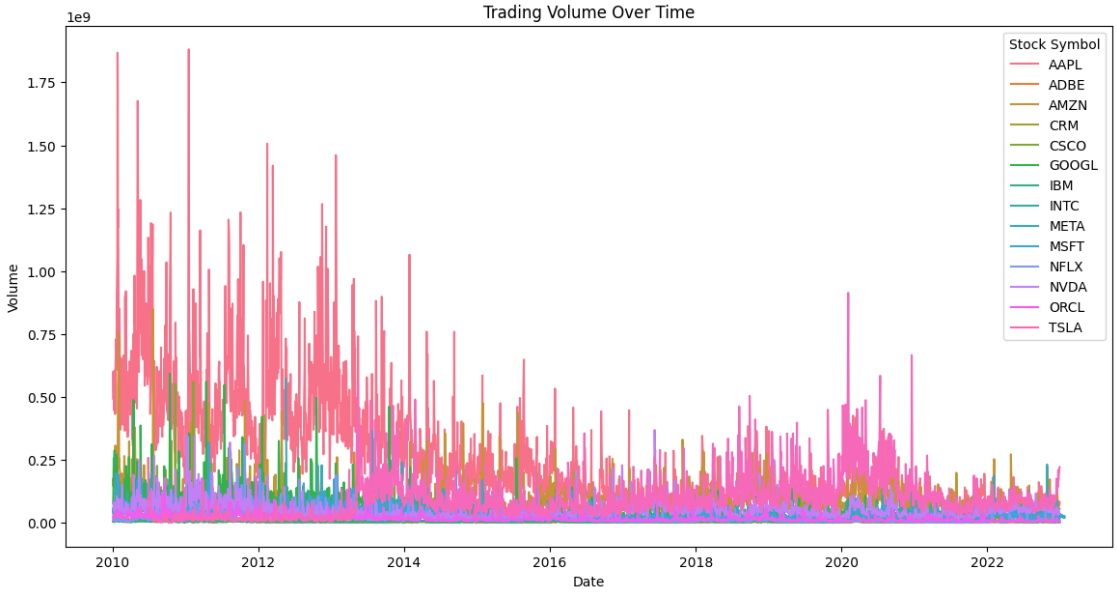
[9]:
```
# Tarih sütununu datetime formatına dönüştürelim
big_tech_stock_prices['date'] = pd.to_datetime(big_tech_stock_prices['date'])
```

[10]:
```
# Hisse senedi fiyatlarını görselleştirelim
plt.figure(figsize=(14, 7))
sns.lineplot(data=big_tech_stock_prices, x='date', y='close',␣
 ↪hue='stock_symbol')
plt.title('Stock Prices Over Time')
plt.xlabel('Date')
plt.ylabel('Close Price')
```

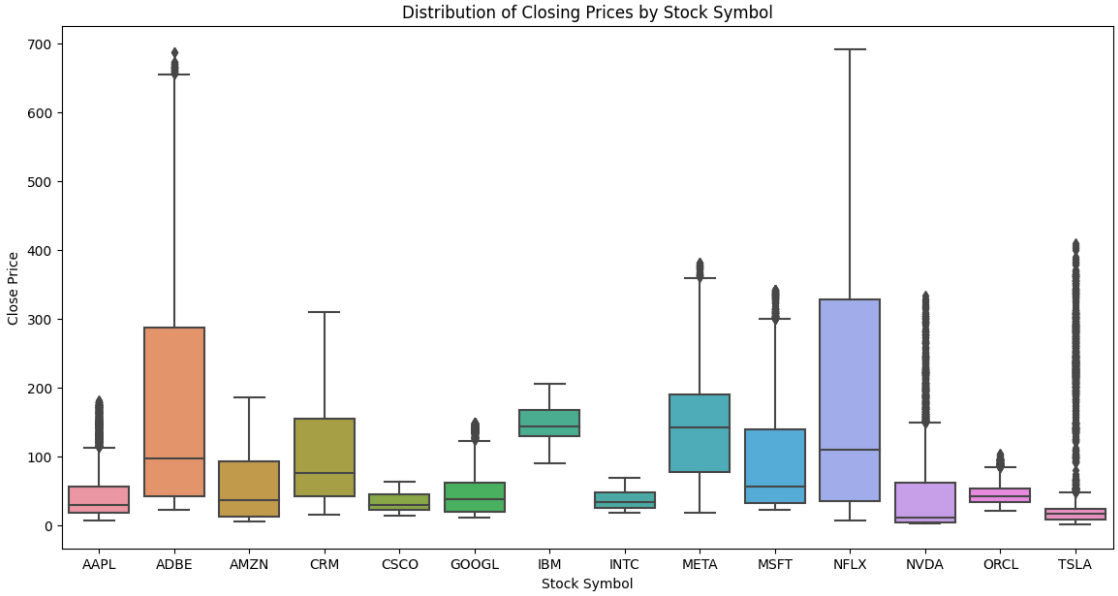
```
plt.legend(title='Stock Symbol')
plt.show()
```



[11]:
```
# Hacim verilerini görselleştirelim
plt.figure(figsize=(14, 7))
sns.lineplot(data=big_tech_stock_prices, x='date', y='volume',␣
 ↪hue='stock_symbol')
plt.title('Trading Volume Over Time')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.legend(title='Stock Symbol')
plt.show()
```

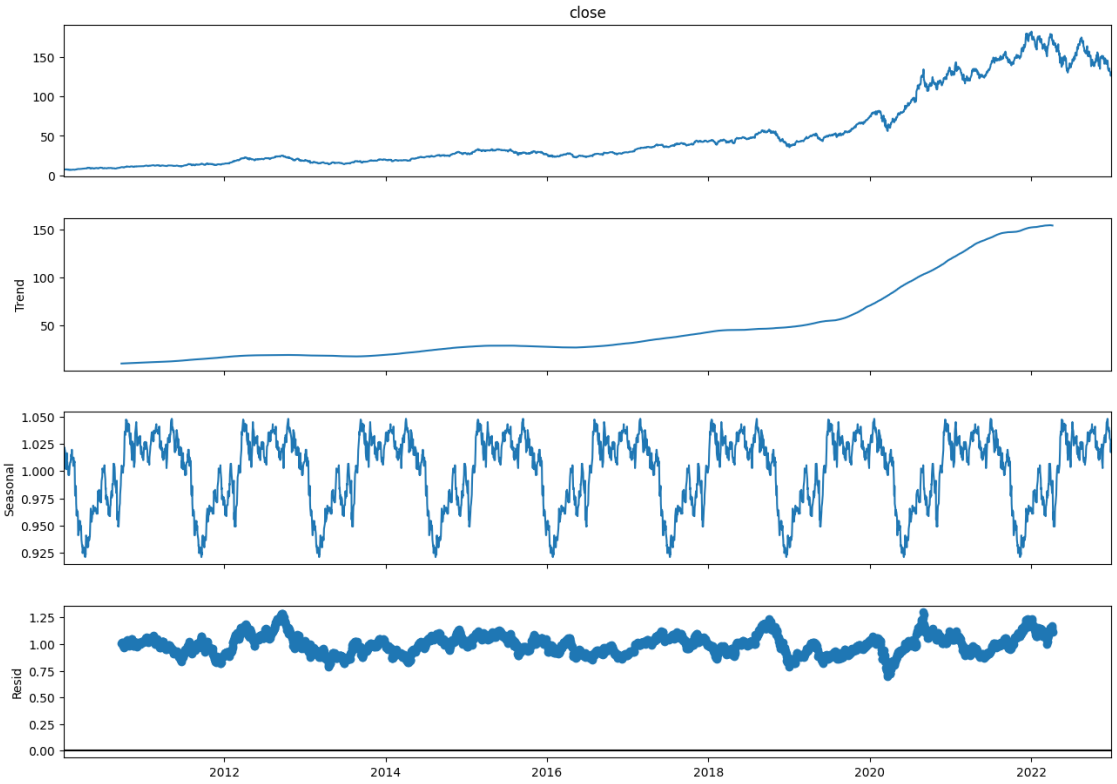


[12]:
```
# Farklı şirketlerin kapanış fiyatlarının dağılımını inceleyelim
plt.figure(figsize=(14, 7))
sns.boxplot(data=big_tech_stock_prices, x='stock_symbol', y='close')
plt.title('Distribution of Closing Prices by Stock Symbol')
plt.xlabel('Stock Symbol')
plt.ylabel('Close Price')
plt.show()
```



[13]:
```
# Belirli bir şirketin (örneğin Apple) zaman serisi analizi
apple_stock = big_tech_stock_prices[big_tech_stock_prices['stock_symbol'] ==␣
 ↪'AAPL']
apple_stock.set_index('date', inplace=True)
```

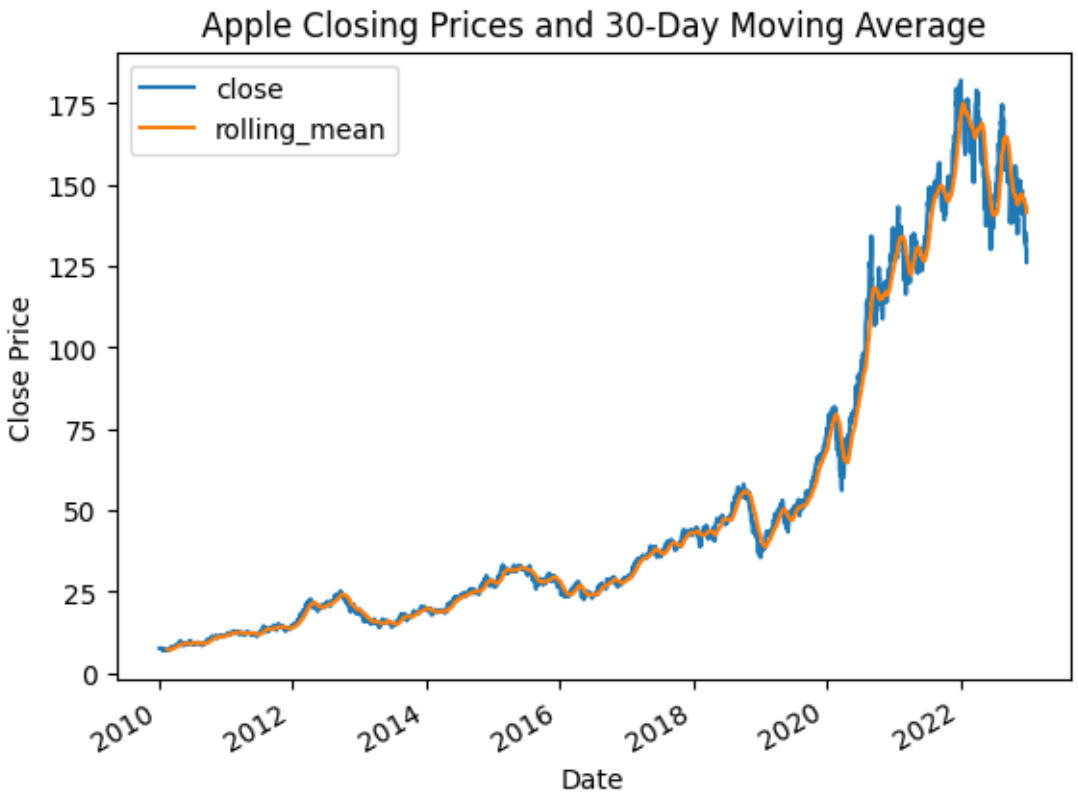
[14]:
```
# Apple hisse senedi fiyatlarının zaman serisi dekompozisyonu
decomposition = seasonal_decompose(apple_stock['close'],␣
 ↪model='multiplicative', period=365)
fig = decomposition.plot()
fig.set_size_inches(14, 10)
plt.show()
```



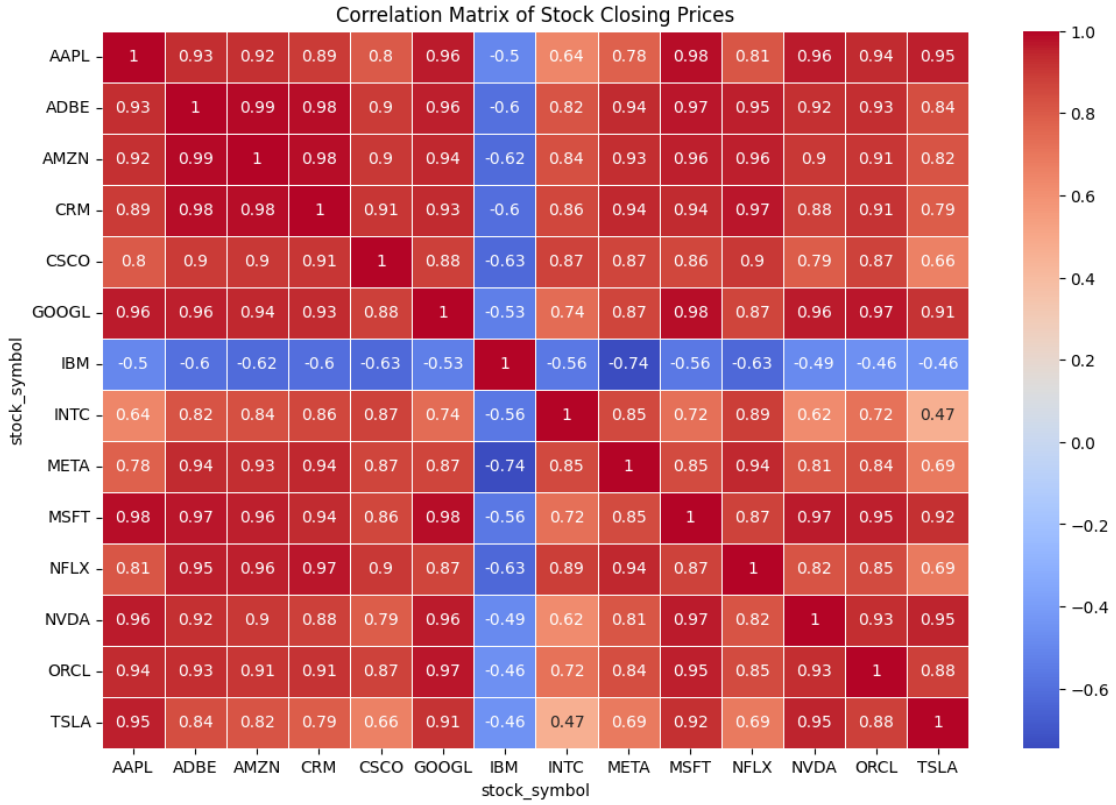
[15]:
```
# Trend ve mevsimselliği görselleştirelim
plt.figure(figsize=(14, 7))
apple_stock['close'].plot()
plt.title('Apple Closing Prices')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.show()
```

[16]:
```
# Apple hisse senedi fiyatlarının hareketli ortalamalarını ekleyelim
apple_stock['rolling_mean'] = apple_stock['close'].rolling(window=30).mean()

plt.figure(figsize=(14, 7))
apple_stock[['close', 'rolling_mean']].plot()
plt.title('Apple Closing Prices and 30-Day Moving Average')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.show()
```

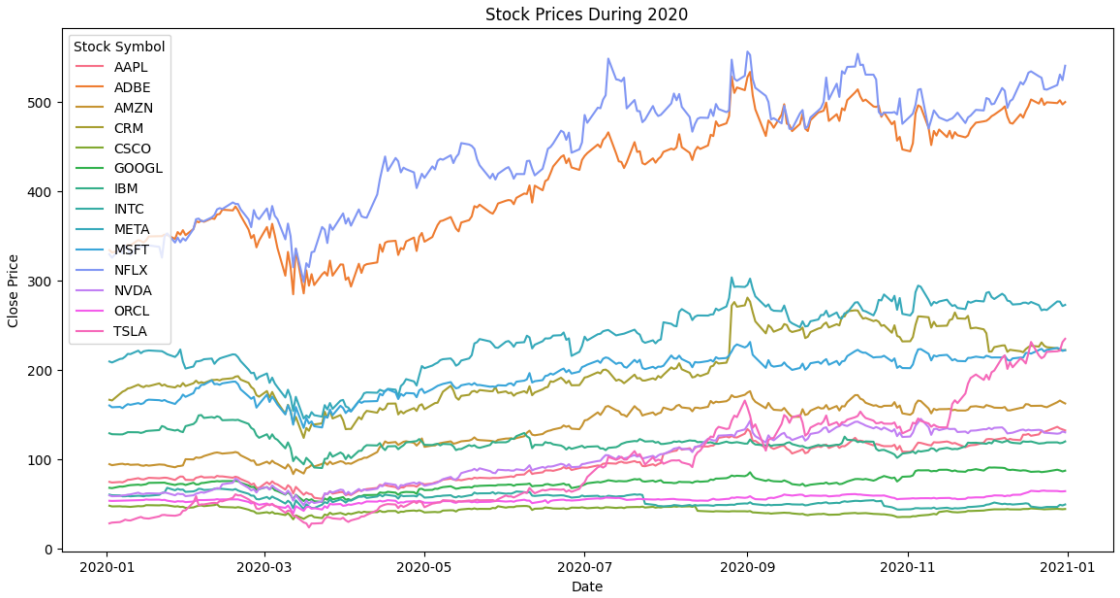
```
<Figure size 1400x700 with 0 Axes>
```



[17]:
```
# Hisse senedi fiyatlarının korelasyon matrisi
pivot_table = big_tech_stock_prices.pivot(index='date', columns='stock_symbol',␣
 ↪values='close')
correlation_matrix = pivot_table.corr()

plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix of Stock Closing Prices')
plt.show()
```



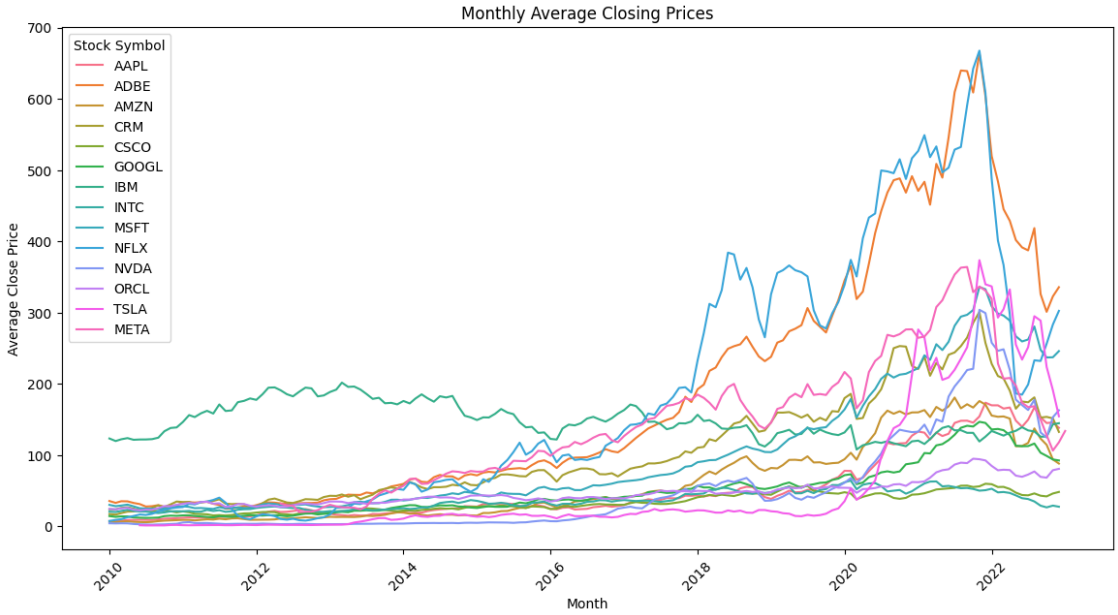
[18]:
```
# Belirli bir dönemde (örneğin 2020 yılı) fiyat değişimlerini inceleme
big_tech_stock_prices_2020 =␣
 ↪big_tech_stock_prices[(big_tech_stock_prices['date'] >= '2020-01-01') &␣
 ↪(big_tech_stock_prices['date'] <= '2020-12-31')]

plt.figure(figsize=(14, 7))
sns.lineplot(data=big_tech_stock_prices_2020, x='date', y='close',␣
 ↪hue='stock_symbol')
plt.title('Stock Prices During 2020')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend(title='Stock Symbol')
plt.show()
```



[19]:
```
# Aylık ortalama fiyatların analizi için month sütununu ekleyelim
big_tech_stock_prices['month'] = big_tech_stock_prices['date'].dt.
 ↪to_period('M').dt.to_timestamp()
```

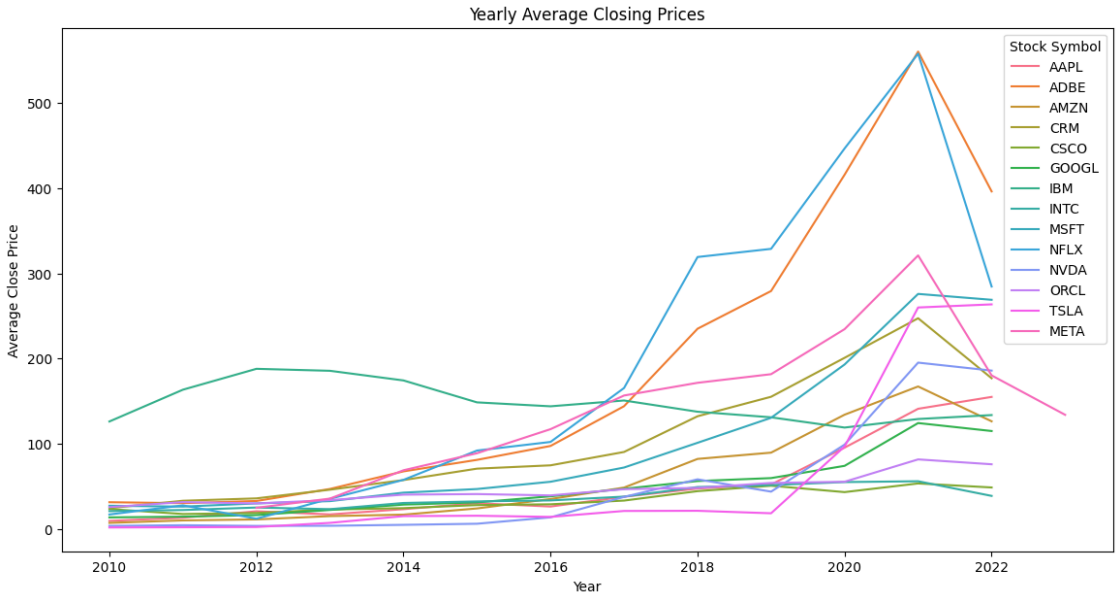
[20]:
```
# Aylık ortalama fiyatları hesaplayalım
monthly_avg_prices = big_tech_stock_prices.groupby(['month', 'stock_symbol']).
 ↪mean().reset_index()
```

[21]:
```
# Aylık ortalama fiyatları görselleştirelim
plt.figure(figsize=(14, 7))
sns.lineplot(data=monthly_avg_prices, x='month', y='close', hue='stock_symbol')
plt.title('Monthly Average Closing Prices')
plt.xlabel('Month')
plt.ylabel('Average Close Price')
plt.xticks(rotation=45)
plt.legend(title='Stock Symbol')
plt.show()
```

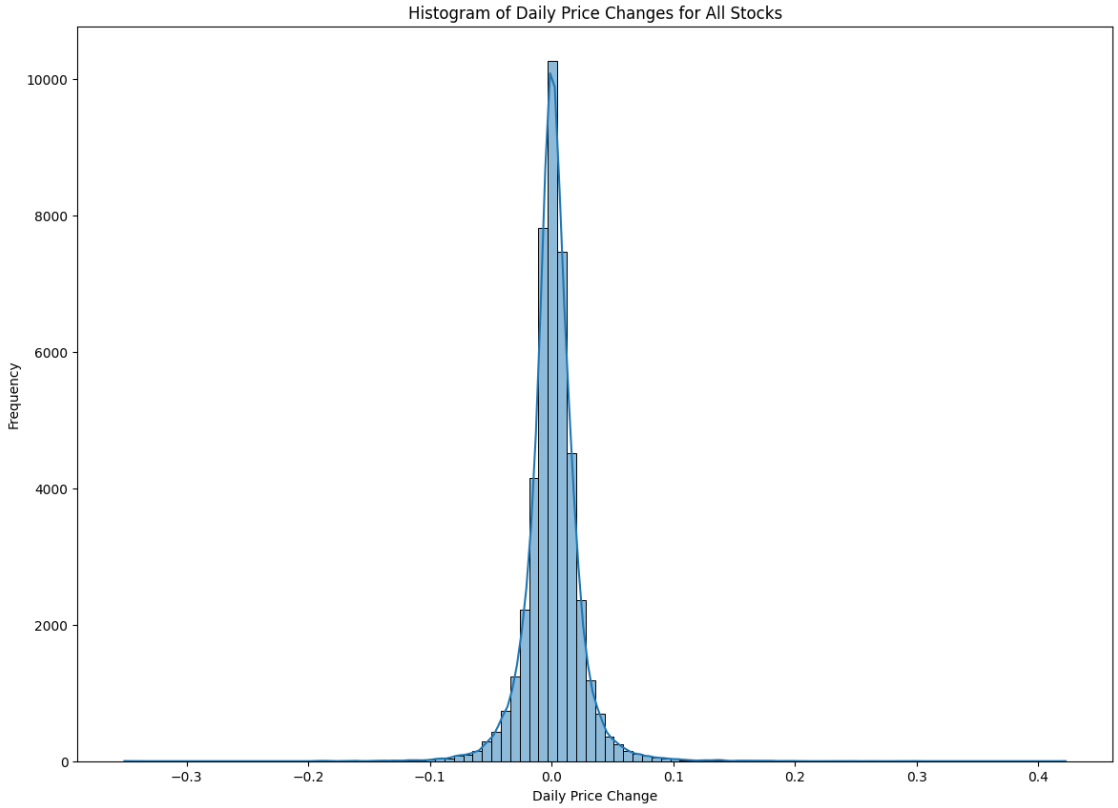


[22]:
```
# Yıllık ortalama fiyatların analizi için year sütununu ekleyelim
big_tech_stock_prices['year'] = big_tech_stock_prices['date'].dt.year
```

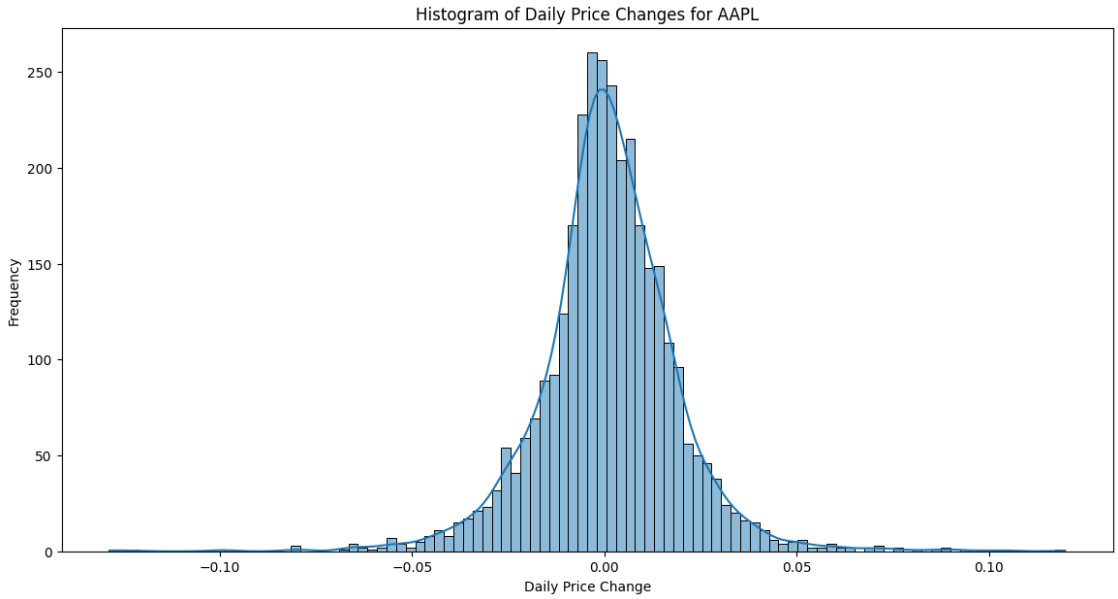
[23]:
```
# Yıllık ortalama fiyatları hesaplayalım
yearly_avg_prices = big_tech_stock_prices.groupby(['year', 'stock_symbol']).
  ↪mean().reset_index()

# Yıllık ortalama fiyatları görselleştirelim
plt.figure(figsize=(14, 7))
sns.lineplot(data=yearly_avg_prices, x='year', y='close', hue='stock_symbol')
plt.title('Yearly Average Closing Prices')
plt.xlabel('Year')
plt.ylabel('Average Close Price')
plt.legend(title='Stock Symbol')
plt.show()
```

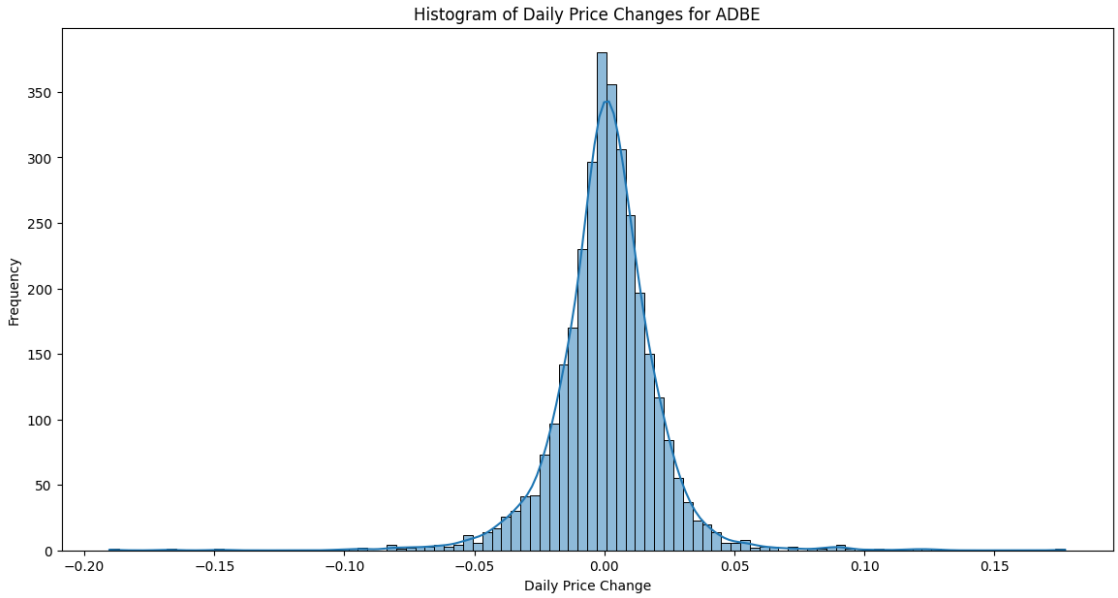


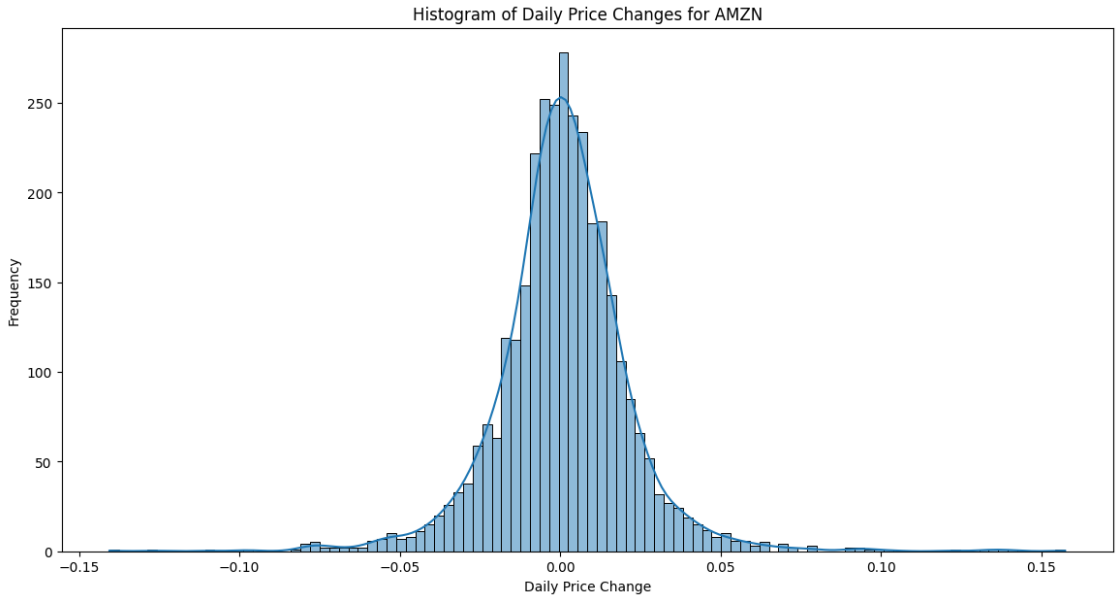
[24]:
```
# Her bir hisse senedi için günlük fiyat değişimlerini hesaplayalım
big_tech_stock_prices['price_change'] = big_tech_stock_prices.
 ↪groupby('stock_symbol')['close'].pct_change()

# Histogramları çizelim
plt.figure(figsize=(14, 10))

# Tüm hisse senetlerinin fiyat değişimlerini tek bir histogramda gösterelim
sns.histplot(big_tech_stock_prices['price_change'].dropna(), bins=100, kde=True)
plt.title('Histogram of Daily Price Changes for All Stocks')
plt.xlabel('Daily Price Change')
plt.ylabel('Frequency')
plt.show()
```

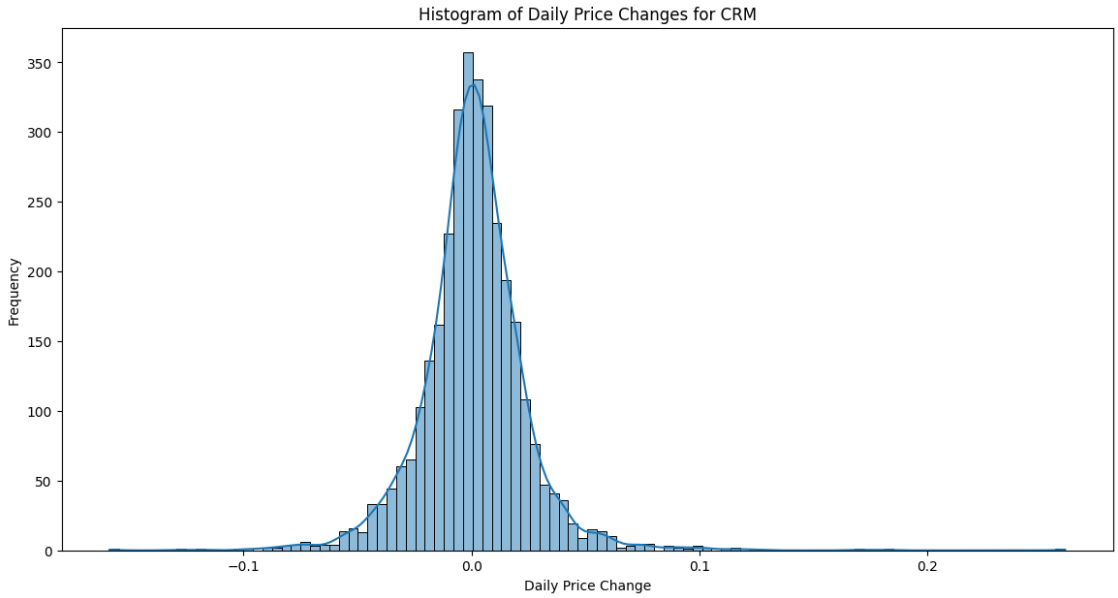


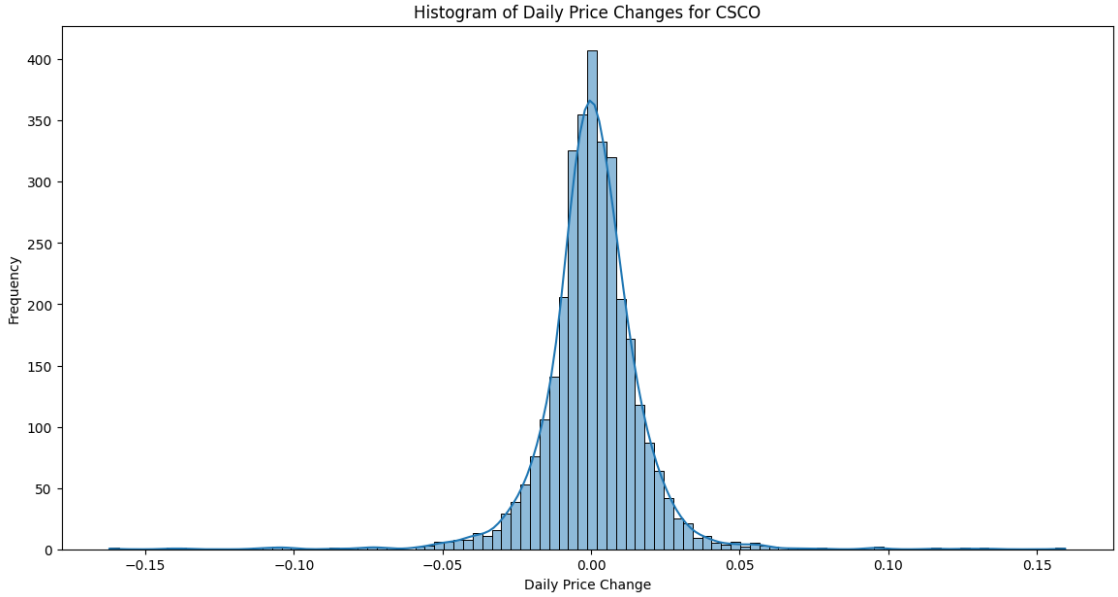
[25]:
```
# Her bir hisse senedi için ayrı ayrı histogramlar çizelim
unique_symbols = big_tech_stock_prices['stock_symbol'].unique()

for symbol in unique_symbols:
    plt.figure(figsize=(14, 7))
    sns.histplot(big_tech_stock_prices[big_tech_stock_prices['stock_symbol'] ==␣
 ↪symbol]['price_change'].dropna(), bins=100, kde=True)
    plt.title(f'Histogram of Daily Price Changes for {symbol}')
    plt.xlabel('Daily Price Change')
    plt.ylabel('Frequency')
    plt.show()
```

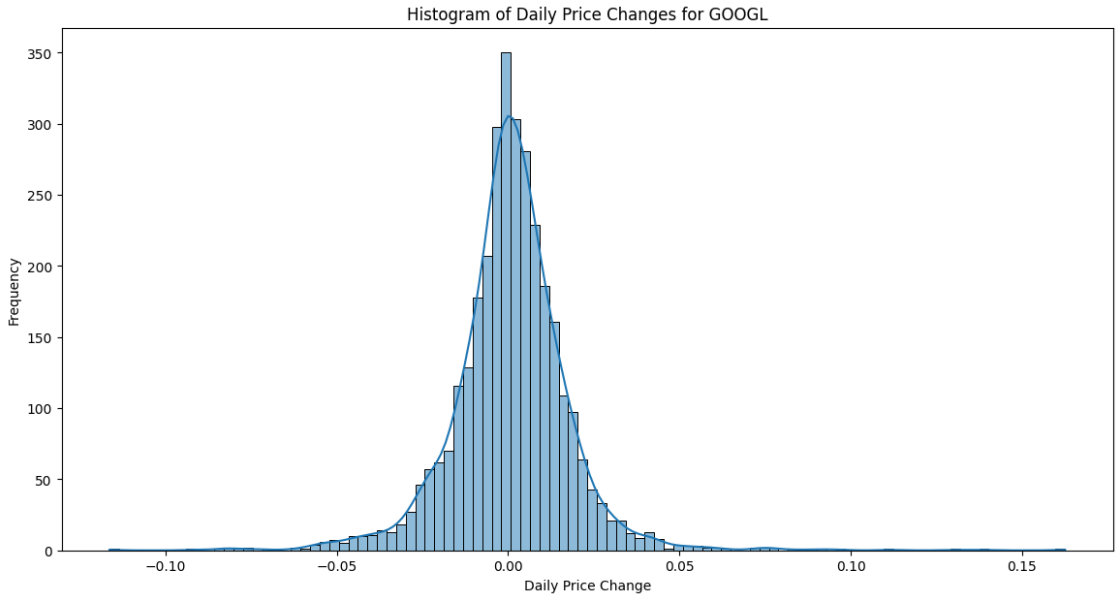

Histogram of Daily Price Changes for AAPL
Frequency
Daily Price Change
0
50
100
150
200
250
−0.10
−0.05
0.00
0.05
0.10

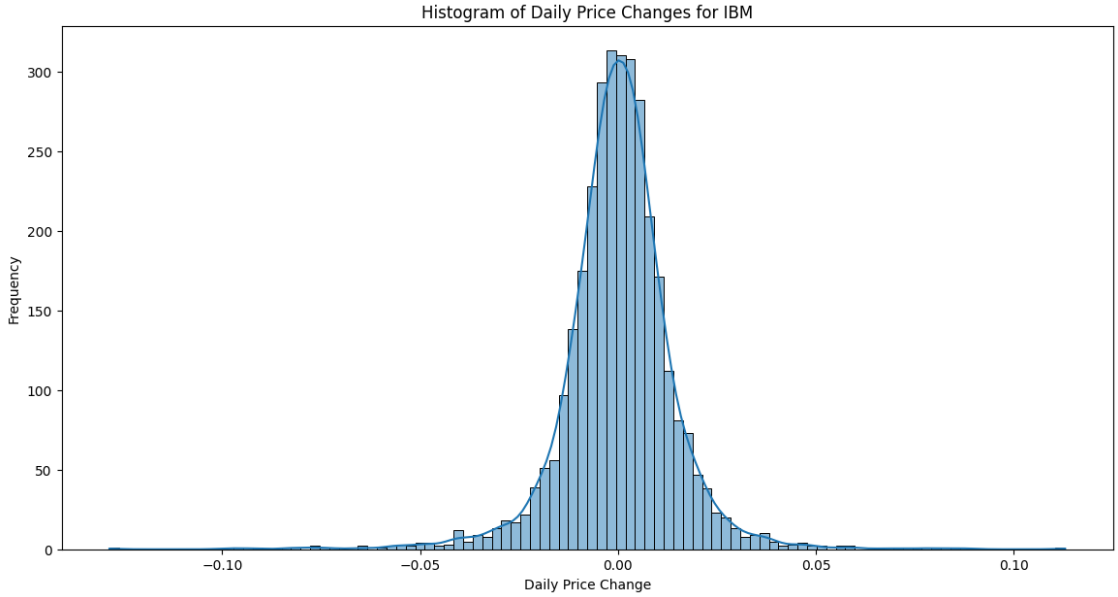

Histogram of Daily Price Changes for ADBE
Frequency
Daily Price Change
0
50
100
150
200
250
300
350
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15

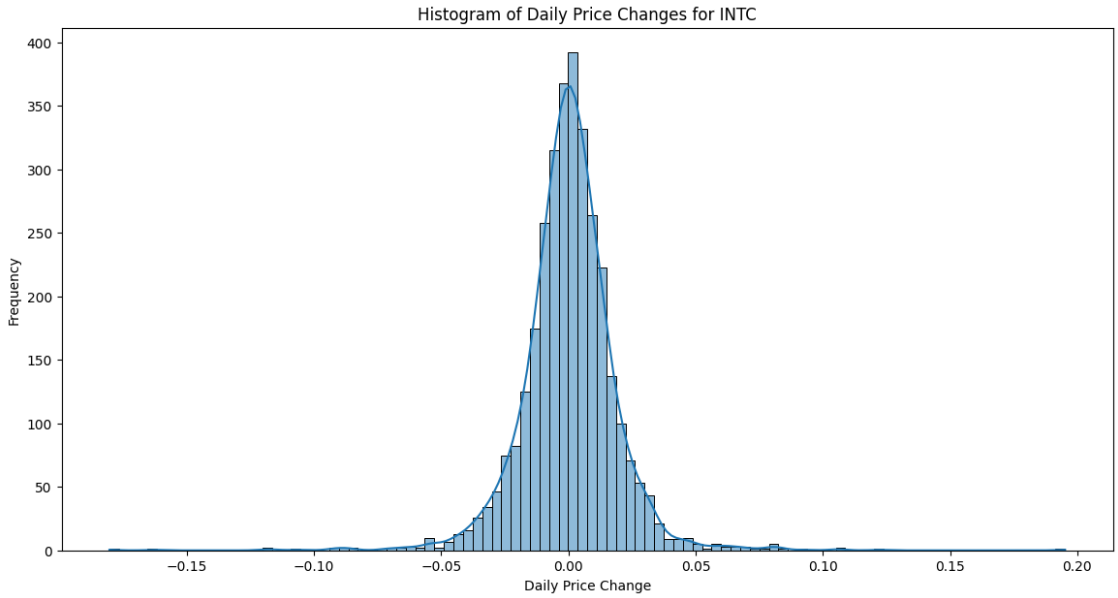

Histogram of Daily Price Changes for AMZN
Frequency
Daily Price Change
0
50
100
150
200
250
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15

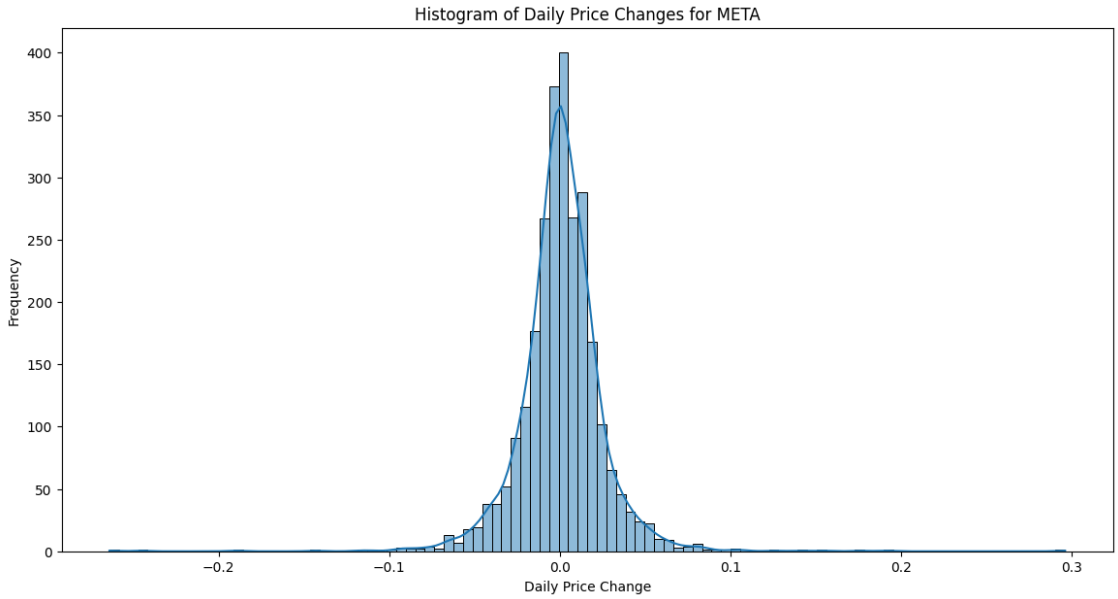

Histogram of Daily Price Changes for CRM
Frequency
Daily Price Change
0
50
100
150
200
250
300
350
−0.1
0.0
0.1
0.2

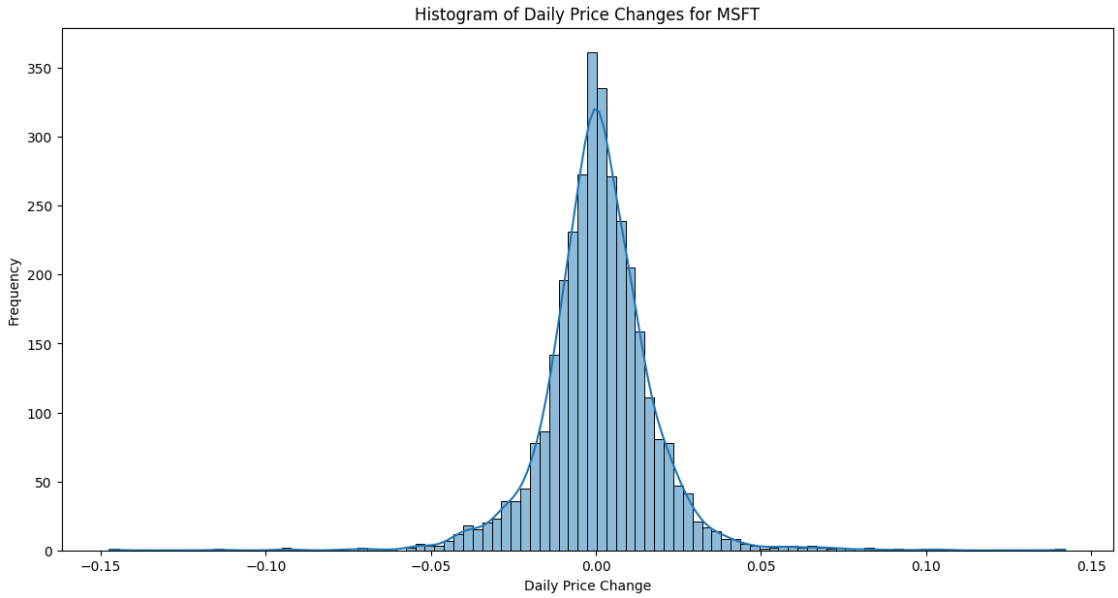

Histogram of Daily Price Changes for CSCO
Frequency
Daily Price Change
0
50
100
150
200
250
300
350
400
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15

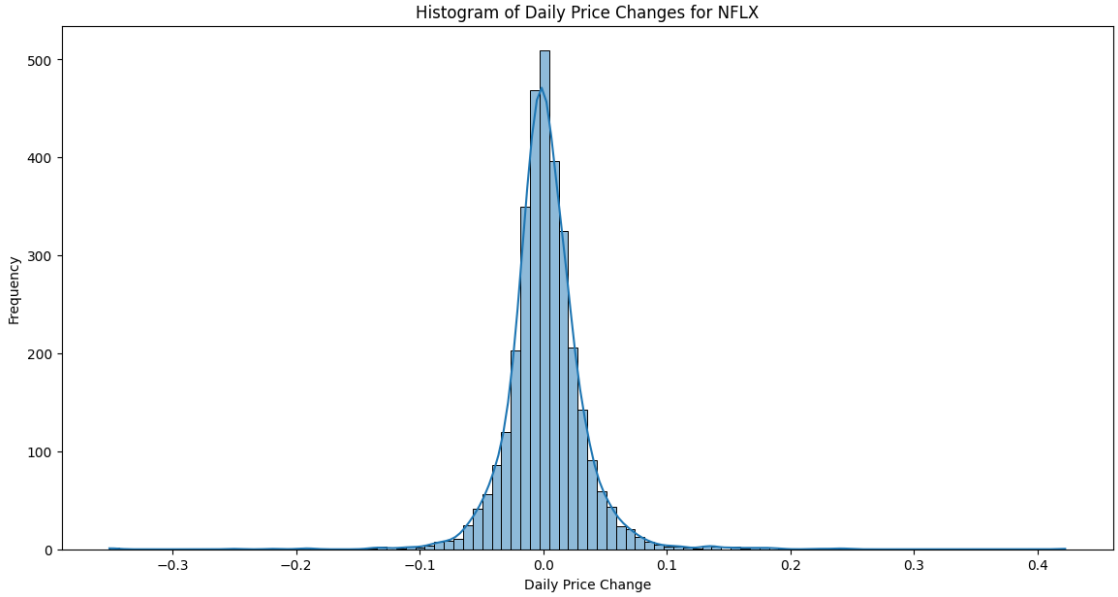

Histogram of Daily Price Changes for GOOGL
Frequency
Daily Price Change
0
50
100
150
200
250
300
350
−0.10
−0.05
0.00
0.05
0.10
0.15

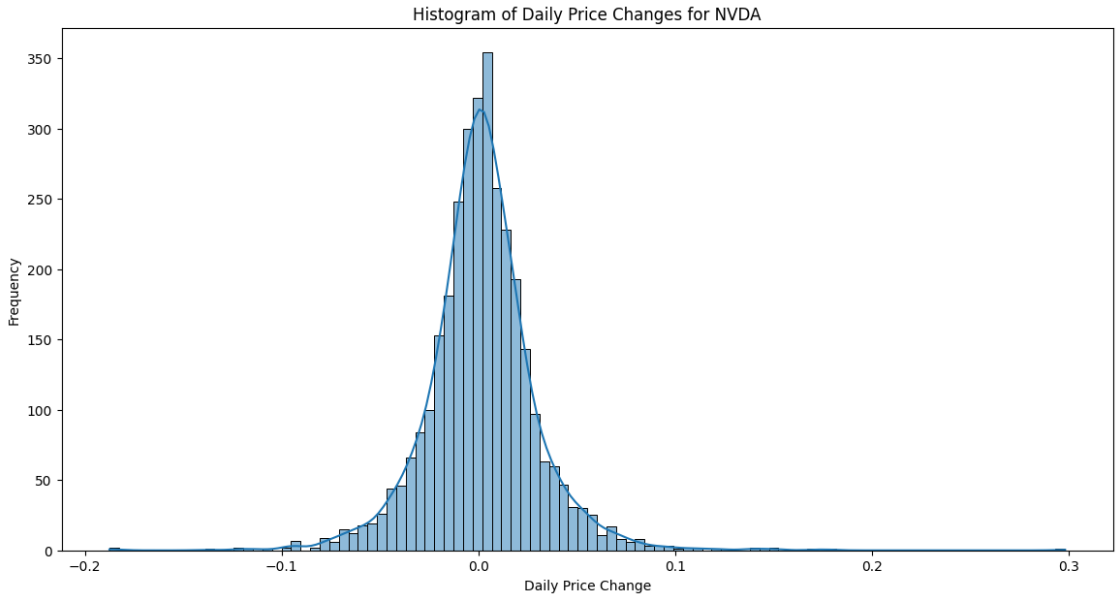

Histogram of Daily Price Changes for IBM
Frequency
Daily Price Change
0
50
100
150
200
250
300
−0.10
−0.05
0.00
0.05
0.10

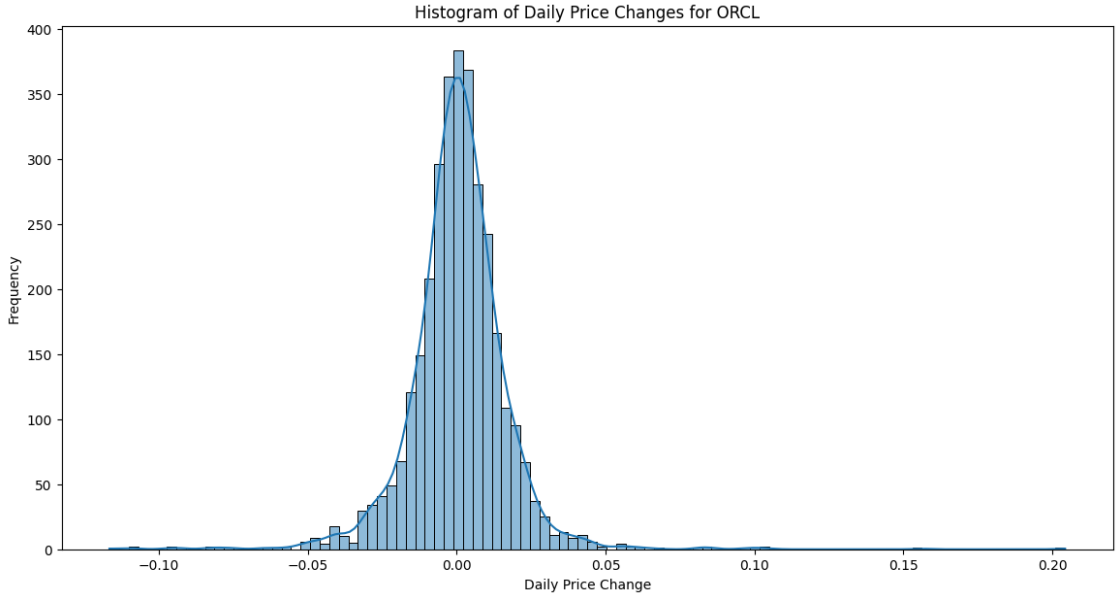

Histogram of Daily Price Changes for INTC
Frequency
Daily Price Change
0
50
100
150
200
250
300
350
400
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20

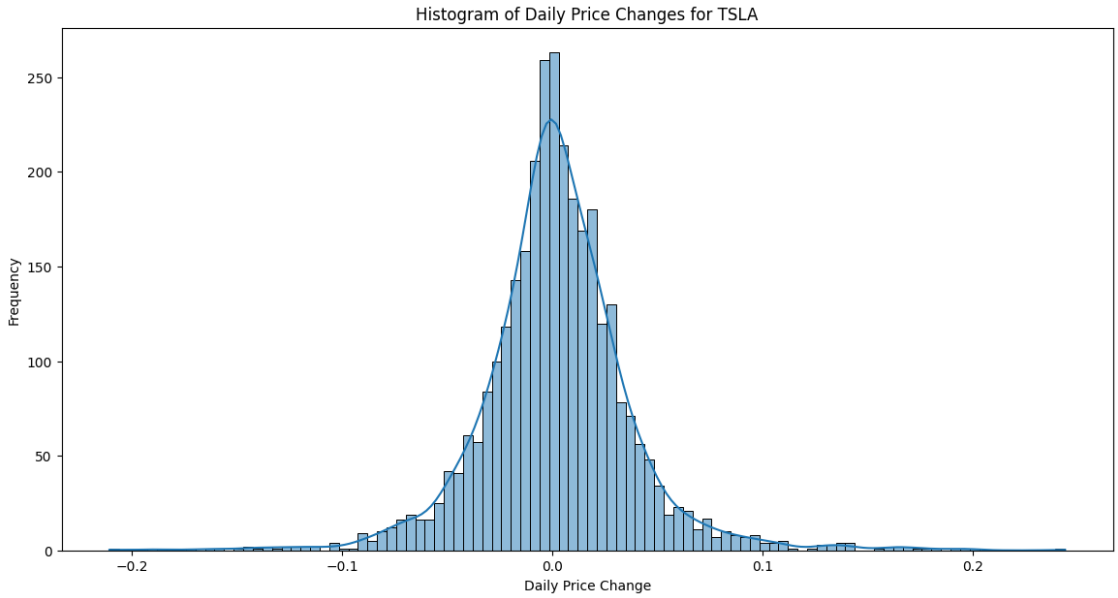

Histogram of Daily Price Changes for META
Frequency
Daily Price Change
0
50
100
150
200
250
300
350
400
−0.2
−0.1
0.0
0.1
0.2
0.3


Histogram of Daily Price Changes for MSFT
Frequency
Daily Price Change
0
50
100
150
200
250
300
350
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15


Histogram of Daily Price Changes for NFLX
Frequency
Daily Price Change
0
100
200
300
400
500
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4

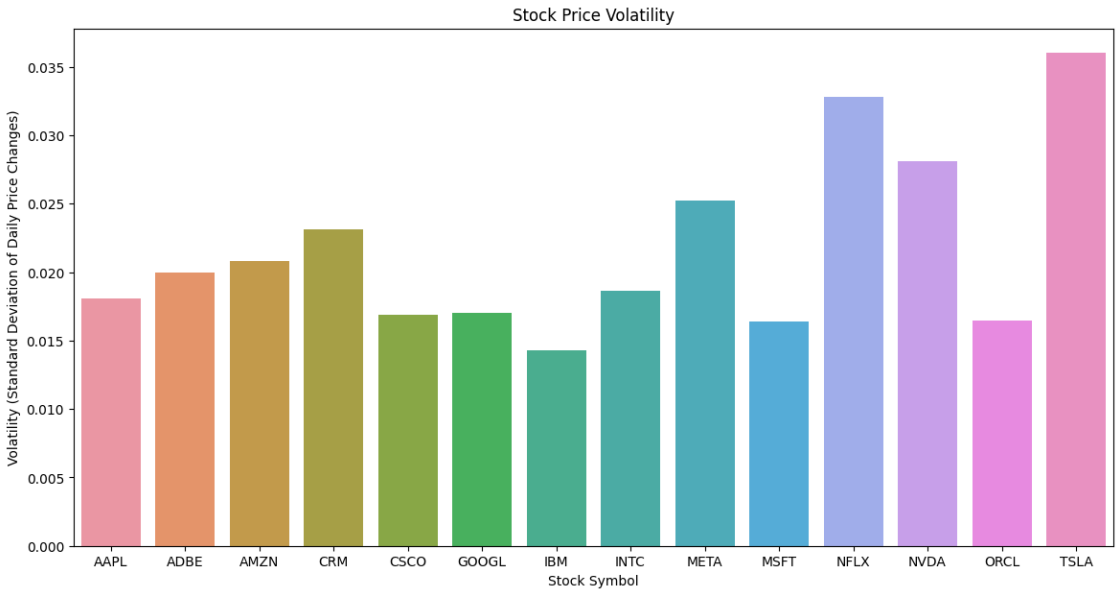

Histogram of Daily Price Changes for NVDA
Frequency
Daily Price Change
0
50
100
150
200
250
300
350
−0.2
−0.1
0.0
0.1
0.2
0.3



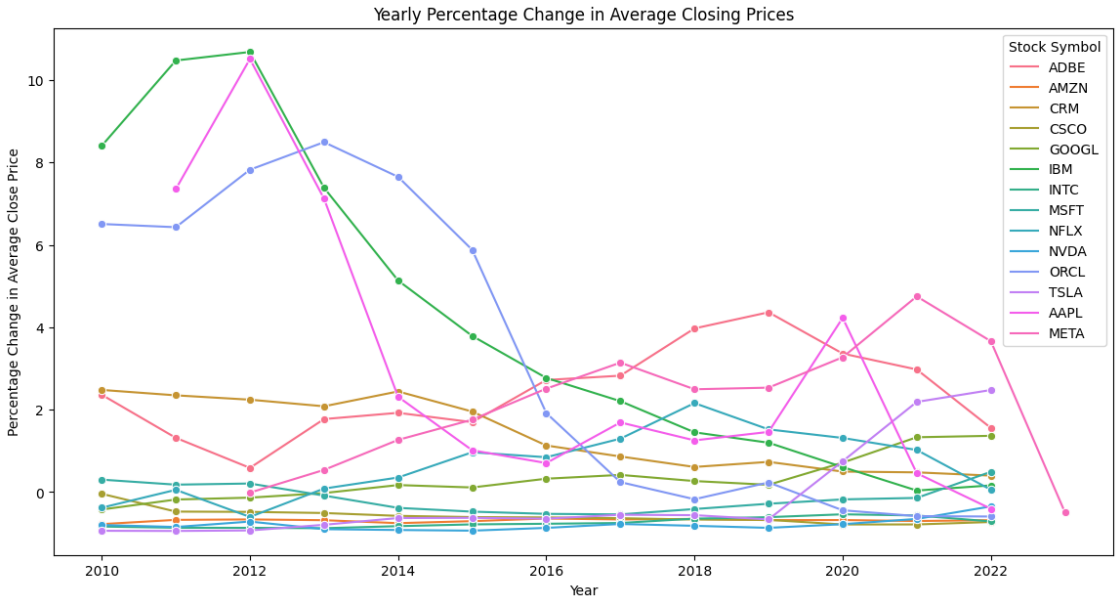


[26]:
```
# Fiyat volatilitesini hesaplayalım (standart sapma)
volatility = big_tech_stock_prices.groupby('stock_symbol')['price_change'].
 ↪std().reset_index()
volatility.columns = ['stock_symbol', 'volatility']

# Volatiliteyi görselleştirelim
plt.figure(figsize=(14, 7))
```

```
sns.barplot(data=volatility, x='stock_symbol', y='volatility')
plt.title('Stock Price Volatility')
plt.xlabel('Stock Symbol')
plt.ylabel('Volatility (Standard Deviation of Daily Price Changes)')
plt.show()
```



[27]:
```
# Yıllık kapanış fiyatlarının yüzdelik değişimlerini hesaplayalım
yearly_price_change = big_tech_stock_prices.groupby(['year',␣
 ↪'stock_symbol'])['close'].mean().pct_change().reset_index()
yearly_price_change = yearly_price_change.dropna()

# Yıllık performansı görselleştirelim
plt.figure(figsize=(14, 7))
sns.lineplot(data=yearly_price_change, x='year', y='close', hue='stock_symbol',␣
 ↪marker='o')
plt.title('Yearly Percentage Change in Average Closing Prices')
plt.xlabel('Year')
plt.ylabel('Percentage Change in Average Close Price')
plt.legend(title='Stock Symbol')
plt.show()
```

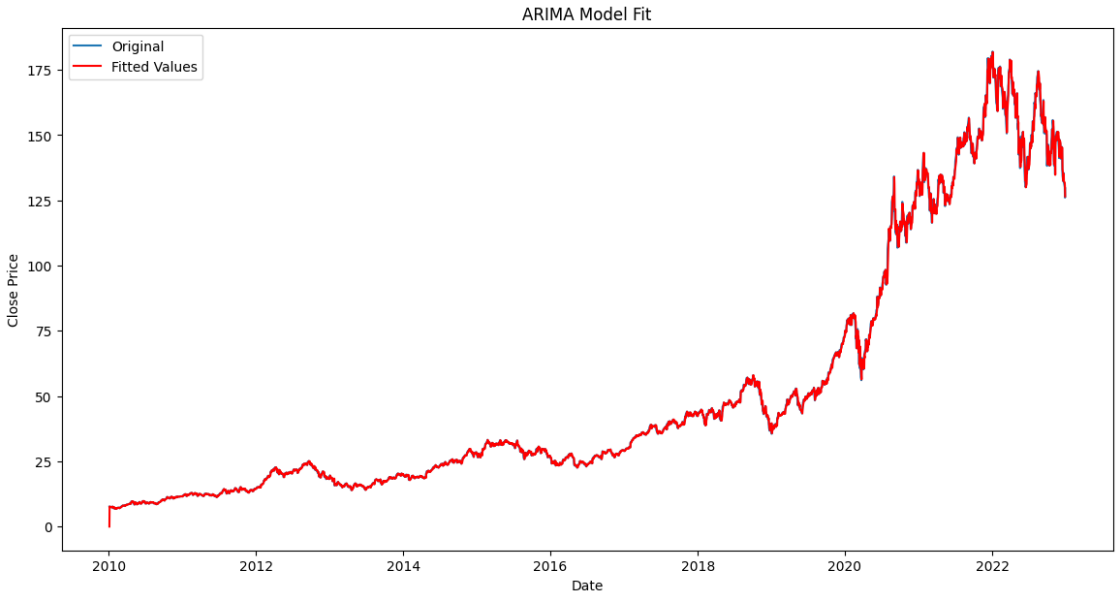


[28]:
```
# ARIMA modeli ile tahmin yapalım
model = ARIMA(apple_stock['close'], order=(5, 1, 0))  # order parametresini␣
 ↪veri setine göre ayarlayın
model_fit = model.fit()
print(model_fit.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  close   No. Observations:                 3271
Model:                 ARIMA(5, 1, 0)   Log Likelihood               -5745.438
Date:                Fri, 28 Jun 2024   AIC                          11502.877
Time:                        15:59:29   BIC                          11539.432
Sample:                             0   HQIC                         11515.968
                                - 3271
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0567      0.008     -7.052      0.000      -0.072      -0.041
ar.L2         -0.0281      0.008     -3.481      0.000      -0.044      -0.012
ar.L3         -0.0305      0.009     -3.387      0.001      -0.048      -0.013
ar.L4         -0.0010      0.008     -0.127      0.899      -0.017       0.015
ar.L5          0.0523      0.008      6.716      0.000       0.037       0.068
sigma2         1.9663      0.018    108.597      0.000       1.931       2.002
==============================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
```

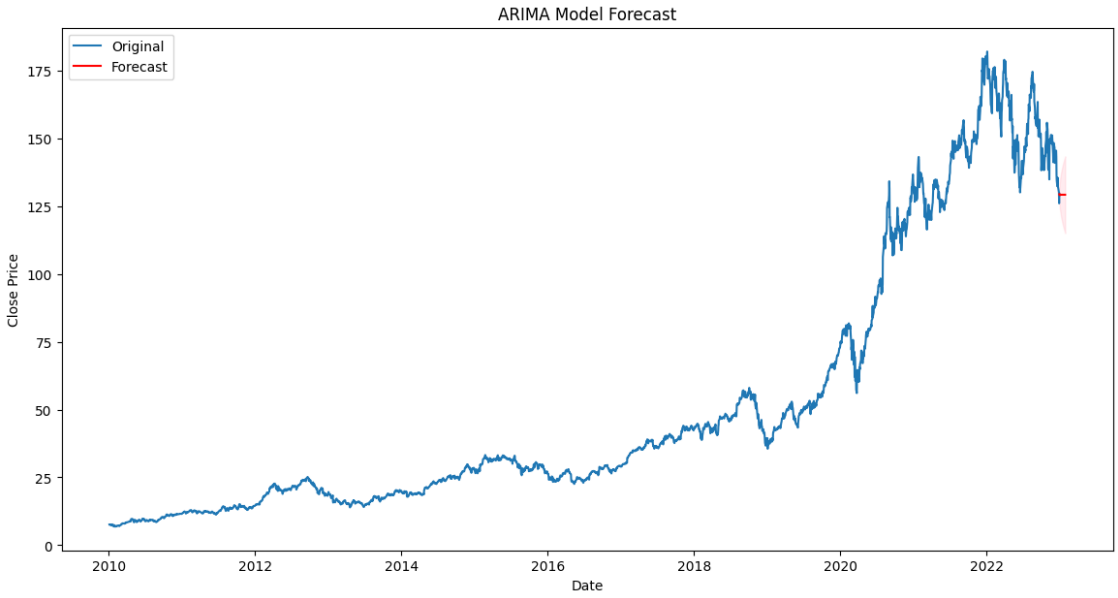
```
23385.05
Prob(Q):                              0.99   Prob(JB):
0.00
Heteroskedasticity (H):              70.64   Skew:
-0.01
Prob(H) (two-sided):                  0.00   Kurtosis:
16.10
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```

[29]:
```
# Tahminleri görselleştirelim
plt.figure(figsize=(14, 7))
plt.plot(apple_stock['close'], label='Original')
plt.plot(model_fit.fittedvalues, color='red', label='Fitted Values')
plt.title('ARIMA Model Fit')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```



[30]:
```
# Gelecekteki kapanış fiyatlarını tahmin edelim
forecast = model_fit.get_forecast(steps=30)
```

```
forecast_index = pd.date_range(start=apple_stock.index[-1], periods=30,␣
 ↪freq='D')
forecast_mean = forecast.predicted_mean
forecast_conf_int = forecast.conf_int()

plt.figure(figsize=(14, 7))
plt.plot(apple_stock['close'], label='Original')
plt.plot(forecast_index, forecast_mean, color='red', label='Forecast')
plt.fill_between(forecast_index, forecast_conf_int.iloc[:, 0],␣
 ↪forecast_conf_int.iloc[:, 1], color='pink', alpha=0.3)
plt.title('ARIMA Model Forecast')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```



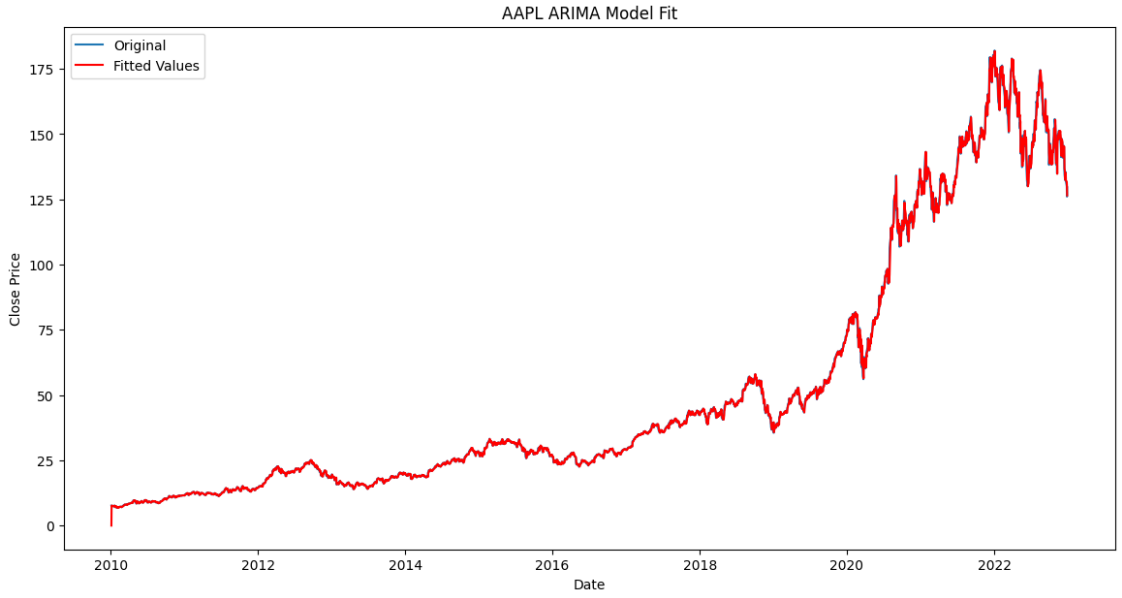
[31]:
```
# Benzersiz hisse senedi sembollerini alalım
unique_symbols = big_tech_stock_prices['stock_symbol'].unique()
```

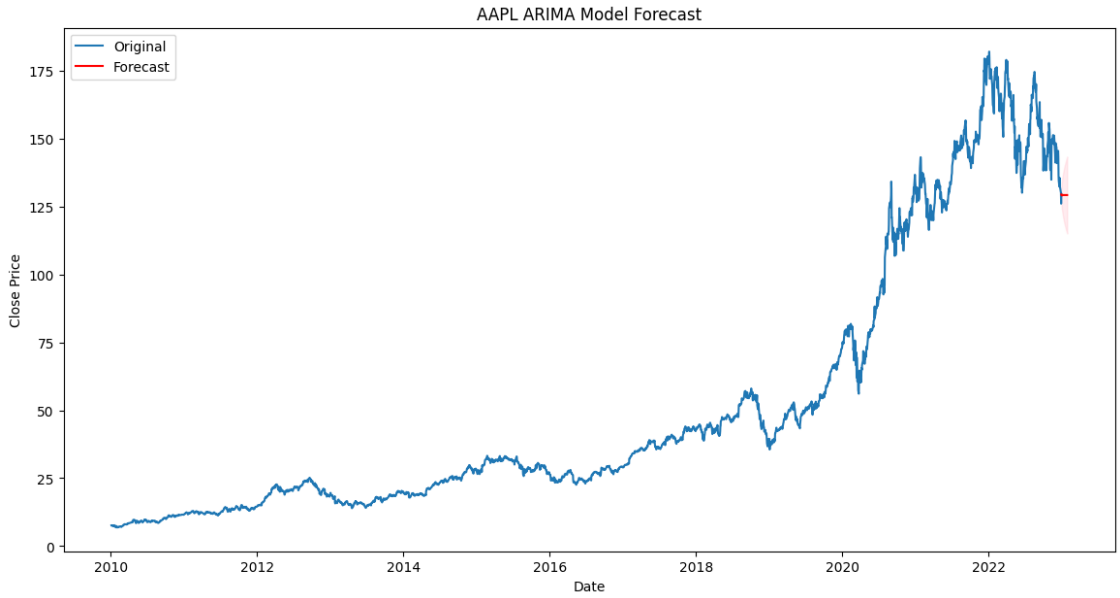
[32]:
```
# Her bir hisse senedi için zaman serisi analizi yapalım
for symbol in unique_symbols:
    stock_data = big_tech_stock_prices[big_tech_stock_prices['stock_symbol'] ==␣
 ↪symbol]
    stock_data.set_index('date', inplace=True)

    print(f"\n### {symbol} ###")
```

```
    # ARIMA modeli ile tahmin yapalım
    model = ARIMA(stock_data['close'], order=(5, 1, 0))  # order parametresini␣
↪veri setine göre ayarlayın
    model_fit = model.fit()
    print(model_fit.summary())

    # Tahminleri görselleştirelim
    plt.figure(figsize=(14, 7))
    plt.plot(stock_data['close'], label='Original')
    plt.plot(model_fit.fittedvalues, color='red', label='Fitted Values')
    plt.title(f'{symbol} ARIMA Model Fit')
    plt.xlabel('Date')
    plt.ylabel('Close Price')
    plt.legend()
    plt.show()

    # Gelecekteki kapanış fiyatlarını tahmin edelim
    forecast = model_fit.get_forecast(steps=30)
    forecast_index = pd.date_range(start=stock_data.index[-1], periods=30,␣
↪freq='D')
    forecast_mean = forecast.predicted_mean
    forecast_conf_int = forecast.conf_int()

    plt.figure(figsize=(14, 7))
    plt.plot(stock_data['close'], label='Original')
    plt.plot(forecast_index, forecast_mean, color='red', label='Forecast')
    plt.fill_between(forecast_index, forecast_conf_int.iloc[:, 0],␣
↪forecast_conf_int.iloc[:, 1], color='pink', alpha=0.3)
    plt.title(f'{symbol} ARIMA Model Forecast')
    plt.xlabel('Date')
    plt.ylabel('Close Price')
    plt.legend()
    plt.show()
```

```
### AAPL ###
                               SARIMAX Results
==============================================================================
Dep. Variable:                  close   No. Observations:                 3271
Model:                 ARIMA(5, 1, 0)   Log Likelihood               -5745.438
Date:                Fri, 28 Jun 2024   AIC                          11502.877
Time:                        15:59:31   BIC                          11539.432
Sample:                             0   HQIC                         11515.968
                                - 3271
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
```

```
------------------------------------------------------------------------------
ar.L1         -0.0567      0.008     -7.052      0.000      -0.072      -0.041
ar.L2         -0.0281      0.008     -3.481      0.000      -0.044      -0.012
ar.L3         -0.0305      0.009     -3.387      0.001      -0.048      -0.013
ar.L4         -0.0010      0.008     -0.127      0.899      -0.017       0.015
ar.L5          0.0523      0.008      6.716      0.000       0.037       0.068
sigma2         1.9663      0.018    108.597      0.000       1.931       2.002
==============================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
23385.05
Prob(Q):                              0.99   Prob(JB):
0.00
Heteroskedasticity (H):              70.64   Skew:
-0.01
Prob(H) (two-sided):                  0.00   Kurtosis:
16.10
==============================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```





ADBE

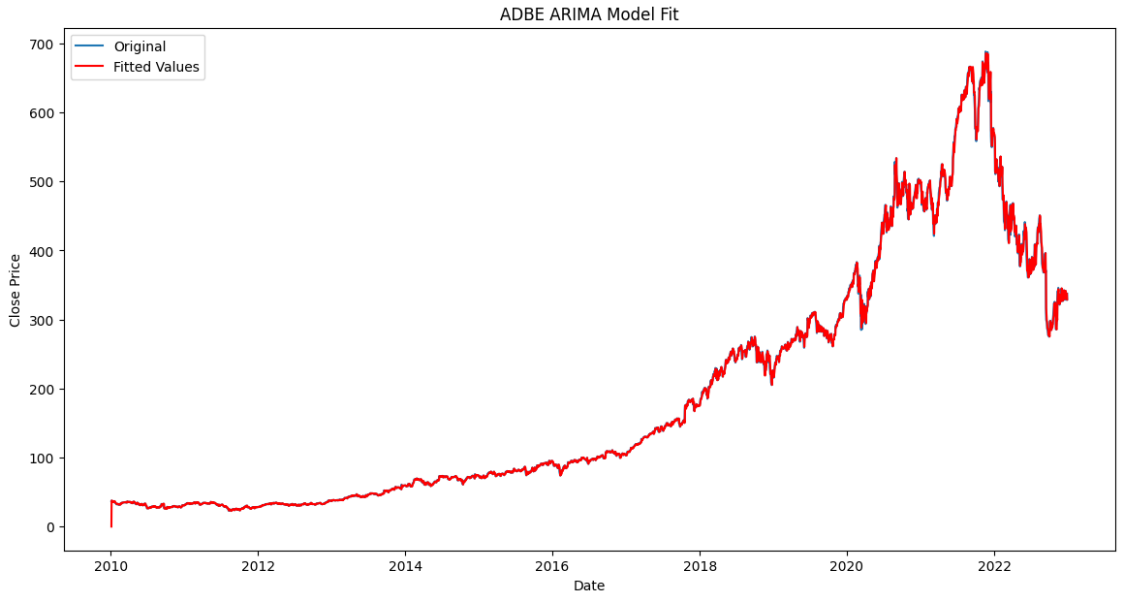
SARIMAX Results

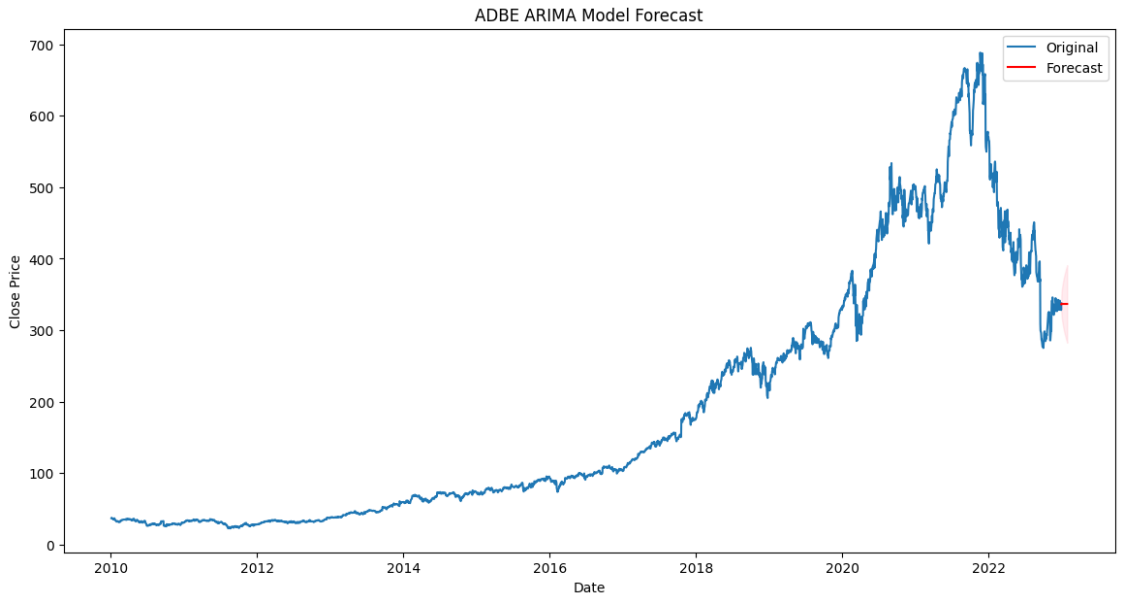
Dep. Variable:	close	No. Observations:	3271
Model:	ARIMA(5, 1, 0)	Log Likelihood	-10329.760
Date:	Fri, 28 Jun 2024	AIC	20671.521
Time:	15:59:33	BIC	20708.076
Sample:	0	HQIC	20684.612
	- 3271		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.1097	0.008	-14.593	0.000	-0.124	-0.095
ar.L2	0.0437	0.008	5.718	0.000	0.029	0.059
ar.L3	-0.0221	0.009	-2.497	0.013	-0.039	-0.005
ar.L4	-0.0385	0.008	-4.642	0.000	-0.055	-0.022
ar.L5	-0.0142	0.008	-1.726	0.084	-0.030	0.002
sigma2	32.4597	0.264	122.735	0.000	31.941	32.978

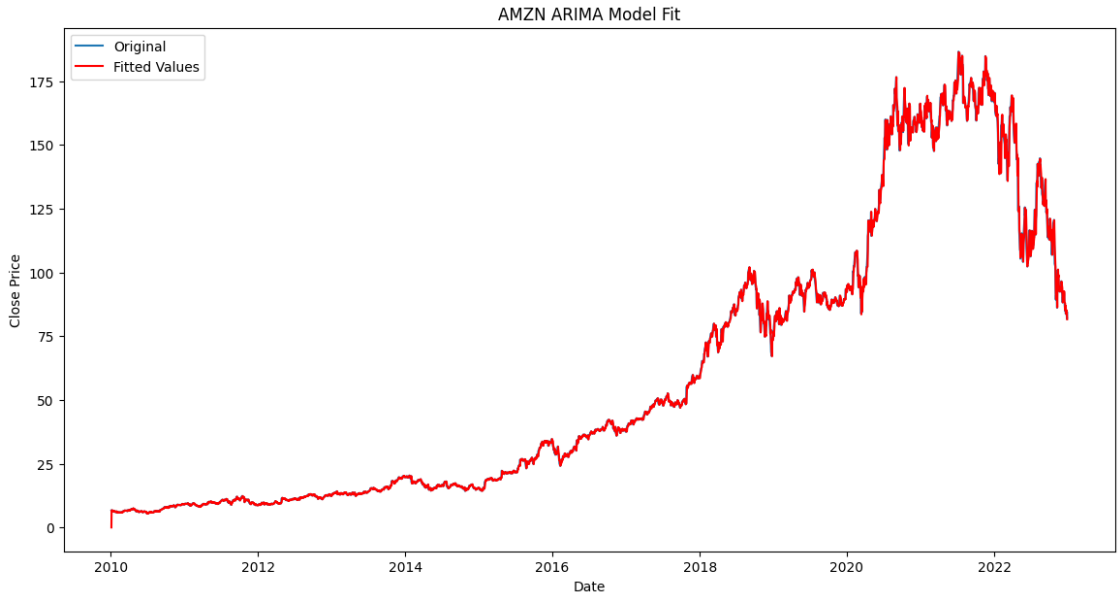
Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	61877.90
Prob(Q):	0.97	Prob(JB):	0.00
Heteroskedasticity (H):	189.18	Skew:	-1.33

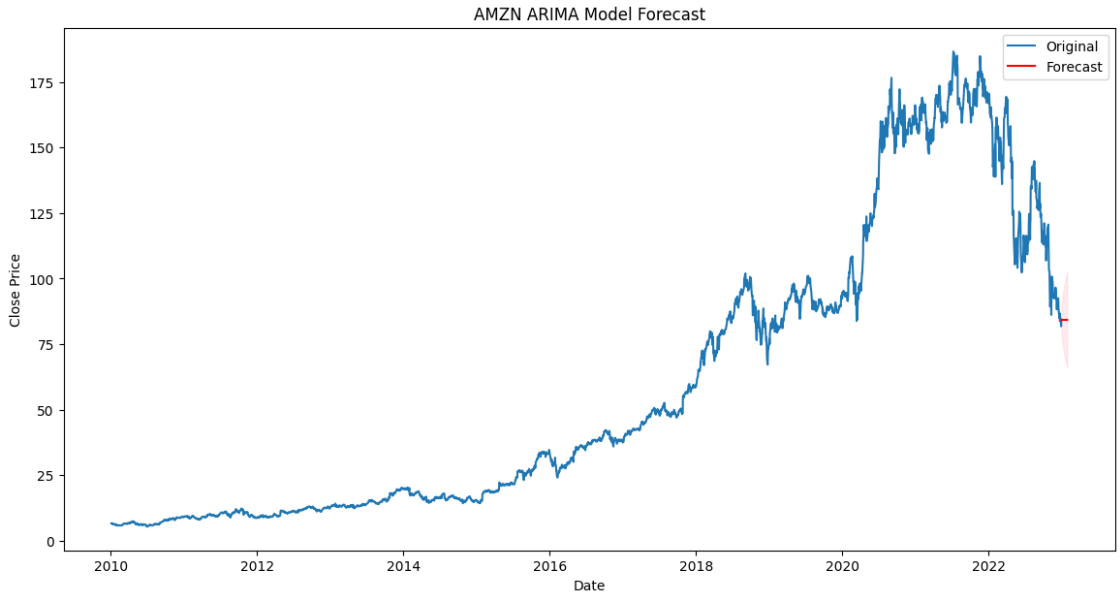
```
Prob(H) (two-sided):                  0.00   Kurtosis:
24.14
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```





```
### AMZN ###
                               SARIMAX Results
==========================================================================
Dep. Variable:                   close   No. Observations:                 3271
Model:                 ARIMA(5, 1, 0)   Log Likelihood               -6416.957
Date:                Fri, 28 Jun 2024   AIC                          12845.915
Time:                        15:59:35   BIC                          12882.470
Sample:                             0   HQIC                         12859.006
                                - 3271
Covariance Type:                  opg
==========================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------
ar.L1         -0.0150      0.009     -1.713      0.087      -0.032       0.002
ar.L2          0.0231      0.010      2.357      0.018       0.004       0.042
ar.L3         -0.0448      0.009     -5.173      0.000      -0.062      -0.028
ar.L4          0.0242      0.009      2.767      0.006       0.007       0.041
ar.L5         -0.0124      0.010     -1.300      0.194      -0.031       0.006
sigma2         2.9649      0.025    117.404      0.000       2.915       3.014
==========================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
48543.47
Prob(Q):                              0.99   Prob(JB):
0.00
Heteroskedasticity (H):             143.52   Skew:
-0.30
Prob(H) (two-sided):                  0.00   Kurtosis:
21.87
==========================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```



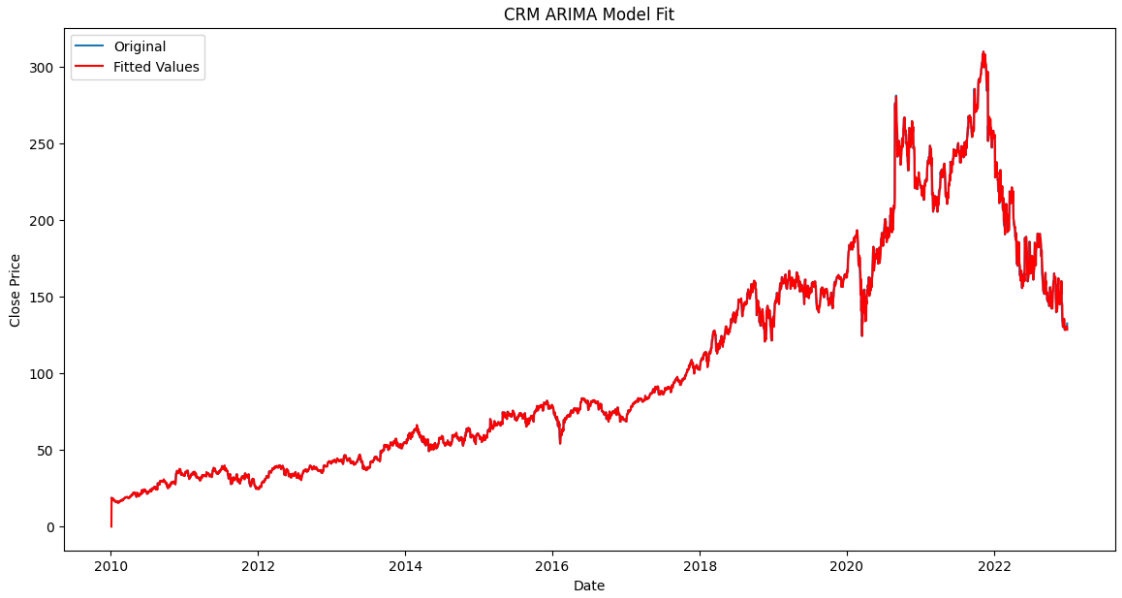


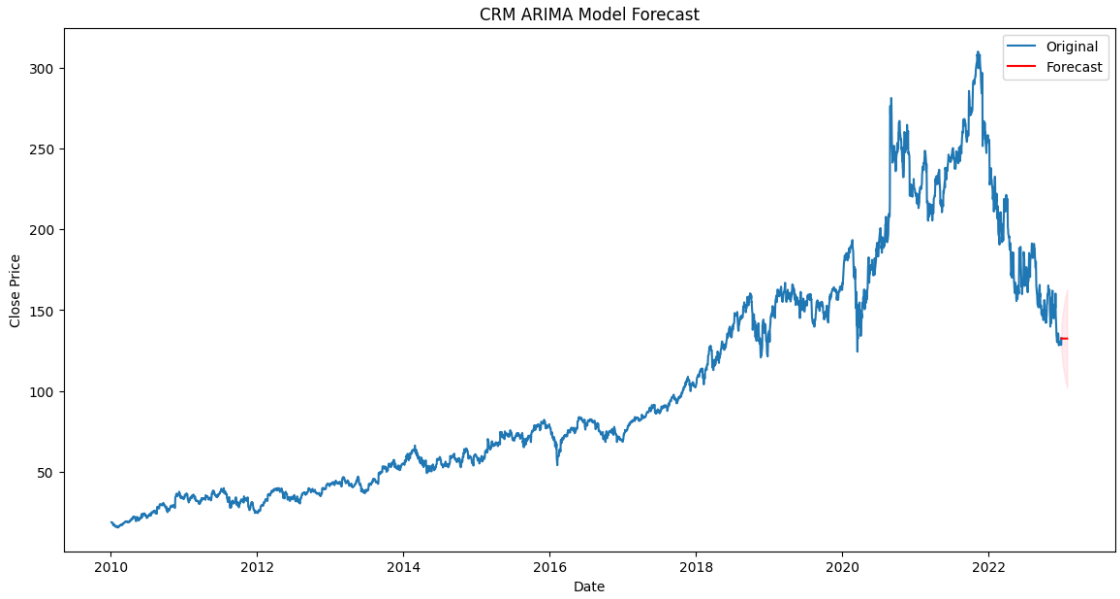
CRM

SARIMAX Results

Dep. Variable:	close	No. Observations:	3271
Model:	ARIMA(5, 1, 0)	Log Likelihood	-8178.802
Date:	Fri, 28 Jun 2024	AIC	16369.604
Time:	15:59:37	BIC	16406.160

```
Sample:                              0   HQIC                          16382.696
                                 - 3271
Covariance Type:                   opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0050      0.009     -0.546      0.585      -0.023       0.013
ar.L2         -0.0080      0.010     -0.835      0.404      -0.027       0.011
ar.L3         -0.0139      0.011     -1.250      0.211      -0.036       0.008
ar.L4         -0.0084      0.009     -0.949      0.343      -0.026       0.009
ar.L5         -0.0142      0.011     -1.327      0.184      -0.035       0.007
sigma2         8.7098      0.048    181.303      0.000       8.616       8.804
==============================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
370034.41
Prob(Q):                              0.96   Prob(JB):
0.00
Heteroskedasticity (H):              30.02   Skew:
1.24
Prob(H) (two-sided):                  0.00   Kurtosis:
55.05
==============================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```





CSCO

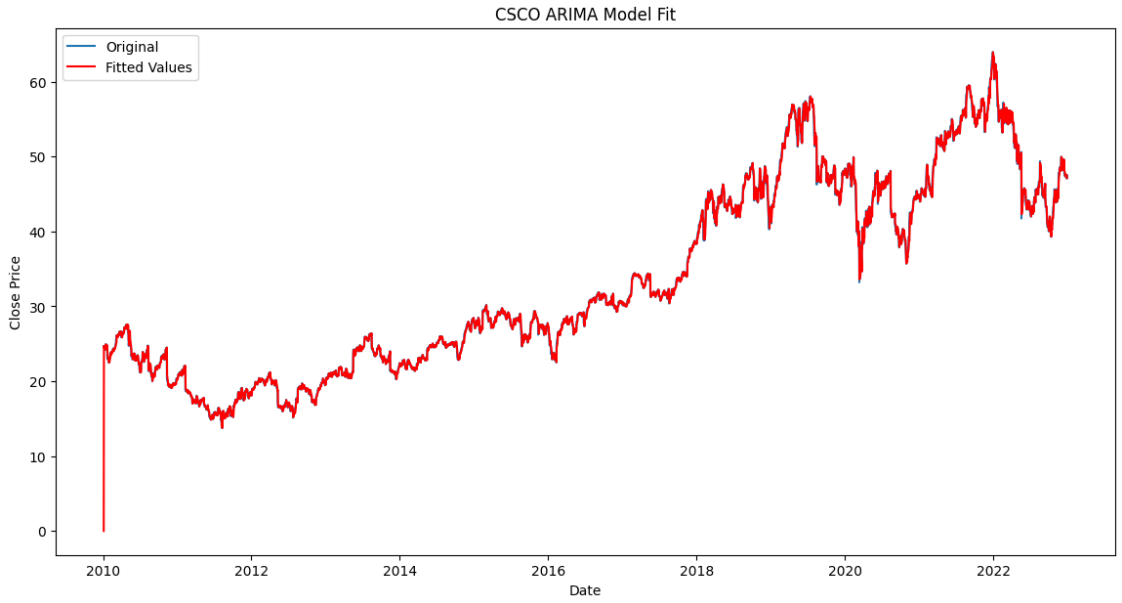
SARIMAX Results

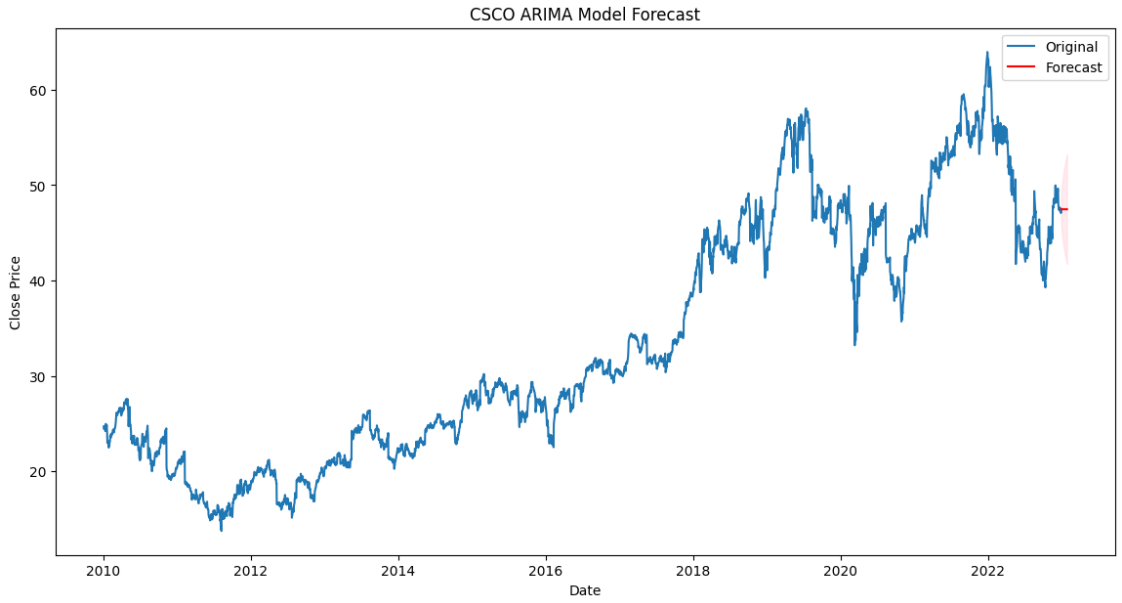
Dep. Variable:	close	No. Observations:	3271
Model:	ARIMA(5, 1, 0)	Log Likelihood	-2908.668
Date:	Fri, 28 Jun 2024	AIC	5829.335
Time:	15:59:39	BIC	5865.891
Sample:	0	HQIC	5842.427
	- 3271		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.0803	0.009	-8.991	0.000	-0.098	-0.063
ar.L2	-0.0087	0.010	-0.857	0.392	-0.029	0.011
ar.L3	0.0043	0.012	0.371	0.710	-0.018	0.027
ar.L4	-0.0229	0.012	-1.874	0.061	-0.047	0.001
ar.L5	-0.0009	0.012	-0.077	0.939	-0.024	0.022
sigma2	0.3468	0.003	113.492	0.000	0.341	0.353

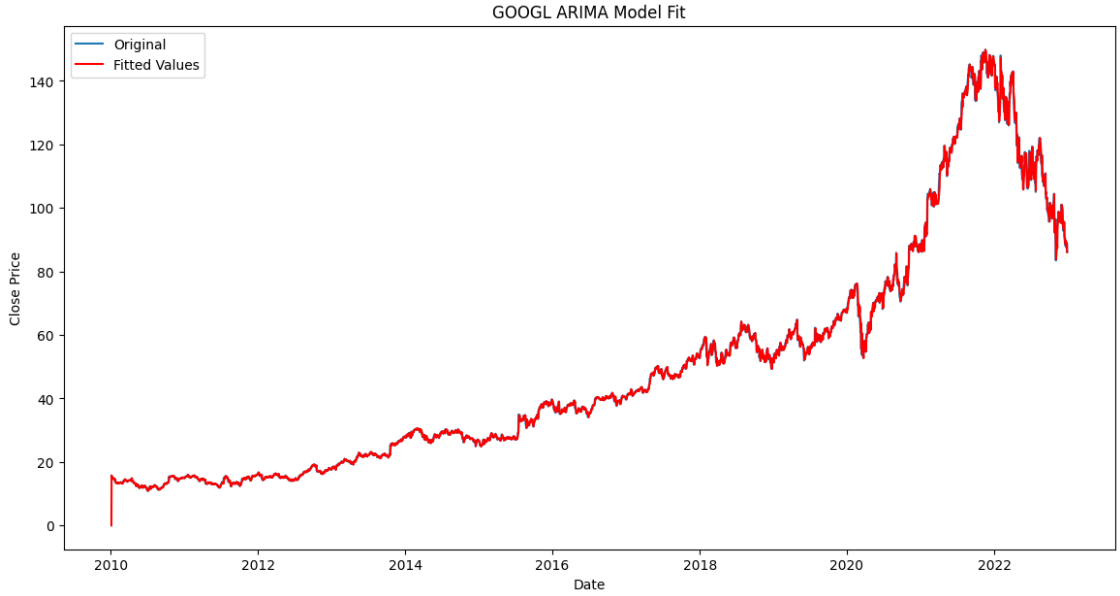
Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	31769.66
Prob(Q):	0.99	Prob(JB):	0.00

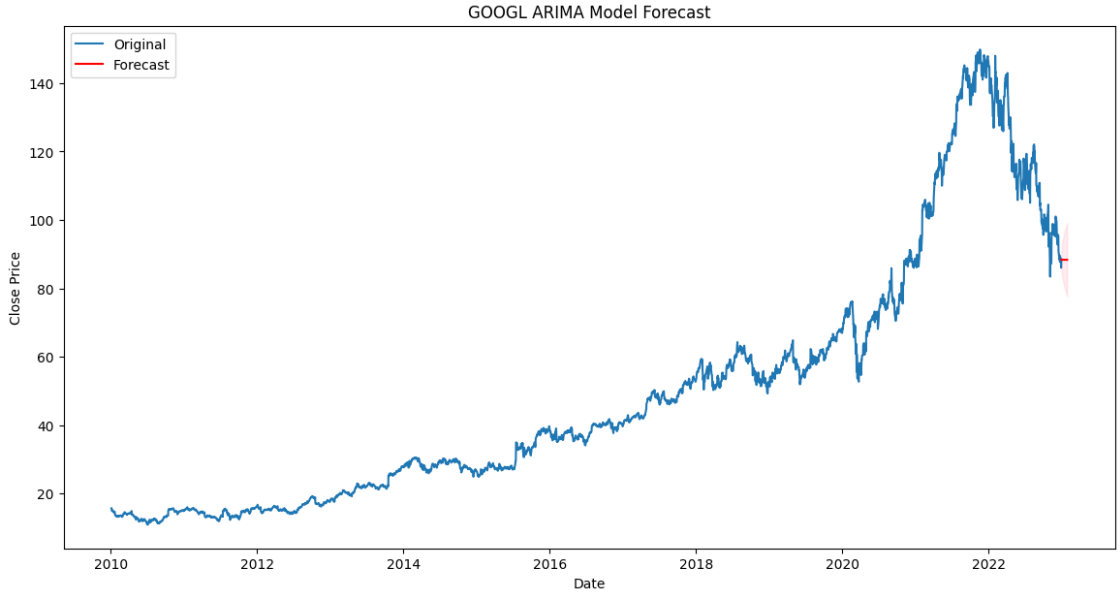
```
Heteroskedasticity (H):                 5.33   Skew:
-1.12
Prob(H) (two-sided):                    0.00   Kurtosis:
18.10
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```





```
### GOOGL ###
                               SARIMAX Results
==========================================================================
Dep. Variable:                   close   No. Observations:                 3271
Model:                 ARIMA(5, 1, 0)   Log Likelihood               -5026.464
Date:                Fri, 28 Jun 2024   AIC                          10064.928
Time:                        15:59:41   BIC                          10101.483
Sample:                             0   HQIC                         10078.019
                                - 3271
Covariance Type:                  opg
==========================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------
ar.L1         -0.0598      0.009     -6.674      0.000      -0.077      -0.042
ar.L2         -0.0074      0.009     -0.834      0.404      -0.025       0.010
ar.L3         -0.0456      0.008     -5.739      0.000      -0.061      -0.030
ar.L4         -0.0169      0.009     -1.810      0.070      -0.035       0.001
ar.L5         -0.0146      0.009     -1.671      0.095      -0.032       0.003
sigma2         1.2667      0.012    107.685      0.000       1.244       1.290
==========================================================================
===
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):
21313.91
Prob(Q):                              0.94   Prob(JB):
0.00
Heteroskedasticity (H):              44.08   Skew:
-0.06
Prob(H) (two-sided):                  0.00   Kurtosis:
15.51
==========================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```



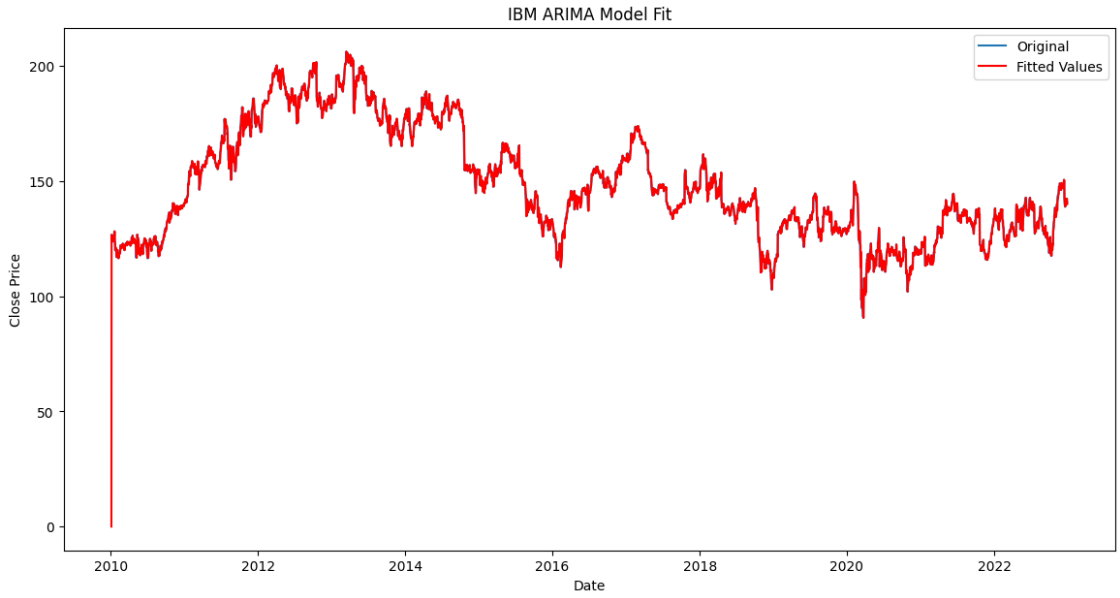


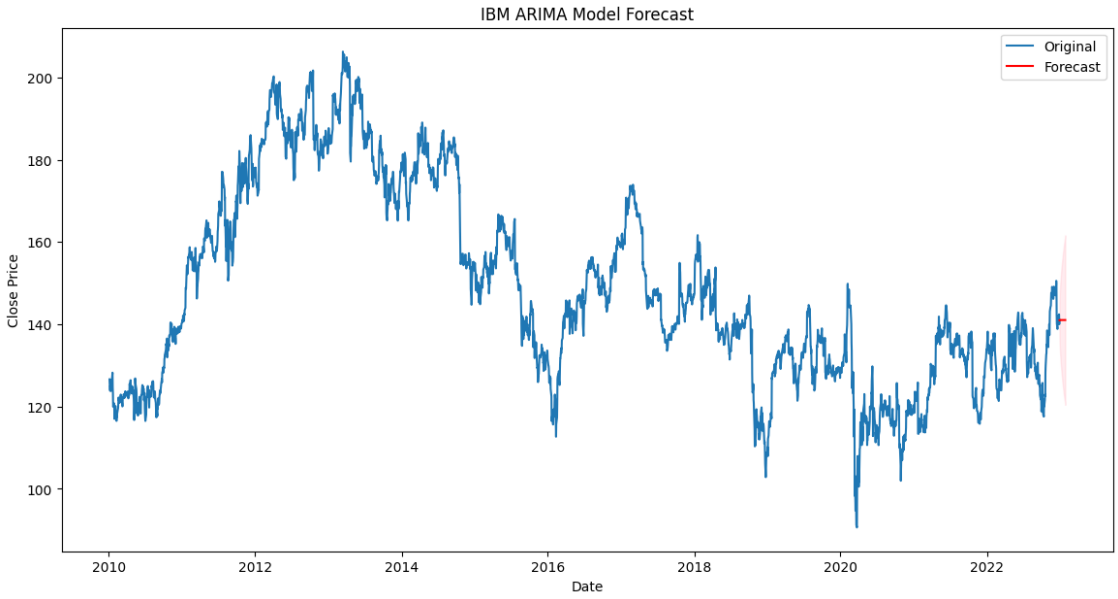
IBM

SARIMAX Results

Dep. Variable:	close	No. Observations:	3271
Model:	ARIMA(5, 1, 0)	Log Likelihood	-6906.575
Date:	Fri, 28 Jun 2024	AIC	13825.150
Time:	15:59:42	BIC	13861.705

```
Sample:                                0   HQIC                           13838.241
                                 - 3271
Covariance Type:                     opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0131      0.014     -0.965      0.335      -0.040       0.014
ar.L2          0.0099      0.013      0.773      0.440      -0.015       0.035
ar.L3         -0.0164      0.013     -1.238      0.216      -0.042       0.010
ar.L4         -0.0269      0.015     -1.846      0.065      -0.055       0.002
ar.L5         -0.0012      0.015     -0.085      0.933      -0.030       0.027
sigma2         4.0001      0.046     86.465      0.000       3.909       4.091
==============================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
7652.60
Prob(Q):                              1.00   Prob(JB):
0.00
Heteroskedasticity (H):               1.21   Skew:
-0.76
Prob(H) (two-sided):                  0.00   Kurtosis:
10.34
==============================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```





INTC

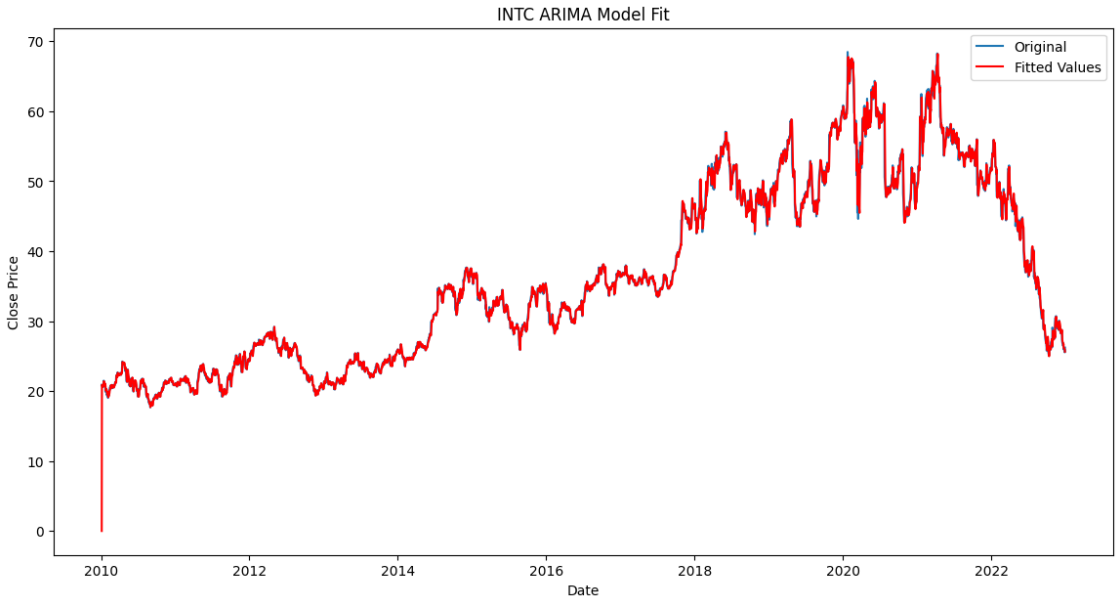
SARIMAX Results

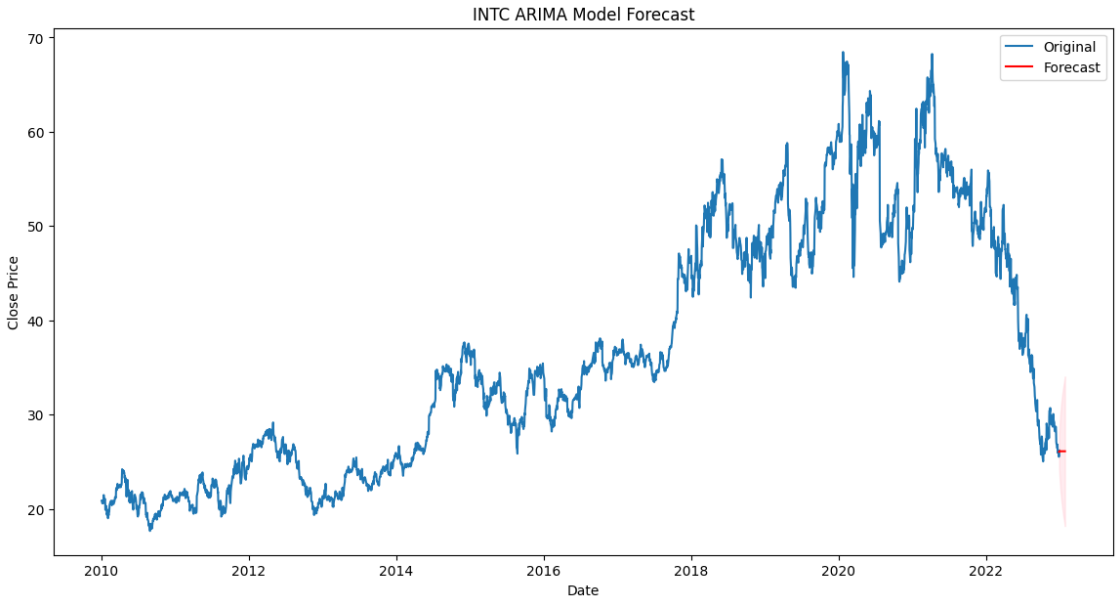
Dep. Variable:	close	No. Observations:	3271
Model:	ARIMA(5, 1, 0)	Log Likelihood	-3905.625
Date:	Fri, 28 Jun 2024	AIC	7823.250
Time:	15:59:44	BIC	7859.805
Sample:	0	HQIC	7836.341
	- 3271		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.1599	0.007	-21.622	0.000	-0.174	-0.145
ar.L2	0.0588	0.010	5.797	0.000	0.039	0.079
ar.L3	0.0455	0.010	4.397	0.000	0.025	0.066
ar.L4	-0.0280	0.011	-2.582	0.010	-0.049	-0.007
ar.L5	0.0005	0.009	0.060	0.952	-0.017	0.019
sigma2	0.6382	0.005	125.865	0.000	0.628	0.648

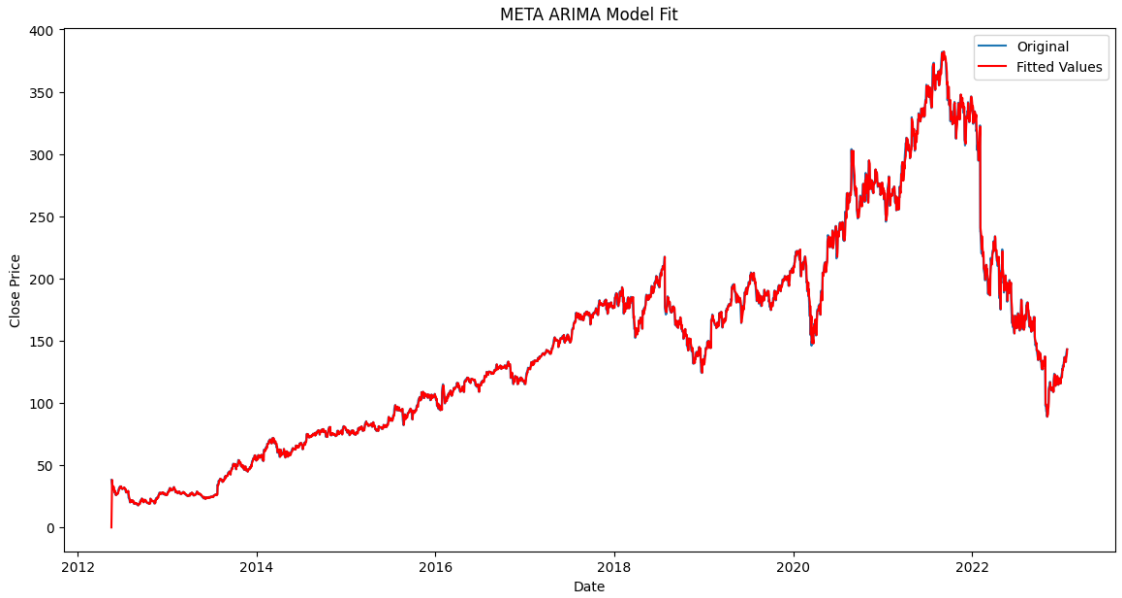
Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	62841.40
Prob(Q):	1.00	Prob(JB):	0.00

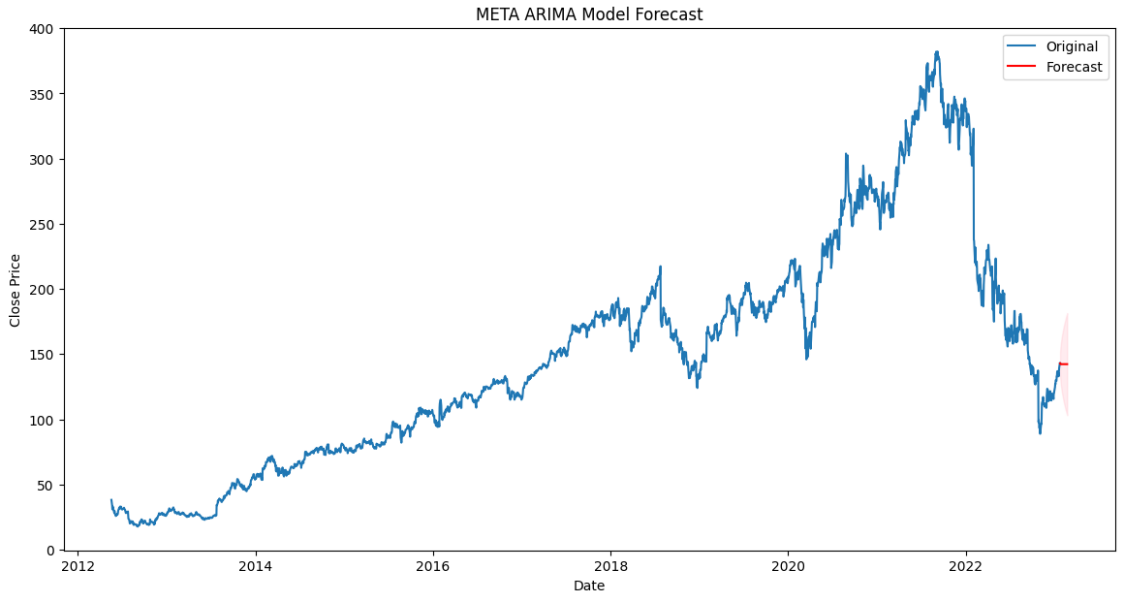
```
Heteroskedasticity (H):                12.82   Skew:
-1.06
Prob(H) (two-sided):                    0.00   Kurtosis:
24.37
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```





```
### META ###
                               SARIMAX Results
==========================================================================================
Dep. Variable:                      close   No. Observations:                 2688
Model:                 ARIMA(5, 1, 0)   Log Likelihood               -7625.455
Date:                Fri, 28 Jun 2024   AIC                            15262.910
Time:                        15:59:46   BIC                            15298.287
Sample:                             0   HQIC                           15275.707
                                - 2688
Covariance Type:                   opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0458      0.014     -3.171      0.002      -0.074      -0.018
ar.L2          0.0152      0.012      1.289      0.197      -0.008       0.038
ar.L3         -0.0588      0.012     -4.866      0.000      -0.083      -0.035
ar.L4         -0.0384      0.012     -3.225      0.001      -0.062      -0.015
ar.L5         -0.0221      0.014     -1.624      0.104      -0.049       0.005
sigma2        17.0803      0.101    169.685      0.000      16.883      17.278
===================================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
602757.96
Prob(Q):                              0.97   Prob(JB):
0.00
Heteroskedasticity (H):              25.15   Skew:
-3.63
Prob(H) (two-sided):                  0.00   Kurtosis:
76.01
===================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```



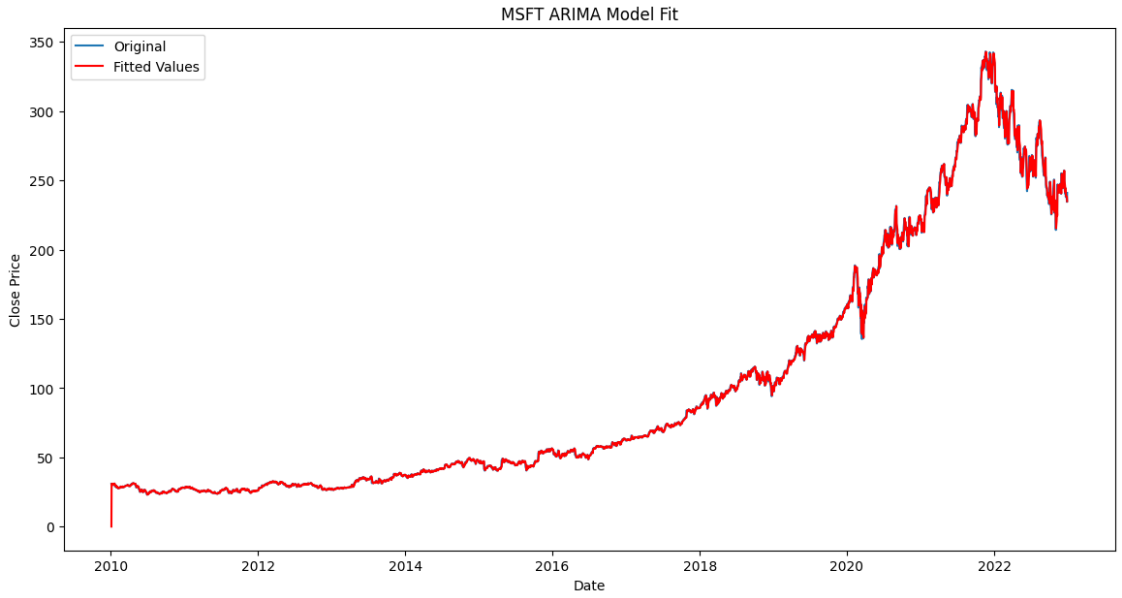


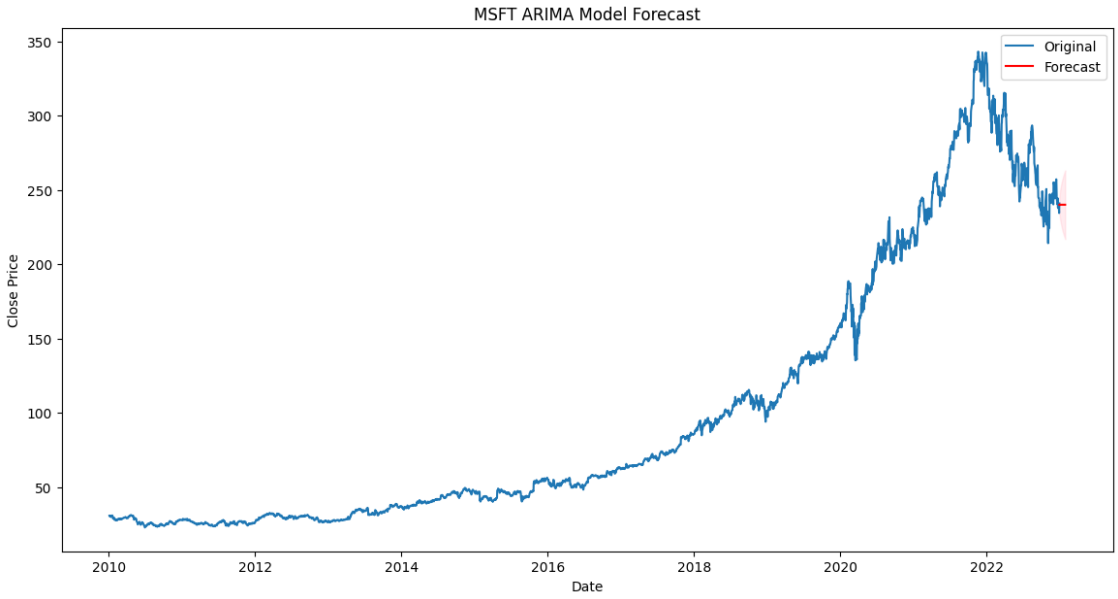
MSFT

SARIMAX Results

Dep. Variable:	close	No. Observations:	3271
Model:	ARIMA(5, 1, 0)	Log Likelihood	-7574.524
Date:	Fri, 28 Jun 2024	AIC	15161.047
Time:	15:59:48	BIC	15197.603

```
Sample:                                0   HQIC                           15174.139
                                 - 3271
Covariance Type:                     opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.1237      0.008    -16.335      0.000      -0.139      -0.109
ar.L2         -0.0078      0.008     -1.021      0.307      -0.023       0.007
ar.L3         -0.0349      0.009     -4.093      0.000      -0.052      -0.018
ar.L4         -0.0083      0.008     -0.987      0.324      -0.025       0.008
ar.L5          0.0276      0.008      3.336      0.001       0.011       0.044
sigma2         6.0186      0.057    105.064      0.000       5.906       6.131
==============================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
22089.36
Prob(Q):                              0.95   Prob(JB):
0.00
Heteroskedasticity (H):              91.28   Skew:
-0.41
Prob(H) (two-sided):                  0.00   Kurtosis:
15.71
==============================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```





NFLX

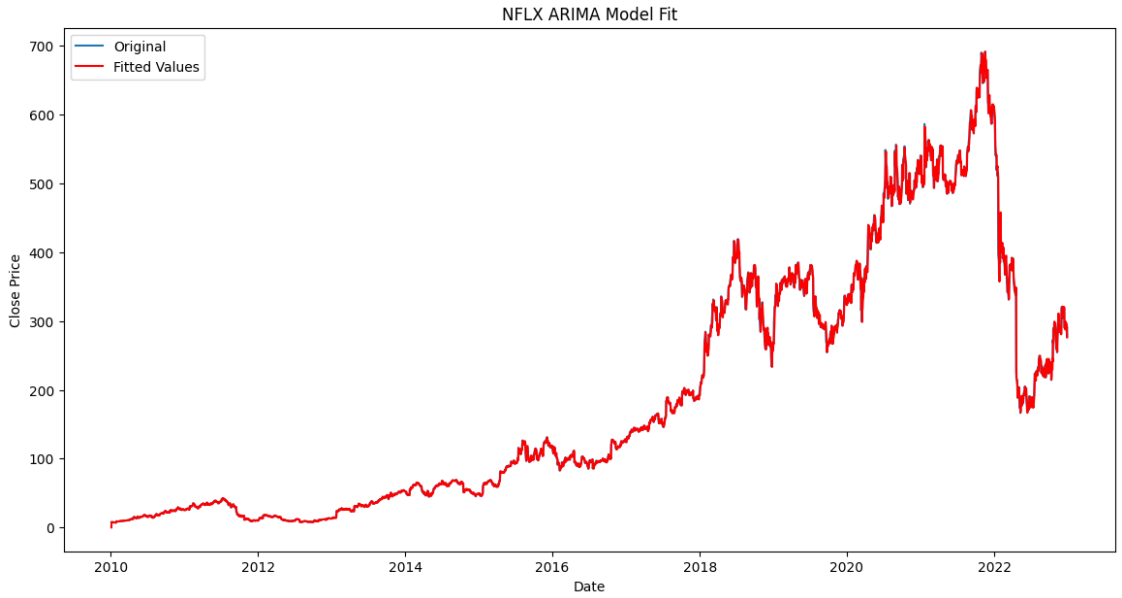
SARIMAX Results

Dep. Variable:	close	No. Observations:	3271
Model:	ARIMA(5, 1, 0)	Log Likelihood	-11038.833
Date:	Fri, 28 Jun 2024	AIC	22089.666
Time:	15:59:50	BIC	22126.221
Sample:	0	HQIC	22102.757
	- 3271		
Covariance Type:	opg		

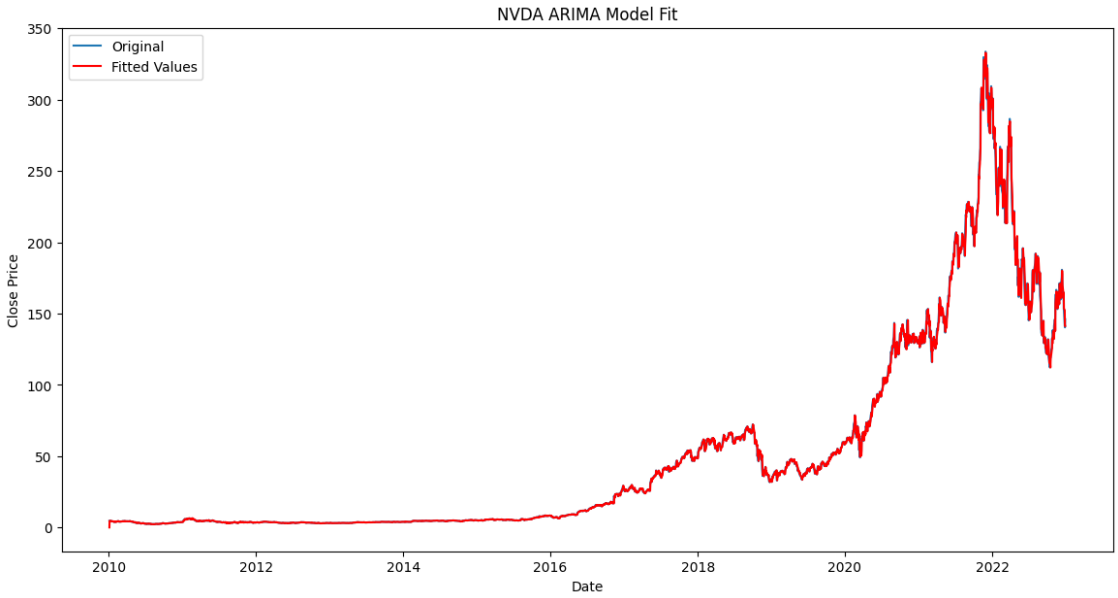
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.0340	0.011	-3.228	0.001	-0.055	-0.013
ar.L2	0.0230	0.010	2.266	0.023	0.003	0.043
ar.L3	0.0123	0.012	1.021	0.307	-0.011	0.036
ar.L4	0.0212	0.009	2.295	0.022	0.003	0.039
ar.L5	-0.0550	0.009	-6.088	0.000	-0.073	-0.037
sigma2	50.0836	0.245	204.618	0.000	49.604	50.563

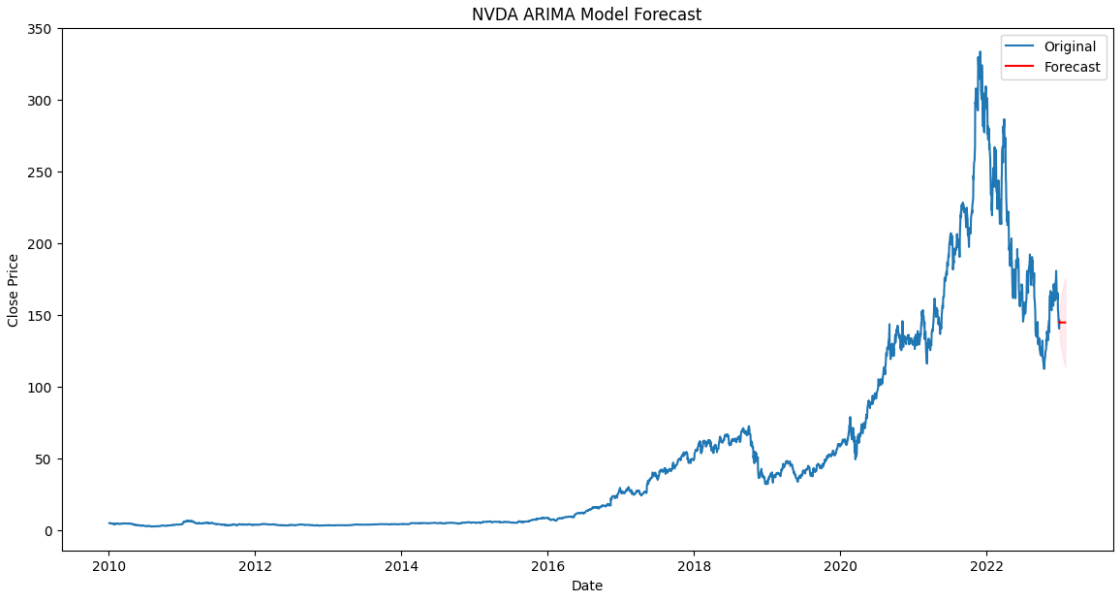
Ljung-Box (L1) (Q):	0.02	Jarque-Bera (JB):	457800.46
Prob(Q):	0.88	Prob(JB):	0.00

```
Heteroskedasticity (H):                156.76   Skew:
-2.28
Prob(H) (two-sided):                     0.00   Kurtosis:
60.79
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```



```
### NVDA ###
                               SARIMAX Results
==========================================================================
Dep. Variable:                   close   No. Observations:                 3271
Model:                 ARIMA(5, 1, 0)   Log Likelihood               -8156.704
Date:                Fri, 28 Jun 2024   AIC                          16325.408
Time:                        15:59:52   BIC                          16361.964
Sample:                             0   HQIC                         16338.500
                                - 3271
Covariance Type:                  opg
==========================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------
ar.L1         -0.0429      0.007     -5.849      0.000      -0.057      -0.029
ar.L2         -0.0294      0.006     -4.978      0.000      -0.041      -0.018
ar.L3          0.0204      0.007      2.808      0.005       0.006       0.035
ar.L4         -0.0303      0.006     -5.353      0.000      -0.041      -0.019
ar.L5          0.0587      0.007      7.988      0.000       0.044       0.073
sigma2         8.5928      0.065    131.416      0.000       8.465       8.721
==========================================================================
===
Ljung-Box (L1) (Q):                   0.05   Jarque-Bera (JB):
57468.96
Prob(Q):                              0.83   Prob(JB):
0.00
Heteroskedasticity (H):            2606.84   Skew:
0.58
Prob(H) (two-sided):                  0.00   Kurtosis:
23.51
==========================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```



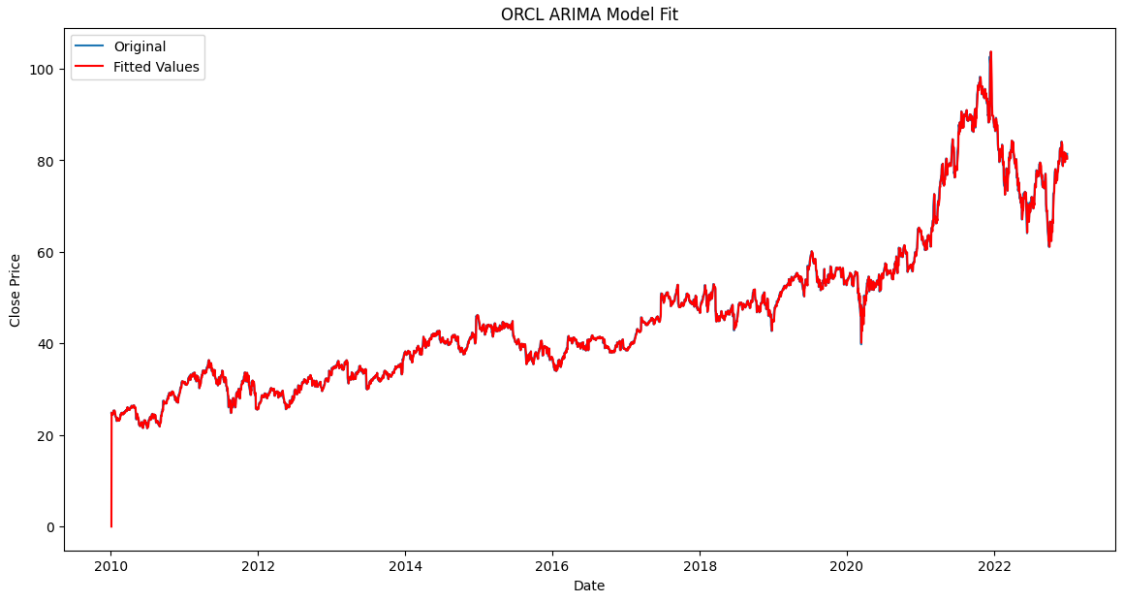


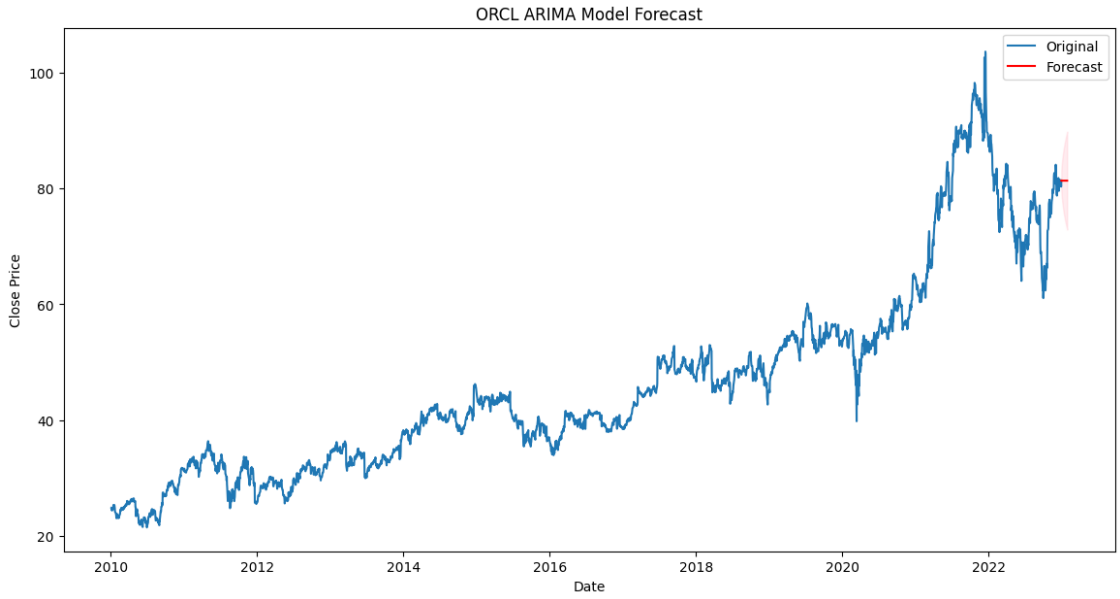
ORCL

SARIMAX Results

Dep. Variable:	close	No. Observations:	3271
Model:	ARIMA(5, 1, 0)	Log Likelihood	-4022.698
Date:	Fri, 28 Jun 2024	AIC	8057.395
Time:	15:59:54	BIC	8093.950

```
Sample:                              0   HQIC                          8070.486
                                 - 3271
Covariance Type:                   opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0591      0.009     -6.908      0.000      -0.076      -0.042
ar.L2         -0.0132      0.010     -1.340      0.180      -0.032       0.006
ar.L3          0.0080      0.009      0.914      0.360      -0.009       0.025
ar.L4          0.0232      0.012      1.907      0.057      -0.001       0.047
ar.L5         -0.0118      0.007     -1.618      0.106      -0.026       0.002
sigma2         0.6856      0.004    154.980      0.000       0.677       0.694
==============================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
149874.76
Prob(Q):                              0.95   Prob(JB):
0.00
Heteroskedasticity (H):               5.99   Skew:
1.23
Prob(H) (two-sided):                  0.00   Kurtosis:
36.08
==============================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```





TSLA

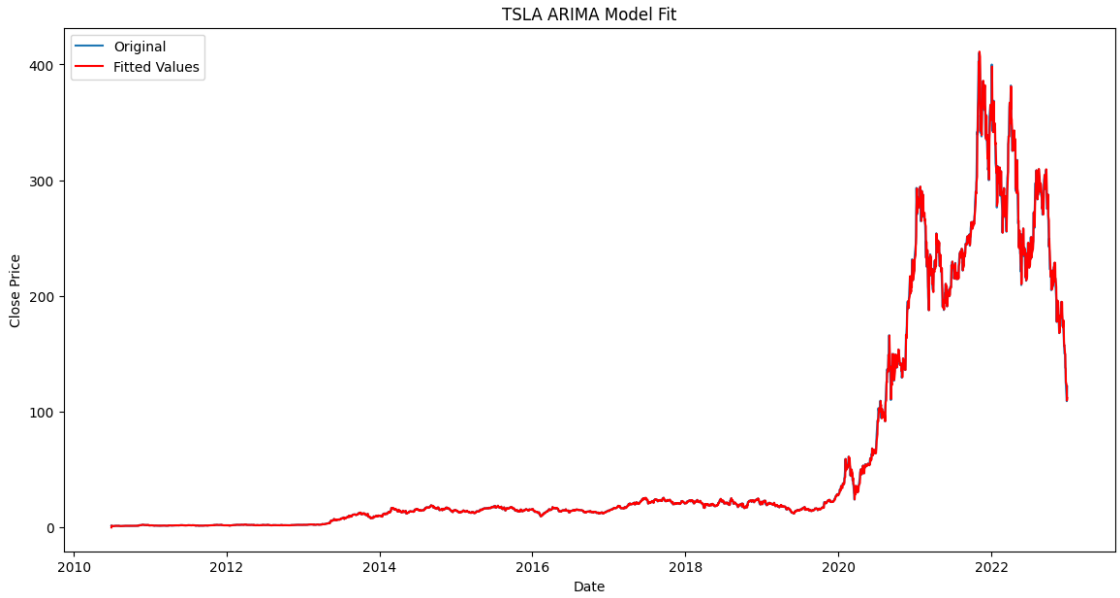
SARIMAX Results

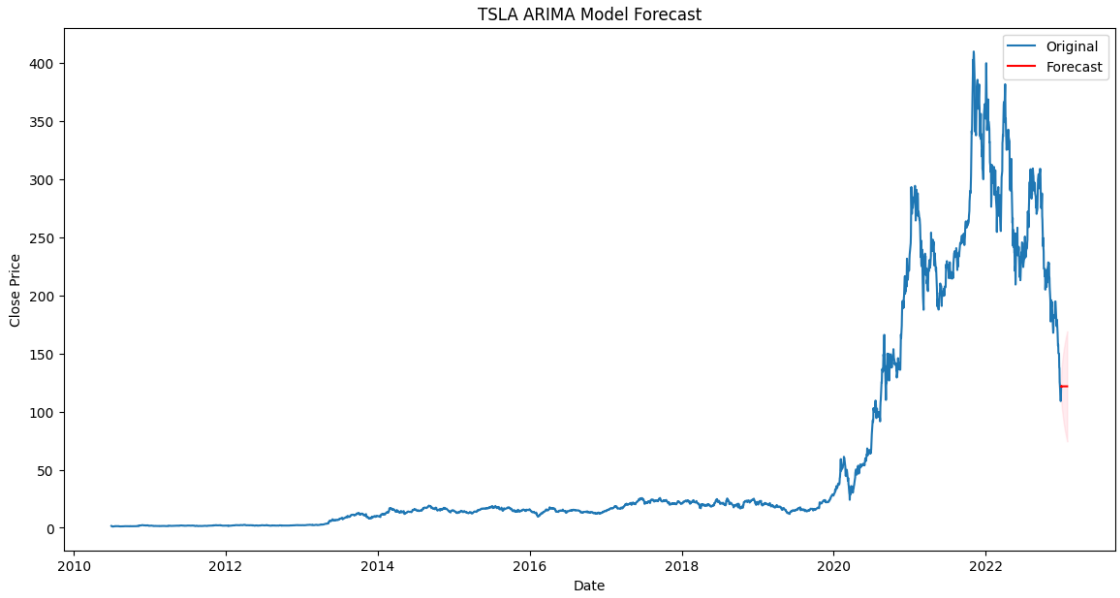
Dep. Variable:	close	No. Observations:	3148
Model:	ARIMA(5, 1, 0)	Log Likelihood	-9130.259
Date:	Fri, 28 Jun 2024	AIC	18272.519
Time:	15:59:56	BIC	18308.844
Sample:	0	HQIC	18285.553
	- 3148		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.0364	0.008	-4.699	0.000	-0.052	-0.021
ar.L2	0.0172	0.007	2.608	0.009	0.004	0.030
ar.L3	-0.0010	0.009	-0.112	0.911	-0.018	0.016
ar.L4	0.0382	0.007	5.357	0.000	0.024	0.052
ar.L5	-0.0126	0.007	-1.834	0.067	-0.026	0.001
sigma2	19.3876	0.136	142.867	0.000	19.122	19.654

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	82423.80
Prob(Q):	0.99	Prob(JB):	0.00

```
Heteroskedasticity (H):            1033.01   Skew:
-0.14
Prob(H) (two-sided):                  0.00   Kurtosis:
28.07
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```





1.4 Risk ve Getiri Hesaplama

Öncelikle her bir hisse senedinin günlük getirisini hesaplayalım ve ardından ortalama getiri ve volatiliteyi hesaplayalım.

[33]:
```
# Günlük getiri hesaplayalım
big_tech_stock_prices['daily_return'] = big_tech_stock_prices.
 ↪groupby('stock_symbol')['close'].pct_change()
```

[34]:
```
# Ortalama getiri ve volatiliteyi hesaplayalım
mean_returns = big_tech_stock_prices.groupby('stock_symbol')['daily_return'].
 ↪mean()
volatilities = big_tech_stock_prices.groupby('stock_symbol')['daily_return'].
 ↪std()
```

[35]:
```
# Sonuçları bir DataFrame'de toplayalım
risk_return_df = pd.DataFrame({'mean_return': mean_returns, 'volatility':␣
 ↪volatilities})
print(risk_return_df)
```

```
              mean_return  volatility
stock_symbol
AAPL             0.001030    0.018102
ADBE             0.000876    0.019975
AMZN             0.000990    0.020811
CRM              0.000864    0.023133
CSCO             0.000343    0.016857
GOOGL            0.000674    0.017032
IBM              0.000135    0.014257
INTC             0.000243    0.018610
META             0.000812    0.025248
MSFT             0.000762    0.016394
NFLX             0.001654    0.032778
NVDA             0.001448    0.028089
ORCL             0.000497    0.016441
TSLA             0.002024    0.036001
```

1.5 Portföy Optimizasyonu

Portföy optimizasyonu için rastgele portföyler oluşturup, her bir portföyün beklenen getirisini ve riskini hesaplayalım. Daha sonra, bu portföylerin performansını değerlendirelim ve etkin sınırı (efficient frontier) bulalım.

[36]:
```
```

```
# Ortalama getiri ve kovaryans matrisini hesaplayalım
mean_returns = big_tech_stock_prices.groupby('stock_symbol')['daily_return'].
 ↪mean()
cov_matrix = big_tech_stock_prices.pivot_table(index='date',␣
 ↪columns='stock_symbol', values='daily_return').cov()
```

[37]:
```
num_portfolios = 10000
results = np.zeros((4, num_portfolios))
weights_record = []

np.random.seed(42)

for i in range(num_portfolios):
    weights = np.random.random(len(mean_returns))
    weights /= np.sum(weights)
    weights_record.append(weights)
    portfolio_return = np.dot(weights, mean_returns)
    portfolio_stddev = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))
    results[0, i] = portfolio_return
    results[1, i] = portfolio_stddev
    results[2, i] = results[0, i] / results[1, i]  # Sharpe Ratio
    results[3, i] = i
```

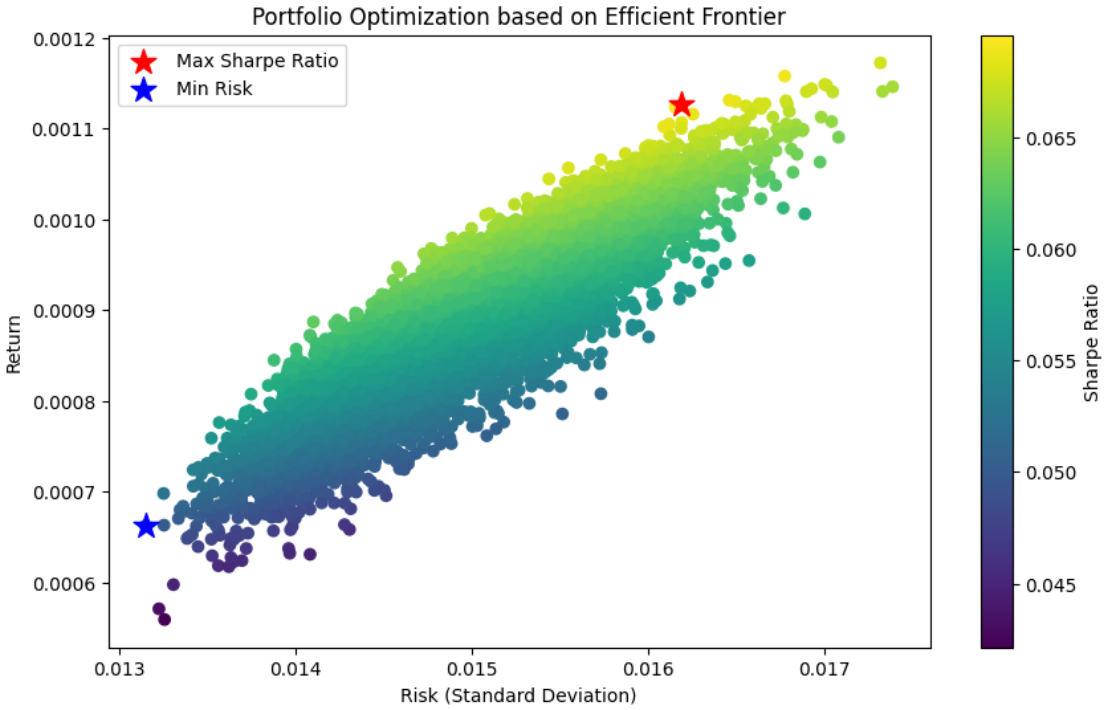
[38]:
```
# Sonuçları bir DataFrame'e aktaralım
results_frame = pd.DataFrame(results.T, columns=['Return', 'Risk', 'Sharpe␣
 ↪Ratio', 'Index'])
```

[39]:
```
# En yüksek Sharpe oranına sahip portföyü bulalım
max_sharpe_idx = results_frame['Sharpe Ratio'].idxmax()
max_sharpe_portfolio = results_frame.iloc[max_sharpe_idx]
max_sharpe_weights = weights_record[int(max_sharpe_portfolio[3])]
```

[40]:
```
# En düşük riske sahip portföyü bulalım
min_risk_idx = results_frame['Risk'].idxmin()
min_risk_portfolio = results_frame.iloc[min_risk_idx]
min_risk_weights = weights_record[int(min_risk_portfolio[3])]
```

[41]:
```
# Etkin sınırı görselleştirelim
plt.figure(figsize=(10, 6))
plt.scatter(results_frame['Risk'], results_frame['Return'],␣
 ↪c=results_frame['Sharpe Ratio'], cmap='viridis')
plt.colorbar(label='Sharpe Ratio')
plt.scatter(max_sharpe_portfolio[1], max_sharpe_portfolio[0], marker='*',␣
 ↪color='r', s=200, label='Max Sharpe Ratio')
plt.scatter(min_risk_portfolio[1], min_risk_portfolio[0], marker='*',␣
 ↪color='b', s=200, label='Min Risk')
plt.title('Portfolio Optimization based on Efficient Frontier')
```

```
plt.xlabel('Risk (Standard Deviation)')
plt.ylabel('Return')
plt.legend()
plt.show()
```



[42]:
```
# Maksimum Sharpe oranına sahip portföyün ağırlıklarını görüntüleyelim
print("Maximum Sharpe Ratio Portfolio Allocation\n")
print("Return:", max_sharpe_portfolio[0])
print("Risk:", max_sharpe_portfolio[1])
print("Sharpe Ratio:", max_sharpe_portfolio[2])
print("\nWeights:\n")
for i, txt in enumerate(mean_returns.index):
    print(f"{txt}: {max_sharpe_weights[i]}")
```

```
Maximum Sharpe Ratio Portfolio Allocation

Return: 0.0011261531815211605
Risk: 0.01618992014385136
Sharpe Ratio: 0.06955890897021212

Weights:

AAPL: 0.145079141641863
ADBE: 0.09957231007802969
```

```
AMZN: 0.044901909665446695
CRM: 0.004715235547238638
CSCO: 0.0002394915805060456
GOOGL: 0.01222050327036604
IBM: 0.0603229939361664
INTC: 0.03157832536513447
META: 0.04442818779683944
MSFT: 0.11727483265468244
NFLX: 0.16076961469796486
NVDA: 0.03465910486544035
ORCL: 0.07804974559639086
TSLA: 0.166188603303931
```

[43]:
```python
# Minimum riske sahip portföyün ağırlıklarını görüntüleyelim
print("\nMinimum Risk Portfolio Allocation\n")
print("Return:", min_risk_portfolio[0])
print("Risk:", min_risk_portfolio[1])
print("\nWeights:\n")
for i, txt in enumerate(mean_returns.index):
    print(f"{txt}: {min_risk_weights[i]}")
```

```
Minimum Risk Portfolio Allocation

Return: 0.0006620880991456467
Risk: 0.013150032175666179

Weights:

AAPL: 0.06416846695909594
ADBE: 0.011902320754213354
AMZN: 0.004506545231634426
CRM: 0.023598914913975123
CSCO: 0.16815192392182066
GOOGL: 0.09594330688067153
IBM: 0.16862455236142118
INTC: 0.07551054848814262
META: 0.054440515009294616
MSFT: 0.08836817841382316
NFLX: 0.0767351016928876
NVDA: 0.01844817897196795
ORCL: 0.11188296852396676
TSLA: 0.0377184778770851
```

1.6 Makroekonomik Verilerin Toplanması

Faiz oranları ve enflasyon verilerini toplayarak bir CSV dosyasına kaydedelim. Bu adım Kaggle notebook'larında internet erişimi olmadığı için lokal olarak yapılmalı ve ardından dosya Kaggle'a

yüklenmelidir.

[44]:
```
# Hisse senedi fiyatlarını ve makroekonomik verileri yükleyelim
big_tech_stock_prices = pd.read_csv('/kaggle/input/
 ↪big-tech-giants-stock-price-data/big_tech_stock_prices.csv')
macro_data = pd.read_csv('/kaggle/input/
 ↪macroeconomic-factors-affecting-us-housing-prices/DATA.csv')
```

[45]:
```
# Sütun adlarını görüntüleyelim
print(macro_data.columns)
```

```
Index(['DATE', 'UNRATE(%)', 'CONSUMER CONF INDEX', 'PPI-CONST MAT.',
       'CPIALLITEMS', 'INFLATION(%)', 'MORTGAGE INT. MONTHLY AVG(%)',
       'MED HOUSEHOLD INCOME', 'CORP. BOND YIELD(%)', 'MONTHLY HOME SUPPLY',
       '% SHARE OF WORKING POPULATION', 'GDP PER CAPITA', 'QUARTERLY REAL GDP',
       'QUARTERLY GDP GROWTH RATE (%)', 'CSUSHPISA'],
      dtype='object')
```

[46]:
```
# Gereksiz sütunları düşürelim ve uygun sütun adlarını kullanarak yeniden␣
 ↪adlandıralım
macro_data = macro_data.rename(columns={
    'UNRATE(%)': 'unemployment_rate',
    'CPIALLITEMS': 'cpi',
    'INFLATION(%)': 'inflation_rate',
    'MORTGAGE INT. MONTHLY AVG(%)': 'mortgage_interest_rate',
    'CORP. BOND YIELD(%)': 'corporate_bond_yield'
})
```

[47]:
```
# Tarih sütununu datetime formatına dönüştürelim
macro_data['DATE'] = pd.to_datetime(macro_data['DATE'])
```

[48]:
```
# Makroekonomik verilerdeki tarih sütununun ismini hisse senedi verileriyle␣
 ↪eşleştirelim
macro_data.rename(columns={'DATE': 'date'}, inplace=True)
```

[49]:
```
# Hisse senedi fiyatlarını yükleyelim
big_tech_stock_prices['date'] = pd.to_datetime(big_tech_stock_prices['date'])
```

[50]:
```
# Hisse senedi ve makro verileri birleştirelim
merged_data = pd.merge(big_tech_stock_prices, macro_data, on='date',␣
 ↪how='inner')
```

[51]:
```
# İlk birkaç satırı görüntüleyelim
print(merged_data.head())
print(merged_data.columns)
```

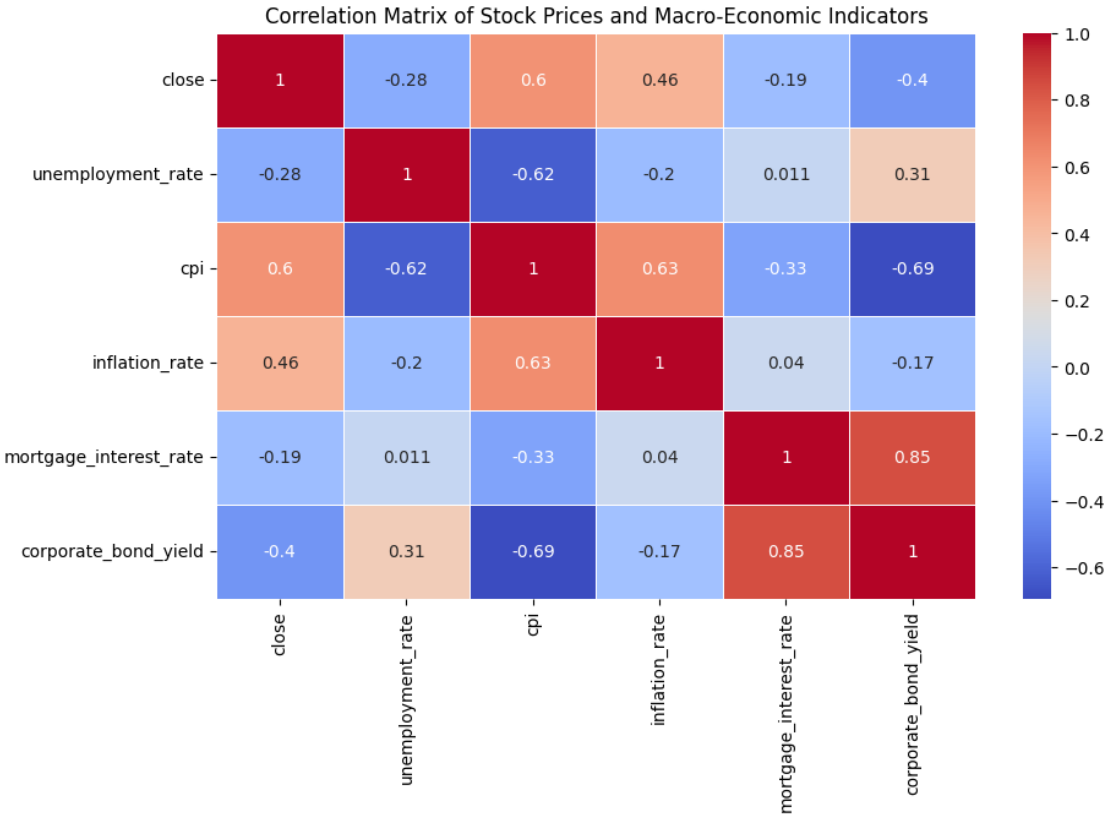
```
  stock_symbol       date      open      high       low     close  adj_close  \
0         AAPL 2010-01-04  7.622500  7.660714  7.585000  7.643214   6.515213
```

```
1         AAPL 2010-01-05  7.664286  7.699643  7.616071  7.656429   6.526476
2         AAPL 2010-01-06  7.656429  7.686786  7.526786  7.534643   6.422664
3         AAPL 2010-01-07  7.562500  7.571429  7.466071  7.520714   6.410790
4         AAPL 2010-01-08  7.510714  7.571429  7.466429  7.570714   6.453412

      volume  unemployment_rate  CONSUMER CONF INDEX  …  inflation_rate  \
0  493729600                9.9                 57.9  …        2.236447
1  601904800                9.6                 63.3  …        2.020986
2  552160000                9.4                 52.9  …        1.053349
3  477131200                9.4                 50.4  …        1.235193
4  447610800                9.5                 53.5  …        1.148105

   mortgage_interest_rate  MED HOUSEHOLD INCOME  corporate_bond_yield  \
0                  5.0980               49276.0                  5.29
1                  4.8875               49276.0                  4.96
2                  4.7375               49276.0                  4.88
3                  4.5640               49276.0                  4.72
4                  4.4275               49276.0                  4.49

   MONTHLY HOME SUPPLY  % SHARE OF WORKING POPULATION  GDP PER CAPITA  \
0                  6.2                      67.128435           48403
1                  9.3                      67.128435           48403
2                  8.3                      67.128435           48403
3                  8.9                      67.128435           48821
4                  8.8                      67.128435           48821

   QUARTERLY REAL GDP  QUARTERLY GDP GROWTH RATE (%)  CSUSHPISA
0           15605.628                       0.967705    164.040
1           15605.628                       0.967705    163.666
2           15605.628                       0.967705    163.400
3           15726.282                       0.773144    163.093
4           15726.282                       0.773144    162.530

[5 rows x 22 columns]
Index(['stock_symbol', 'date', 'open', 'high', 'low', 'close', 'adj_close',
       'volume', 'unemployment_rate', 'CONSUMER CONF INDEX', 'PPI-CONST MAT.',
       'cpi', 'inflation_rate', 'mortgage_interest_rate',
       'MED HOUSEHOLD INCOME', 'corporate_bond_yield', 'MONTHLY HOME SUPPLY',
       '% SHARE OF WORKING POPULATION', 'GDP PER CAPITA', 'QUARTERLY REAL GDP',
       'QUARTERLY GDP GROWTH RATE (%)', 'CSUSHPISA'],
      dtype='object')
```

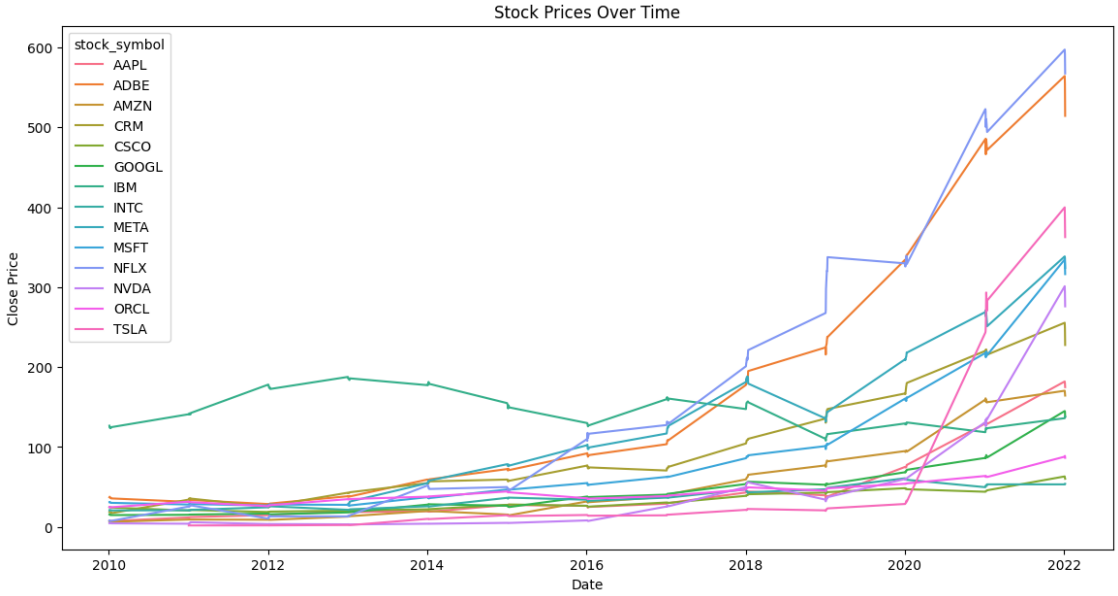
[52]:
```
# Makroekonomik göstergelerin hisse senedi fiyatlarıyla korelasyonunu␣
 ↪hesaplayalım
correlation_matrix = merged_data[['close', 'unemployment_rate', 'cpi',␣
 ↪'inflation_rate', 'mortgage_interest_rate', 'corporate_bond_yield']].corr()
print(correlation_matrix)
```

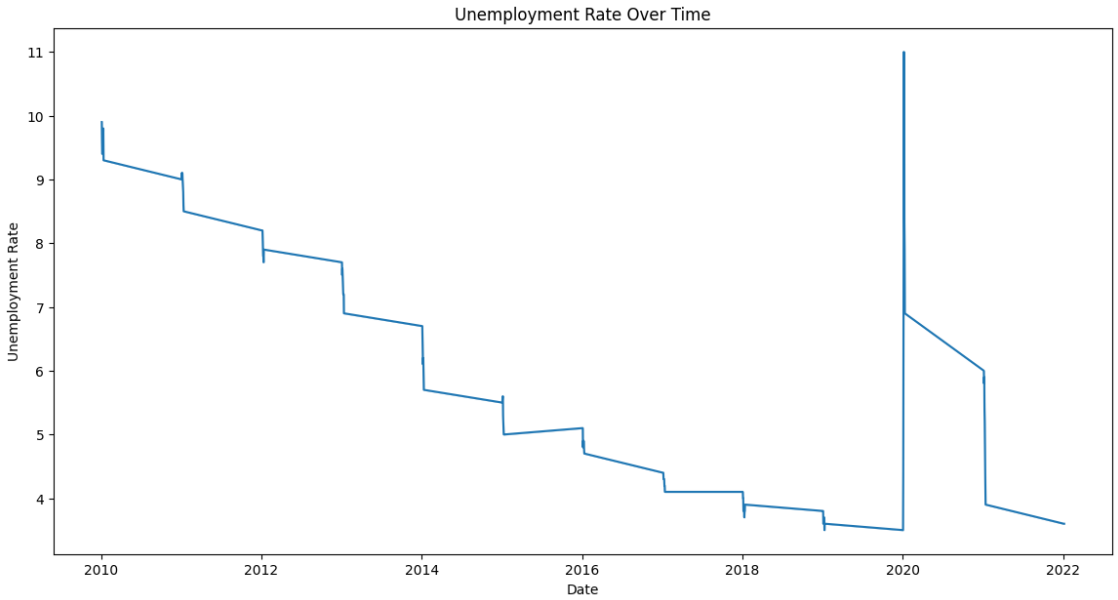
```
                           close  unemployment_rate       cpi  inflation_rate  \
close                   1.000000          -0.283223  0.603614        0.457935
unemployment_rate      -0.283223           1.000000 -0.622760       -0.199709
cpi                     0.603614          -0.622760  1.000000        0.626033
inflation_rate          0.457935          -0.199709  0.626033        1.000000
mortgage_interest_rate -0.192615           0.010875 -0.330658        0.039829
corporate_bond_yield   -0.399648           0.313109 -0.693106       -0.170279

                       mortgage_interest_rate  corporate_bond_yield
close                                -0.192615             -0.399648
unemployment_rate                     0.010875              0.313109
cpi                                  -0.330658             -0.693106
inflation_rate                        0.039829             -0.170279
mortgage_interest_rate                1.000000              0.846179
corporate_bond_yield                  0.846179              1.000000
```

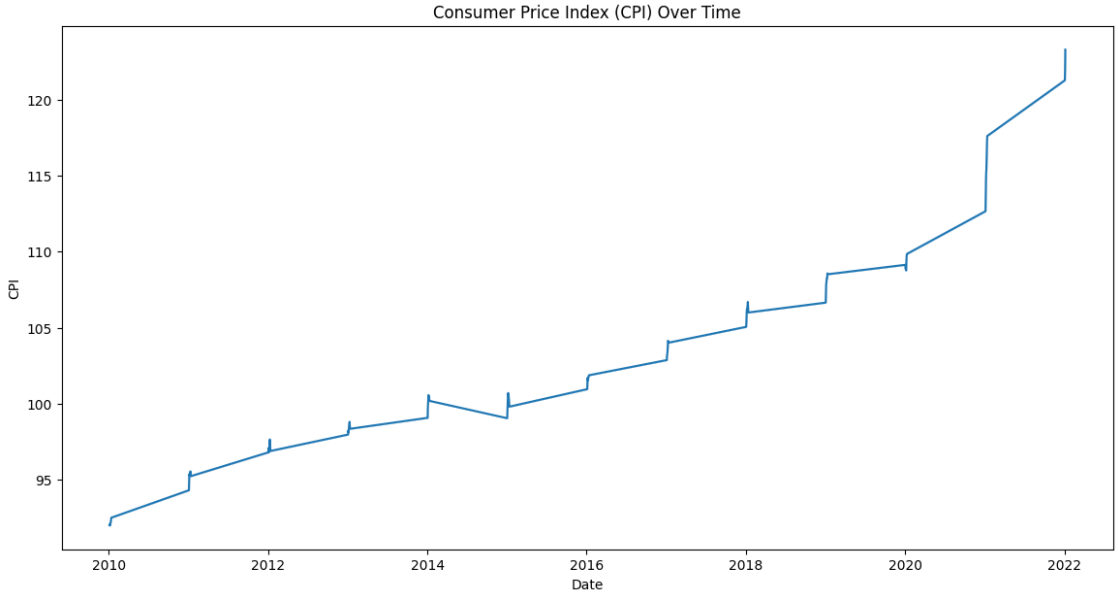
[53]:
```
# Korelasyon matrisini görselleştirelim
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix of Stock Prices and Macro-Economic Indicators')
plt.show()
```

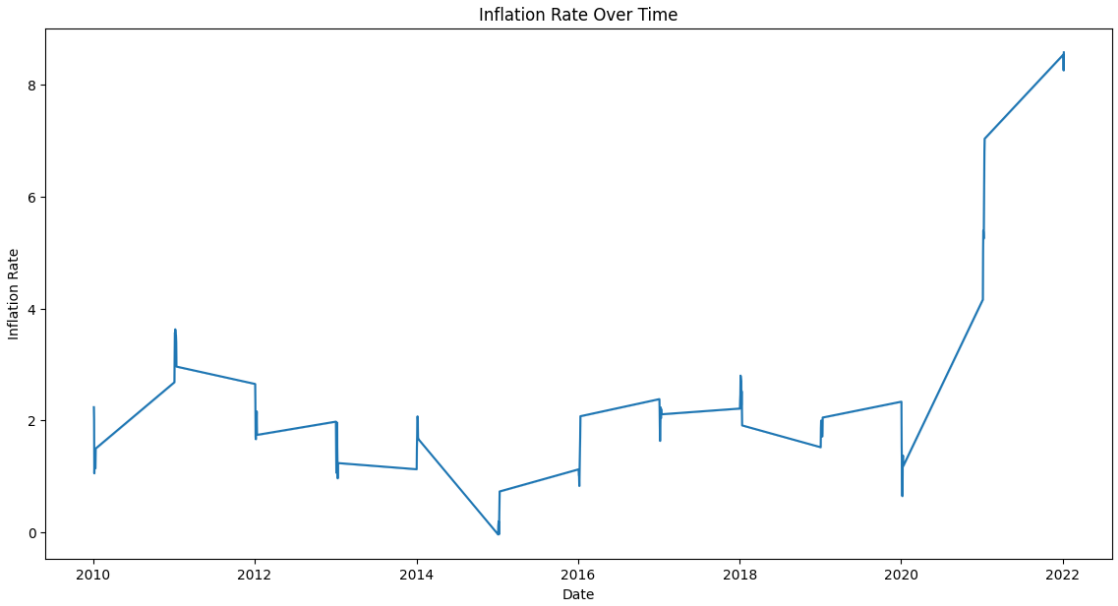


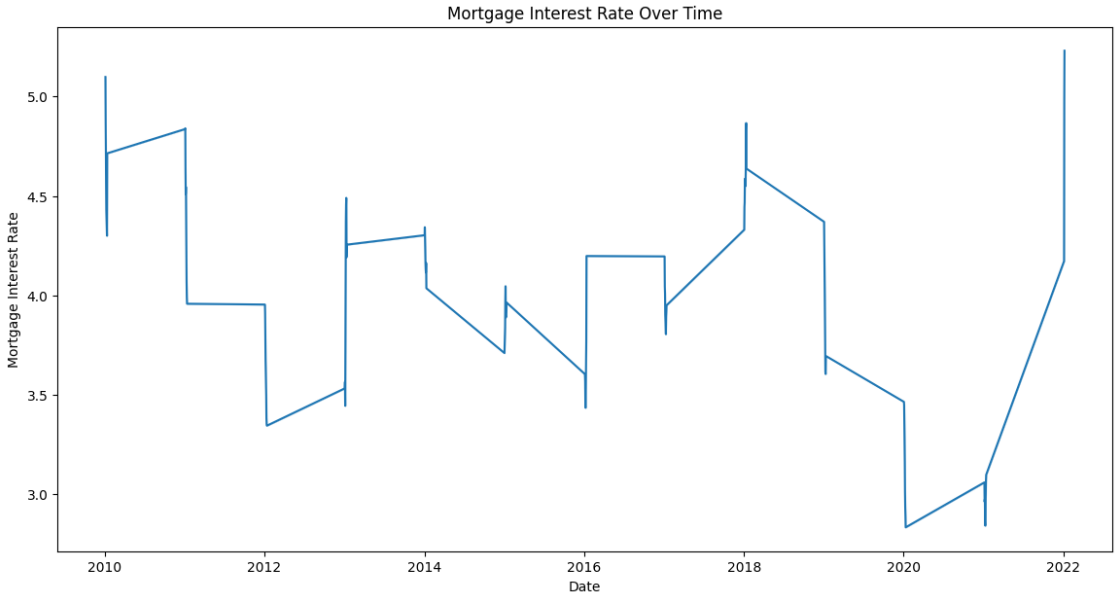
[54]:
```
# Makroekonomik göstergelerin zaman içindeki değişimlerini görselleştirelim
plt.figure(figsize=(14, 7))
sns.lineplot(data=merged_data, x='date', y='close', hue='stock_symbol')
plt.title('Stock Prices Over Time')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.show()

plt.figure(figsize=(14, 7))
sns.lineplot(data=merged_data, x='date', y='unemployment_rate')
plt.title('Unemployment Rate Over Time')
plt.xlabel('Date')
plt.ylabel('Unemployment Rate')
plt.show()

plt.figure(figsize=(14, 7))
sns.lineplot(data=merged_data, x='date', y='cpi')
plt.title('Consumer Price Index (CPI) Over Time')
plt.xlabel('Date')
plt.ylabel('CPI')
plt.show()

plt.figure(figsize=(14, 7))
sns.lineplot(data=merged_data, x='date', y='inflation_rate')
plt.title('Inflation Rate Over Time')
plt.xlabel('Date')
plt.ylabel('Inflation Rate')
plt.show()

plt.figure(figsize=(14, 7))
sns.lineplot(data=merged_data, x='date', y='mortgage_interest_rate')
plt.title('Mortgage Interest Rate Over Time')
plt.xlabel('Date')
plt.ylabel('Mortgage Interest Rate')
plt.show()

plt.figure(figsize=(14, 7))
sns.lineplot(data=merged_data, x='date', y='corporate_bond_yield')
plt.title('Corporate Bond Yield Over Time')
plt.xlabel('Date')
plt.ylabel('Corporate Bond Yield')
plt.show()
```

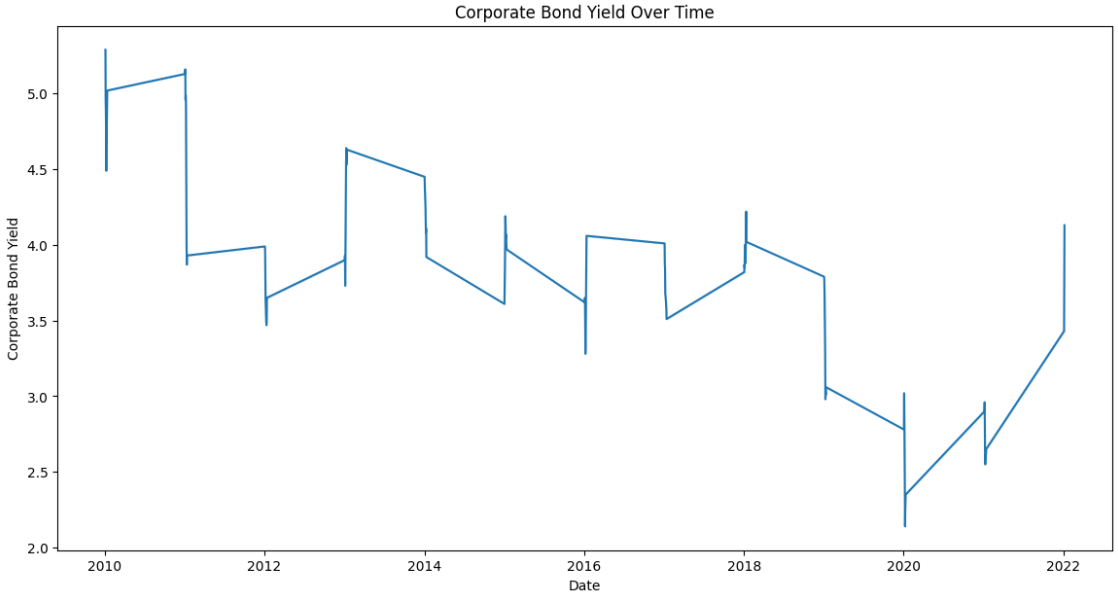

Stock Prices Over Time
stock_symbol
AAPL
ADBE
AMZN
CRM
CSCO
GOOGL
IBM
INTC
META
MSFT
NFLX
NVDA
ORCL
TSLA
Close Price
Date
0
100
200
300
400
500
600
2010
2012
2014
2016
2018
2020
2022


Unemployment Rate Over Time
Unemployment Rate
Date
4
5
6
7
8
9
10
11
2010
2012
2014
2016
2018
2020
2022


Consumer Price Index (CPI) Over Time
CPI
120
115
110
105
100
95
2010
2012
2014
2016
2018
2020
2022
Date


Inflation Rate Over Time
Inflation Rate
8
6
4
2
0
2010
2012
2014
2016
2018
2020
2022
Date





[55]:
```
# Model için verileri hazırlayalım
X = merged_data[['unemployment_rate', 'cpi', 'inflation_rate',␣
  ↪'mortgage_interest_rate', 'corporate_bond_yield']]
y = merged_data['close']
```

[56]:
```
# Veriyi eğitim ve test setlerine bölelim
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[57]:
```
# Regresyon modelini eğitelim
model = LinearRegression()
model.fit(X_train, y_train)
```

[57]:
```
LinearRegression()
```

[58]:
```
# Modeli değerlendirelim
y_pred = model.predict(X_test)
r2_score = model.score(X_test, y_test)

print(f"R^2 Score: {r2_score}")
```

```
R^2 Score: 0.39069984795185186
```

[59]:
```
# Model katsayılarını ve etkilerini görüntüleyelim
coefficients = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
print(coefficients)
```

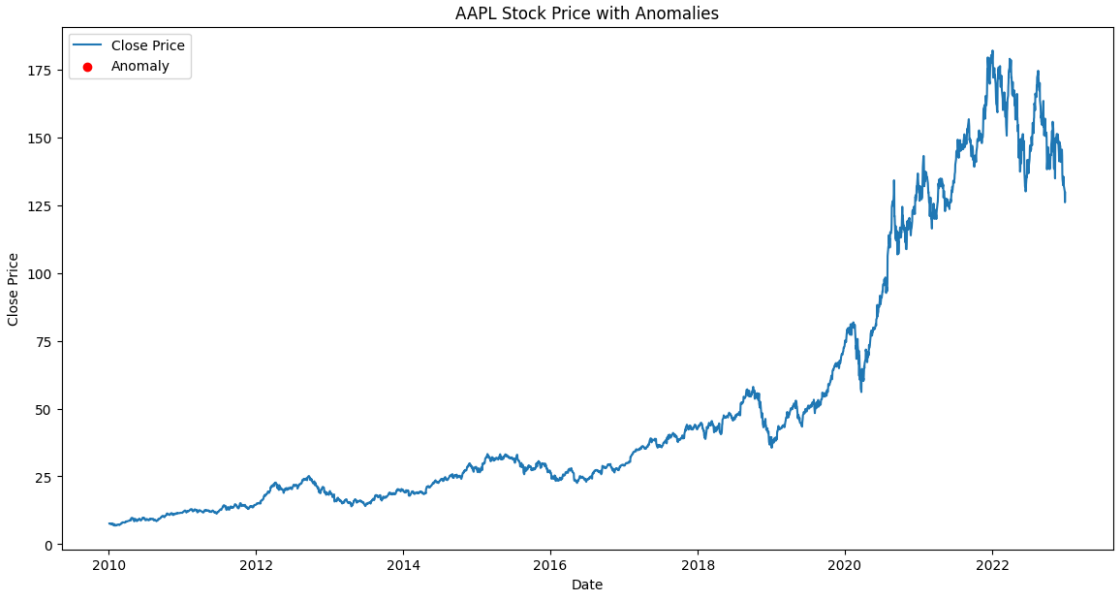
```
                        Coefficient
unemployment_rate          5.640217
cpi                        8.748596
inflation_rate             2.774549
mortgage_interest_rate    -9.828158
corporate_bond_yield       9.676488
```

1.7 Anomali Tespiti

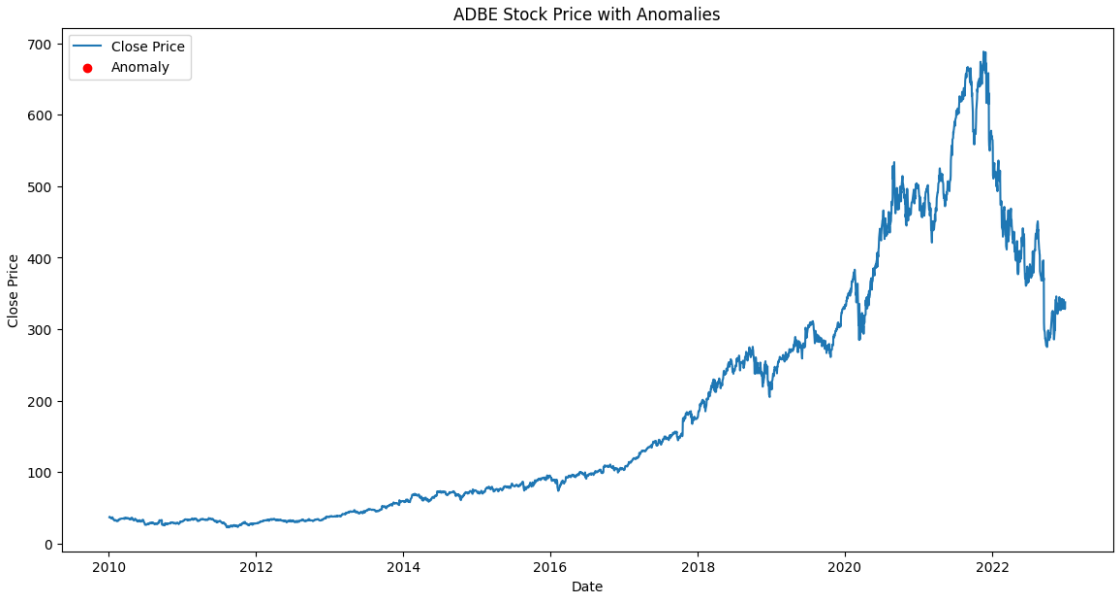
Zaman serisi verilerinde anormallikleri tespit etmek için çeşitli yöntemler kullanabiliriz. Örneğin, z-score veya IQR yöntemlerini kullanarak anormallikleri belirleyebiliriz. Aşağıda z-score yöntemini kullanarak anomali tespiti yapacağız.

[60]:
```
# Her bir hisse senedi için anomali tespiti yapalım
for symbol in unique_symbols:
    stock_data = big_tech_stock_prices[big_tech_stock_prices['stock_symbol'] ==␣
↪symbol]
    stock_data.set_index('date', inplace=True)

    # Z-score yöntemiyle anomali tespiti
    stock_data['z_score'] = (stock_data['close'] - stock_data['close'].mean()) /
↪ stock_data['close'].std()
    stock_data['anomaly'] = np.where(stock_data['z_score'].abs() > 3, True,␣
↪False)

    # Anomalileri görselleştirelim
    plt.figure(figsize=(14, 7))
```

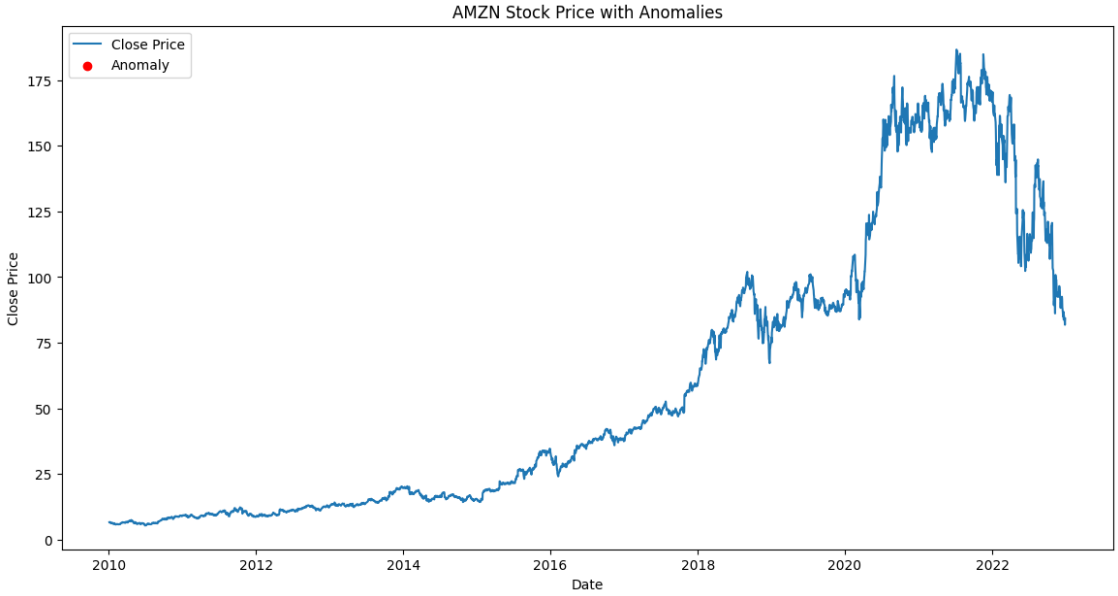
```
    plt.plot(stock_data.index, stock_data['close'], label='Close Price')
    plt.scatter(stock_data[stock_data['anomaly']].index,␣
↪stock_data[stock_data['anomaly']]['close'], color='red', label='Anomaly')
    plt.title(f'{symbol} Stock Price with Anomalies')
    plt.xlabel('Date')
    plt.ylabel('Close Price')
    plt.legend()
    plt.show()

    # Anomalileri görüntüleyelim
    anomalies = stock_data[stock_data['anomaly']]
    print(f"Anomalies for {symbol}:")
    print(anomalies[['close', 'z_score']])
    print("\n")
```




```
Anomalies for AAPL:
Empty DataFrame
Columns: [close, z_score]
Index: []
```

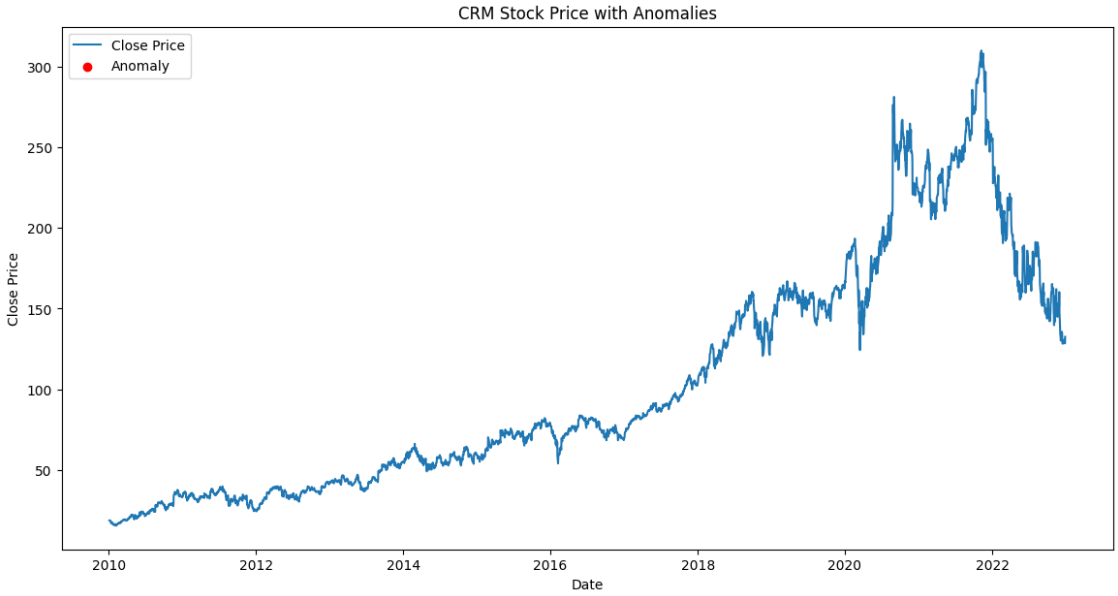



```
Anomalies for ADBE:
Empty DataFrame
Columns: [close, z_score]
Index: []
```

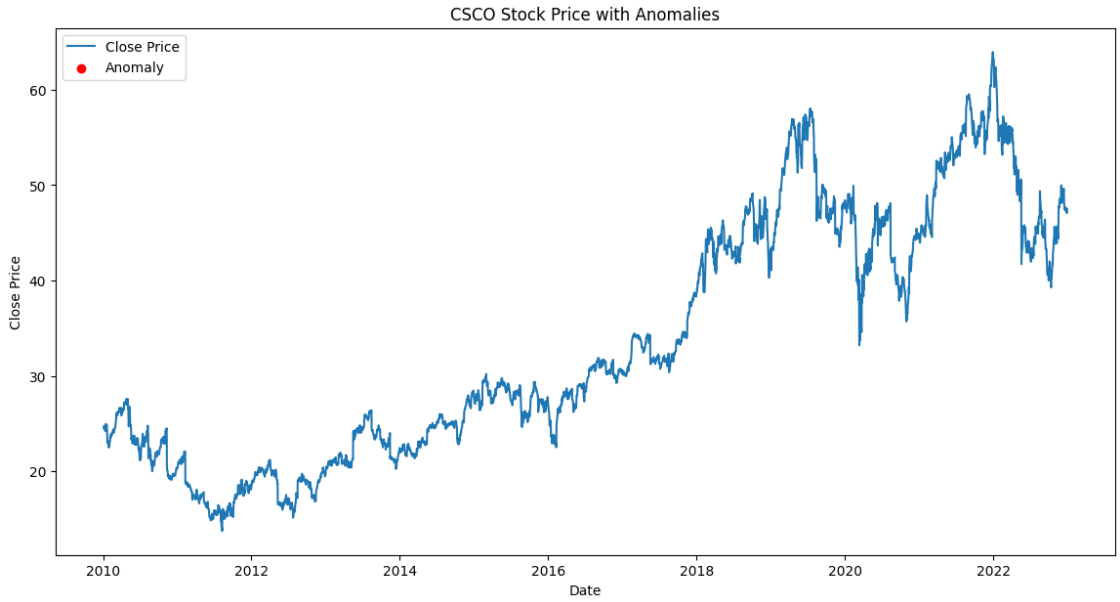



```
Anomalies for AMZN:
```

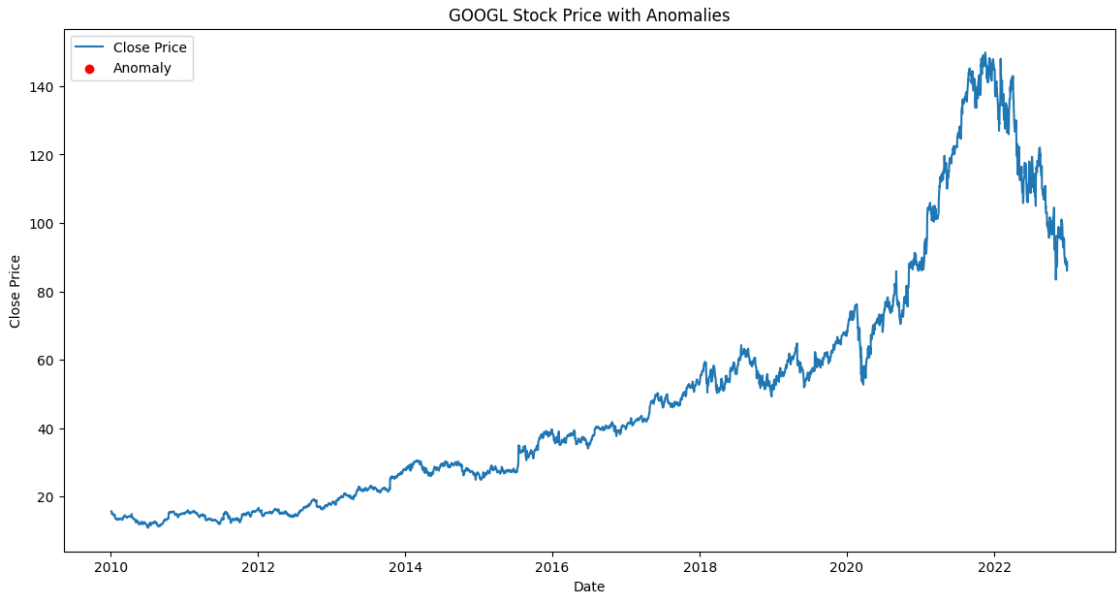
```
Empty DataFrame
Columns: [close, z_score]
Index: []
```




```
Anomalies for CRM:
Empty DataFrame
Columns: [close, z_score]
Index: []
```

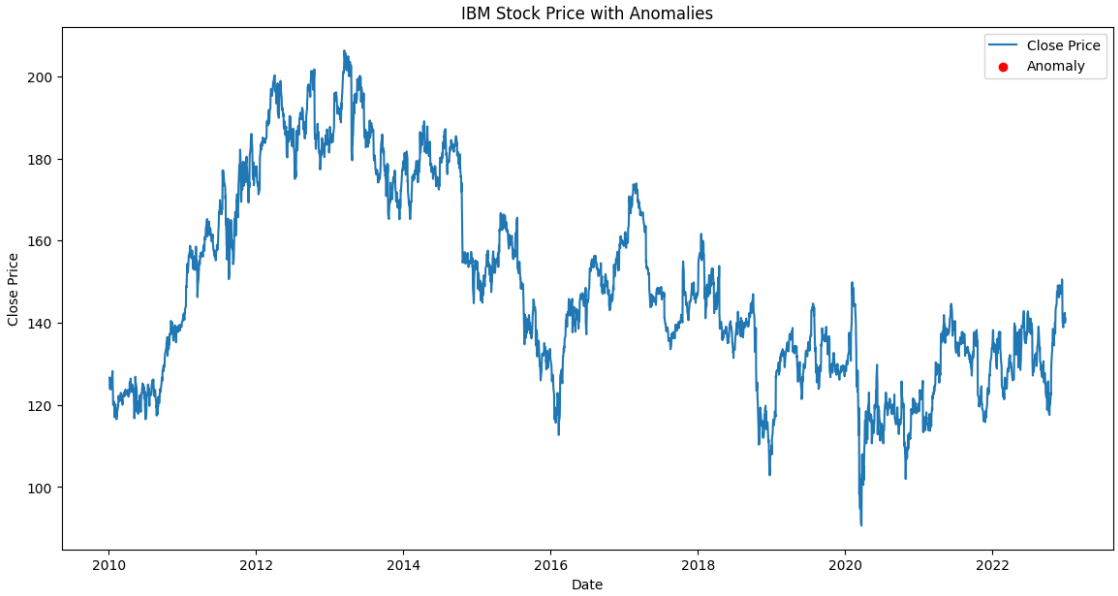



```
Anomalies for CSCO:
Empty DataFrame
Columns: [close, z_score]
Index: []
```

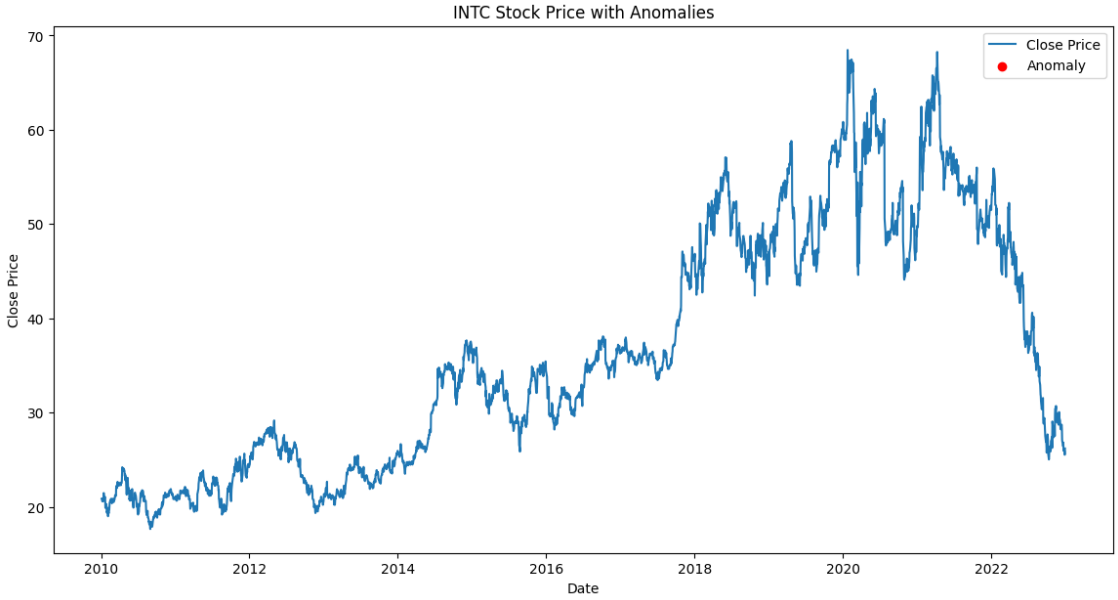



```
Anomalies for GOOGL:
```

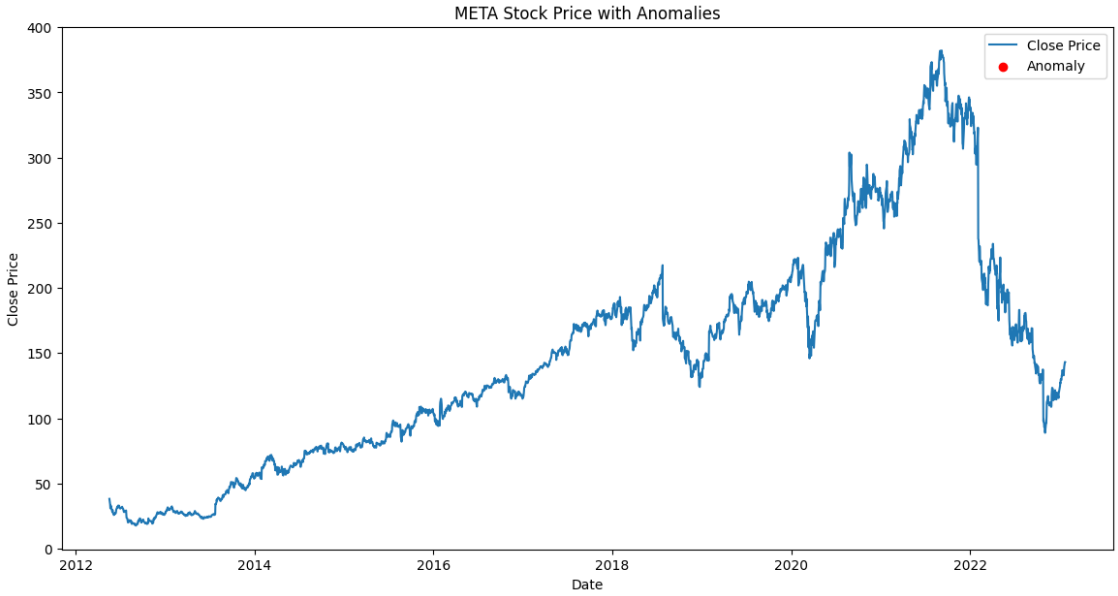
```
Empty DataFrame
Columns: [close, z_score]
Index: []
```




```
Anomalies for IBM:
Empty DataFrame
Columns: [close, z_score]
Index: []
```

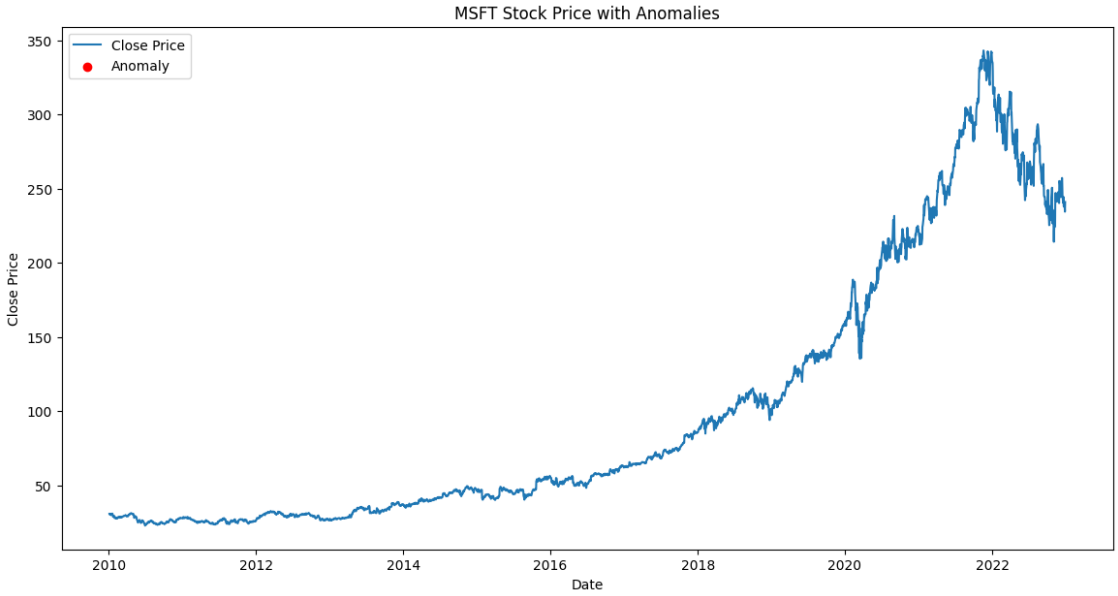



```
Anomalies for INTC:
Empty DataFrame
Columns: [close, z_score]
Index: []
```

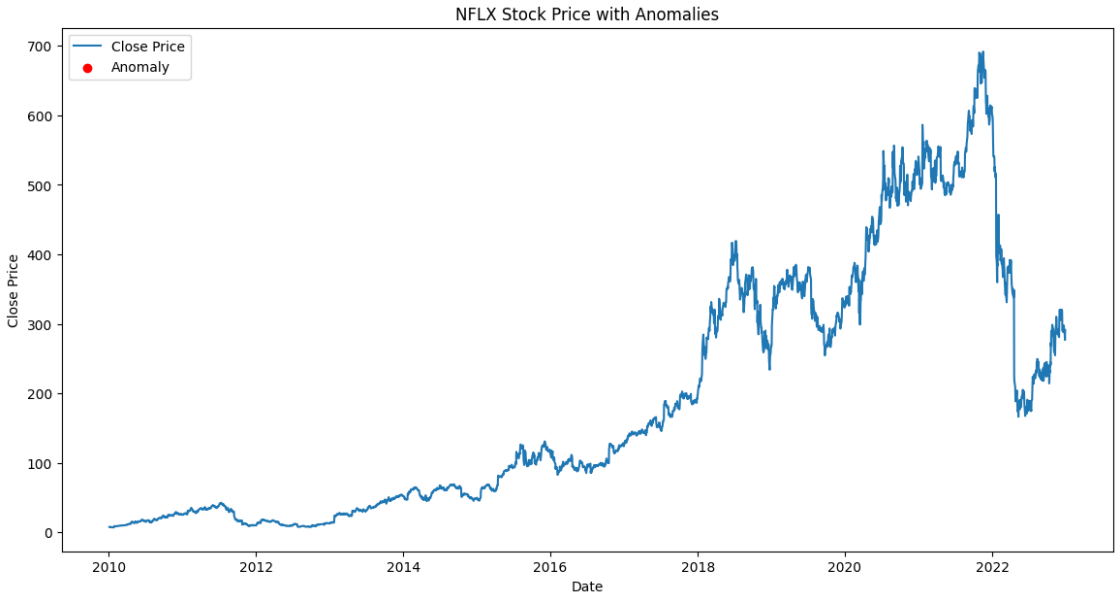



```
Anomalies for META:
```

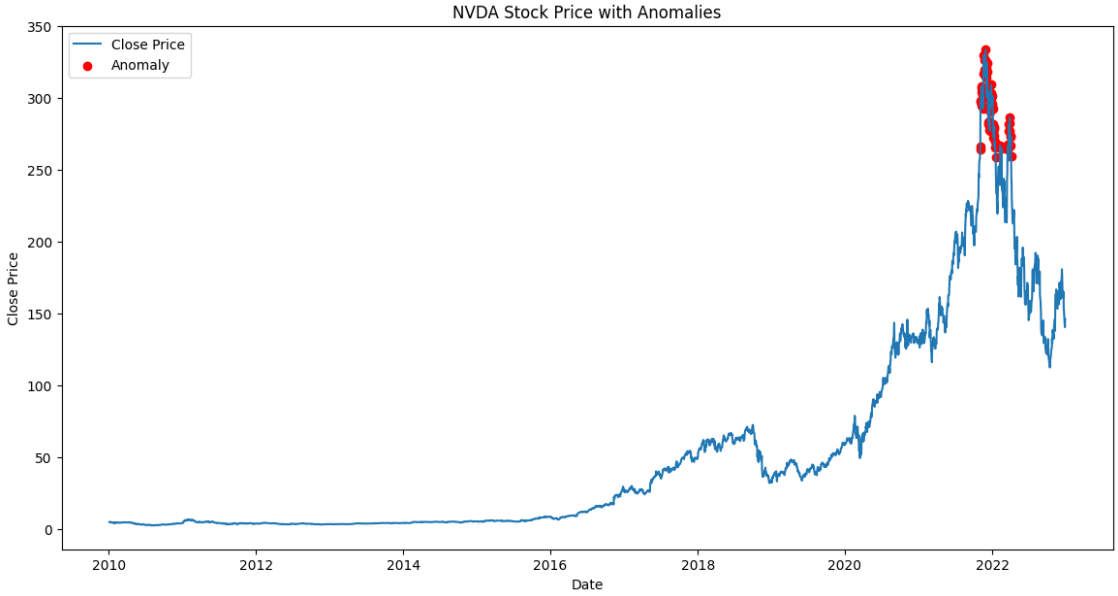
```
Empty DataFrame
Columns: [close, z_score]
Index: []
```




```
Anomalies for MSFT:
Empty DataFrame
Columns: [close, z_score]
Index: []
```

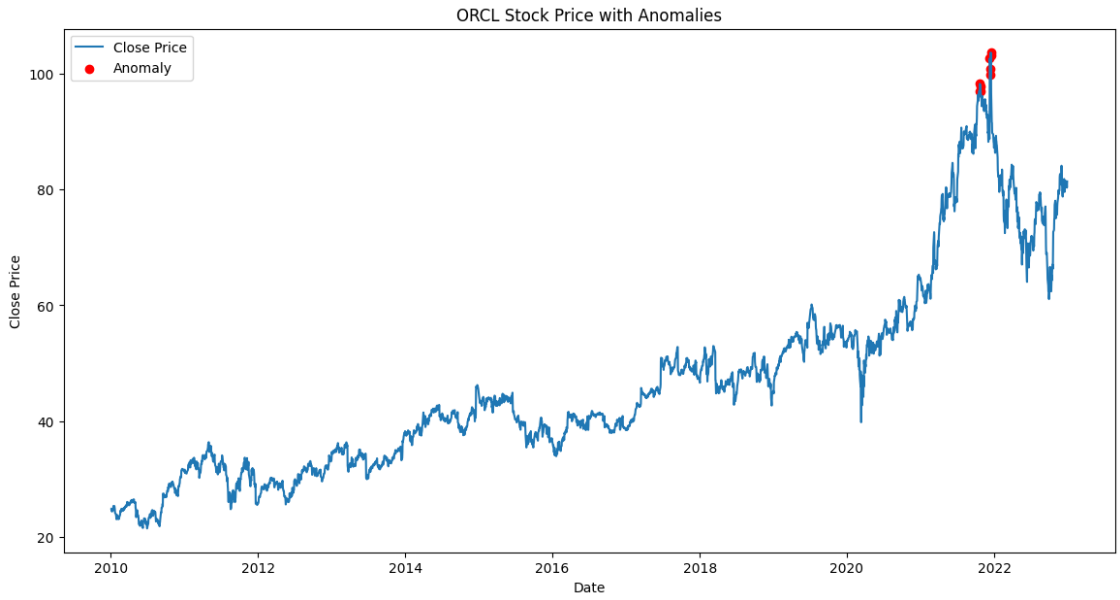



```
Anomalies for NFLX:
Empty DataFrame
Columns: [close, z_score]
Index: []
```



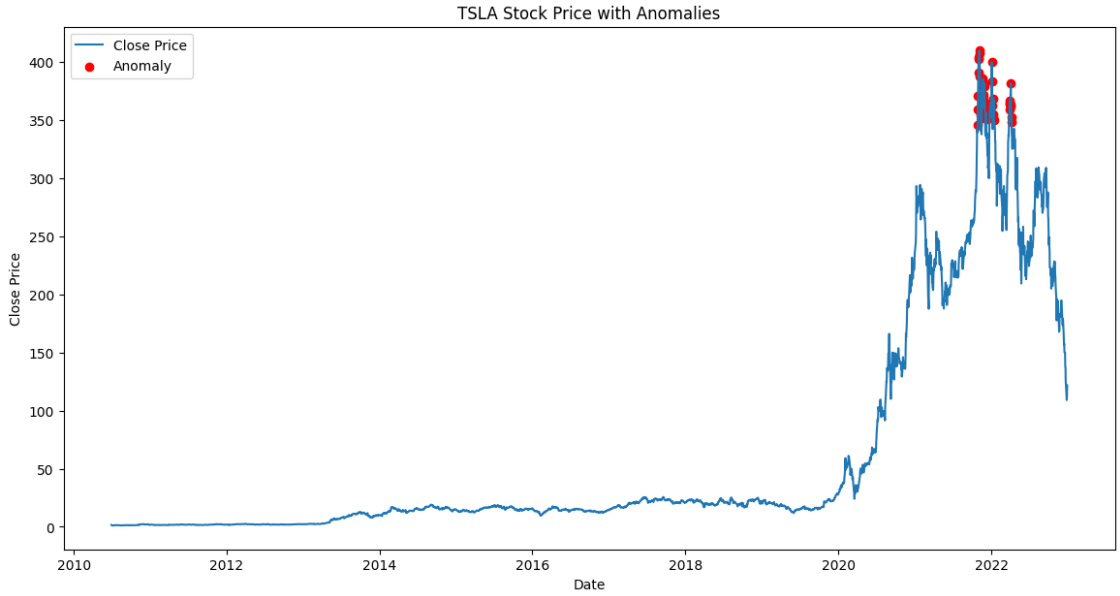

```
Anomalies for NVDA:
```

```
                close   z_score
date
2021-11-02  264.010010  3.072152
2021-11-03  265.980011  3.100505
2021-11-04  298.010010  3.561498
2021-11-05  297.519989  3.554446
2021-11-08  308.040009  3.705856
…                …         …
2022-03-30  276.899994  3.257672
2022-03-31  272.859985  3.199526
2022-04-01  267.119995  3.116913
2022-04-04  273.600006  3.210176
2022-04-05  259.309998  3.004507

[68 rows x 2 columns]
```




```
Anomalies for ORCL:
                close   z_score
date
2021-10-19   97.059998  3.012349
2021-10-22   98.250000  3.082917
2021-10-25   97.889999  3.061569
2021-10-26   96.980003  3.007605
2021-12-10  102.629997  3.342655
2021-12-13  100.889999  3.239472
2021-12-14   99.889999  3.180171
```

```
2021-12-15  103.650002  3.403142
2021-12-16  103.220001  3.377643
```




```
Anomalies for TSLA:
                close   z_score
date
2021-10-27  345.953339  3.005389
2021-10-28  359.013336  3.142079
2021-10-29  371.333344  3.271025
2021-11-01  402.863342  3.601028
2021-11-02  390.666656  3.473374
2021-11-03  404.619995  3.619414
2021-11-04  409.970001  3.675409
2021-11-05  407.363342  3.648127
2021-11-08  387.646667  3.441765
2021-11-10  355.983337  3.110366
2021-11-11  354.503326  3.094876
2021-11-16  351.576660  3.064244
2021-11-17  363.003326  3.183840
2021-11-18  365.459991  3.209552
2021-11-19  379.019989  3.351476
2021-11-22  385.623322  3.420588
2021-11-23  369.676666  3.253685
2021-11-24  372.000000  3.278002
2021-11-26  360.640015  3.159105
2021-11-29  378.996674  3.351232
```

```
2021-11-30  381.586670  3.378339
2021-12-01  365.000000  3.204738
2021-12-02  361.533325  3.168454
2021-12-07  350.583344  3.053848
2021-12-08  356.320007  3.113890
2021-12-23  355.666656  3.107052
2021-12-27  364.646667  3.201040
2021-12-28  362.823334  3.181956
2021-12-29  362.063324  3.174001
2021-12-30  356.779999  3.118704
2021-12-31  352.260010  3.071397
2022-01-03  399.926666  3.570292
2022-01-04  383.196655  3.395190
2022-01-05  362.706665  3.180735
2022-01-06  354.899994  3.099028
2022-01-10  352.706665  3.076071
2022-01-11  354.799988  3.097981
2022-01-12  368.739990  3.243882
2022-01-14  349.869995  3.046382
2022-03-28  363.946655  3.193713
2022-03-29  366.523346  3.220682
2022-03-30  364.663330  3.201214
2022-03-31  359.200012  3.144033
2022-04-01  361.529999  3.168419
2022-04-04  381.816681  3.380747
2022-04-05  363.753326  3.191690
2022-04-06  348.586670  3.032950
2022-04-07  352.420013  3.073071
```

1.8 Volatilite Modellemesi (Tüm Şirketler)

[61]:
```
!pip install arch
```

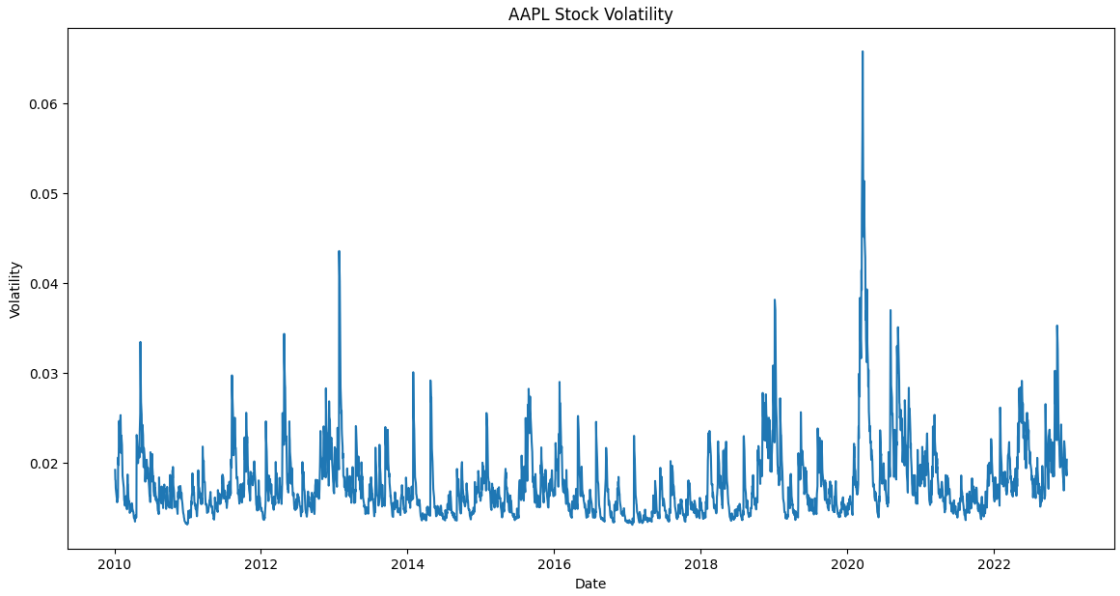
```
Collecting arch
  Downloading
arch-7.0.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(13 kB)
Requirement already satisfied: numpy>=1.22.3 in /opt/conda/lib/python3.10/site-
packages (from arch) (1.26.4)
Requirement already satisfied: scipy>=1.8 in /opt/conda/lib/python3.10/site-
packages (from arch) (1.11.4)
Requirement already satisfied: pandas>=1.4 in /opt/conda/lib/python3.10/site-
packages (from arch) (2.2.2)
Requirement already satisfied: statsmodels>=0.12 in
/opt/conda/lib/python3.10/site-packages (from arch) (0.14.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
```

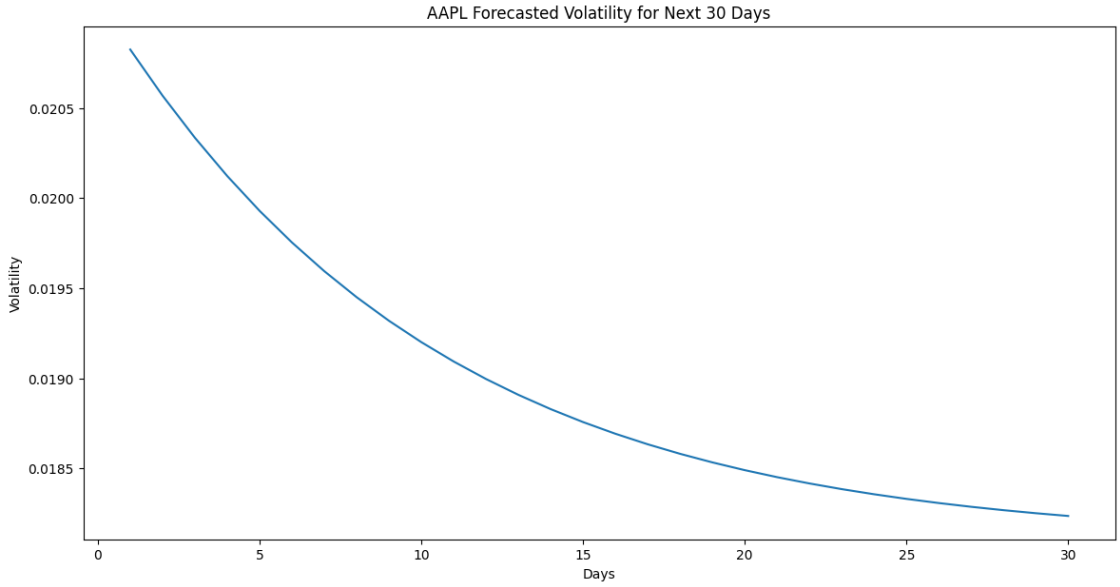
```
/opt/conda/lib/python3.10/site-packages (from pandas>=1.4->arch) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-
packages (from pandas>=1.4->arch) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.10/site-
packages (from pandas>=1.4->arch) (2023.4)
Requirement already satisfied: patsy>=0.5.4 in /opt/conda/lib/python3.10/site-
packages (from statsmodels>=0.12->arch) (0.5.6)
Requirement already satisfied: packaging>=21.3 in
/opt/conda/lib/python3.10/site-packages (from statsmodels>=0.12->arch) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/opt/conda/lib/python3.10/site-packages (from
packaging>=21.3->statsmodels>=0.12->arch) (3.1.1)
Requirement already satisfied: six in /opt/conda/lib/python3.10/site-packages
(from patsy>=0.5.4->statsmodels>=0.12->arch) (1.16.0)
Downloading
arch-7.0.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (983 kB)
                         983.4/983.4 kB
17.4 MB/s eta 0:00:00
Installing collected packages: arch
Successfully installed arch-7.0.0
```

[62]:
```python
from arch import arch_model

# Her bir hisse senedi için volatilite modellemesi yapalım
for symbol in unique_symbols:
    stock_data = big_tech_stock_prices[big_tech_stock_prices['stock_symbol'] ==␣
 ↪symbol]
    stock_data.set_index('date', inplace=True)

    # Günlük getiriyi hesaplayalım
    stock_data['return'] = stock_data['close'].pct_change().dropna()

    # GARCH(1,1) modeli
    model = arch_model(stock_data['return'].dropna(), vol='Garch', p=1, q=1)
    model_fit = model.fit(disp='off')
    print(f"Summary for {symbol}:")
    print(model_fit.summary())

    # Volatilite tahminleri
    volatility = model_fit.conditional_volatility

    # Volatiliteyi görselleştirelim
    plt.figure(figsize=(14, 7))
    plt.plot(volatility)
    plt.title(f'{symbol} Stock Volatility')
    plt.xlabel('Date')
    plt.ylabel('Volatility')
```

```
    plt.show()

    # Gelecekteki volatiliteyi tahmin edelim
    forecast_horizon = 30
    forecast = model_fit.forecast(horizon=forecast_horizon)
    forecast_volatility = np.sqrt(forecast.variance.values[-1, :])

    plt.figure(figsize=(14, 7))
    plt.plot(range(1, forecast_horizon+1), forecast_volatility)
    plt.title(f'{symbol} Forecasted Volatility for Next 30 Days')
    plt.xlabel('Days')
    plt.ylabel('Volatility')
    plt.show()
```

```
Summary for AAPL:
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                  return   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:                8715.58
Distribution:                  Normal   AIC:                           -17423.2
Method:            Maximum Likelihood   BIC:                           -17398.8
                                        No. Observations:                 3270
Date:                Fri, Jun 28 2024   Df Residuals:                     3269
Time:                        16:00:53   Df Model:                            1
                            Mean Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
mu         1.6334e-03  2.830e-04      5.772  7.843e-09 [1.079e-03,2.188e-03]
                          Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega      3.2758e-05  1.455e-06     22.522 2.545e-112 [2.991e-05,3.561e-05]
alpha[1]       0.1000  1.293e-02      7.734  1.045e-14   [7.466e-02,  0.125]
beta[1]        0.8000  9.918e-03     80.660      0.000     [  0.781,  0.819]
============================================================================

Covariance estimator: robust
```



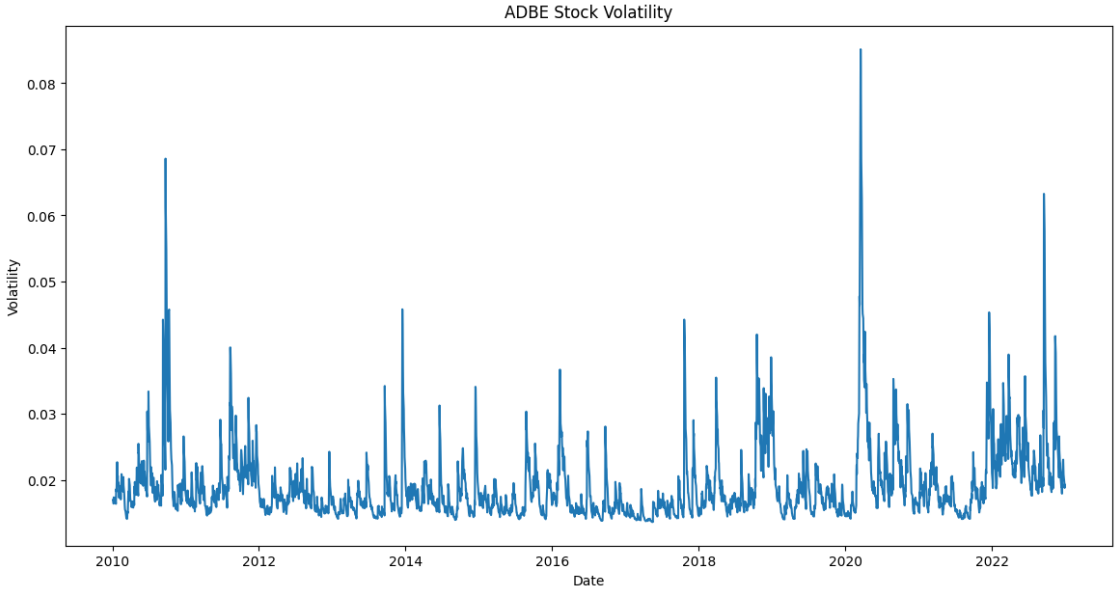


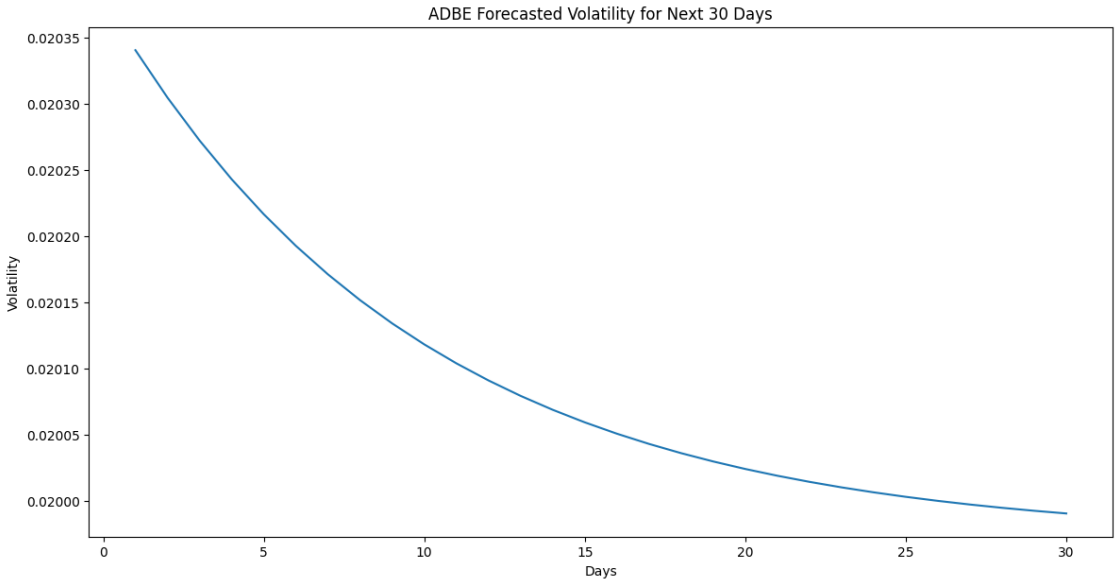
Summary for ADBE:

Constant Mean - GARCH Model Results

Dep. Variable:	return	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	8449.05
Distribution:	Normal	AIC:	-16890.1
Method:	Maximum Likelihood	BIC:	-16865.7

```
                                  No. Observations:                 3270
Date:                Fri, Jun 28 2024   Df Residuals:                 3269
Time:                        16:00:54   Df Model:                        1
                            Mean Model
================================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
--------------------------------------------------------------------------------
mu         1.4660e-03  3.113e-04      4.710  2.483e-06 [8.559e-04,2.076e-03]
                          Volatility Model
================================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
--------------------------------------------------------------------------------
omega      3.9138e-05  5.699e-06      6.867  6.537e-12 [2.797e-05,5.031e-05]
alpha[1]       0.1169  1.620e-02      7.216  5.355e-13   [8.516e-02,  0.149]
beta[1]        0.7850  2.383e-02     32.936 6.717e-238     [  0.738,  0.832]
================================================================================

Covariance estimator: robust
```





Summary for AMZN:

Constant Mean - GARCH Model Results

Dep. Variable:	return	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	8192.52
Distribution:	Normal	AIC:	-16377.0
Method:	Maximum Likelihood	BIC:	-16352.7
		No. Observations:	3270
Date:	Fri, Jun 28 2024	Df Residuals:	3269
Time:	16:00:55	Df Model:	1

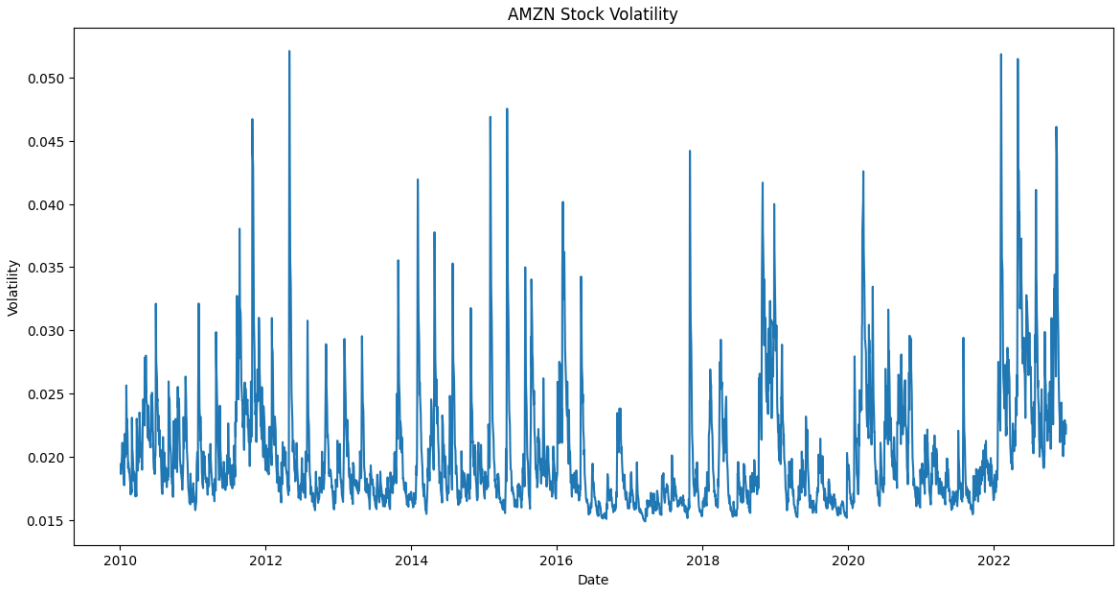
Mean Model

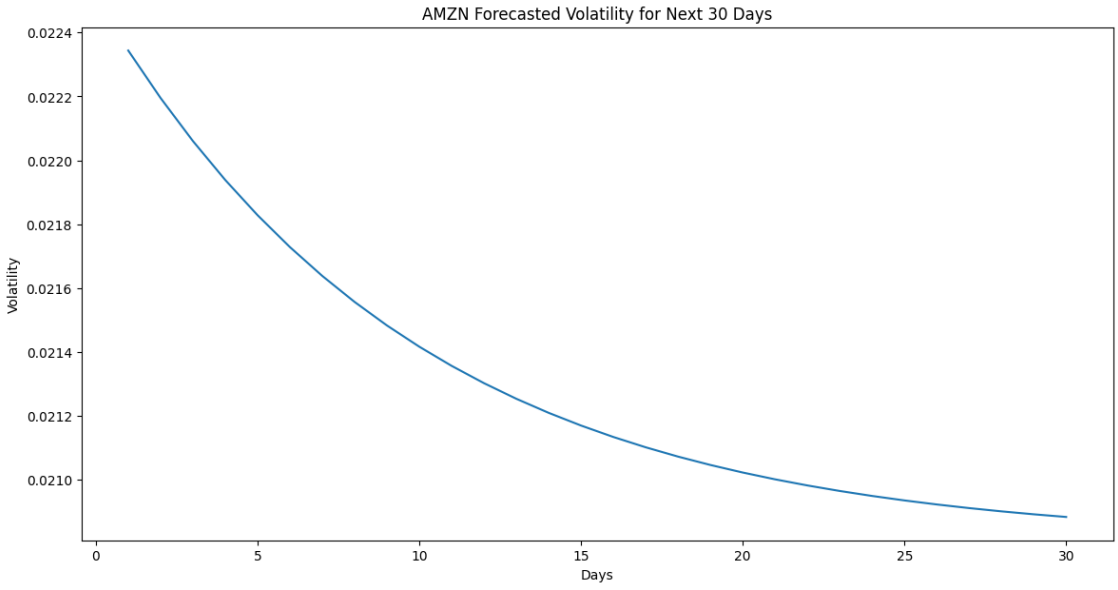
	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	1.4858e-03	3.260e-04	4.557	5.187e-06	[8.467e-04,2.125e-03]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	4.3303e-05	3.721e-06	11.636	2.701e-31	[3.601e-05,5.060e-05]
alpha[1]	0.1000	1.965e-02	5.090	3.587e-07	[6.149e-02, 0.139]
beta[1]	0.8000	1.587e-02	50.398	0.000	[0.769, 0.831]

Covariance estimator: robust



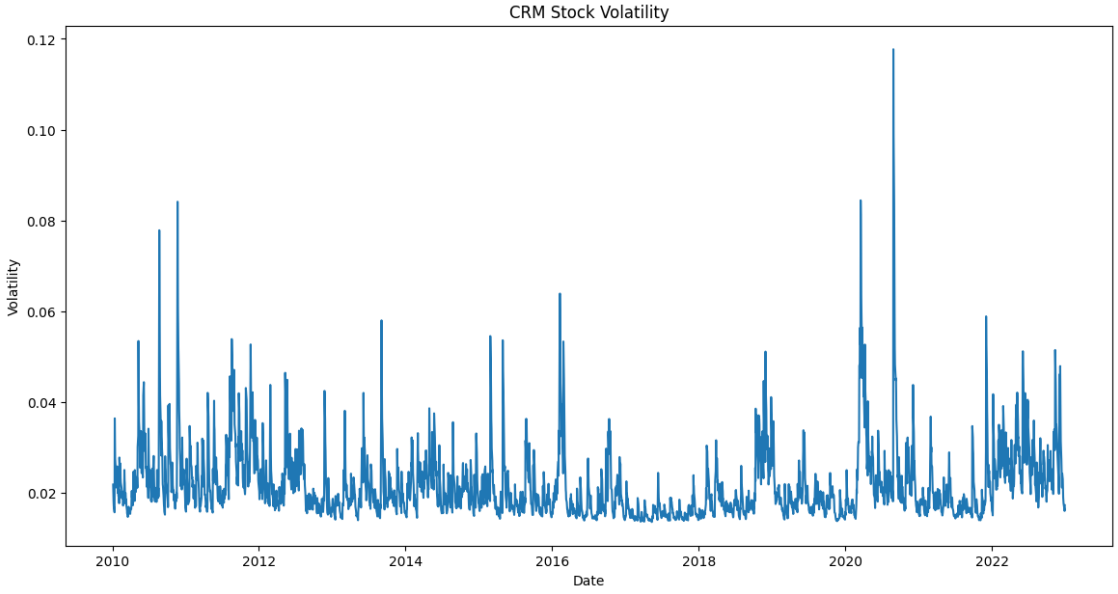


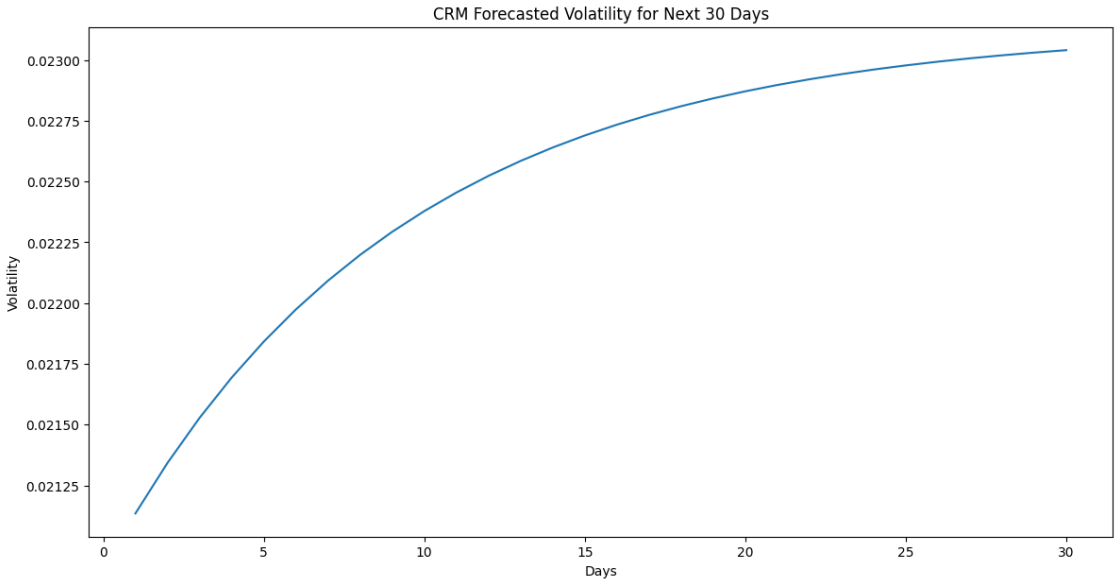
Summary for CRM:

Constant Mean - GARCH Model Results

Dep. Variable:	return	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	7894.08
Distribution:	Normal	AIC:	-15780.2
Method:	Maximum Likelihood	BIC:	-15755.8

```
                                 No. Observations:                 3270
Date:                Fri, Jun 28 2024   Df Residuals:                 3269
Time:                        16:00:56   Df Model:                        1
                            Mean Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
mu         1.3200e-03  3.999e-04      3.301  9.644e-04 [5.362e-04,2.104e-03]
                          Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega      5.3500e-05  1.382e-05      3.872  1.080e-04 [2.642e-05,8.058e-05]
alpha[1]       0.2000  5.552e-02      3.602  3.153e-04   [9.119e-02,  0.309]
beta[1]        0.7000  7.121e-02      9.830  8.364e-23     [  0.560,  0.840]
============================================================================

Covariance estimator: robust
```





Summary for CSCO:

Constant Mean - GARCH Model Results

Dep. Variable:	return	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	8910.14
Distribution:	Normal	AIC:	-17812.3
Method:	Maximum Likelihood	BIC:	-17787.9
		No. Observations:	3270
Date:	Fri, Jun 28 2024	Df Residuals:	3269
Time:	16:00:57	Df Model:	1

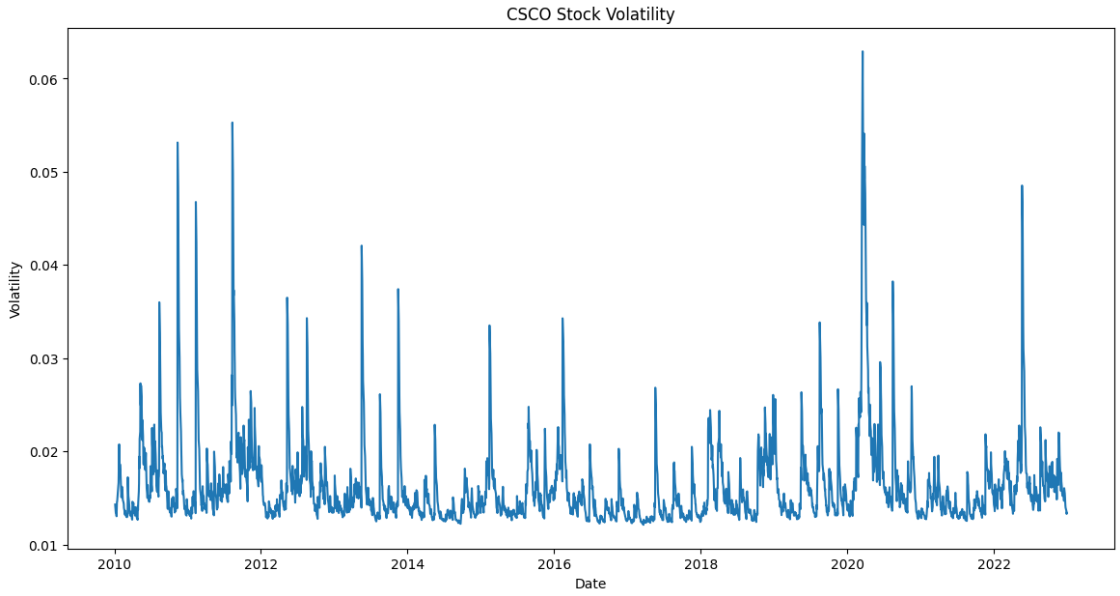
Mean Model

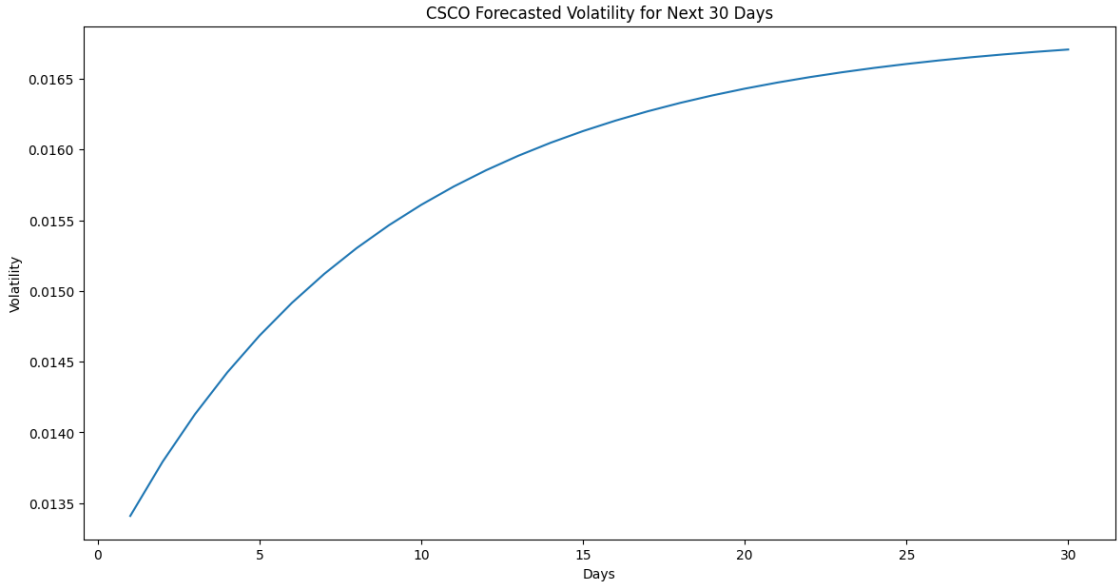
	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	4.0814e-04	3.302e-04	1.236	0.216	[-2.390e-04,1.055e-03]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	2.8406e-05	7.256e-07	39.151	0.000	[2.698e-05,2.983e-05]
alpha[1]	0.1000	2.947e-02	3.393	6.906e-04	[4.224e-02, 0.158]
beta[1]	0.8000	2.840e-02	28.169	1.385e-174	[0.744, 0.856]

Covariance estimator: robust



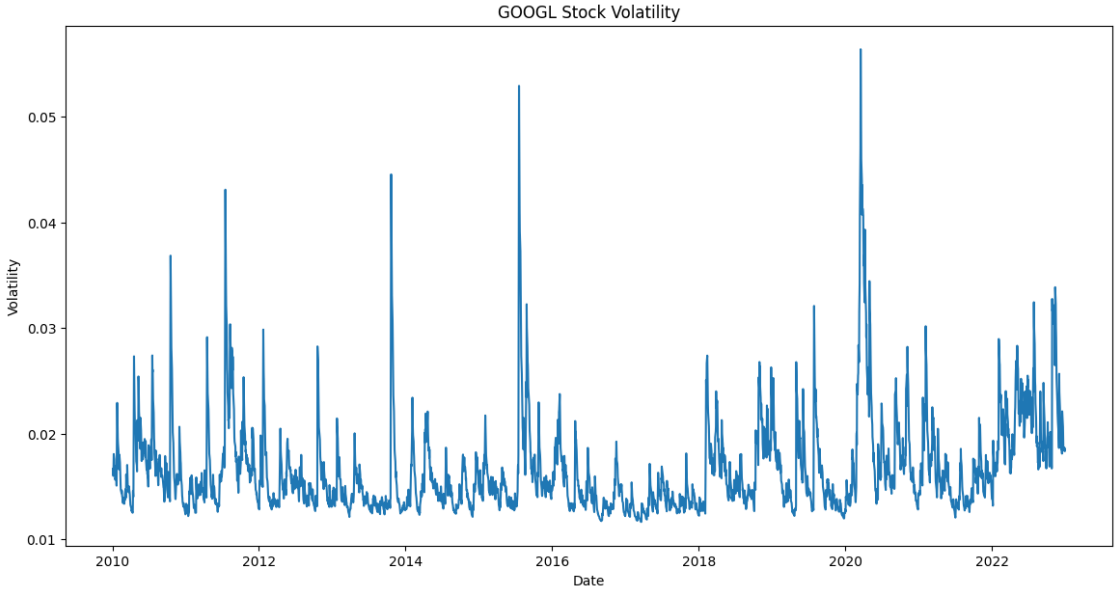


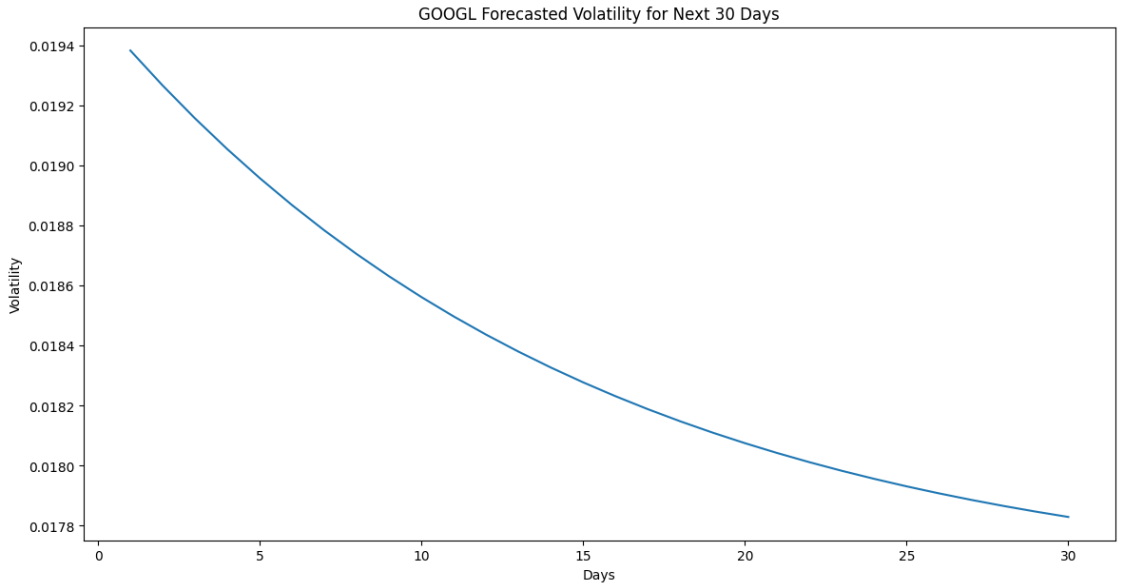
Summary for GOOGL:

Constant Mean - GARCH Model Results

Dep. Variable:	return	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	8847.66
Distribution:	Normal	AIC:	-17687.3
Method:	Maximum Likelihood	BIC:	-17663.0

```
                                     No. Observations:                 3270
Date:                Fri, Jun 28 2024   Df Residuals:                     3269
Time:                        16:00:58   Df Model:                            1
                            Mean Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
mu         7.0922e-04  2.867e-04      2.474  1.337e-02 [1.473e-04,1.271e-03]
                          Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega      2.0991e-05  2.053e-11  1.023e+06      0.000 [2.099e-05,2.099e-05]
alpha[1]       0.0957  2.917e-02      3.280  1.037e-03   [3.852e-02,  0.153]
beta[1]        0.8364  2.433e-02     34.374 6.247e-259     [  0.789,  0.884]
============================================================================

Covariance estimator: robust
```





Summary for IBM:

Constant Mean - GARCH Model Results

Dep. Variable:	return	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	9501.24
Distribution:	Normal	AIC:	-18994.5
Method:	Maximum Likelihood	BIC:	-18970.1
		No. Observations:	3270
Date:	Fri, Jun 28 2024	Df Residuals:	3269
Time:	16:00:59	Df Model:	1

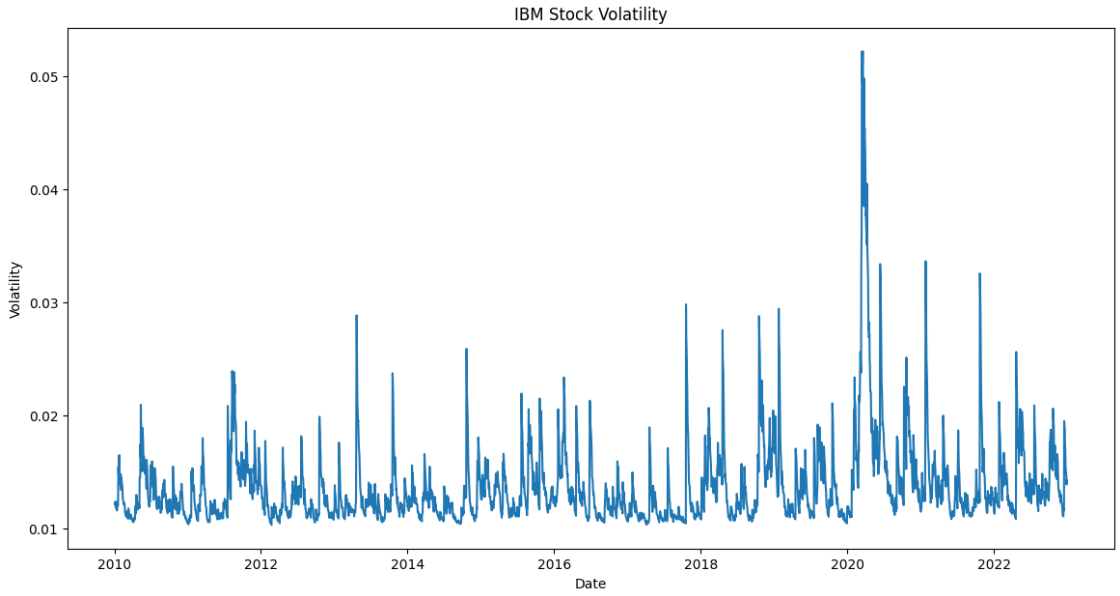
Mean Model

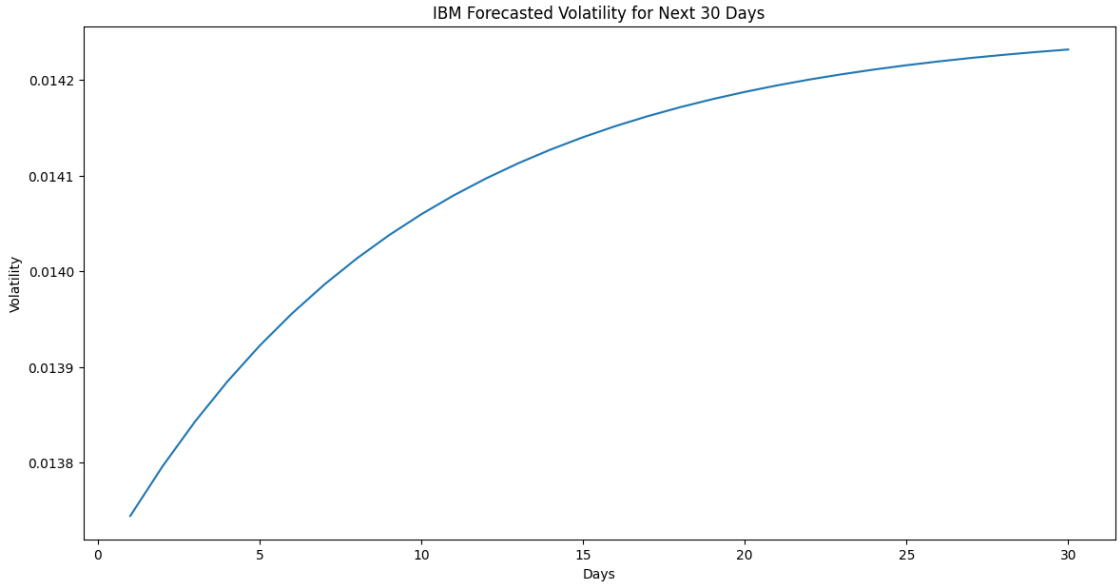
	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	2.1451e-04	2.328e-04	0.921	0.357	[-2.418e-04,6.708e-04]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	2.0321e-05	2.388e-12	8.510e+06	0.000	[2.032e-05,2.032e-05]
alpha[1]	0.1000	2.580e-02	3.876	1.061e-04	[4.944e-02, 0.151]
beta[1]	0.8000	2.365e-02	33.827	7.992e-251	[0.754, 0.846]

Covariance estimator: robust



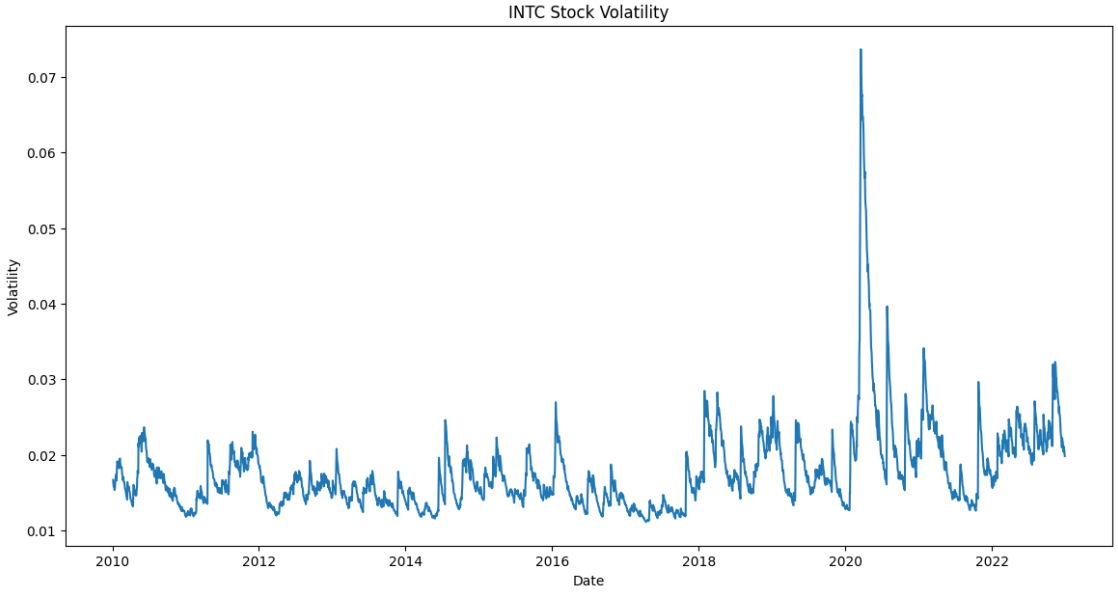


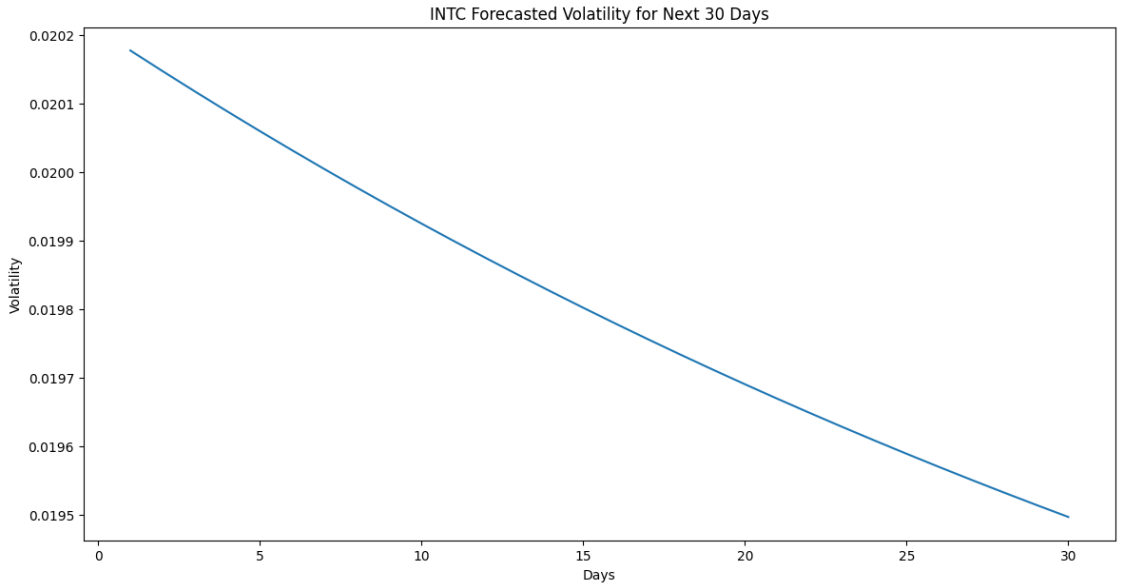
Summary for INTC:

Constant Mean - GARCH Model Results

Dep. Variable:	return	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	8659.92
Distribution:	Normal	AIC:	-17311.8
Method:	Maximum Likelihood	BIC:	-17287.5

```
                                     No. Observations:                 3270
Date:                Fri, Jun 28 2024   Df Residuals:                     3269
Time:                        16:01:00   Df Model:                            1
                            Mean Model
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
mu         2.4911e-04  1.822e-04      1.367      0.172 [-1.080e-04,6.062e-04]
                          Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega      6.9255e-06  1.429e-11  4.847e+05      0.000 [6.926e-06,6.926e-06]
alpha[1]       0.0500  2.049e-02      2.441  1.467e-02 [9.845e-03,9.015e-02]
beta[1]        0.9300  1.980e-02     46.974      0.000     [  0.891,  0.969]
============================================================================

Covariance estimator: robust
```





Summary for META:

Constant Mean - GARCH Model Results

Dep. Variable:	return	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	6255.35
Distribution:	Normal	AIC:	-12502.7
Method:	Maximum Likelihood	BIC:	-12479.1
		No. Observations:	2687
Date:	Fri, Jun 28 2024	Df Residuals:	2686
Time:	16:01:01	Df Model:	1

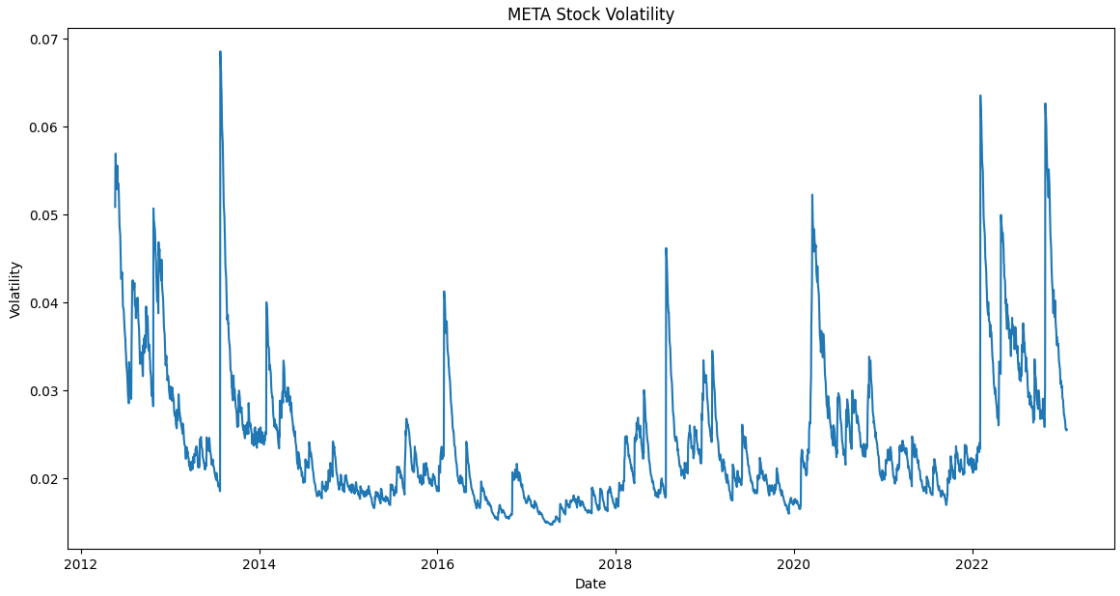
Mean Model

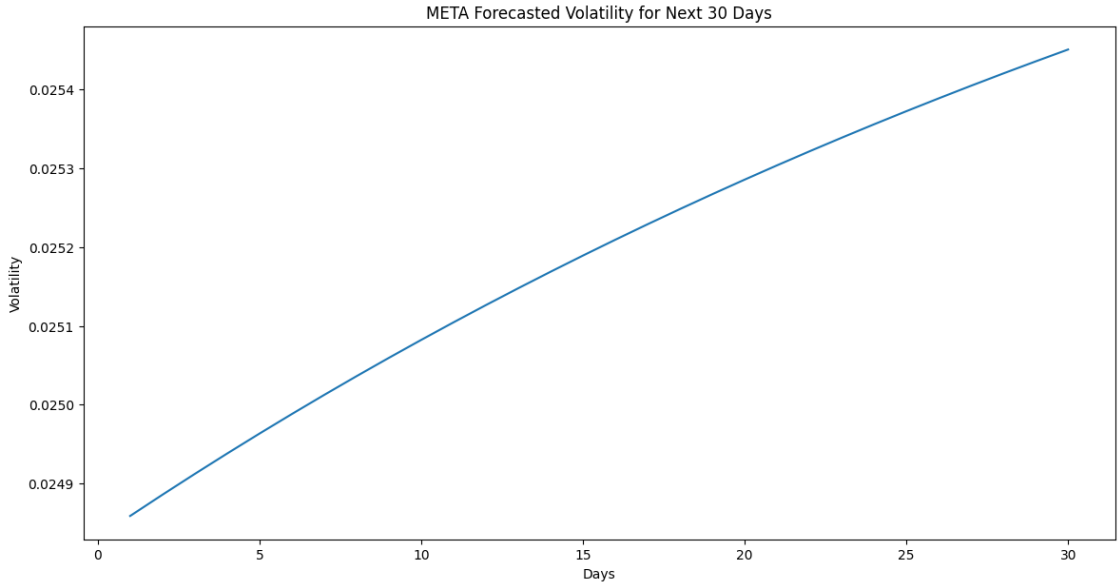
	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	9.2039e-04	4.786e-04	1.923	5.448e-02	[-1.771e-05,1.858e-03]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	1.3654e-05	5.109e-12	2.673e+06	0.000	[1.365e-05,1.365e-05]
alpha[1]	0.0501	3.606e-02	1.389	0.165	[-2.060e-02, 0.121]
beta[1]	0.9300	3.448e-02	26.971	3.251e-160	[0.862, 0.998]

Covariance estimator: robust



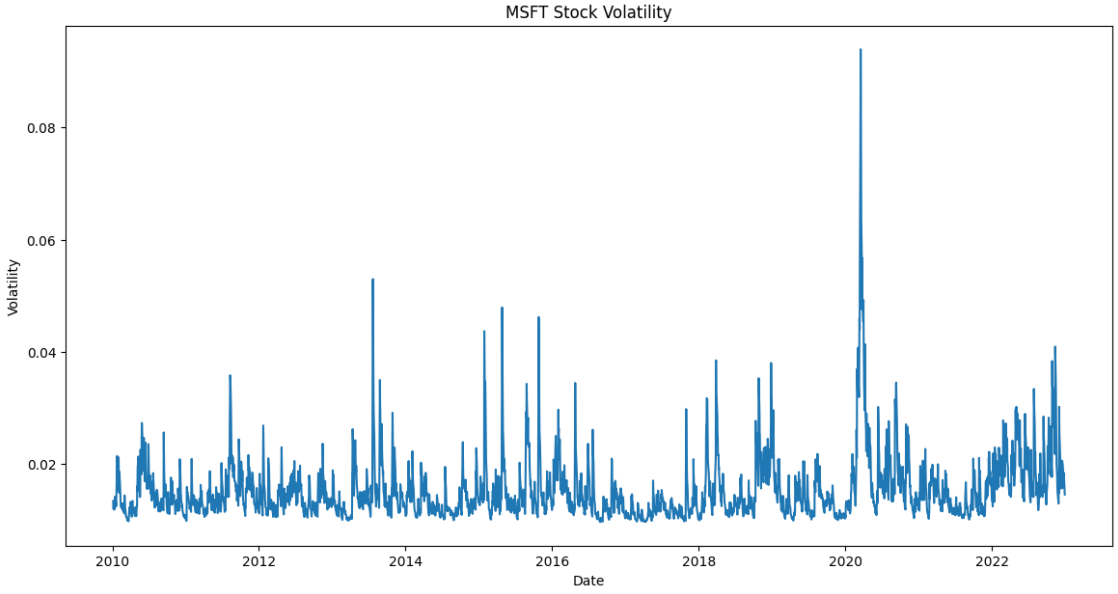


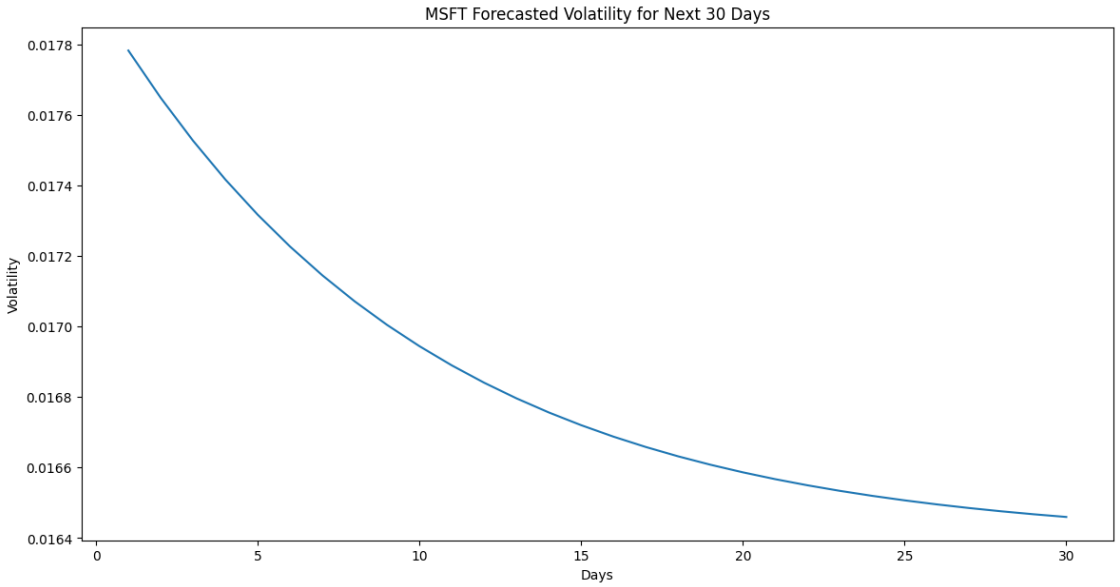
Summary for MSFT:

Constant Mean - GARCH Model Results

Dep. Variable:	return	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	9091.62
Distribution:	Normal	AIC:	-18175.2
Method:	Maximum Likelihood	BIC:	-18150.9

```
                                  No. Observations:                 3270
Date:                Fri, Jun 28 2024   Df Residuals:                     3269
Time:                        16:01:02   Df Model:                            1
                            Mean Model
================================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
--------------------------------------------------------------------------------
mu         1.1395e-03  2.526e-04      4.511  6.438e-06 [6.444e-04,1.634e-03]
                          Volatility Model
================================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
--------------------------------------------------------------------------------
omega      2.6869e-05  8.389e-07     32.030 4.208e-225 [2.522e-05,2.851e-05]
alpha[1]       0.2000  3.514e-02      5.691  1.262e-08     [  0.131,  0.269]
beta[1]        0.7000  2.948e-02     23.748 1.146e-124     [  0.642,  0.758]
================================================================================

Covariance estimator: robust
```





Summary for NFLX:

Constant Mean - GARCH Model Results

Dep. Variable:	return	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	6622.67
Distribution:	Normal	AIC:	-13237.3
Method:	Maximum Likelihood	BIC:	-13213.0
		No. Observations:	3270
Date:	Fri, Jun 28 2024	Df Residuals:	3269
Time:	16:01:03	Df Model:	1

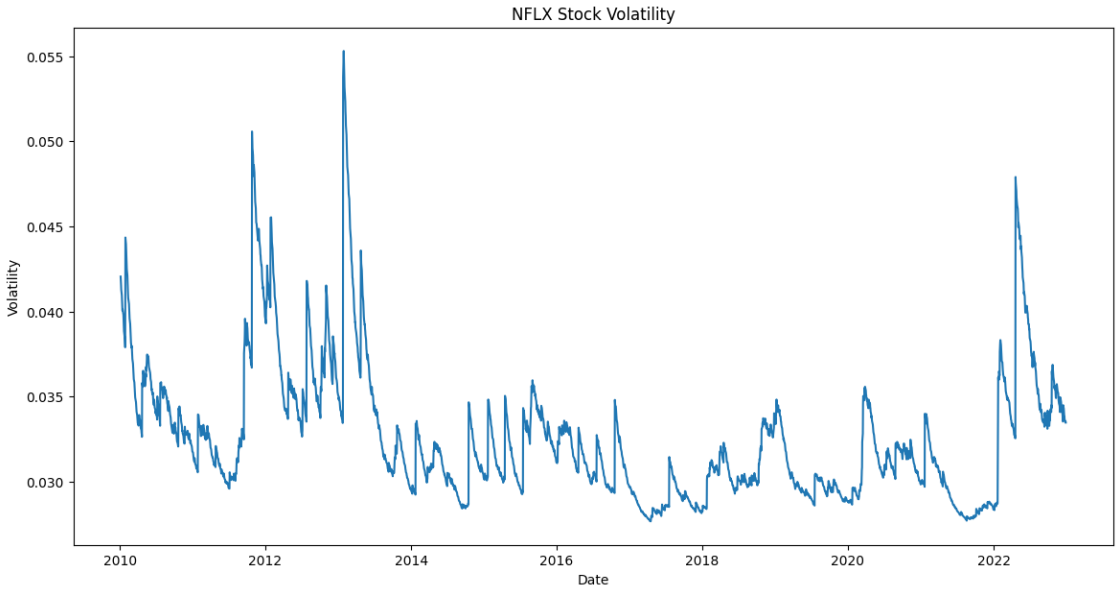
Mean Model

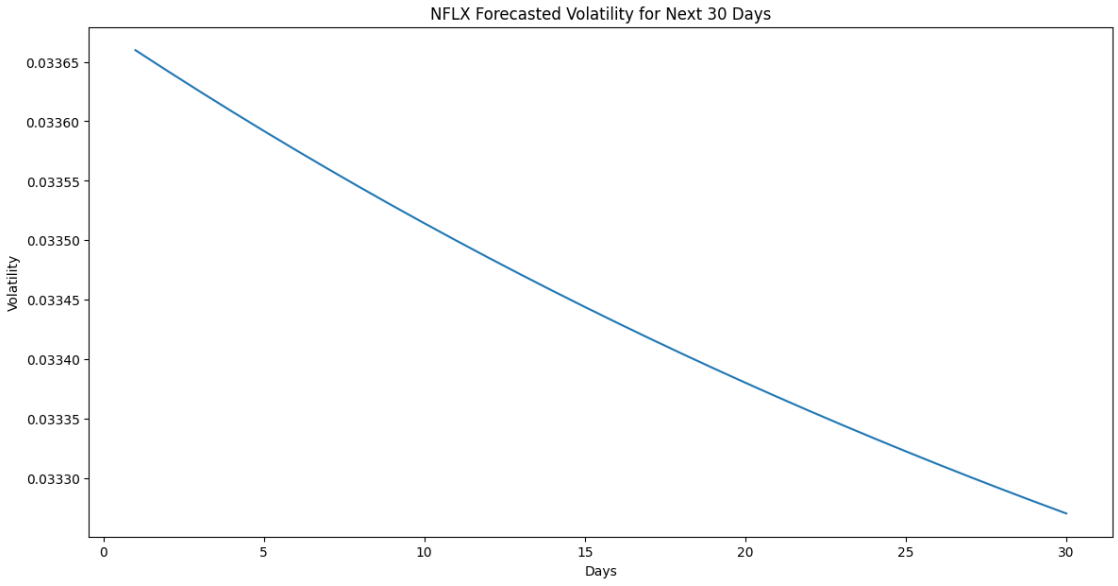
	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	1.7870e-03	4.997e-04	3.576	3.484e-04	[8.076e-04,2.766e-03]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	2.1484e-05	1.711e-11	1.256e+06	0.000	[2.148e-05,2.148e-05]
alpha[1]	1.0000e-02	2.879e-03	3.473	5.146e-04	[4.357e-03,1.564e-02]
beta[1]	0.9700	3.243e-03	299.100	0.000	[0.964, 0.976]

Covariance estimator: robust



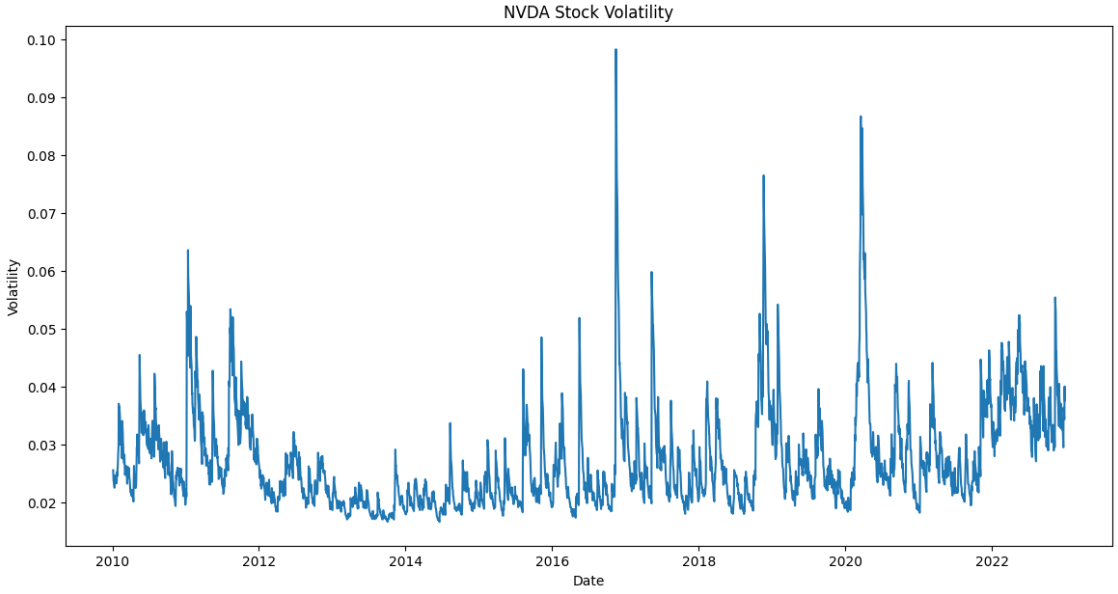


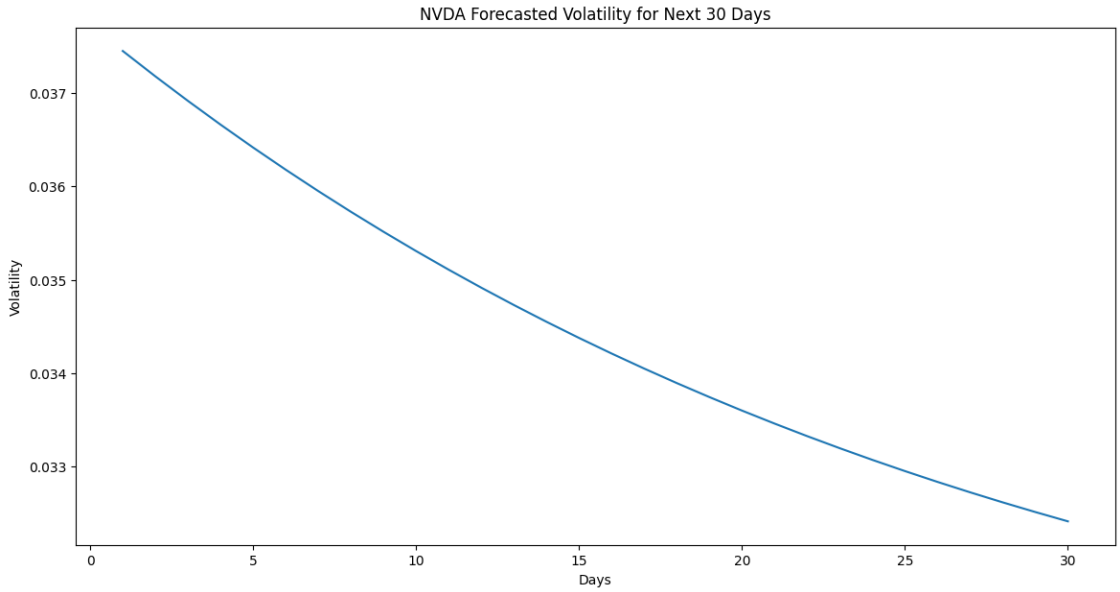
Summary for NVDA:

Constant Mean - GARCH Model Results

Dep. Variable:	return	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	7285.69
Distribution:	Normal	AIC:	-14563.4
Method:	Maximum Likelihood	BIC:	-14539.0

```
                                    No. Observations:                 3270
Date:                Fri, Jun 28 2024   Df Residuals:                     3269
Time:                        16:01:03   Df Model:                            1
                            Mean Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
mu         2.0079e-03  4.867e-04      4.126  3.699e-05 [1.054e-03,2.962e-03]
                          Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega      3.5771e-05  4.165e-06      8.587  8.888e-18 [2.761e-05,4.394e-05]
alpha[1]       0.1033  3.032e-02      3.406  6.599e-04   [4.384e-02,  0.163]
beta[1]        0.8568  2.441e-02     35.102 6.317e-270     [  0.809,  0.905]
============================================================================

Covariance estimator: robust
```





Summary for ORCL:

Constant Mean - GARCH Model Results

Dep. Variable:	return	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	9086.00
Distribution:	Normal	AIC:	-18164.0
Method:	Maximum Likelihood	BIC:	-18139.6
		No. Observations:	3270
Date:	Fri, Jun 28 2024	Df Residuals:	3269
Time:	16:01:04	Df Model:	1

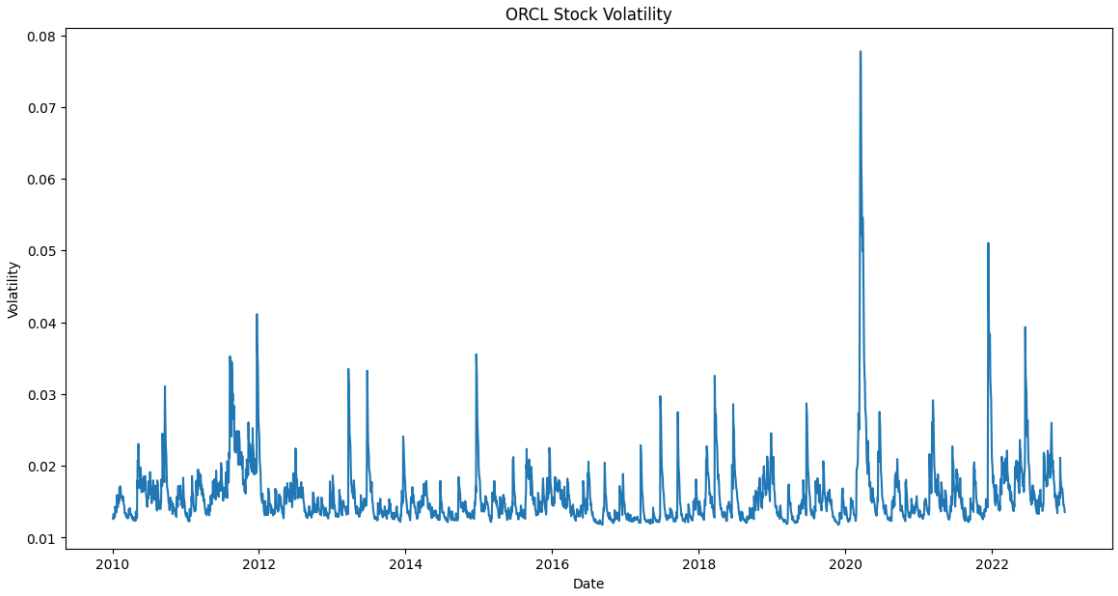
Mean Model

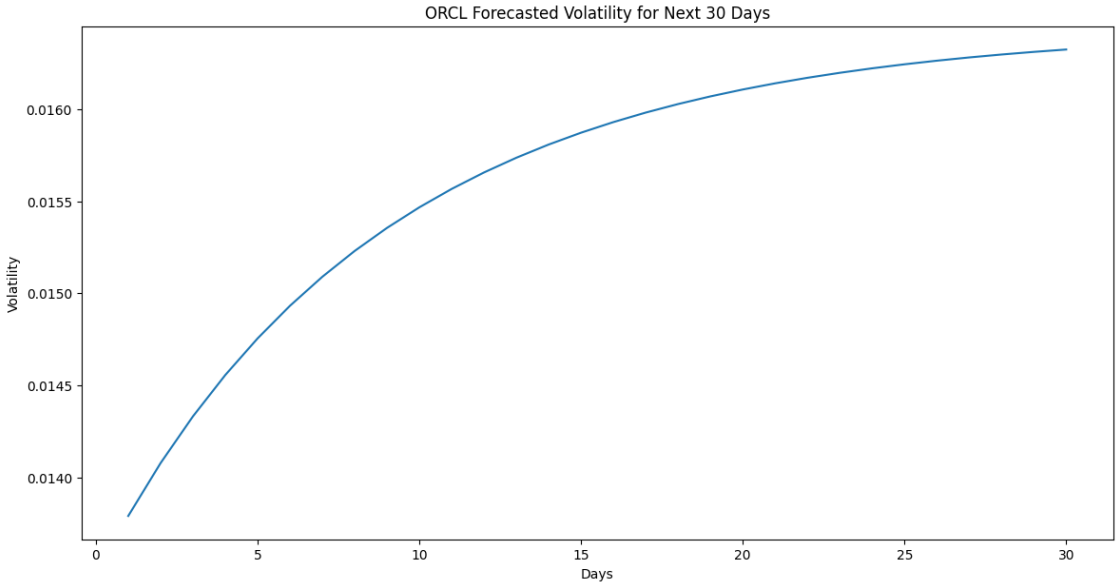
	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	6.3706e-04	2.637e-04	2.416	1.571e-02	[1.202e-04,1.154e-03]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	2.7025e-05	5.731e-07	47.154	0.000	[2.590e-05,2.815e-05]
alpha[1]	0.1000	2.182e-02	4.583	4.587e-06	[5.723e-02, 0.143]
beta[1]	0.8000	1.880e-02	42.562	0.000	[0.763, 0.837]

Covariance estimator: robust





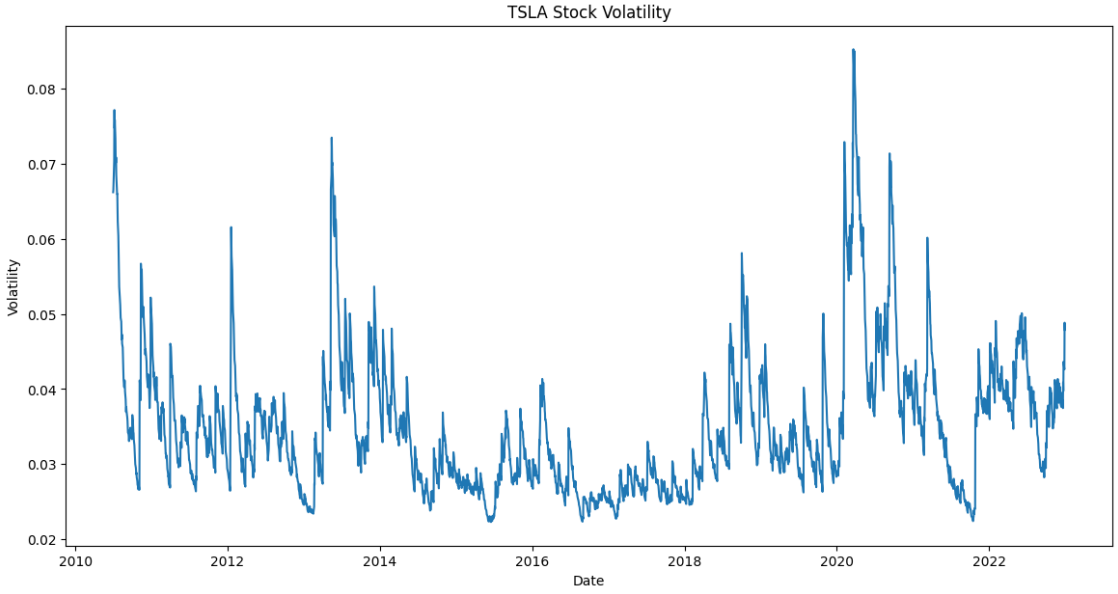
Summary for TSLA:

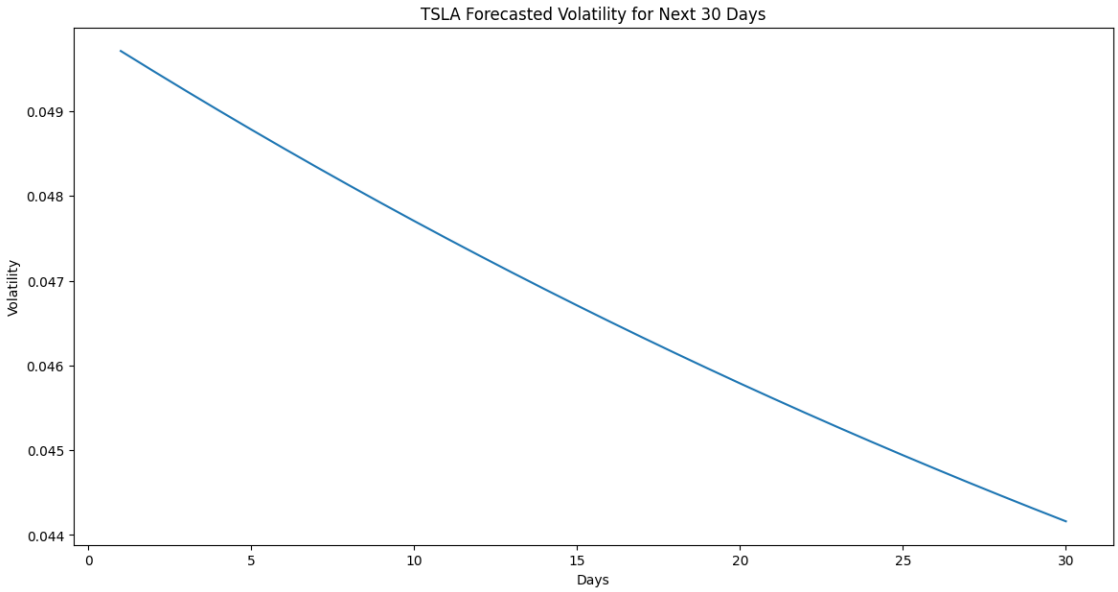
Constant Mean - GARCH Model Results

Dep. Variable:	return	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	6177.22
Distribution:	Normal	AIC:	-12346.4
Method:	Maximum Likelihood	BIC:	-12322.2

```
                                  No. Observations:                 3147
Date:                Fri, Jun 28 2024   Df Residuals:                     3146
Time:                        16:01:05   Df Model:                            1
                            Mean Model
================================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
--------------------------------------------------------------------------------
mu         1.7528e-03  5.834e-04      3.005  2.660e-03 [6.094e-04,2.896e-03]
                          Volatility Model
================================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
--------------------------------------------------------------------------------
omega      2.5918e-05  2.199e-06     11.784  4.713e-32 [2.161e-05,3.023e-05]
alpha[1]       0.0500  1.317e-02      3.797  1.467e-04 [2.419e-02,7.581e-02]
beta[1]        0.9300  1.311e-02     70.942      0.000     [  0.904,  0.956]
================================================================================
```

Covariance estimator: robust



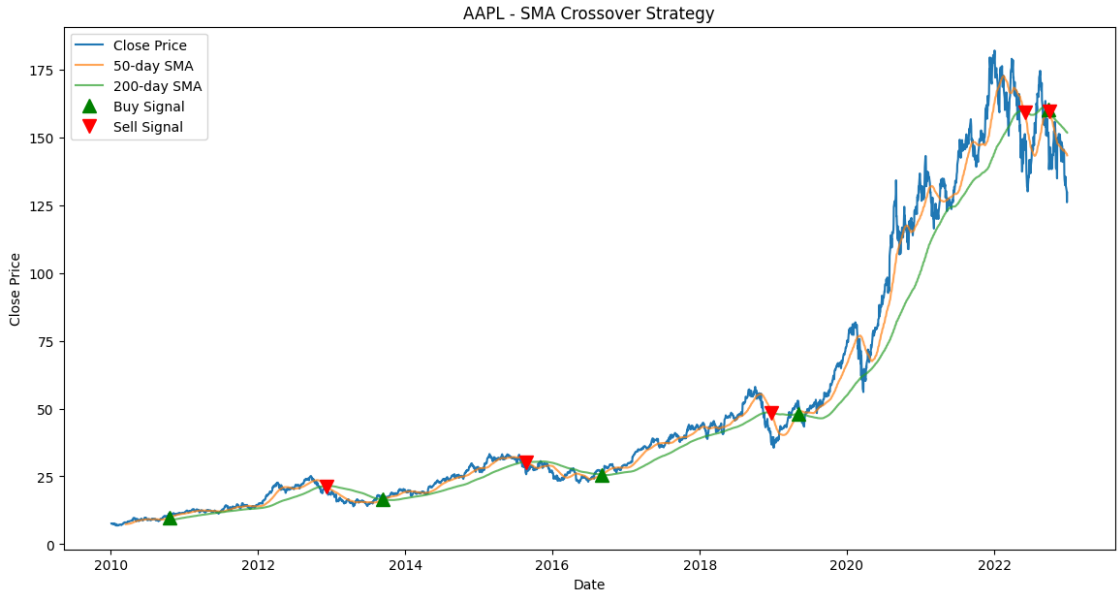


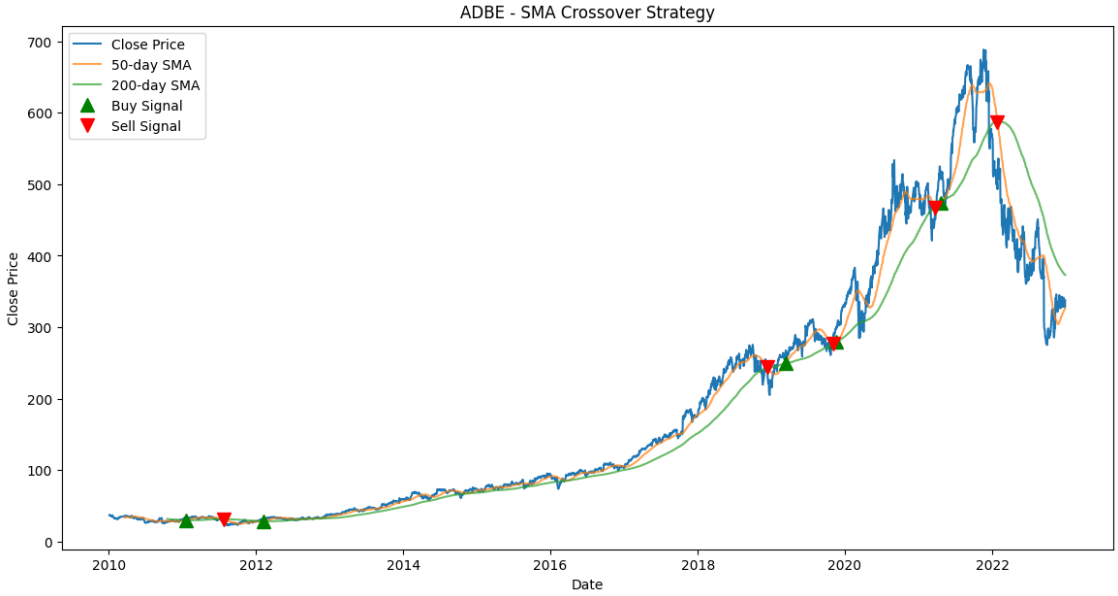
1.9 Otomatik Alım Satım Algoritmaları (Tüm Şirketler)

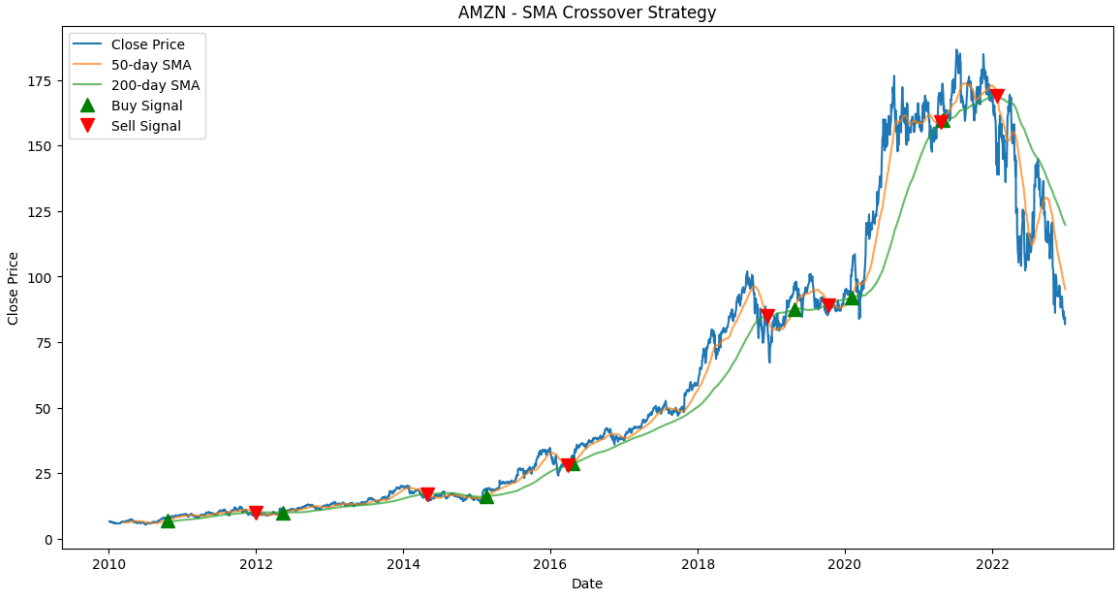
Basit bir hareketli ortalama crossover stratejisi kullanarak alım-satım sinyalleri oluşturacağız.

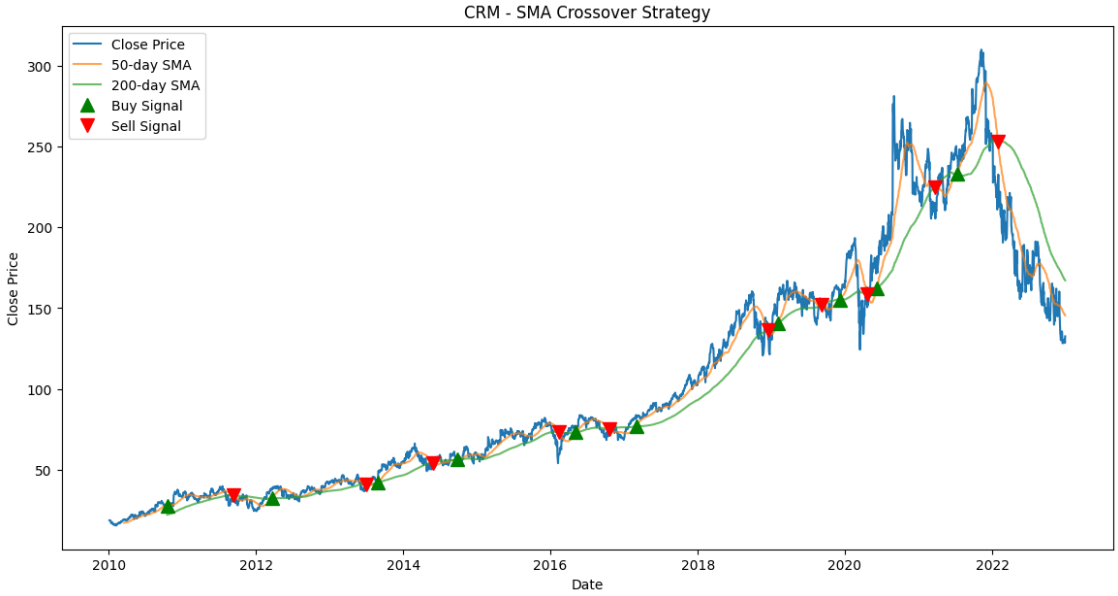
[63]:
```
# Her bir hisse senedi için alım-satım sinyalleri oluşturmak
for symbol in unique_symbols:
    stock_data = big_tech_stock_prices[big_tech_stock_prices['stock_symbol'] ==␣
 ↪symbol]
    stock_data.set_index('date', inplace=True)

    # Hareketli ortalamaları hesaplayalım
    stock_data['SMA50'] = stock_data['close'].rolling(window=50).mean()
    stock_data['SMA200'] = stock_data['close'].rolling(window=200).mean()

    # Alım-satım sinyalleri
    stock_data['Signal'] = 0.0
    stock_data['Signal'][50:] = np.where(stock_data['SMA50'][50:] >␣
 ↪stock_data['SMA200'][50:], 1.0, 0.0)
    stock_data['Position'] = stock_data['Signal'].diff()

    # Sinyalleri görselleştirelim
    plt.figure(figsize=(14, 7))
    plt.plot(stock_data['close'], label='Close Price')
    plt.plot(stock_data['SMA50'], label='50-day SMA', alpha=0.7)
    plt.plot(stock_data['SMA200'], label='200-day SMA', alpha=0.7)
```

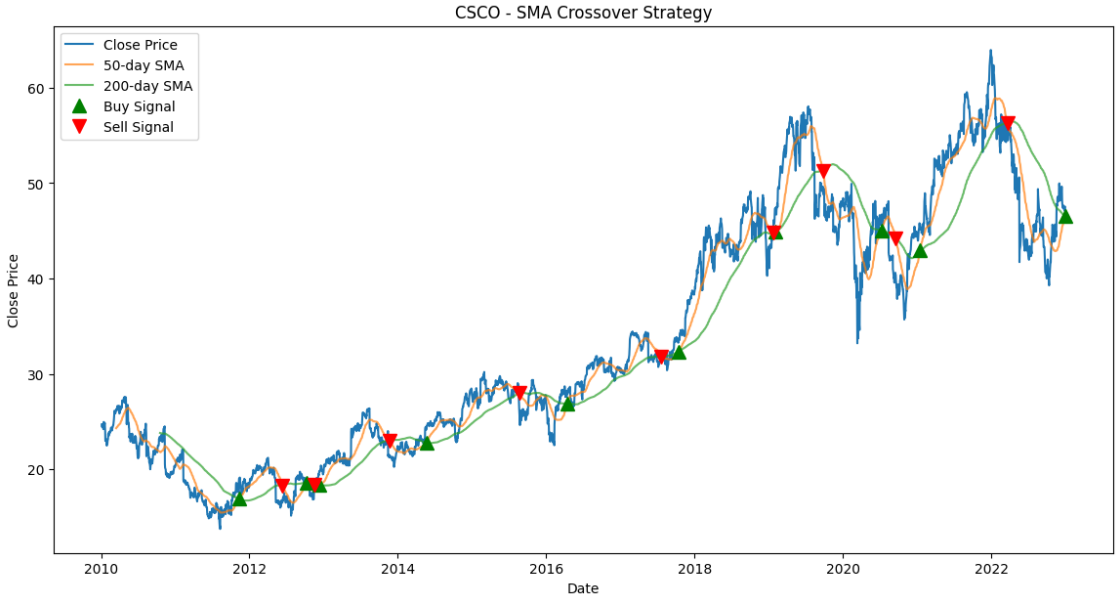
```
    plt.plot(stock_data[stock_data['Position'] == 1].index,␣
↪stock_data['SMA50'][stock_data['Position'] == 1], '^', markersize=10,␣
↪color='g', lw=0, label='Buy Signal')
    plt.plot(stock_data[stock_data['Position'] == -1].index,␣
↪stock_data['SMA50'][stock_data['Position'] == -1], 'v', markersize=10,␣
↪color='r', lw=0, label='Sell Signal')
    plt.title(f'{symbol} - SMA Crossover Strategy')
    plt.xlabel('Date')
    plt.ylabel('Close Price')
    plt.legend()
    plt.show()
```

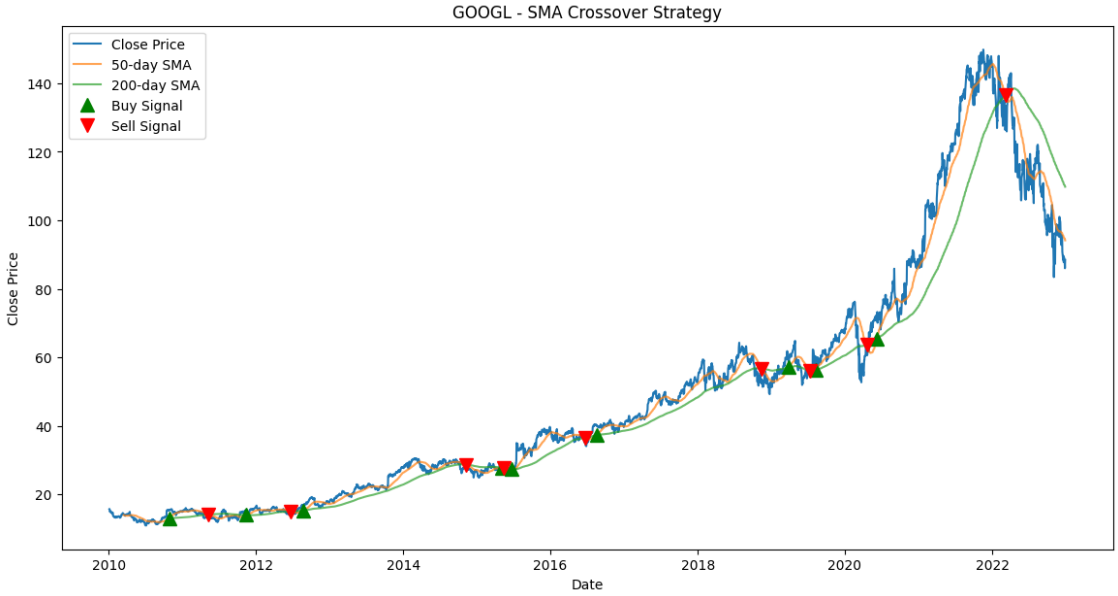


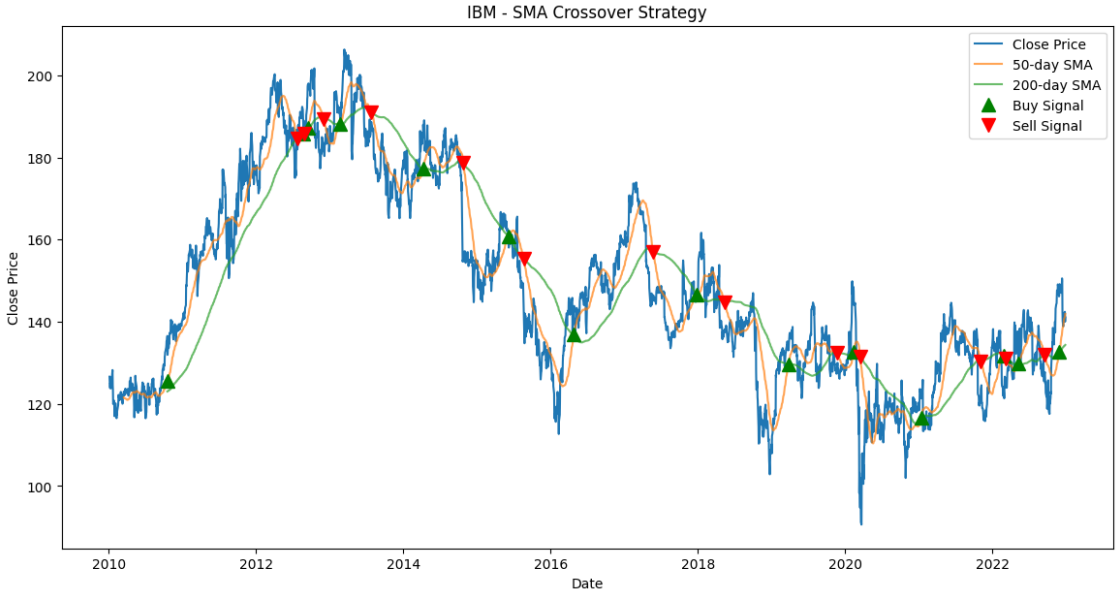

ADBE - SMA Crossover Strategy
Close Price
50-day SMA
200-day SMA
Buy Signal
Sell Signal
Close Price
Date
0
100
200
300
400
500
600
700
2010
2012
2014
2016
2018
2020
2022

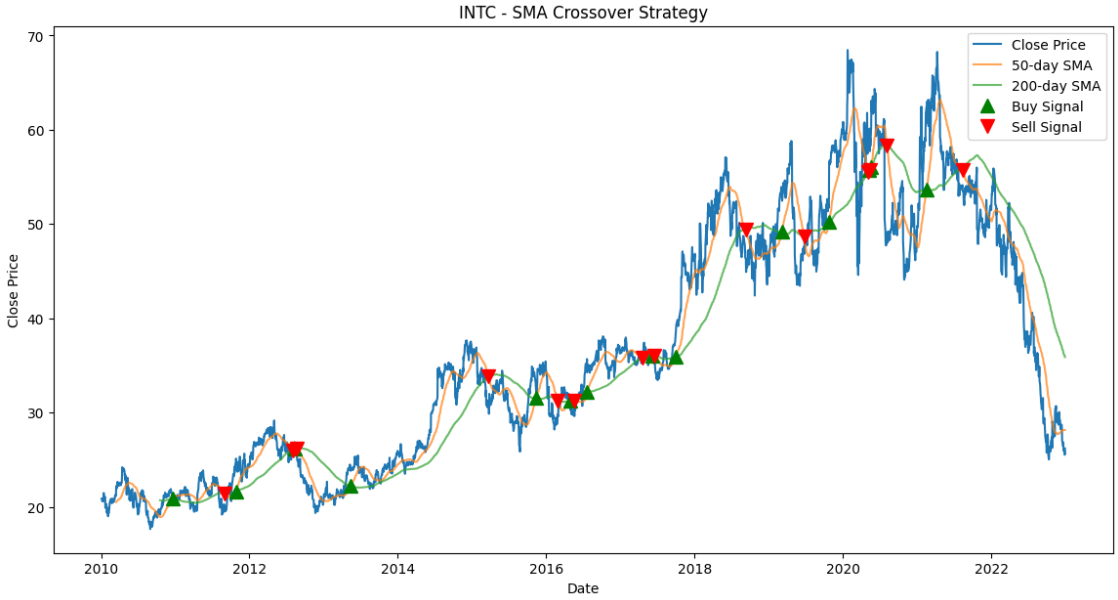

AMZN - SMA Crossover Strategy
Close Price
50-day SMA
200-day SMA
Buy Signal
Sell Signal
Close Price
Date
0
25
50
75
100
125
150
175
2010
2012
2014
2016
2018
2020
2022

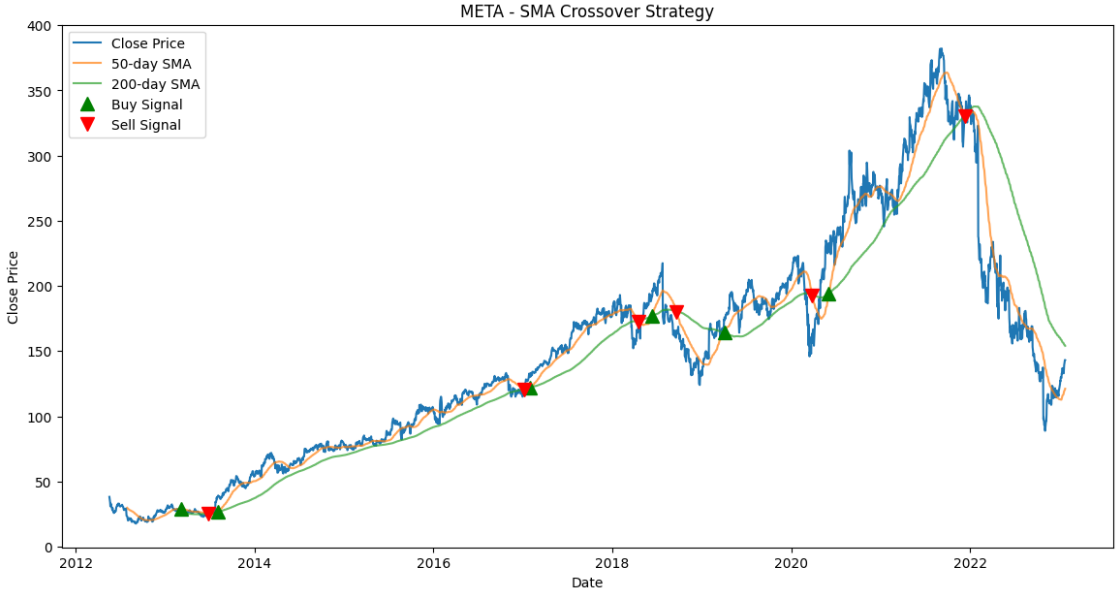

CRM - SMA Crossover Strategy
Close Price
50-day SMA
200-day SMA
Buy Signal
Sell Signal
Close Price
Date
300
250
200
150
100
50
2010
2012
2014
2016
2018
2020
2022

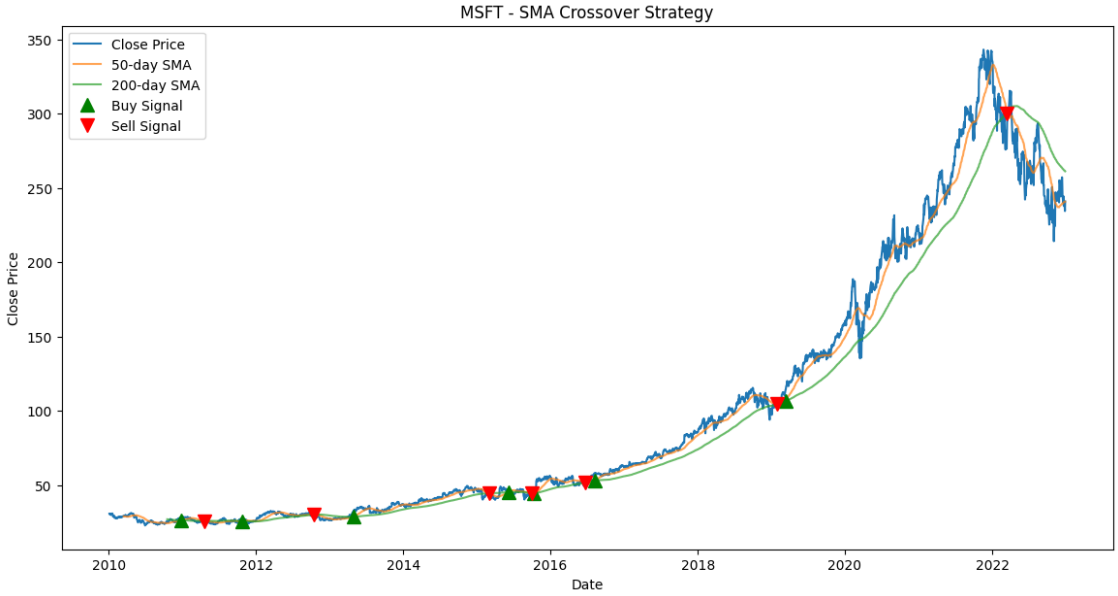

CSCO - SMA Crossover Strategy
Close Price
50-day SMA
200-day SMA
Buy Signal
Sell Signal
Close Price
Date
60
50
40
30
20
2010
2012
2014
2016
2018
2020
2022


GOOGL - SMA Crossover Strategy
Close Price
50-day SMA
200-day SMA
Buy Signal
Sell Signal
Close Price
Date
20
40
60
80
100
120
140
2010
2012
2014
2016
2018
2020
2022

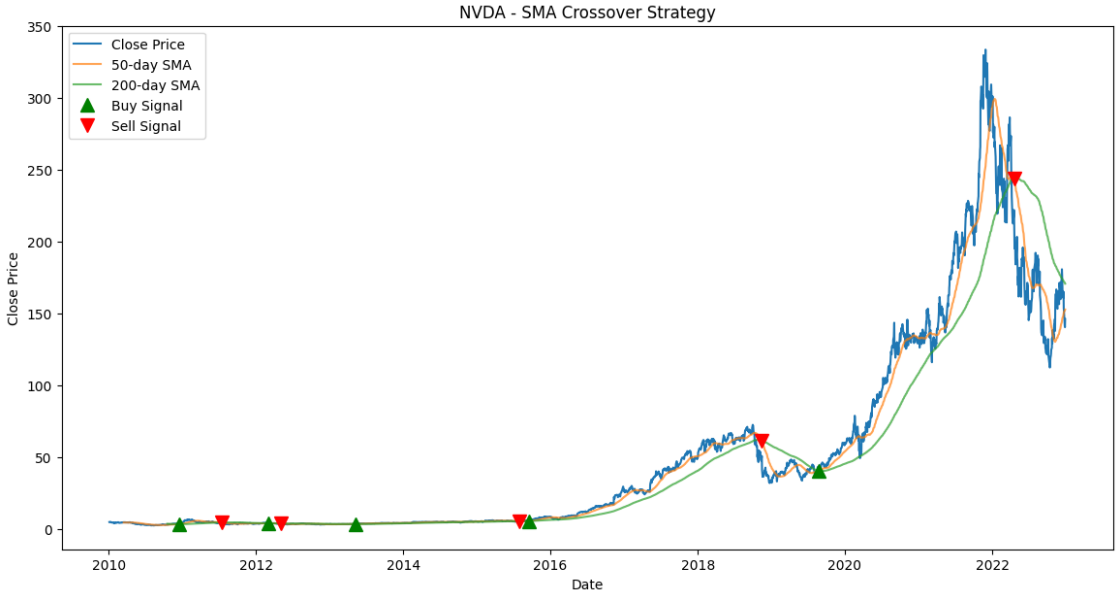

IBM - SMA Crossover Strategy
Close Price
50-day SMA
200-day SMA
Buy Signal
Sell Signal
Close Price
Date
100
120
140
160
180
200
2010
2012
2014
2016
2018
2020
2022

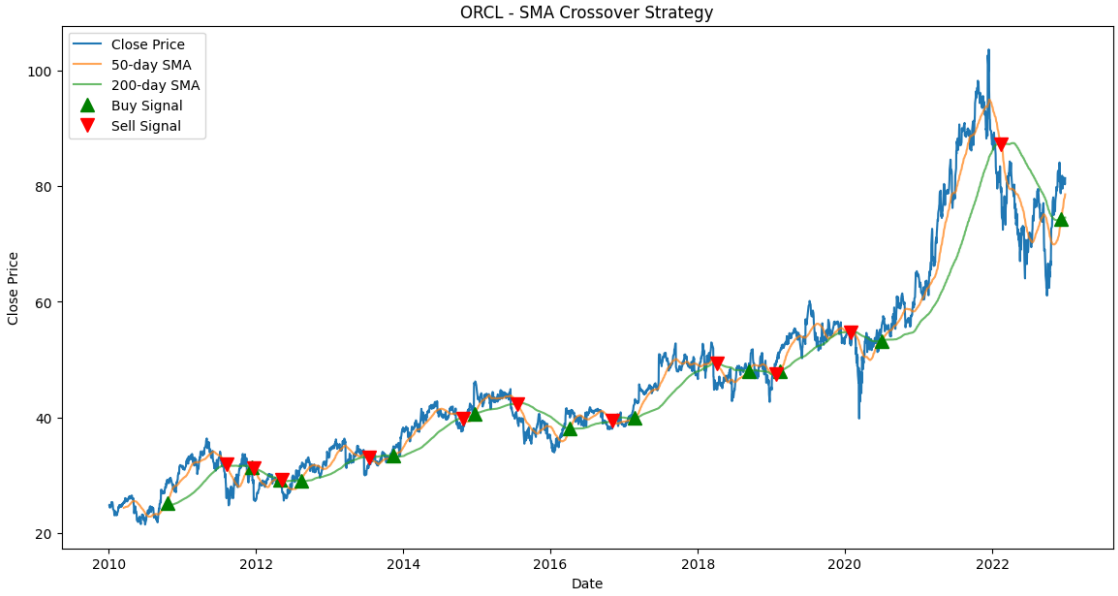

INTC - SMA Crossover Strategy
Close Price
50-day SMA
200-day SMA
Buy Signal
Sell Signal
Close Price
Date
70
60
50
40
30
20
2010
2012
2014
2016
2018
2020
2022

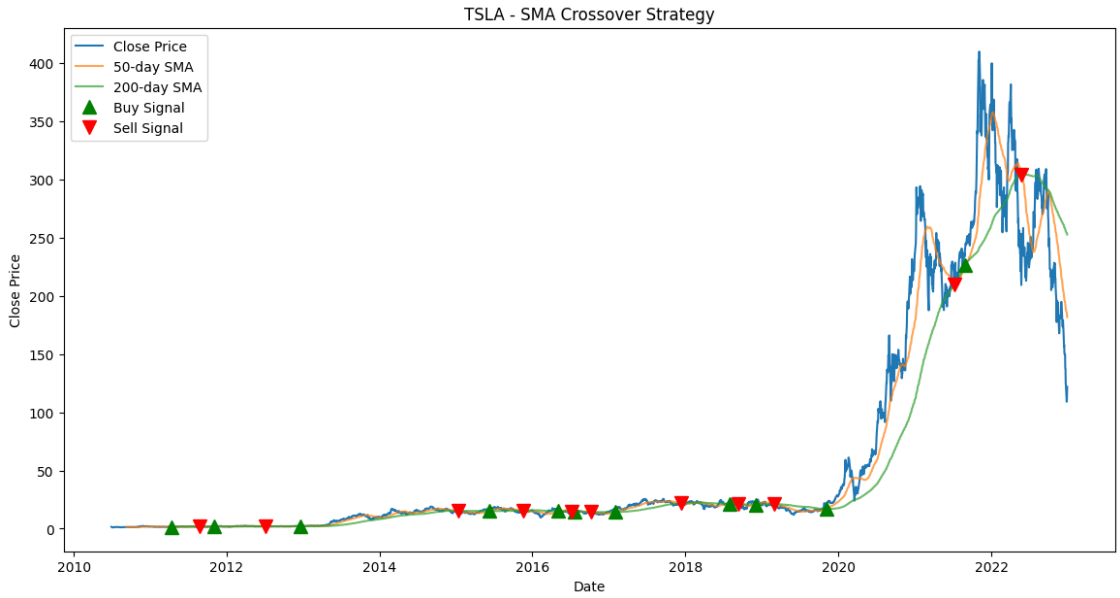

META - SMA Crossover Strategy
Close Price
50-day SMA
200-day SMA
Buy Signal
Sell Signal
Close Price
Date
400
350
300
250
200
150
100
50
0
2012
2014
2016
2018
2020
2022


MSFT - SMA Crossover Strategy
Close Price
50-day SMA
200-day SMA
Buy Signal
Sell Signal
Close Price
Date
350
300
250
200
150
100
50
2010
2012
2014
2016
2018
2020
2022


NFLX - SMA Crossover Strategy
Close Price
50-day SMA
200-day SMA
Buy Signal
Sell Signal
Close Price
Date
700
600
500
400
300
200
100
0
2010
2012
2014
2016
2018
2020
2022


NVDA - SMA Crossover Strategy
Close Price
50-day SMA
200-day SMA
Buy Signal
Sell Signal
Close Price
Date
350
300
250
200
150
100
50
0
2010
2012
2014
2016
2018
2020
2022


ORCL - SMA Crossover Strategy
Close Price
50-day SMA
200-day SMA
Buy Signal
Sell Signal
Close Price
Date
100
80
60
40
20
2010
2012
2014
2016
2018
2020
2022



1.10 Model Açıklanabilirliği (SHAP)

SHAP (SHapley Additive exPlanations) değerlerini kullanarak model kararlarını açıklayabiliriz. Aşağıda örnek bir regresyon modeli kullanarak SHAP değerlerini hesaplayacağız.

[64]:
```
# Model için verileri hazırlayalım
X = merged_data[['unemployment_rate', 'cpi', 'inflation_rate',␣
 ↪'mortgage_interest_rate', 'corporate_bond_yield']]
y = merged_data['close']
```

[65]:
```
# Veriyi eğitim ve test setlerine bölelim
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[66]:
```
# RandomForest modelini eğitelim
model = RandomForestRegressor()
model.fit(X_train, y_train)
```

[66]:
```
RandomForestRegressor()
```

[67]:
```
import shap

# SHAP değerlerini hesaplayalım
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)
```

[68]:
```
# SHAP özet grafiğini görselleştirelim
shap.summary_plot(shap_values, X_test)
```