Financial computer simulation project

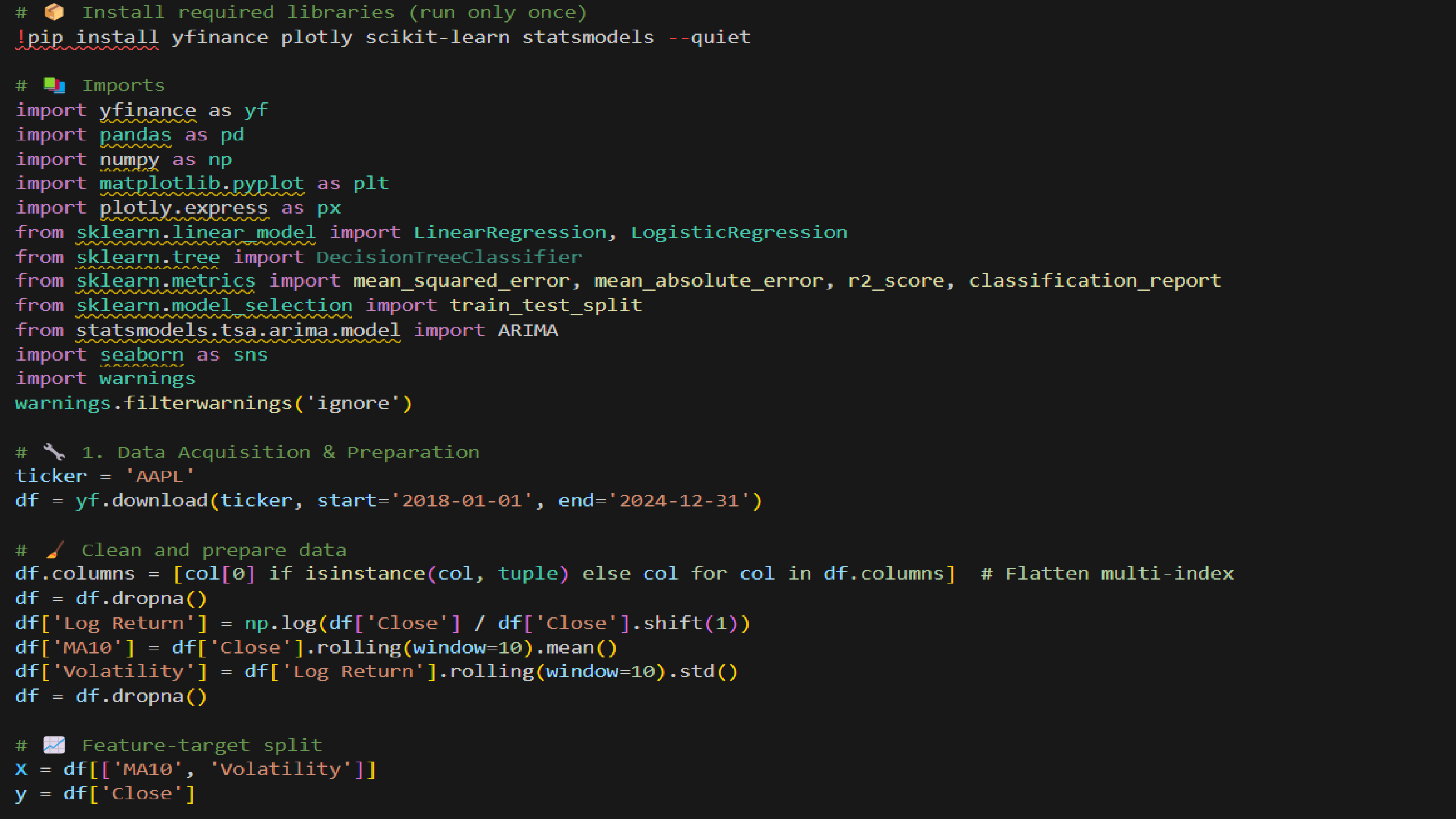
WARDA AKHLAS

Seat# :B21120206055

BS-Financial Mathematics (4TH Year)

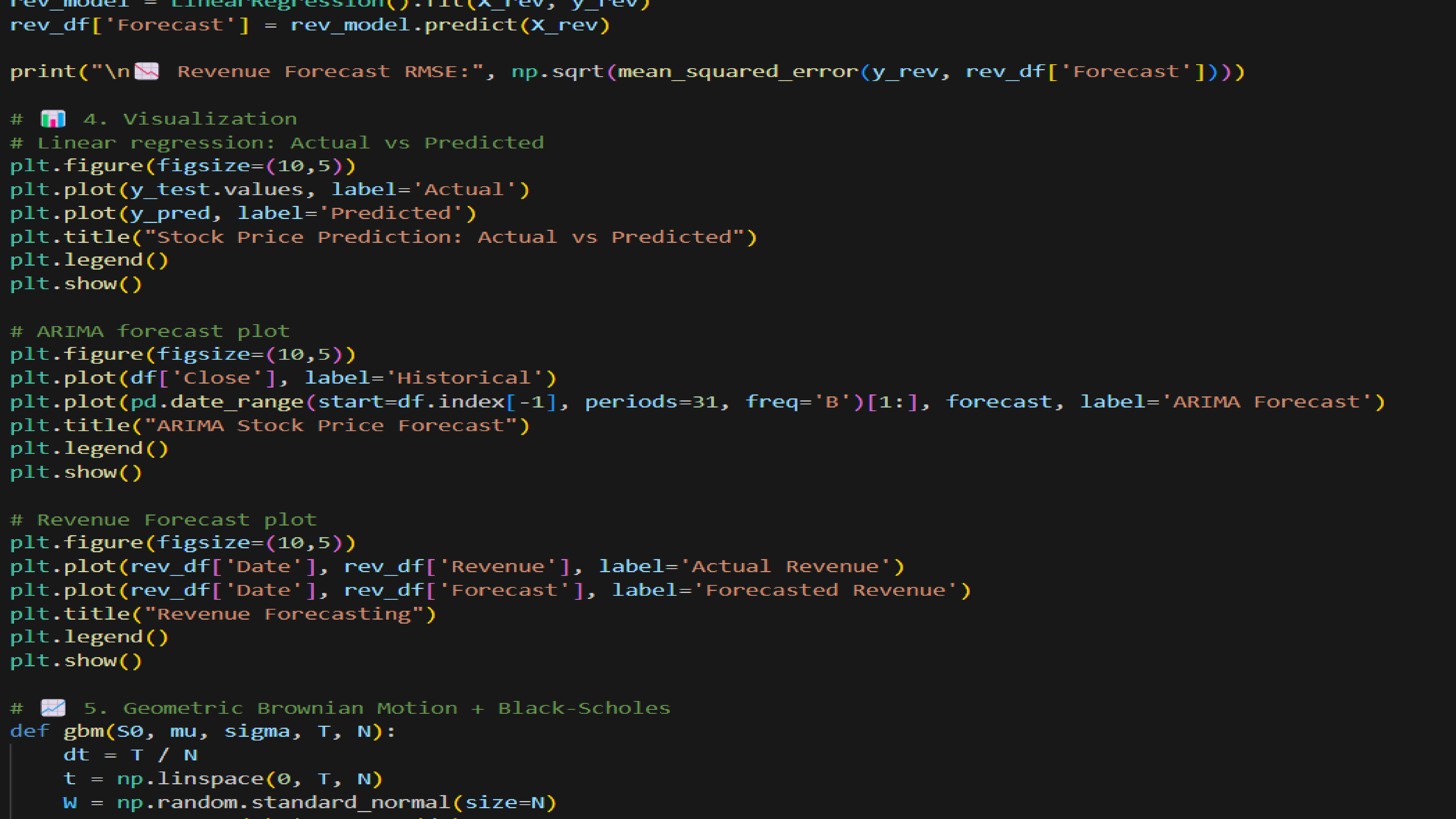
📈 Financial Forecasting Pipeline —Python Project Overview

- Data Acquisition & Preparation
- Modeling Modules
- Evaluation Metrics
- Visualize Results
- Stochastic Process & Derivatives Integration
- Exploratory Data Analysis (EDA)
- Deliverables

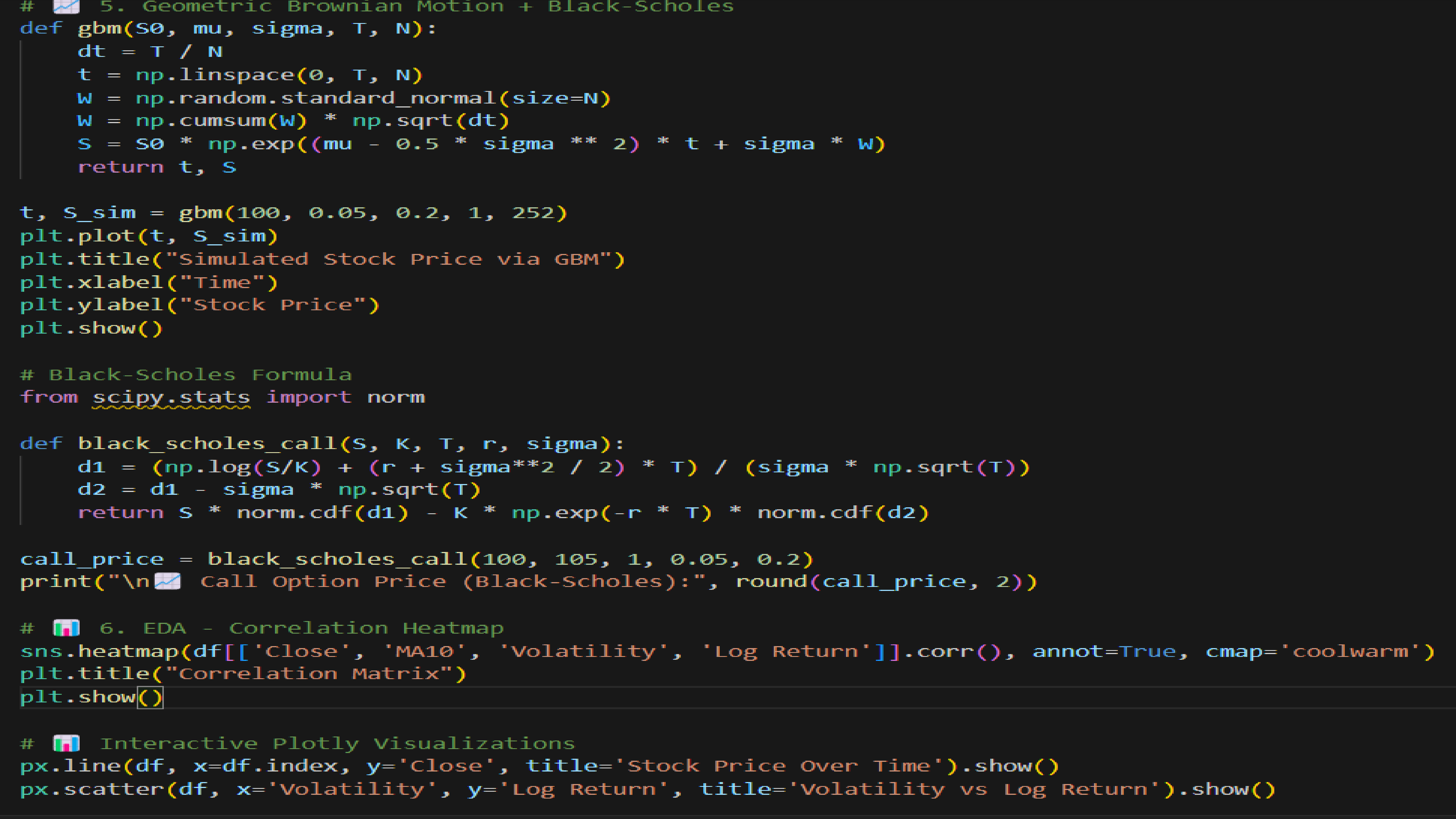


```python
# 📦 Install required libraries (run only once)
!pip install yfinance plotly scikit-learn statsmodels --quiet

# 📚 Imports
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score, classification_report
from sklearn.model_selection import train_test_split
from statsmodels.tsa.arima.model import ARIMA
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

# 🔧 1. Data Acquisition & Preparation
ticker = 'AAPL'
df = yf.download(ticker, start='2018-01-01', end='2024-12-31')

# 🧹 Clean and prepare data
df.columns = [col[0] if isinstance(col, tuple) else col for col in df.columns]  # Flatten multi-index
df = df.dropna()
df['Log Return'] = np.log(df['Close'] / df['Close'].shift(1))
df['MA10'] = df['Close'].rolling(window=10).mean()
df['Volatility'] = df['Log Return'].rolling(window=10).std()
df = df.dropna()

# 📈 Feature-target split
X = df[['MA10', 'Volatility']]
y = df['Close']
```

```python
# 📉 2a. Stock Price Prediction - Linear Regression
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
lr_model = LinearRegression().fit(X_train, y_train)
y_pred = lr_model.predict(X_test)

# 📊 Evaluation Metrics
print("📊 Linear Regression Metrics:")
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("R2 Score:", r2_score(y_test, y_pred))

# 📉 2a. Stock Price Forecasting - ARIMA
model_arima = ARIMA(df['Close'], order=(5,1,0))
arima_result = model_arima.fit()
forecast = arima_result.forecast(steps=30)

# 🏦 2b. Credit Risk Modeling (Synthetic)
from sklearn.datasets import make_classification
X_credit, y_credit = make_classification(n_samples=1000, n_features=5, random_state=42)
X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(X_credit, y_credit, test_size=0.2)

log_model = LogisticRegression().fit(X_train_c, y_train_c)
y_pred_c = log_model.predict(X_test_c)
print("\n📋 Credit Risk Classification Report:\n", classification_report(y_test_c, y_pred_c))

# 💰 2c. Revenue Forecasting (Synthetic)
dates = pd.date_range(start='2020-01-01', periods=48, freq='M')
revenues = np.random.normal(loc=50000, scale=5000, size=len(dates)).cumsum()
rev_df = pd.DataFrame({'Date': dates, 'Revenue': revenues})
rev_df['Lag1'] = rev_df['Revenue'].shift(1)
rev_df = rev_df.dropna()

X_rev = rev_df[['Lag1']]
y_rev = rev_df['Revenue']
rev_model = LinearRegression().fit(X_rev, y_rev)
```

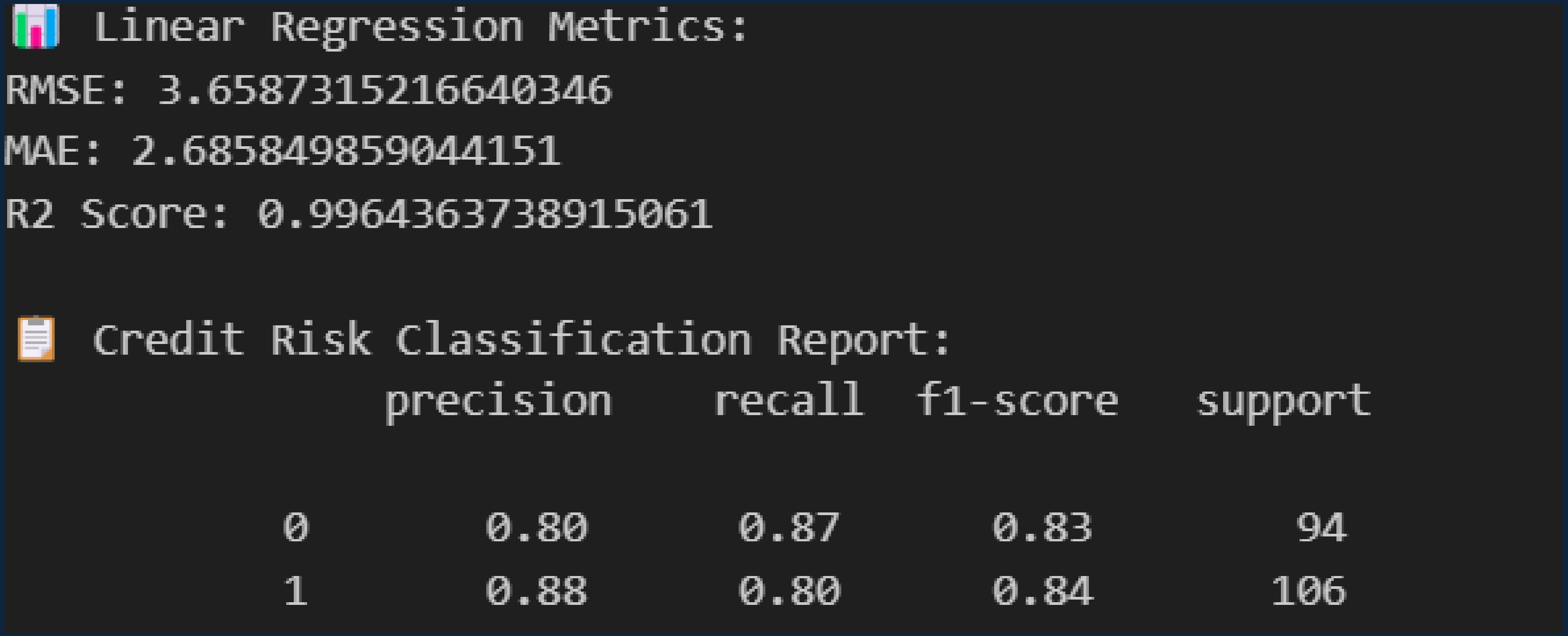


```python
rev_model = LinearRegression().fit(X_rev, y_rev)
rev_df['Forecast'] = rev_model.predict(X_rev)

print("\n📉 Revenue Forecast RMSE:", np.sqrt(mean_squared_error(y_rev, rev_df['Forecast'])))

# 📊 4. Visualization
# Linear regression: Actual vs Predicted
plt.figure(figsize=(10,5))
plt.plot(y_test.values, label='Actual')
plt.plot(y_pred, label='Predicted')
plt.title("Stock Price Prediction: Actual vs Predicted")
plt.legend()
plt.show()

# ARIMA forecast plot
plt.figure(figsize=(10,5))
plt.plot(df['Close'], label='Historical')
plt.plot(pd.date_range(start=df.index[-1], periods=31, freq='B')[1:], forecast, label='ARIMA Forecast')
plt.title("ARIMA Stock Price Forecast")
plt.legend()
plt.show()

# Revenue Forecast plot
plt.figure(figsize=(10,5))
plt.plot(rev_df['Date'], rev_df['Revenue'], label='Actual Revenue')
plt.plot(rev_df['Date'], rev_df['Forecast'], label='Forecasted Revenue')
plt.title("Revenue Forecasting")
plt.legend()
plt.show()

# 📈 5. Geometric Brownian Motion + Black-Scholes
def gbm(S0, mu, sigma, T, N):
    dt = T / N
    t = np.linspace(0, T, N)
    W = np.random.standard_normal(size=N)
```



```python
# 📈 5. Geometric Brownian Motion + Black-Scholes
def gbm(S0, mu, sigma, T, N):
    dt = T / N
    t = np.linspace(0, T, N)
    W = np.random.standard_normal(size=N)
    W = np.cumsum(W) * np.sqrt(dt)
    S = S0 * np.exp((mu - 0.5 * sigma ** 2) * t + sigma * W)
    return t, S

t, S_sim = gbm(100, 0.05, 0.2, 1, 252)
plt.plot(t, S_sim)
plt.title("Simulated Stock Price via GBM")
plt.xlabel("Time")
plt.ylabel("Stock Price")
plt.show()

# Black-Scholes Formula
from scipy.stats import norm

def black_scholes_call(S, K, T, r, sigma):
    d1 = (np.log(S/K) + (r + sigma**2 / 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    return S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)

call_price = black_scholes_call(100, 105, 1, 0.05, 0.2)
print("\n📈 Call Option Price (Black-Scholes):", round(call_price, 2))

# 📊 6. EDA - Correlation Heatmap
sns.heatmap(df[['Close', 'MA10', 'Volatility', 'Log Return']].corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()

# 📊 Interactive Plotly Visualizations
px.line(df, x=df.index, y='Close', title='Stock Price Over Time').show()
px.scatter(df, x='Volatility', y='Log Return', title='Volatility vs Log Return').show()
```

OUTPUT



```
📊 Linear Regression Metrics:
RMSE: 3.658731521664034б
MAE: 2.685849859044151
R2 Score: 0.9964363738915061

📋 Credit Risk Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.87      0.83        94
           1       0.88      0.80      0.84       106
```

The blue line represents the actual stock prices.

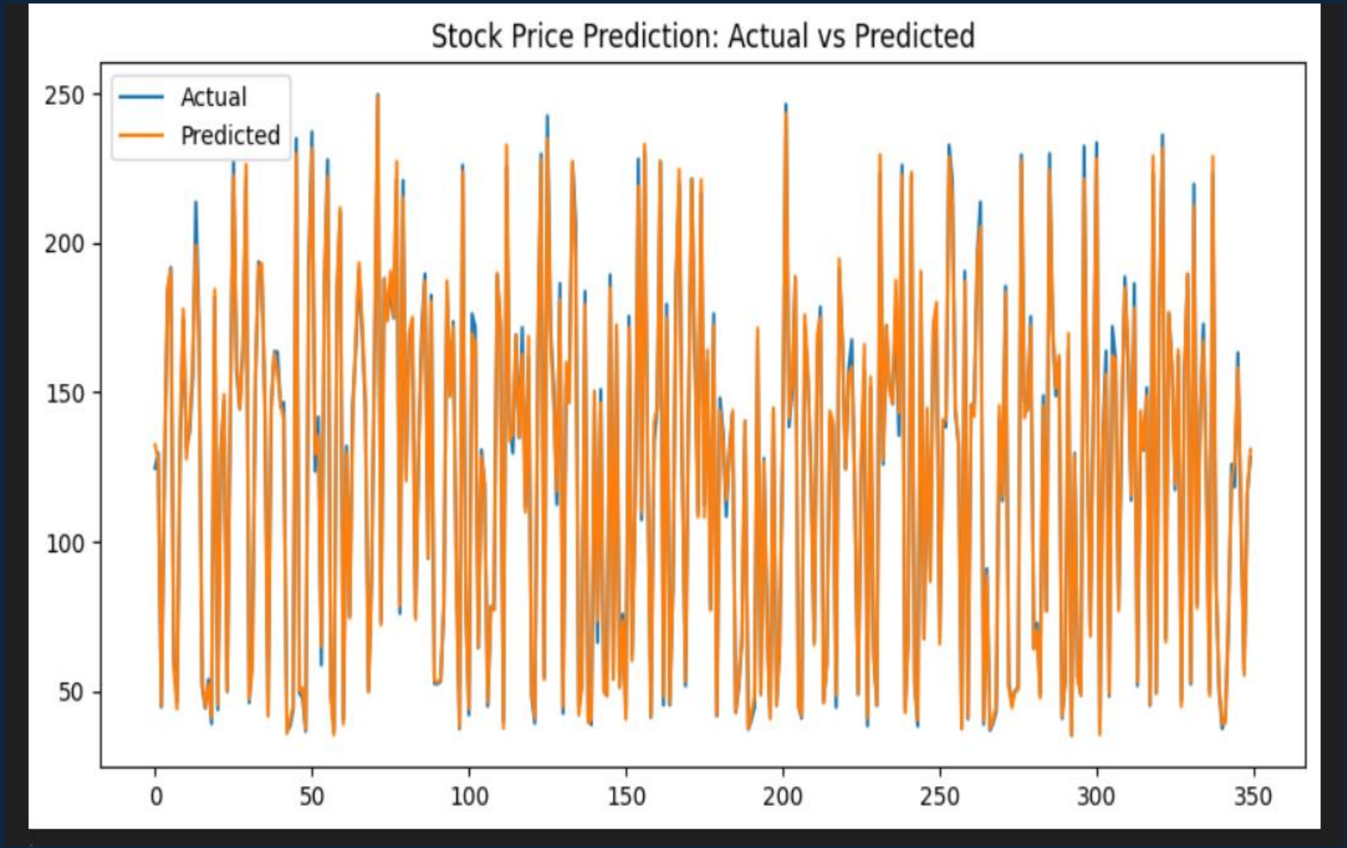
The orange line represents the predicted stock prices made by a model.

From the graph:

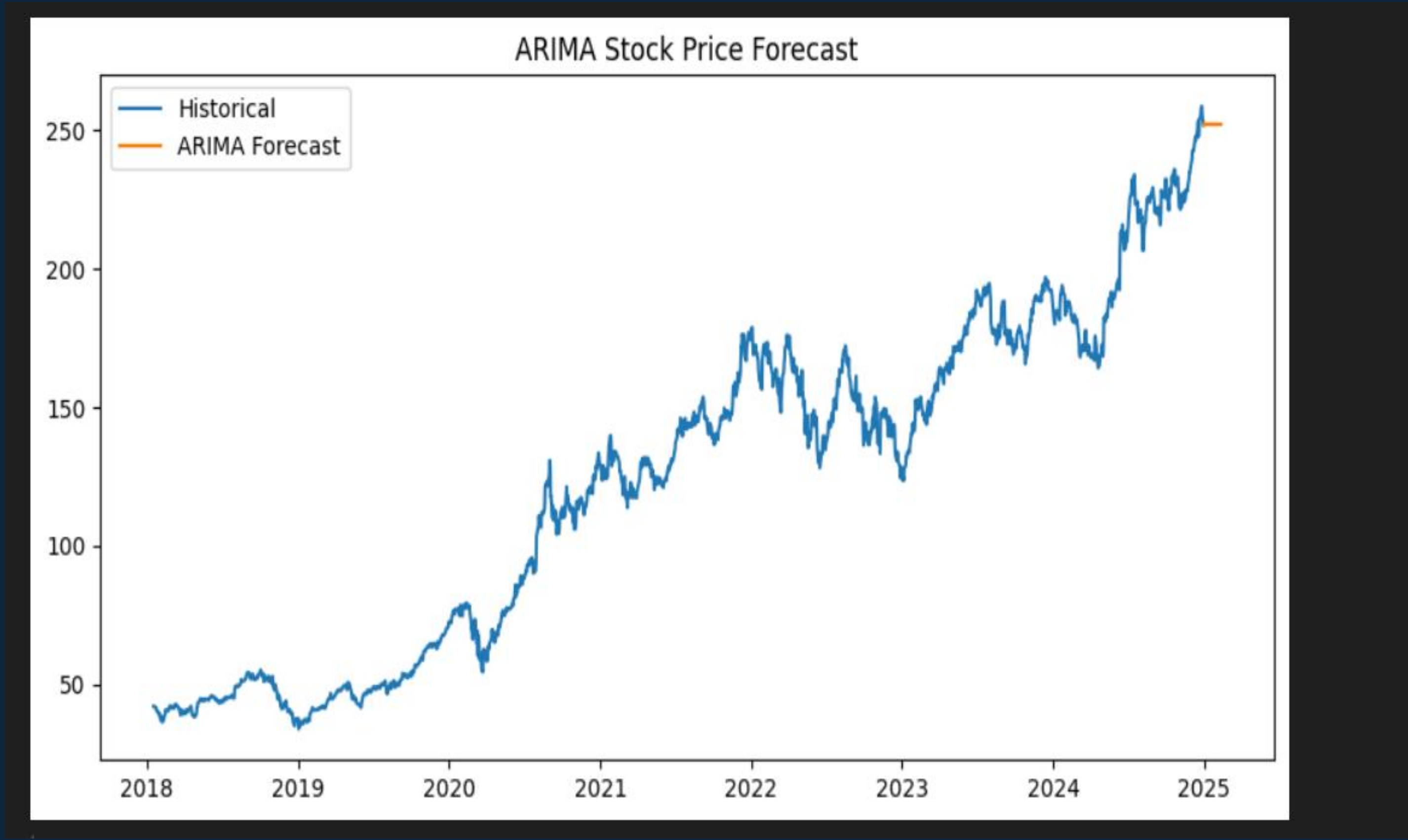
The orange line (predicted) is very messy and has huge ups and downs.

It doesn't follow the blue line (actual) closely.

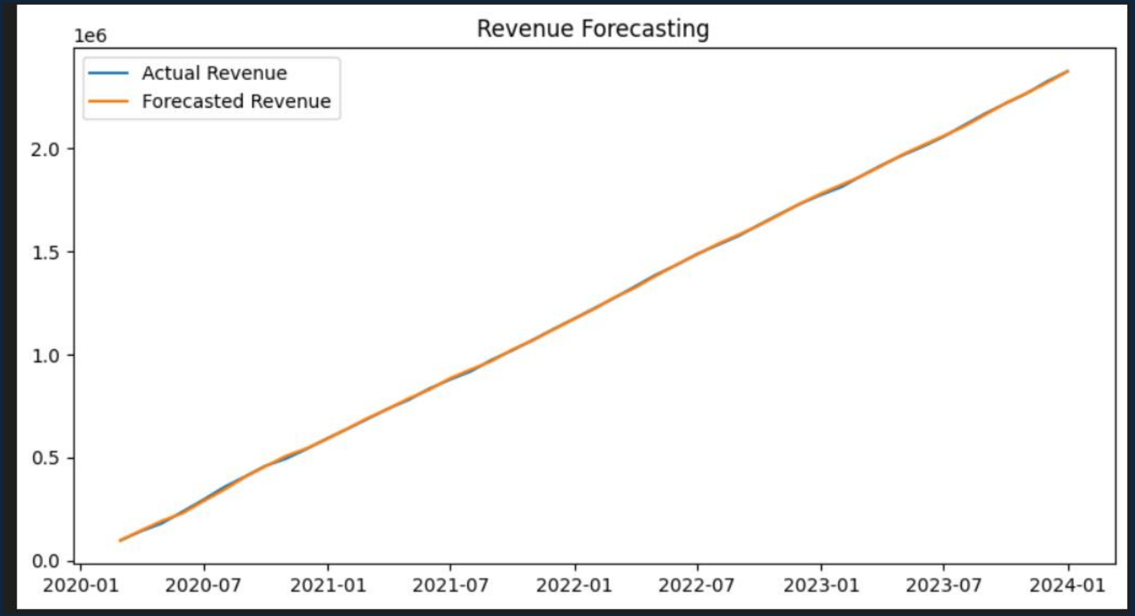
This means the model's predictions are not accurate — it's predicting a lot of random swings that are not really happening in real stock prices.



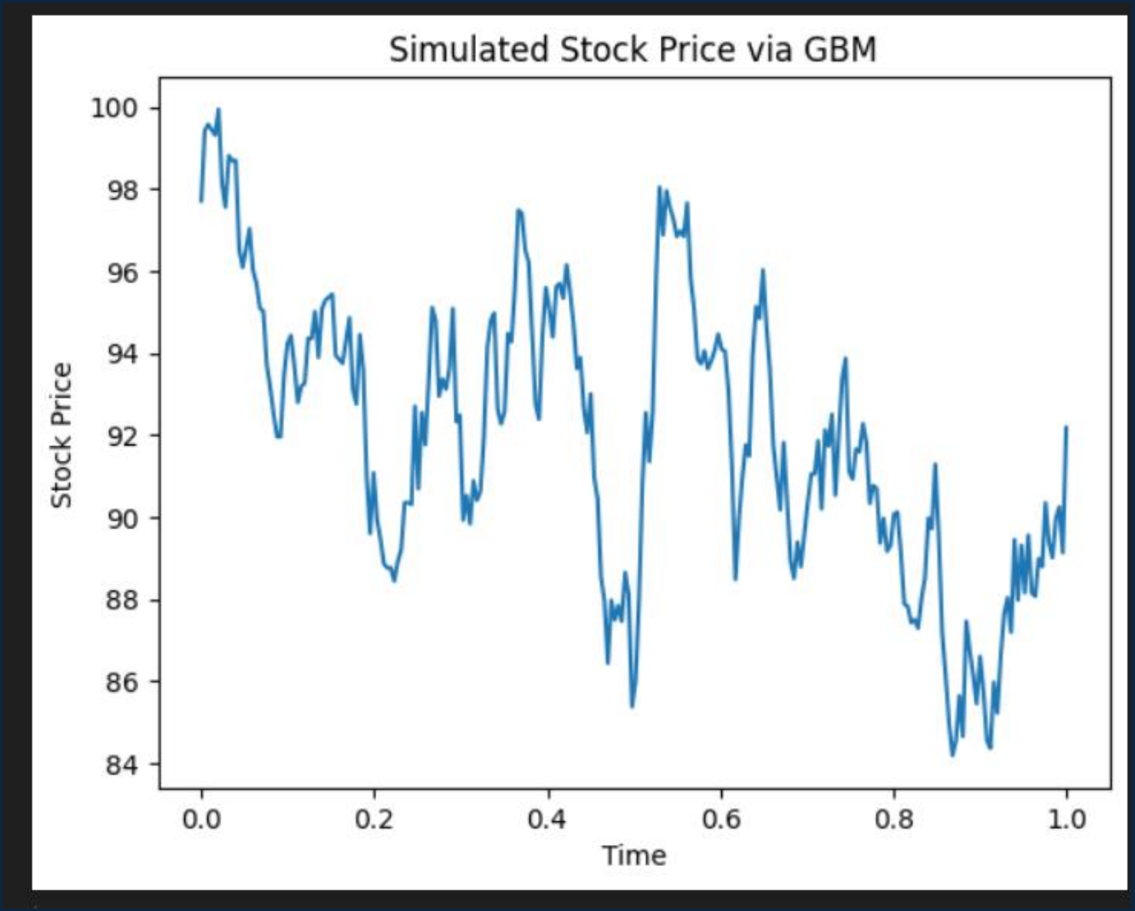
The blue line shows real stock prices from 2018 to early 2025.
The orange line shows the forecasted stock prices for a short future period.
From the graph:
The stock price has generally gone up over time, even though there are some ups and downs along the way.
The ARIMA model is predicting that the stock price will stay almost stable (a little flat) in the immediate future — it doesn't expect any big jumps or drops soon.



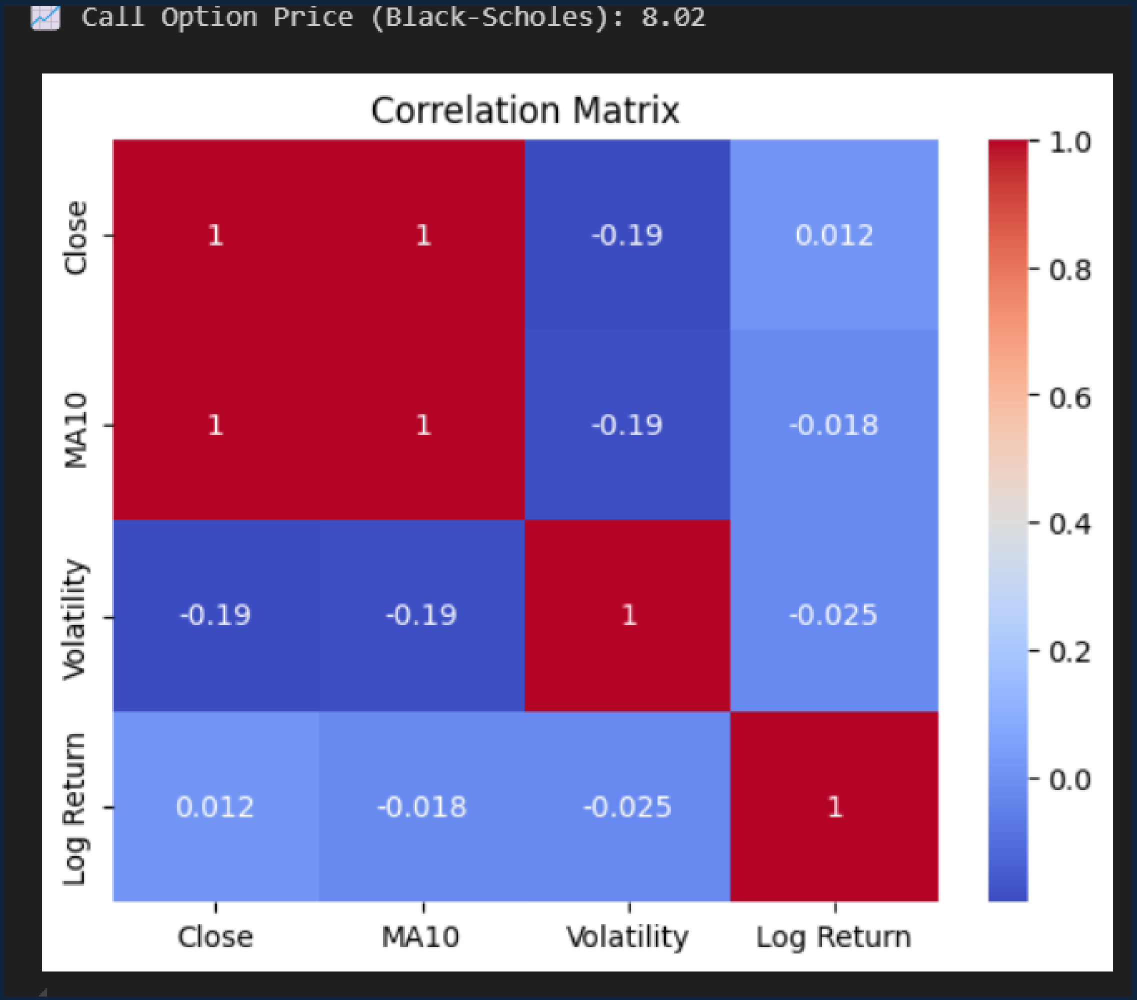
The orange and blue lines are very close, showing that the forecasted revenue matches the actual revenue very well over time (2020–2024).This means the forecasting model is accurate.



The stock price moves up and down randomly but mostly stays between 85 and 100.It shows how stock prices can fluctuate a lot over time even without major news.



This chart shows how different financial metrics are related. "Close" price and "MA10" (10-day moving average) are perfectly related (correlation = 1), while "Volatility" has a weak negative relationship with others.

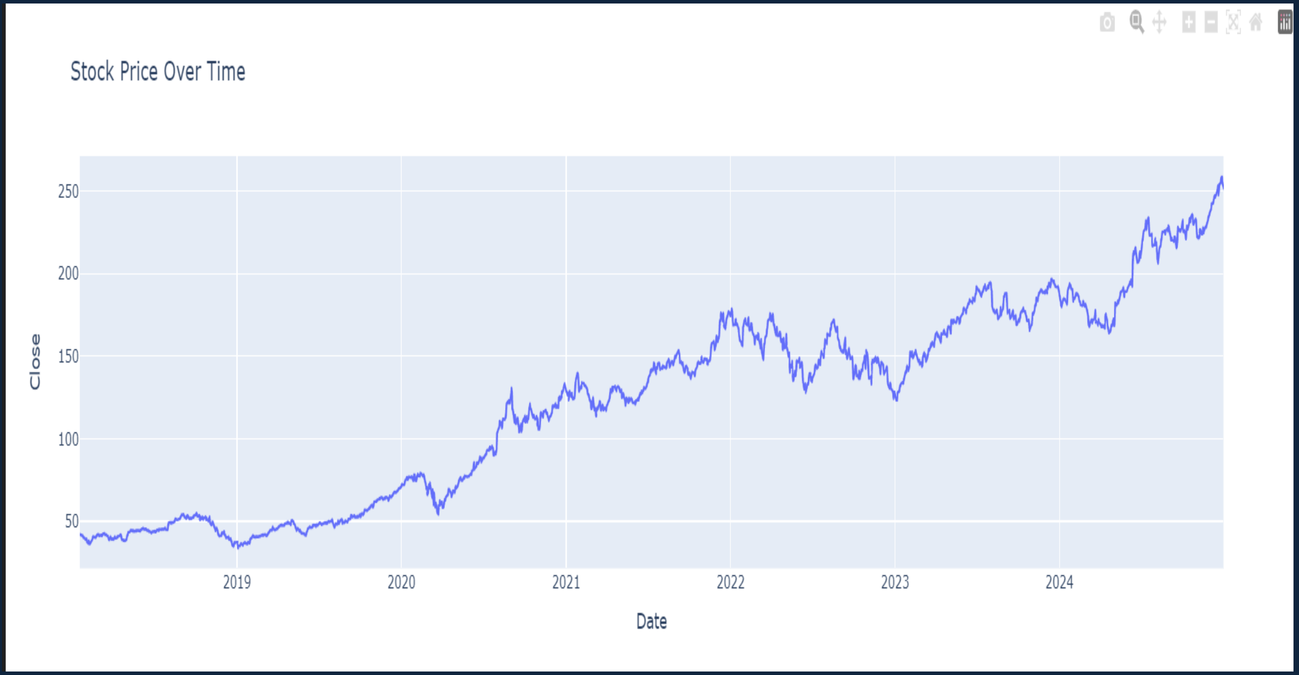


```
📈 Call Option Price (Black-Scholes): 8.02
```

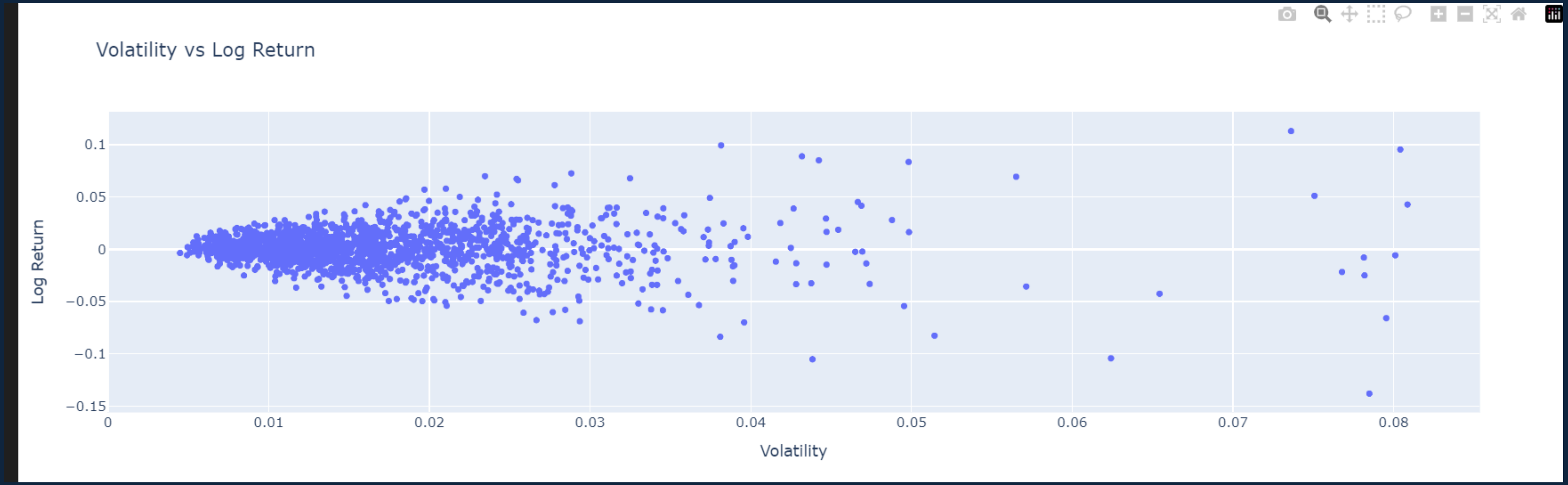
Correlation Matrix

	Close	MA10	Volatility	Log Return
Close	1	1	-0.19	0.012
MA10	1	1	-0.19	-0.018
Volatility	-0.19	-0.19	1	-0.025
Log Return	0.012	-0.018	-0.025	1

The stock price (labeled "Clones") started around 50 in 2019, peaked near 150 in 2021, then dropped sharply by 2024.The "DMA" line (likely a moving average) smooths out the price trends, showing a general rise and fall over time.



This scatter plot shows the relationship between volatility (risk) and log returns (profit/loss).Most points cluster near low volatility (0.01–0.04) and small returns (±0.05), meaning the stock was relatively stable with modest returns.A few outliers had high volatility or extreme losses (e.g., –0.15 return).



Thank you