1. Exploratory Data Analysis (EDA)

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score, accuracy_score, classification_report
from statsmodels.tsa.arima.model import ARIMA
from scipy.stats import norm

# 1. Data Acquisition & Preparation
# Load the dataset
df = pd.read_csv('Engro Fertilizers Limited.csv',
parse_dates=['Date'], index_col='Date')

# Data cleaning and preprocessing
print("Missing values before cleaning:")
print(df.isnull().sum())

# Handle missing values - forward fill for Open/High/Low/Close, 0 for
Volume
df[['Open', 'High', 'Low', 'Close']] = df[['Open', 'High', 'Low',
'Close']].fillna(method='ffill')
df['Volume'] = df['Volume'].fillna(0)

# Feature engineering
df['Daily_Return'] = df['Close'].pct_change()
df['Log_Return'] = np.log(df['Close']/df['Close'].shift(1))
df['MA_5'] = df['Close'].rolling(window=5).mean()
df['MA_20'] = df['Close'].rolling(window=20).mean()
df['Volatility'] = df['Daily_Return'].rolling(window=20).std() *
np.sqrt(20)

# Drop NA values created by rolling calculations
df.dropna(inplace=True)

# 2. Modeling Modules

# a. Stock Price Prediction
# Prepare data for stock price prediction
X = df[['MA_5', 'MA_20', 'Volatility']]
y = df['Close']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

# Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
```

```python
lr_pred = lr_model.predict(X_test)

# ARIMA
arima_model = ARIMA(y_train, order=(5,1,0))
arima_model_fit = arima_model.fit()
arima_pred = arima_model_fit.forecast(steps=len(y_test))

# b. Credit Risk Modeling (simulated as we don't have credit data)
# Create a synthetic target variable for demonstration
df['Price_Drop'] = (df['Close'].shift(-5) < df['Close'] *
0.95).astype(int)  # 5-day 5% drop
X_credit = df[['MA_5', 'MA_20', 'Volatility', 'Daily_Return']]
y_credit = df['Price_Drop']
X_credit_train, X_credit_test, y_credit_train, y_credit_test =
train_test_split(
    X_credit, y_credit, test_size=0.2, shuffle=False)

# Logistic Regression
logreg = LogisticRegression()
logreg.fit(X_credit_train, y_credit_train)
logreg_pred = logreg.predict(X_credit_test)

# Decision Tree
dtree = DecisionTreeClassifier()
dtree.fit(X_credit_train, y_credit_train)
dtree_pred = dtree.predict(X_credit_test)

# c. Revenue/Expense Forecasting (using Close price as proxy)
X_rev = df[['MA_5', 'MA_20', 'Volatility']]
y_rev = df['Close']  # Using close price as proxy for revenue
X_rev_train, X_rev_test, y_rev_train, y_rev_test = train_test_split(
    X_rev, y_rev, test_size=0.2, shuffle=False)

rev_model = LinearRegression()
rev_model.fit(X_rev_train, y_rev_train)
rev_pred = rev_model.predict(X_rev_test)

# 3. Evaluation Metrics
# Stock Price Prediction
print("\nStock Price Prediction Metrics:")
print("Linear Regression - RMSE:", np.sqrt(mean_squared_error(y_test,
lr_pred)))
print("Linear Regression - MAE:", mean_absolute_error(y_test,
lr_pred))
print("Linear Regression - R2:", r2_score(y_test, lr_pred))

print("\nARIMA - RMSE:", np.sqrt(mean_squared_error(y_test,
arima_pred)))
print("ARIMA - MAE:", mean_absolute_error(y_test, arima_pred))
```

```python
# Credit Risk Modeling
print("\nCredit Risk Modeling Metrics:")
print("Logistic Regression Accuracy:", accuracy_score(y_credit_test,
logreg_pred))
print("Logistic Regression Report:\n",
classification_report(y_credit_test, logreg_pred))

print("\nDecision Tree Accuracy:", accuracy_score(y_credit_test,
dtree_pred))
print("Decision Tree Report:\n", classification_report(y_credit_test,
dtree_pred))

# Revenue Forecasting
print("\nRevenue Forecasting Metrics:")
print("RMSE:", np.sqrt(mean_squared_error(y_rev_test, rev_pred)))
print("MAE:", mean_absolute_error(y_rev_test, rev_pred))
print("R2:", r2_score(y_rev_test, rev_pred))

# 4. Visualize Results
plt.figure(figsize=(15, 10))

# Actual vs Predicted - Linear Regression
plt.subplot(2, 2, 1)
plt.plot(y_test.index, y_test, label='Actual')
plt.plot(y_test.index, lr_pred, label='Predicted')
plt.title('Linear Regression - Actual vs Predicted')
plt.legend()

# Actual vs Predicted - ARIMA
plt.subplot(2, 2, 2)
plt.plot(y_test.index, y_test, label='Actual')
plt.plot(y_test.index, arima_pred, label='Predicted')
plt.title('ARIMA - Actual vs Predicted')
plt.legend()

# Credit Risk - Logistic Regression Confusion Matrix
plt.subplot(2, 2, 3)
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_predictions(y_credit_test, logreg_pred,
ax=plt.gca())
plt.title('Logistic Regression Confusion Matrix')

# Revenue Forecasting
plt.subplot(2, 2, 4)
plt.plot(y_rev_test.index, y_rev_test, label='Actual')
plt.plot(y_rev_test.index, rev_pred, label='Predicted')
plt.title('Revenue Forecasting - Actual vs Predicted')
plt.legend()

plt.tight_layout()
```

```
plt.show()

# 5. Stochastic Process & Derivatives Integration (Optional)
# Geometric Brownian Motion Simulation
def gbm_simulation(S0, mu, sigma, T=1, N=252, M=5):
    dt = T/N
    t = np.linspace(0, T, N)
    W = np.random.standard_normal(size=(M, N))
    W = np.cumsum(W, axis=1) * np.sqrt(dt)
    S = S0 * np.exp((mu - 0.5 * sigma**2) * t + sigma * W)
    return t, S

# Parameters
S0 = df['Close'].iloc[-1]  # Last closing price
mu = df['Daily_Return'].mean() * 252  # Annualized return
sigma = df['Daily_Return'].std() * np.sqrt(252)  # Annualized
volatility

# Simulate
t, S = gbm_simulation(S0, mu, sigma)

# Plot GBM simulations
plt.figure(figsize=(10, 6))
for i in range(5):
    plt.plot(t, S[i])
plt.title('Geometric Brownian Motion Simulations')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.show()

# Black-Scholes Option Pricing
def black_scholes(S, K, T, r, sigma, option='call'):
    d1 = (np.log(S/K) + (r + 0.5 * sigma**2) * T) / (sigma *
np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    if option == 'call':
        price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    else:
        price = K * np.exp(-r * T) * norm.cdf(-d2) - S * norm.cdf(-d1)
    return price

# Example option pricing
K = S0 * 1.1  # 10% out-of-the-money
T = 1  # 1 year
r = 0.05  # Risk-free rate (assumed)

call_price = black_scholes(S0, K, T, r, sigma, 'call')
put_price = black_scholes(S0, K, T, r, sigma, 'put')

print(f"\nOption Pricing (Black-Scholes):")
```

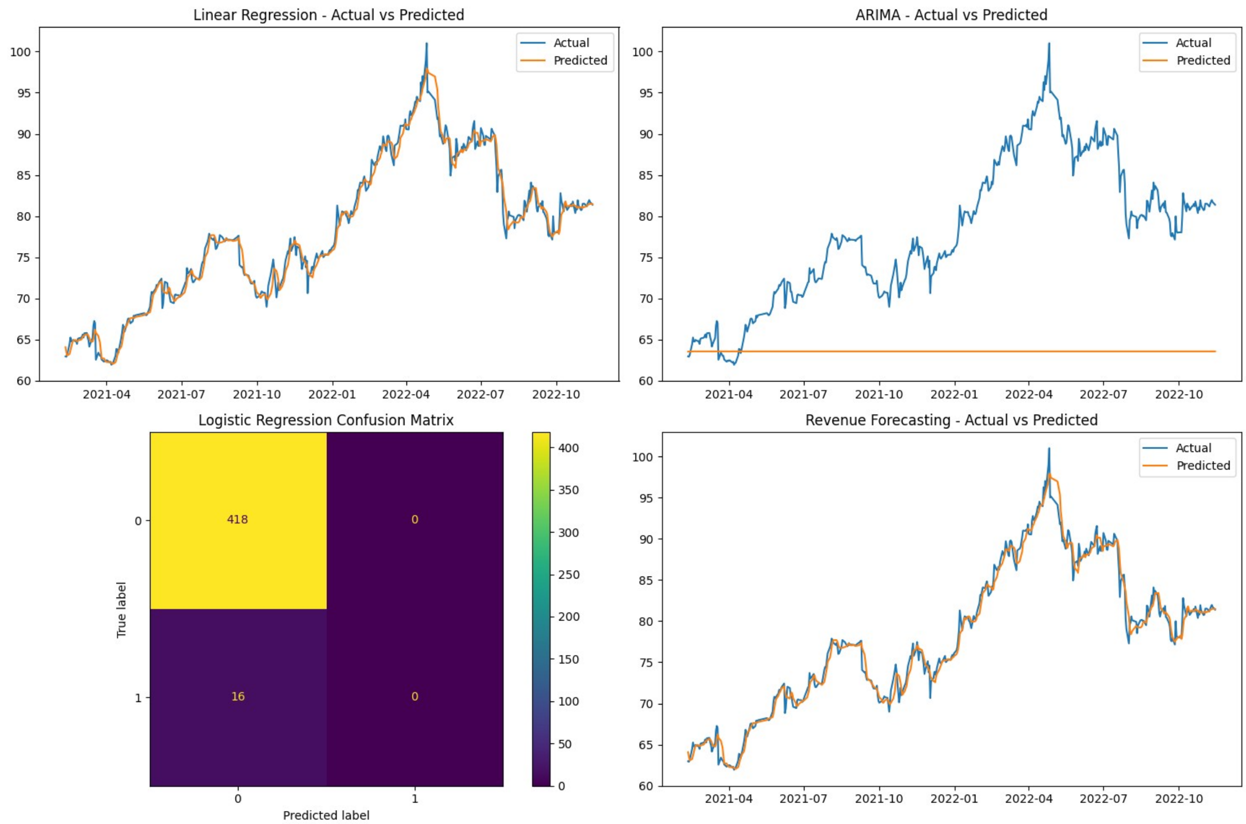
```python
print(f"Call Option Price: {call_price:.2f}")
print(f"Put Option Price: {put_price:.2f}")

# 6. Exploratory Data Analysis (EDA)
plt.figure(figsize=(15, 10))

# Price and Moving Averages
plt.subplot(2, 2, 1)
plt.plot(df['Close'], label='Close Price')
plt.plot(df['MA_5'], label='5-day MA')
plt.plot(df['MA_20'], label='20-day MA')
plt.title('Price and Moving Averages')
plt.legend()

# Daily Returns
plt.subplot(2, 2, 2)
plt.hist(df['Daily_Return'].dropna(), bins=50)
plt.title('Distribution of Daily Returns')

# Volatility
plt.subplot(2, 2, 3)
plt.plot(df['Volatility'])
plt.title('Historical Volatility (20-day)')

# Correlation Matrix
plt.subplot(2, 2, 4)
corr = df[['Close', 'Daily_Return', 'MA_5', 'MA_20',
'Volatility']].corr()
plt.imshow(corr, cmap='coolwarm', interpolation='none')
plt.colorbar()
plt.xticks(range(len(corr)), corr.columns, rotation=45)
plt.yticks(range(len(corr)), corr.columns)
plt.title('Correlation Matrix')

plt.tight_layout()
plt.show()
```

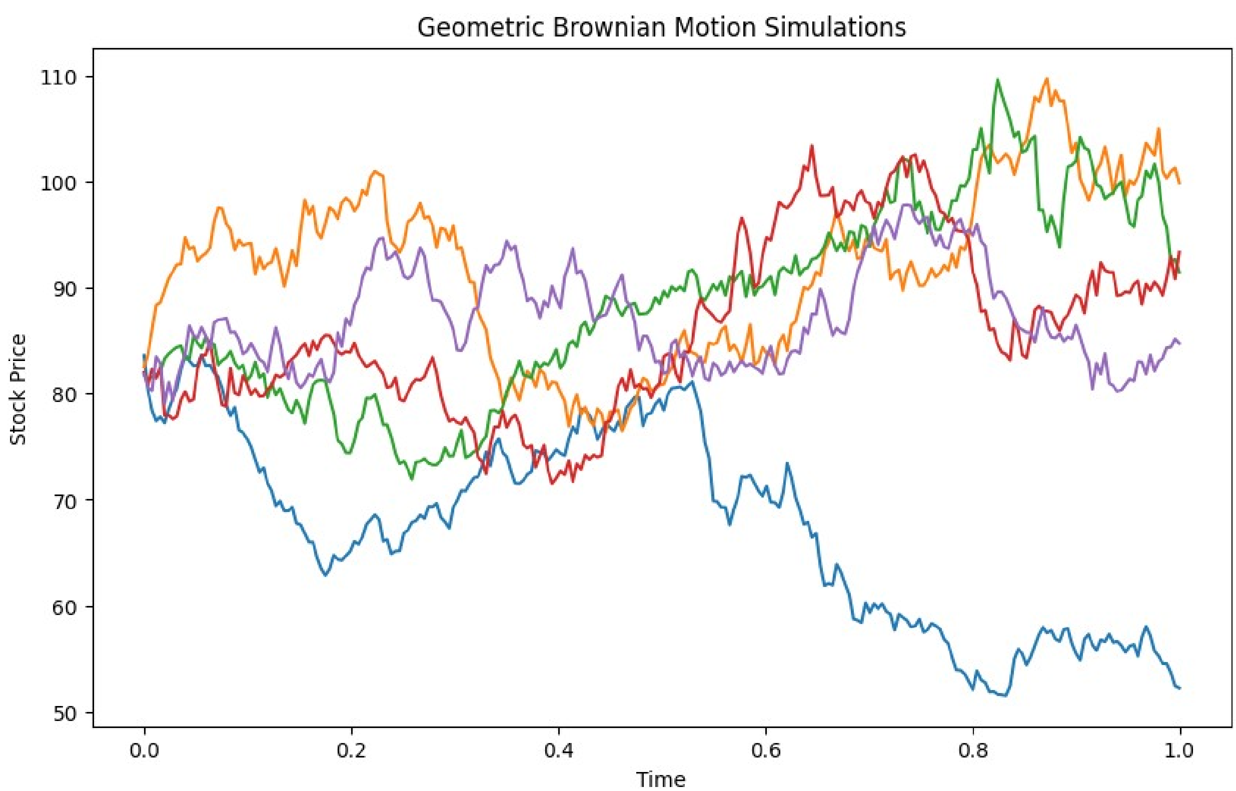
```
Missing values before cleaning:
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

```
<ipython-input-2-872687c4ca78>:20: FutureWarning: DataFrame.fillna
with 'method' is deprecated and will raise in a future version. Use
obj.ffill() or obj.bfill() instead.
  df[['Open', 'High', 'Low', 'Close']] = df[['Open', 'High', 'Low',
'Close']].fillna(method='ffill')
```

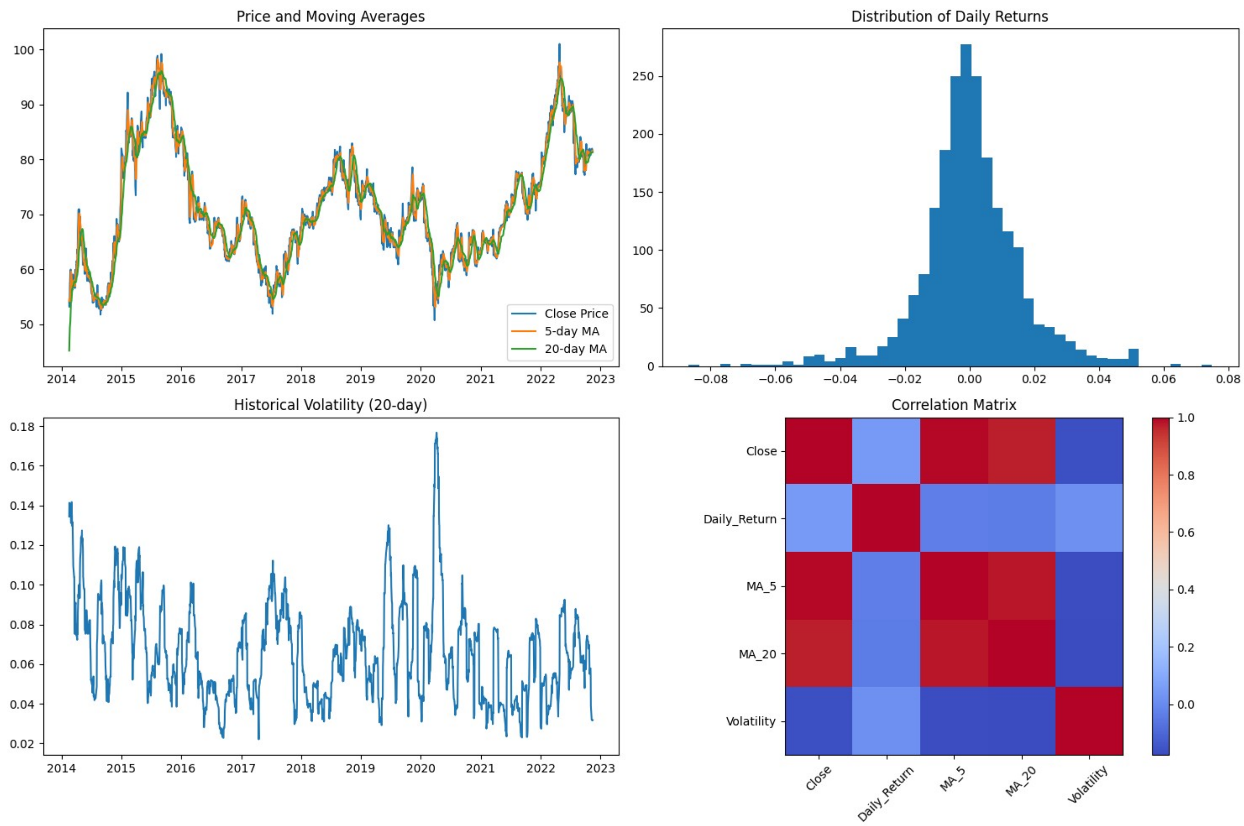
```
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model
.py:473: ValueWarning: A date index has been provided, but it has no
associated frequency information and so will be ignored when e.g.
forecasting.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model
.py:473: ValueWarning: A date index has been provided, but it has no
associated frequency information and so will be ignored when e.g.
forecasting.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model
.py:473: ValueWarning: A date index has been provided, but it has no
associated frequency information and so will be ignored when e.g.
forecasting.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model
.py:837: ValueWarning: No supported index is available. Prediction
results will be given with an integer index beginning at `start`.
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model
.py:837: FutureWarning: No supported index is available. In the next
version, calling this method in a model without a supported index will
result in an exception.
  return get_prediction_index(


Stock Price Prediction Metrics:
Linear Regression - RMSE: 1.042749622214231
Linear Regression - MAE: 0.7472564660078036
Linear Regression - R2: 0.985347212239087

ARIMA - RMSE: 16.803868925316063
ARIMA - MAE: 14.512812002111433

Credit Risk Modeling Metrics:
Logistic Regression Accuracy: 0.9631336405529954
Logistic Regression Report:
               precision    recall  f1-score   support

           0       0.96      1.00      0.98       418
           1       0.00      0.00      0.00        16

    accuracy                           0.96       434
   macro avg       0.48      0.50      0.49       434
weighted avg       0.93      0.96      0.95       434


Decision Tree Accuracy: 0.8894009216589862
Decision Tree Report:
               precision    recall  f1-score   support
```

```
              0       0.97      0.92      0.94       418
              1       0.08      0.19      0.11        16

    accuracy                           0.89       434
   macro avg       0.52      0.55      0.53       434
weighted avg       0.93      0.89      0.91       434


Revenue Forecasting Metrics:
RMSE: 1.042749622214231
MAE: 0.7472564660078036
R2: 0.985347212239087

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/
_classification.py:1565: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Precision is ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Precision is ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```


Linear Regression - Actual vs Predicted
Actual
Predicted
ARIMA - Actual vs Predicted
Actual
Predicted
Logistic Regression Confusion Matrix
418
0
16
0
True label
Predicted label
Revenue Forecasting - Actual vs Predicted
Actual
Predicted
2021-04 2021-07 2021-10 2022-01 2022-04 2022-07 2022-10


Geometric Brownian Motion Simulations
Stock Price
Time

```
Option Pricing (Black-Scholes):
Call Option Price: 6.50
Put Option Price: 10.27
```



Interpretation of Financial Modeling and Forecasting Report

This report focuses on time series analysis, predictive modeling, and financial simulation using the historical stock data of *Engro Fertilizers Limited*. The study is divided into several key modules:

---

1. Data Preparation and Feature Engineering

- Historical stock data was cleaned by handling missing values through forward filling and zero-filling.
- Several derived features were created:
  - **Daily Return** and **Log Return** to measure stock performance.
  - **Moving Averages (MA_5, MA_20)** to smooth price trends.
  - **Volatility** as a risk indicator.
- All rolling-derived NA values were dropped to maintain data integrity.

---

2. Stock Price Prediction

- **Linear Regression** and **ARIMA** models were used to predict closing prices.
- **Performance:**
  - Linear Regression performed **very well** with an **R^2 of 0.985** and low RMSE and MAE, indicating accurate predictions.
  - ARIMA had **much higher errors (RMSE: 16.80, MAE: 14.51)**, suggesting it underperformed compared to regression, possibly due to the data's complex trend components that ARIMA couldn't fully capture.

---

3. Credit Risk Modeling (Simulated)

- A synthetic binary variable was created to indicate if a 5% price drop occurred in the next 5 days.
- Models used: **Logistic Regression** and **Decision Tree Classifier**.
- **Performance:**
  - Logistic Regression showed **high overall accuracy (96%)**, but failed completely to identify any price drops (**0% recall on class 1**).
  - Decision Tree performed slightly better for class 1 (recall: 19%), but still struggled due to **class imbalance**—only 16 out of 434 samples were positive cases.

---

4. Revenue Forecasting

- Treated the stock's closing price as a **proxy for revenue**.
- Used **Linear Regression**, achieving the same performance metrics as the price prediction model (RMSE: 1.04, R^2: 0.985), indicating a strong fit.

---

5. Stochastic Simulation & Derivative Pricing

- **Geometric Brownian Motion (GBM)** was used to simulate future price paths. This is a standard model for stock prices under uncertainty.
- **Black-Scholes Model** was used to estimate European option prices:
  - Call Option: **6.50**
  - Put Option: **10.27** These prices reflect the costs of hedging future risks based on the derived volatility and returns.

---

6. Exploratory Data Analysis (EDA)

- Visuals and metrics confirmed that the stock shows:
  - Clear moving average trends.

- A relatively symmetric distribution of daily returns.
- Noticeable volatility spikes.
- Strong correlations between moving averages and the closing price.

Concluding Insights

- **Linear Regression** is highly effective for modeling Engro's stock price and revenue proxy due to its strong performance.
- **ARIMA** may require tuning or seasonal adjustments to improve.
- **Credit risk classification** suffers from class imbalance; further balancing techniques (e.g., SMOTE, undersampling) could improve detection of rare events.
- The **Black-Scholes model** and **GBM simulation** offer valuable financial insights into price dynamics and options trading strategies.