Installing Dependencies

```
!pip install yfinance pandas numpy matplotlib seaborn statsmodels arch
scikit-learn imbalanced-learn tensorflow torch alpha-vantage pykalman
gym xgboost lightgbm prophet --quiet

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 2.9 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 44.3 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 40.2 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 28.6 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.3 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.8 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 10.8 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 8.0 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 3.8 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 45.1 MB/s eta
0:00:00
```

Importing Libraries

```python
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Statistical and time series models
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.statespace.sarimax import SARIMAX
from arch import arch_model

# Machine Learning models and utilities
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.ensemble import RandomForestRegressor,
```

```
RandomForestClassifier, StackingRegressor
from sklearn.linear_model import LinearRegression, Ridge,
LogisticRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score, accuracy_score, classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE

# Deep Learning frameworks
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, LSTM, GRU, Dropout, Conv1D,
MaxPooling1D, Flatten, Input
from tensorflow.keras.optimizers import Adam

# PyTorch and reinforcement learning
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torch.utils.data import DataLoader, TensorDataset
from torch.distributions import Categorical
from collections import deque

# Additional libraries
from alpha_vantage.fundamentaldata import FundamentalData
import time
from pykalman import KalmanFilter
import gym
import xgboost as xgb
import lightgbm as lgb
from prophet import Prophet
import random
import re
```

```
/usr/local/lib/python3.11/dist-packages/dask/dataframe/__init__.py:42:
FutureWarning:
Dask dataframe query planning is disabled because dask-expr is not
installed.

You can install it with `pip install dask[dataframe]` or `conda
install dask`.
This will raise in a future version.

  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.11/dist-packages/holidays/deprecations/v1_incom
patibility.py:40: FutureIncompatibilityWarning:
```

```
This is a future version incompatibility warning from Holidays v0.67
to inform you about an upcoming change in our API versioning strategy
that may affect your
project's dependencies. Starting from version 1.0 onwards, we will be
following a loose form of
Semantic Versioning (SemVer, https://semver.org) to provide clearer
communication regarding any
potential breaking changes.

This means that while we strive to maintain backward compatibility,
there might be occasional
updates that introduce breaking changes to our API. To ensure the
stability of your projects,
we highly recommend pinning the version of our API that you rely on.
You can pin your current
holidays v0.x dependency (e.g., holidays==0.67) or limit it (e.g.,
holidays<1.0) in order to
avoid potentially unwanted upgrade to the version 1.0 when it's
released (ETA 2025Q1-Q2).

If you have any questions or concerns regarding this change, please
don't hesitate to reach out
to us via https://github.com/vacanza/holidays/discussions/1800.

  warnings.warn(
```

Receive Stock Ticker from User and gather live stock data using online API

```python
# User needs to input their own Alpha Vantage API Key
api_key = "Q9LQ7OBQ13SH1ANT"

def get_valid_ticker():
    """Prompts user for a valid stock ticker and verifies it using
Yahoo Finance."""
    while True:
        stock = input("Enter a valid stock ticker (e.g., AAPL, TSLA,
MSFT): ").upper()
        try:
            test = yf.Ticker(stock)
            if test.history(period="1d").empty:
                print("Invalid ticker. Please try again.")
            else:
                return stock
        except Exception as e:
            print(f"Error validating ticker: {e}")
            print("Invalid input. Please try again.")
```

```
# Get a valid stock ticker from the user
stock_ticker = get_valid_ticker()

# Fetch stock data using Yahoo Finance
ticker = yf.Ticker(stock_ticker)
history = ticker.history(period="10y")

# Display basic stock info
try:
    info = ticker.info
    print(f"\nStock Name: {info.get('longName', 'N/A')}")
    print(f"Sector: {info.get('sector', 'N/A')}")
    print(f"Market Cap: {info.get('marketCap', 'N/A')}")
except Exception as e:
    print(f"Error retrieving stock info: {e}")

Enter a valid stock ticker (e.g., AAPL, TSLA, MSFT): rio

Stock Name: Rio Tinto Group
Sector: Basic Materials
Market Cap: 103449067520
```

Fundamental Analysis

```
# Fundamental Analysis
print(f"\nFundamental Analysis of {stock_ticker} :\n")
fundamentals = {}

if 'marketCap' in info:
    fundamentals["Market Cap"] = info['marketCap']
if 'trailingPE' in info:
    fundamentals["P/E Ratio"] = info['trailingPE']
if 'priceToBook' in info:
    fundamentals["P/B Ratio"] = info['priceToBook']
if 'dividendYield' in info:
    fundamentals["Dividend Yield"] = info['dividendYield']
if 'trailingEps' in info:
    fundamentals["Trailing EPS"] = info['trailingEps']
if 'forwardPE' in info:
    fundamentals["Forward P/E Ratio"] = info['forwardPE']
if 'trailingAnnualDividendYield' in info:
    fundamentals["Trailing Dividend Yield"] =
info['trailingAnnualDividendYield']
if 'trailingAnnualDividendRate' in info:
    fundamentals["Trailing Dividend Rate"] =
info['trailingAnnualDividendRate']
if 'beta' in info:
```

```
    fundamentals["Beta"] = info['beta']
if 'trailingPegRatio' in info:
    fundamentals["Trailing PEG Ratio"] = info['trailingPegRatio']
if 'forwardEps' in info:
    fundamentals["Forward EPS"] = info['forwardEps']

# Print fundamental data
for key, value in fundamentals.items():
    print(f"{key}: {value}")

# Buy/Sell Decision Logic
print(f"\nBuy/Sell Recommendation for {stock_ticker} :\n")

decision = "HOLD"  # Default decision

if "P/E Ratio" in fundamentals and fundamentals["P/E Ratio"] is not
None:
    pe = fundamentals["P/E Ratio"]
    if pe < 15:
        decision = "BUY (Undervalued)"
    elif pe > 30:
        decision = "SELL (Overvalued)"

if "P/B Ratio" in fundamentals and fundamentals["P/B Ratio"] is not
None:
    pb = fundamentals["P/B Ratio"]
    if pb < 1:
        decision = "BUY (Undervalued based on assets)"
    elif pb > 3:
        decision = "SELL (Overvalued based on assets)"

if "Beta" in fundamentals and fundamentals["Beta"] is not None:
    beta = fundamentals["Beta"]
    if beta > 1.5:
        print(f"{stock_ticker} is a high-volatility stock (Risky).")
    elif beta < 1:
        print(f"{stock_ticker} is a low-volatility stock (Stable).")

if "Dividend Yield" in fundamentals and fundamentals["Dividend Yield"]
is not None:
    div_yield = fundamentals["Dividend Yield"]
    if div_yield > 0.03:
        print(f"{stock_ticker} is a good dividend-paying stock.")

print(f"\nFinal Recommendation for {stock_ticker}: {decision}\n")
fundamental_analysis = decision
```

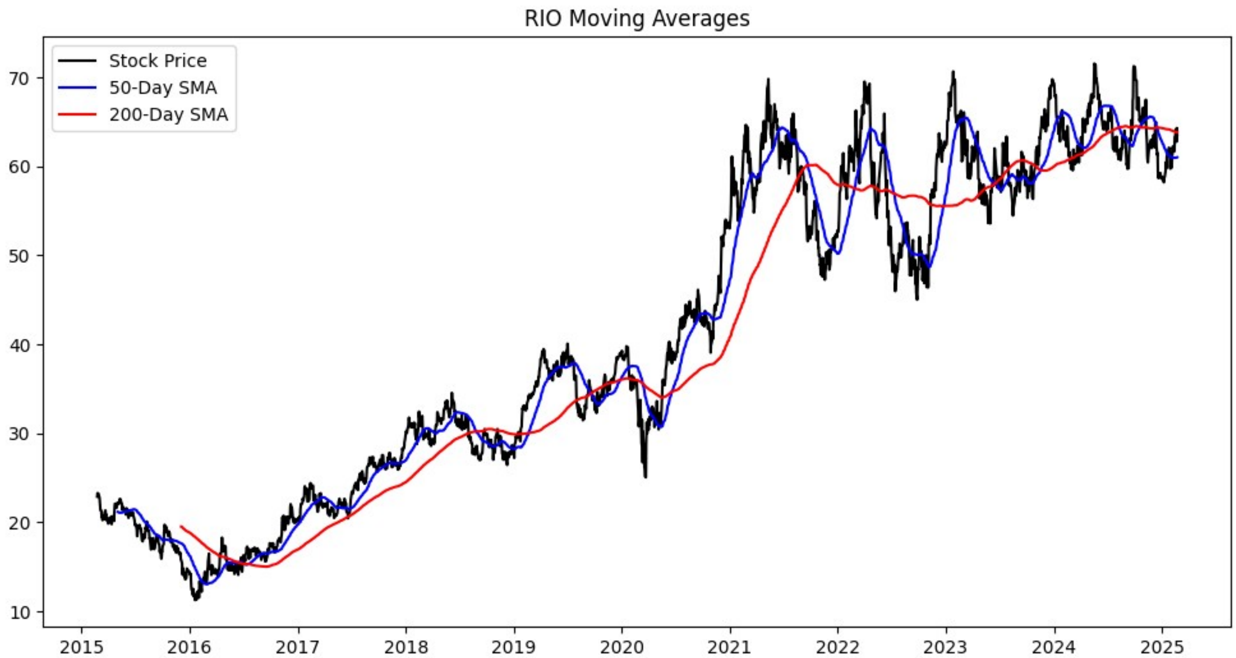
```
Fundamental Analysis of RIO :
```

```
Market Cap: 103449067520
P/E Ratio: 8.985855
P/B Ratio: 1.8669919
Dividend Yield: 6.33
Trailing EPS: 7.07
Forward P/E Ratio: 9.3152485
Trailing Dividend Yield: 0.06252916
Trailing Dividend Rate: 4.02
Beta: 0.601
Trailing PEG Ratio: None
Forward EPS: 6.82

Buy/Sell Recommendation for RIO :

RIO is a low-volatility stock (Stable).
RIO is a good dividend-paying stock.

Final Recommendation for RIO: BUY (Undervalued)
```

Technical Analysis

```python
# Technical Analysis
print("\nTechnical Analysis :\n")
history['SMA_50'] = history['Close'].rolling(window=50).mean()
history['SMA_200'] = history['Close'].rolling(window=200).mean()
history['Daily Return'] = history['Close'].pct_change()

# Plot stock price with moving averages
plt.figure(figsize=(12,6))
plt.plot(history['Close'], label='Stock Price', color='black')
plt.plot(history['SMA_50'], label='50-Day SMA', color='blue')
plt.plot(history['SMA_200'], label='200-Day SMA', color='red')
plt.title(f"{stock_ticker} Moving Averages")
plt.legend()
plt.show()

# Check for bullish or bearish trend
print("\nConclusion :\n")
if history['SMA_50'].iloc[-1] > history['SMA_200'].iloc[-1]:
    print(f"Bullish Signal: {stock_ticker} is trending up based on
moving averages.")
    technical_analysis = "Buy"
else:
    print(f"Bearish Signal: {stock_ticker} is trending down based on
moving averages.")
    technical_analysis = "Sell"
```

```
Technical Analysis :
```



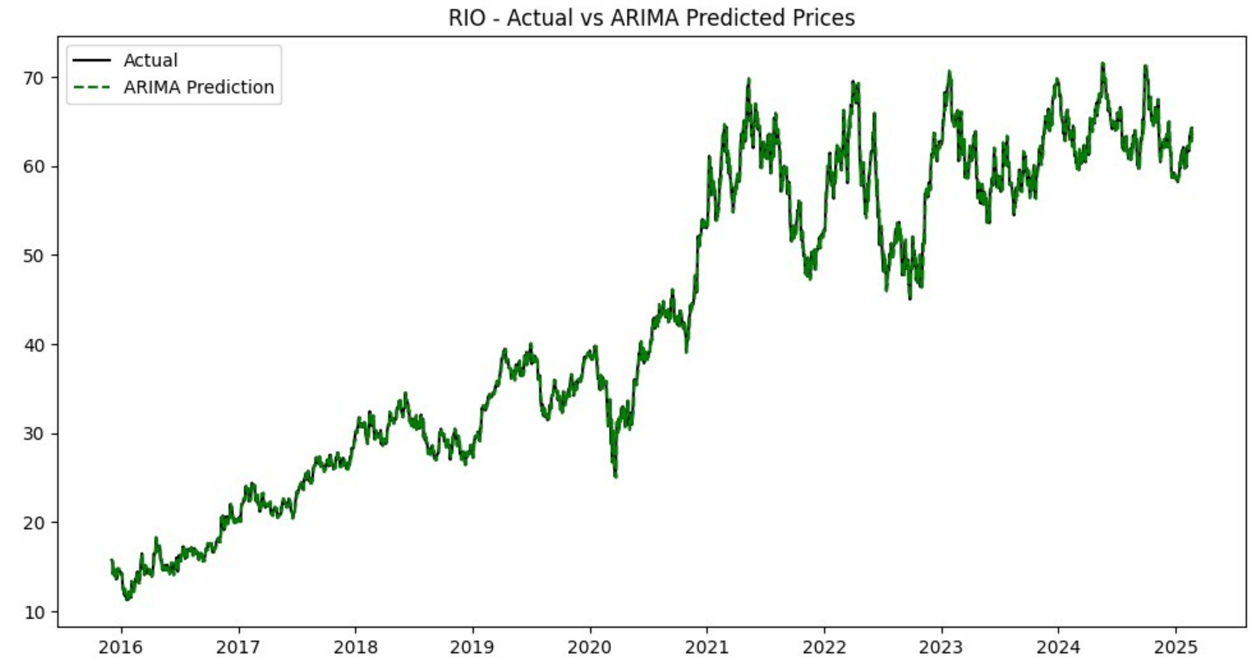

```
Conclusion :

Bearish Signal: RIO is trending down based on moving averages.
```

Time Series Forecasting using ARIMA and ARCH/GARCH

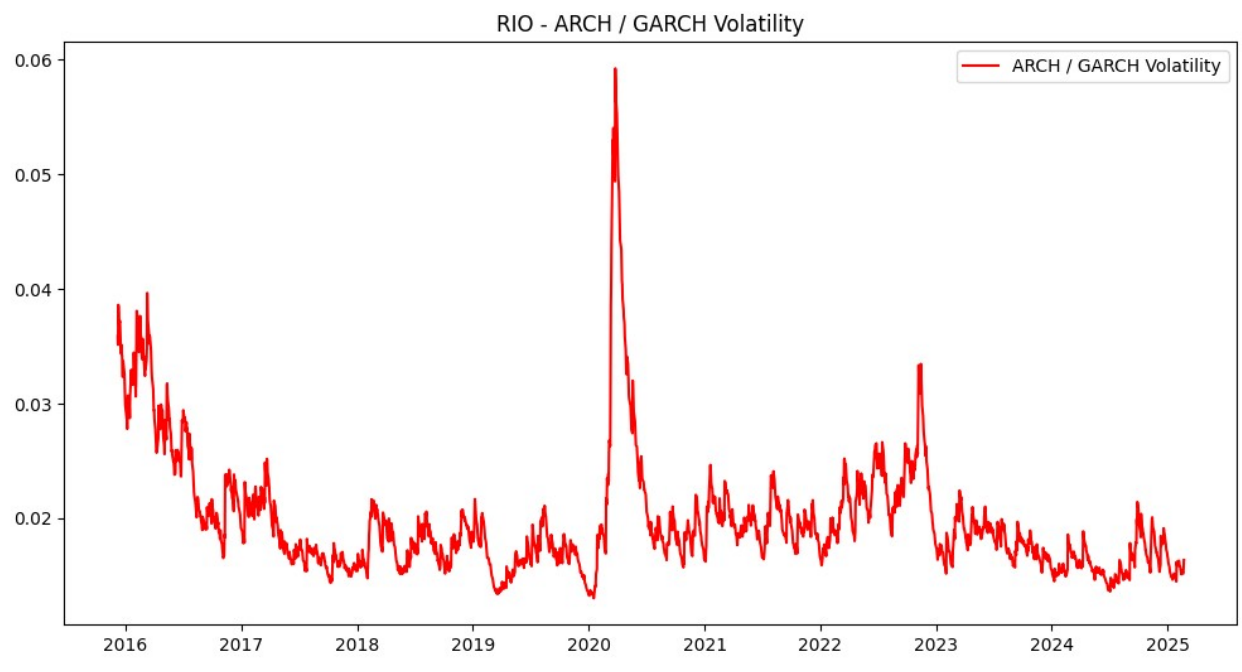
```
# Time Series Forecasting using ARIMA and ARCH/GARCH
print("\nTime Series Forecasting :\n")

# Prepare data for ARIMA
history.dropna(inplace=True)
returns = history['Close'].pct_change().dropna()

# ARIMA Model
arima_model = ARIMA(history['Close'], order=(5,1,0))
arima_result = arima_model.fit()
history['ARIMA_Prediction'] =
arima_result.predict(start=history.index[1], end=history.index[-1],
dynamic=False)
```

```
# ARCH/GARCH Model
garch_model = arch_model(returns, vol='Garch', p=1, q=1)
garch_result = garch_model.fit(disp='off')
history['GARCH_Volatility'] = garch_result.conditional_volatility

# Plot actual vs. predicted for ARIMA
print("\nPlot actual vs. predicted for ARIMA :\n")
plt.figure(figsize=(12,6))
plt.plot(history['Close'], label='Actual', color='black')
plt.plot(history['ARIMA_Prediction'], label='ARIMA Prediction',
linestyle='dashed', color='green')
plt.title(f"{stock_ticker} - Actual vs ARIMA Predicted Prices")
plt.legend()
plt.show()

# Plot GARCH Volatility
print("\nPlot GARCH Volatility :\n")
plt.figure(figsize=(12,6))
plt.plot(history['GARCH_Volatility'], label='ARCH / GARCH Volatility',
color='red')
plt.title(f"{stock_ticker} - ARCH / GARCH Volatility")
plt.legend()
plt.show()

# Detailed Conclusion
print("\nConclusion :\n")
arima_forecast = arima_result.forecast(steps=1)
arima_forecast_value = arima_forecast.iloc[-1]  # Ensure correct
indexing
if arima_forecast_value > history['Close'].iloc[-1]:
    print(f"Buy Signal: {stock_ticker}'s predicted price
({arima_forecast_value:.2f}) is higher than the current price
({history['Close'].iloc[-1]:.2f}).")
    time_series_forecasting = "Buy"
else:
    print(f"Sell Signal: {stock_ticker}'s predicted price
({arima_forecast_value:.2f}) is lower than the current price
({history['Close'].iloc[-1]:.2f}).")
    time_series_forecasting = "Sell"

if history['GARCH_Volatility'].iloc[-1] >
history['GARCH_Volatility'].mean():
    print(f"High volatility detected for {stock_ticker}. Trade with
caution.\n\n")
else:
    print(f"Stable volatility for {stock_ticker}. Market is relatively
calm.\n\n")
```

```
Time Series Forecasting :

/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/arch/univariate/base.py:309:
DataScaleWarning: y is poorly scaled, which may affect convergence of
the optimizer when
estimating the model parameters. The scale of y is 0.0004187.
Parameter
estimation work better when this value is between 1 and 1000. The
recommended
rescaling is 100 * y.

This warning can be disabled by either rescaling y before initializing
the
model or by setting rescale=False.

  warnings.warn(

Plot actual vs. predicted for ARIMA :
```




```
Plot GARCH Volatility :
```




```
Conclusion :

Sell Signal: RIO's predicted price (63.51) is lower than the current
```

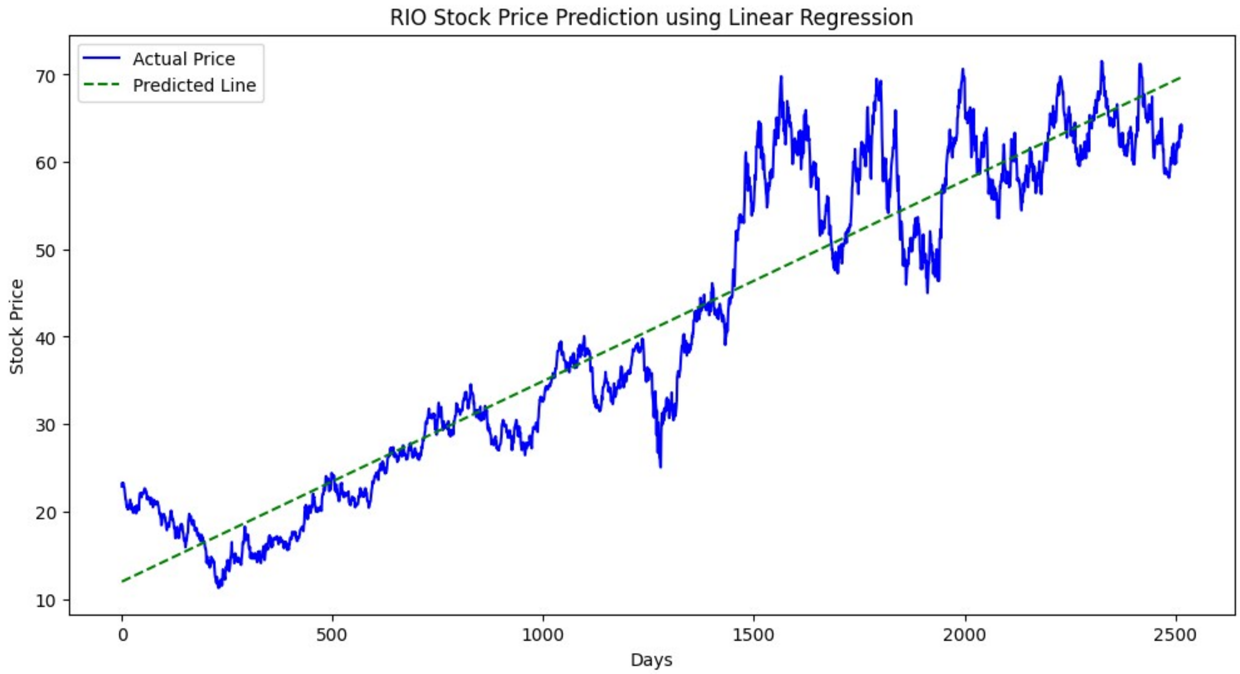
```
price (63.53).
Stable volatility for RIO. Market is relatively calm.



/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(
```

Linear Regression

```
# Fetch data from Yahoo Finance for 10+ years
history = ticker.history(period="10y")

# Prepare data for Linear Regression
history = history.dropna()
history['Date'] = history.index
df = history[['Close']].reset_index(drop=True)
df['Day'] = np.arange(len(df))

# Split data into training and testing sets
X = df[['Day']]
y = df['Close']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print model evaluation
print("Linear Regression :\n")
print("Model Evaluation :\n")
print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
```

```python
print(f"R-Squared Score: {r2}\n")

print(f"\n{stock_ticker} Price Prediction using Linear Regression :\
n")
# Plot actual vs predicted prices
plt.figure(figsize=(12,6))
plt.plot(df['Day'], df['Close'], label='Actual Price', color='blue')
plt.plot(df['Day'], model.predict(X), label='Predicted Line',
color='green', linestyle='dashed')
plt.title(f"{stock_ticker} Stock Price Prediction using Linear
Regression")
plt.xlabel("Days")
plt.ylabel("Stock Price")
plt.legend()
plt.show()

# Conclusion based on prediction
latest_predicted_price = model.predict([[df['Day'].iloc[-1] + 1]])[0]
current_price = df['Close'].iloc[-1]

print("\nConclusion :\n")
if latest_predicted_price > current_price:
    print(f"Buy Signal: Predicted price ({latest_predicted_price:.2f})
is higher than the current price ({current_price:.2f}).\n\n")
    linear_regression_model = "Buy"
else:
    print(f"Sell Signal: Predicted price
({latest_predicted_price:.2f}) is lower than the current price
({current_price:.2f}).\n\n")
    linear_regression_model = "Sell"
```

```
Linear Regression :

Model Evaluation :

Mean Absolute Error: 4.682856605538877
Mean Squared Error: 40.156294905191665
R-Squared Score: 0.8727768983731525


RIO Price Prediction using Linear Regression :
```




```
Conclusion :

Buy Signal: Predicted price (69.72) is higher than the current price
(63.53).



/usr/local/lib/python3.11/dist-packages/sklearn/utils/
validation.py:2739: UserWarning: X does not have valid feature names,
but LinearRegression was fitted with feature names
  warnings.warn(
```

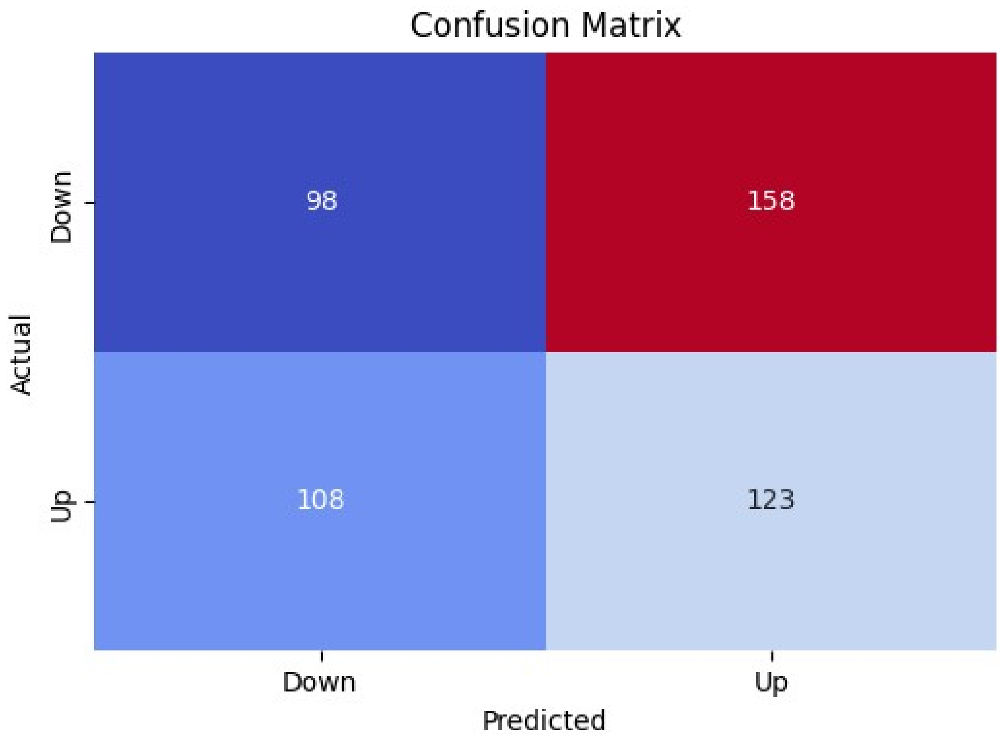
Logistic Regression Model

```python
# Ensure 'Daily Return' is computed correctly
if 'Daily Return' not in history.columns:
    history['Daily Return'] = history['Close'].pct_change()

# Add more technical indicators
history['Momentum'] = history['Close'] - history['Close'].shift(4)
history['Volatility'] = history['Daily
Return'].rolling(window=5).std()
history['SMA_10'] = history['Close'].rolling(window=10).mean()

# Ensure SMA_50 and SMA_200 are computed
history['SMA_50'] = history['Close'].rolling(window=50).mean()
history['SMA_200'] = history['Close'].rolling(window=200).mean()
```

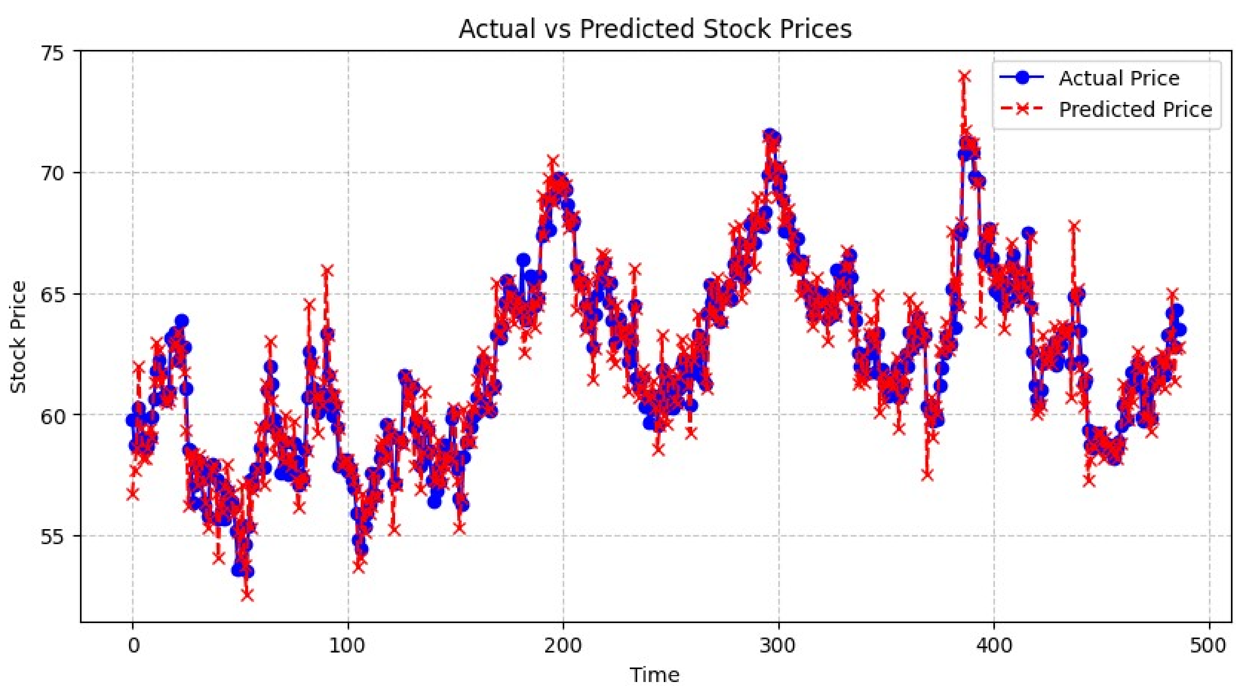
```
# Logistic Regression Model for Predicting Stock Movement
print("\nLogistic Regression Model \n")
history['Target'] = (history['Close'].shift(-1) >
history['Close']).astype(int)

features = ['SMA_50', 'SMA_200', 'Daily Return', 'Momentum',
'Volatility', 'SMA_10']

# Drop rows where any feature or target is NaN
history.dropna(subset=features + ['Target'], inplace=True)

X = history[features]
y = history['Target']

# Balance dataset using SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_resampled)
X_train, X_test, y_train, y_test = train_test_split(X_scaled,
y_resampled, test_size=0.2, random_state=42)

# Hyperparameter tuning with GridSearchCV
param_grid = {'C': [0.01, 0.1, 1, 10, 100]}
grid_search = GridSearchCV(LogisticRegression(), param_grid, cv=5,
scoring='accuracy')
grid_search.fit(X_train, y_train)

model = grid_search.best_estimator_
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Map predicted movements to actual prices
predicted_prices = history['Close'].iloc[-len(y_test):].values * (1 +
(y_pred * 2 - 1) * history['Daily Return'].iloc[-len(y_test):].values)

print("Best Hyperparameters:", grid_search.best_params_)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n\n", classification_report(y_test,
y_pred))

print(f"\nConfusion Matrix :\n")
plt.figure(figsize=(6,4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d',
cmap='coolwarm', cbar=False, xticklabels=['Down', 'Up'],
yticklabels=['Down', 'Up'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

```python
plt.title("Confusion Matrix")
plt.show()

print(f"\n{stock_ticker} Price Prediction using Logistic Regression :\
n")
plt.figure(figsize=(10,5))
plt.plot(history['Close'].iloc[-len(y_test):].values, label="Actual
Price", color="blue", marker='o', linestyle='-')
plt.plot(predicted_prices, label="Predicted Price", color="red",
marker='x', linestyle='--')
plt.xlabel("Time")
plt.ylabel("Stock Price")
plt.legend()
plt.title("Actual vs Predicted Stock Prices")
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()

print("\nConclusion :\n")
if y_pred[-1] == 1:
    print(f"Buy Recommendation: The model predicts an upward trend for
{stock_ticker}.\n")
    logistic_regression_model = "Buy"
else:
    print(f"Sell Recommendation: The model predicts a downward trend
for {stock_ticker}.\n")
    logistic_regression_model = "Sell"
```

```
Logistic Regression Model

Best Hyperparameters: {'C': 100}
Accuracy: 0.4537987679671458

Classification Report:

              precision    recall  f1-score   support

           0       0.48      0.38      0.42       256
           1       0.44      0.53      0.48       231

    accuracy                           0.45       487
   macro avg       0.46      0.46      0.45       487
weighted avg       0.46      0.45      0.45       487


Confusion Matrix :
```




```
RIO Price Prediction using Logistic Regression :
```



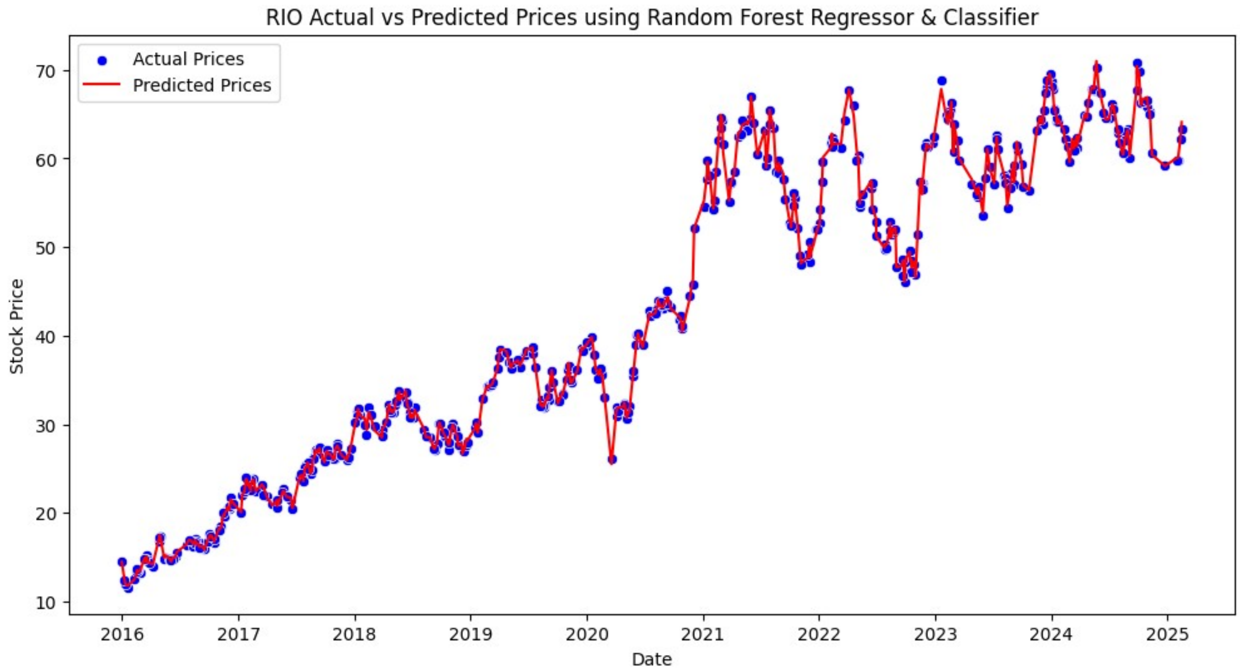
```
Conclusion :

Buy Recommendation: The model predicts an upward trend for RIO.
```

Random Forest Regressor & Classifier

```
# Fetch historical stock data
def fetch_stock_data(stock_ticker):
    ticker = yf.Ticker(stock_ticker)
    history = ticker.history(period="10y")
    return history

# Get stock ticker from user
history = fetch_stock_data(stock_ticker)

# Prepare data for prediction
history['Returns'] = history['Close'].pct_change()
history['SMA_50'] = history['Close'].rolling(window=50).mean()
history['SMA_200'] = history['Close'].rolling(window=200).mean()
history.dropna(inplace=True)

# Feature selection and target variable
features = ['SMA_50', 'SMA_200', 'Open', 'High', 'Low', 'Volume']
X = history[features]
y_reg = history['Close']  # For regression
y_clf = (history['Returns'] > 0).astype(int)  # Classification: 1 if
return > 0 else 0

# Split data
X_train, X_test, y_train_reg, y_test_reg = train_test_split(X, y_reg,
test_size=0.2, random_state=42)
X_train_clf, X_test_clf, y_train_clf, y_test_clf = train_test_split(X,
y_clf, test_size=0.2, random_state=42)

# Train Random Forest Regressor
regressor = RandomForestRegressor(n_estimators=100, random_state=42)
regressor.fit(X_train, y_train_reg)
y_pred_reg = regressor.predict(X_test)

# Train Random Forest Classifier
classifier = RandomForestClassifier(n_estimators=100, random_state=42)
classifier.fit(X_train_clf, y_train_clf)
y_pred_clf = classifier.predict(X_test_clf)

# Performance evaluation
print(f"\n{stock_ticker} Price Prediction using Random Forest
```

```
Regressor & Classifier :")
print("\nRegression Model Performance :\n")
print(f"Mean Absolute Error: {mean_absolute_error(y_test_reg,
y_pred_reg)}")
print(f"Mean Squared Error: {mean_squared_error(y_test_reg,
y_pred_reg)}")

print("\nClassification Model Performance :\n")
print(f"Accuracy: {accuracy_score(y_test_clf, y_pred_clf)}\n")

print(f"{stock_ticker} Actual vs Predicted Prices using Random Forest
Regressor & Classifier :\n")
# Improved graph with scatter plot and trend lines
plt.figure(figsize=(12,6))
sns.scatterplot(x=y_test_reg.index, y=y_test_reg, label='Actual
Prices', color='blue')
sns.lineplot(x=y_test_reg.index, y=y_pred_reg, label='Predicted
Prices', color='red')
plt.title(f"{stock_ticker} Actual vs Predicted Prices using Random
Forest Regressor & Classifier")
plt.xlabel("Date")
plt.ylabel("Stock Price")
plt.legend()
plt.show()

# Conclusion
print("\nConclusion :\n")
if y_pred_clf[-1] == 1:
    print(f"Buy Signal: {stock_ticker} is predicted to go up.\n")
    random_forest = "Buy"
else:
    print(f"Sell Signal: {stock_ticker} is predicted to go down.\n")
    random_forest = "Sell"
```

```
RIO Price Prediction using Random Forest Regressor & Classifier :

Regression Model Performance :

Mean Absolute Error: 0.21172219093503633
Mean Squared Error: 0.08887298514180927

Classification Model Performance :

Accuracy: 0.5107758620689655

RIO Actual vs Predicted Prices using Random Forest Regressor &
Classifier :
```




```
Conclusion :

Sell Signal: RIO is predicted to go down.
```

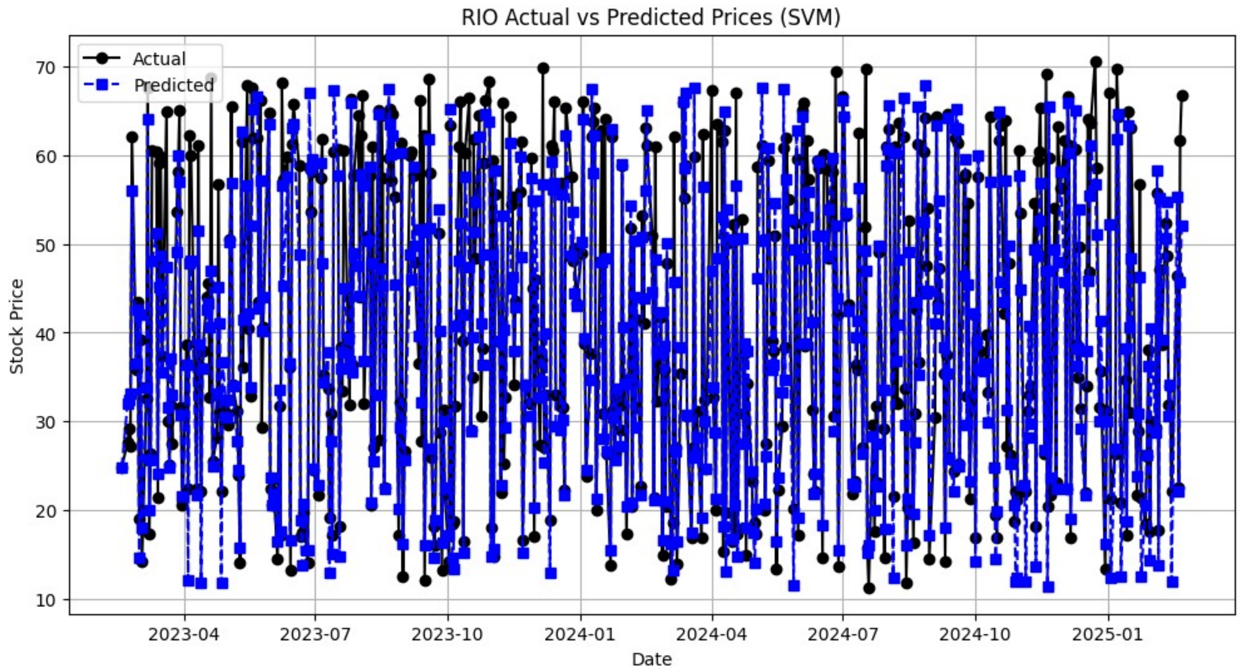
Support Vector Machine (SVM)

```
# Fetch 10-year historical data
history = yf.download(stock_ticker, period="10y")

# Prepare data for SVM
history.dropna(inplace=True)
history['Date'] = history.index
history['Date'] = history['Date'].map(pd.Timestamp.toordinal)

X = history[['Date']]
y = history['Close']

# Normalize data
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)
```

```
# Train SVM model
svm_model = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=0.1)
svm_model.fit(X_train, y_train)

# Predictions
y_pred = svm_model.predict(X_test)

# Plot actual vs predicted
print(f"\n{stock_ticker} Actual vs Predicted Prices using SVM :\n")
plt.figure(figsize=(12,6))
plt.plot(history.index[-len(y_test):], y_test.values, label='Actual',
color='black', marker='o')
plt.plot(history.index[-len(y_test):], y_pred, label='Predicted',
color='blue', linestyle='dashed', marker='s')
plt.title(f"{stock_ticker} Actual vs Predicted Prices (SVM)")
plt.xlabel("Date")
plt.ylabel("Stock Price")
plt.legend()
plt.grid(True)
plt.show()

# Evaluation
print("\nConclusion :\n")
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared Score: {r2}\n")

# Conclusion: Buy or Sell
latest_predicted =
svm_model.predict(scaler.transform([[history['Date'].iloc[-
1]]])).item()
latest_actual = history['Close'].iloc[-1].item()

if latest_predicted > latest_actual:
    print(f"Prediction: Price is expected to rise. Consider BUYING
{stock_ticker}.")
    svm_model = "Buy"
else:
    print(f"Prediction: Price is expected to drop. Consider SELLING
{stock_ticker}.")
    svm_model = "Sell"
```

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

YF.download() has changed argument auto_adjust default to True


/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:14
08: DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples, ), for
```

```
example using ravel().
  y = column_or_1d(y, warn=True)


RIO Actual vs Predicted Prices using SVM :
```




```
Conclusion :

Mean Squared Error: 41.64553485597158
R-squared Score: 0.8680586910272767

Prediction: Price is expected to rise. Consider BUYING RIO.
```

XGBoost & LightGBM

```
# Fetch 10-year historical data
data = history[['Close']].dropna()
data = data[-2520:]  # Approx. 252 trading days per year * 10 years
data['Returns'] = data['Close'].pct_change()
data.dropna(inplace=True)
data['Future_Close'] = data['Close'].shift(-1)  # Renamed to avoid
issues
data.dropna(inplace=True)

X = data[['Close', 'Returns']]
```

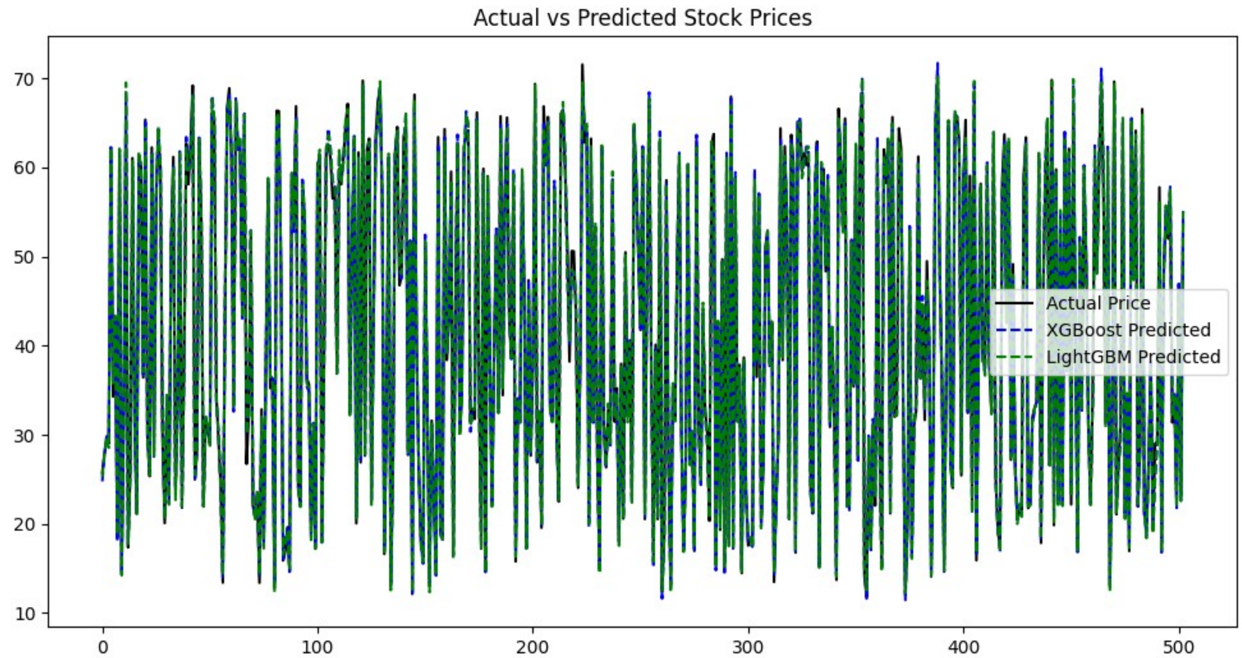
```python
y = data['Future_Close']

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

# Convert MultiIndex columns to a standard Index
if isinstance(X_train.columns, pd.MultiIndex):
    X_train.columns = ['_'.join(col).strip() for col in
X_train.columns]
    X_test.columns = ['_'.join(col).strip() for col in X_test.columns]

# Rename columns to remove special characters
X_train.columns = [re.sub(r'[^a-zA-Z0-9_]', '', col) for col in
X_train.columns]
X_test.columns = [re.sub(r'[^a-zA-Z0-9_]', '', col) for col in
X_test.columns]

# XGBoost Model
xgb_model = xgb.XGBRegressor(objective='reg:squarederror',
n_estimators=100)
xgb_model.fit(X_train, y_train)
xgb_preds = xgb_model.predict(X_test)

xgb_rmse = np.sqrt(mean_squared_error(y_test, xgb_preds))
xgb_r2 = r2_score(y_test, xgb_preds)
print("\nXGBoost Model :\n")
print(f"XGBoost RMSE: {xgb_rmse}")
print(f"XGBoost R2 Score: {xgb_r2}\n")

# LightGBM Model
lgb_model = lgb.LGBMRegressor(n_estimators=100)
lgb_model.fit(X_train, y_train)
lgb_preds = lgb_model.predict(X_test)

lgb_rmse = np.sqrt(mean_squared_error(y_test, lgb_preds))
lgb_r2 = r2_score(y_test, lgb_preds)
print("\nLightGBM Model :\n")
print(f"LightGBM RMSE: {lgb_rmse}")
print(f"LightGBM R2 Score: {lgb_r2}\n")

# Price Graph Over 10 Years
print(f"\n{stock_ticker} Price Trend Over 10 Years :\n")
plt.figure(figsize=(12,6))
plt.plot(data.index, data['Close'], label="Stock Price (10 years)",
color='black')
plt.title(f"{stock_ticker} Price Trend Over 10 Years")
plt.xlabel("Date")
plt.ylabel("Close Price")
plt.legend()
plt.show()
```

```python
# Graph Comparing Actual vs Predicted Prices
print(f"\n{stock_ticker} Actual vs Predicted Prices :\n")
plt.figure(figsize=(12,6))
plt.plot(y_test.values, label="Actual Price", color='black')
plt.plot(xgb_preds, label="XGBoost Predicted", linestyle='dashed',
color='blue')
plt.plot(lgb_preds, label="LightGBM Predicted", linestyle='dashed',
color='green')
plt.title("Actual vs Predicted Stock Prices")
plt.legend()
plt.show()

# Conclusion on Buy or Sell Decision
latest_close = history['Close'].iloc[-1].item()  # Ensure scalar value
xgb_latest_pred = xgb_preds[-1]
lgb_latest_pred = lgb_preds[-1]

print("\nConclusion :\n")
if xgb_latest_pred > latest_close and lgb_latest_pred > latest_close:
    print(f"Both models predict an upward trend. Suggested action: BUY
{stock_ticker}.")
    xgboost_lightgbm_model = "Buy"
elif xgb_latest_pred < latest_close and lgb_latest_pred <
latest_close:
    print(f"Both models predict a downward trend. Suggested action:
SELL {stock_ticker}.")
    xgboost_lightgbm_model = "Sell"
else:
    print(f"Models have mixed predictions. Suggested action: HOLD
{stock_ticker}, observe further.")
    xgboost_lightgbm_model = "Hold"
```

```
XGBoost Model :

XGBoost RMSE: 0.9516588623027524
XGBoost R2 Score: 0.9971960524812682

[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead
of testing was 0.000440 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 510
[LightGBM] [Info] Number of data points in the train set: 2011, number
of used features: 2
[LightGBM] [Info] Start training from score 40.847535

LightGBM Model :

LightGBM RMSE: 0.8861747413413361
```

```
LightGBM R2 Score: 0.9975686581674562

RIO Price Trend Over 10 Years :
```


```
RIO Actual vs Predicted Prices :
```



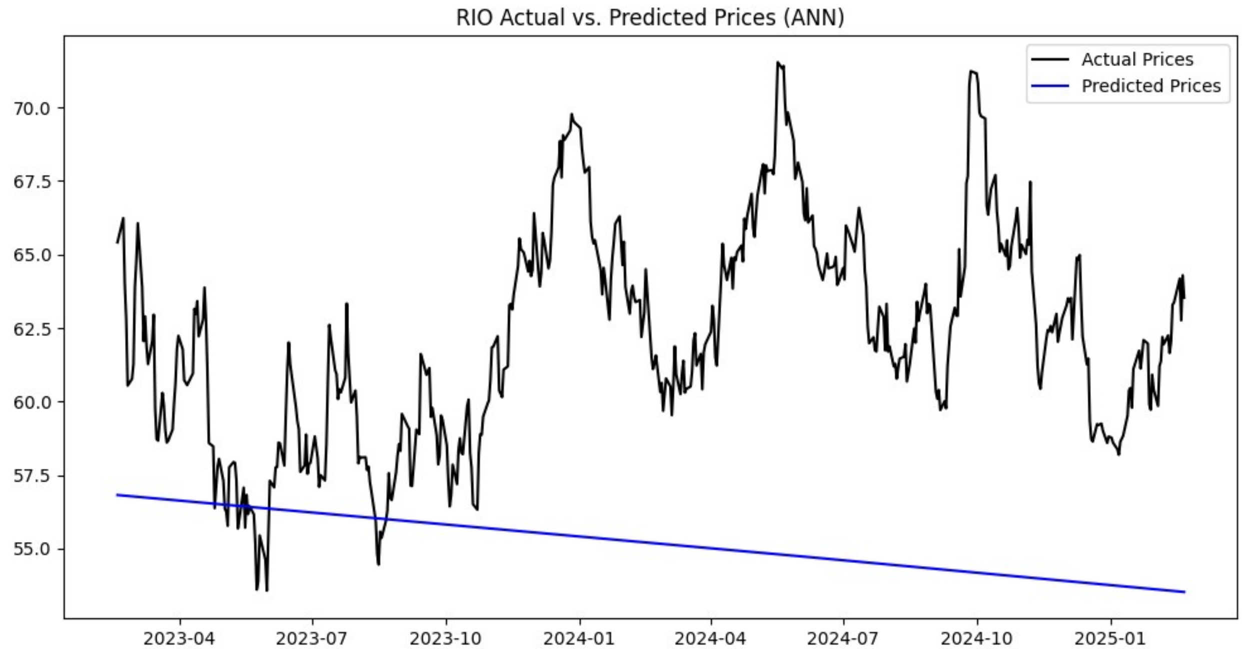

```
Conclusion :

Both models predict a downward trend. Suggested action: SELL RIO.
```

Artificial Neural Networks (ANN)

```
# Fetch 10 years of data
history = ticker.history(period="10y")
history['Date'] = history.index
history['Ordinal Date'] = history['Date'].map(pd.Timestamp.toordinal)
features = ['Ordinal Date']
X = history[features].values
Y = history['Close'].values

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
Y_scaled = scaler.fit_transform(Y.reshape(-1, 1))

X_train, X_test, Y_train, Y_test = train_test_split(X_scaled,
Y_scaled, test_size=0.2, shuffle=False)

# Define ANN Model
model = keras.Sequential([
    keras.layers.Dense(64, activation='relu',
input_shape=(X_train.shape[1],)),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(1)
```

```
])

model.compile(optimizer='adam', loss='mse')
model.fit(X_train, Y_train, epochs=100, batch_size=16, verbose=0)

# Predictions
Y_pred = model.predict(X_test)
Y_pred_rescaled = scaler.inverse_transform(Y_pred)
Y_test_rescaled = scaler.inverse_transform(Y_test)

# Plot Actual vs. Predicted Prices
print(f"\n{stock_ticker} Actual vs. Predicted Prices (ANN) :\n")
plt.figure(figsize=(12,6))
plt.plot(history.index[-len(Y_test):], Y_test_rescaled, label='Actual
Prices', color='black')
plt.plot(history.index[-len(Y_pred):], Y_pred_rescaled,
label='Predicted Prices', color='blue')
plt.title(f"{stock_ticker} Actual vs. Predicted Prices (ANN)")
plt.legend()
plt.show()

# Buy/Sell Conclusion
print("\nConclusion :\n")
if Y_pred_rescaled[-1] > Y_test_rescaled[-1]:
    print(f"BUY Signal: {stock_ticker}'s price is predicted to
increase.")
    ann_model = "Buy"

else:
    print(f"SELL Signal: {stock_ticker}'s price is predicted to
decrease.")
    ann_model = "Sell"
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/
dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an
`Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step

RIO Actual vs. Predicted Prices (ANN) :
```




```
Conclusion :

SELL Signal: RIO's price is predicted to decrease.
```

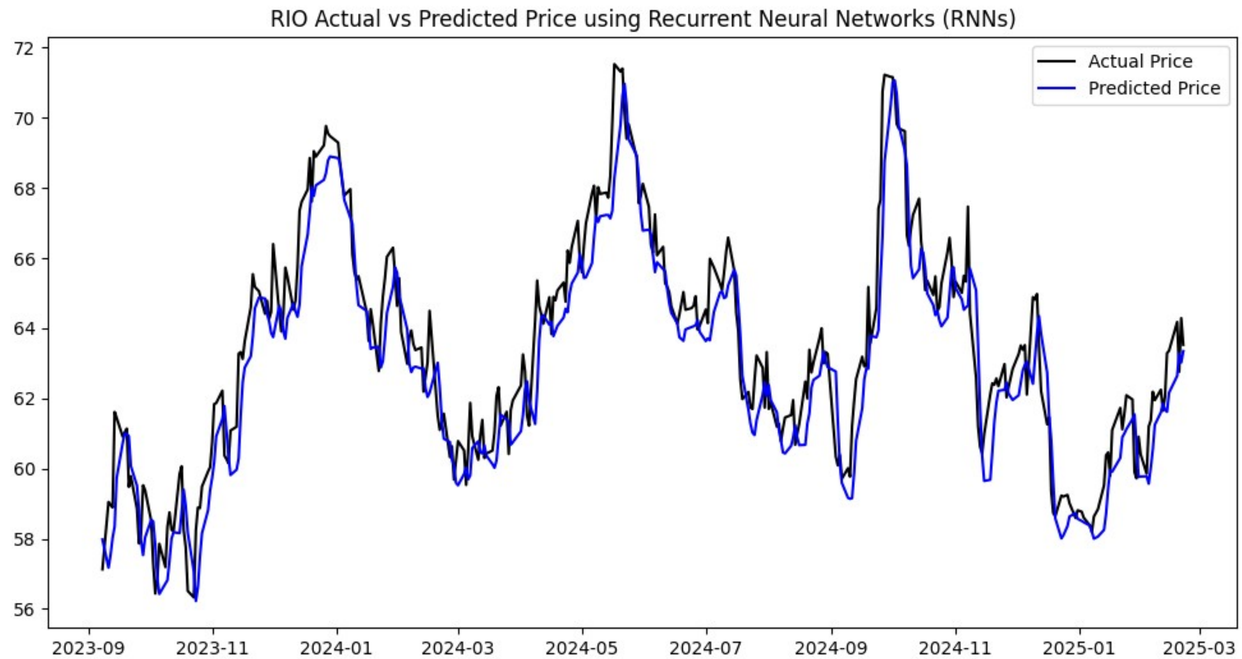
Recurrent Neural Networks (RNN)

```python
# Prepare data for RNN
scaler = MinMaxScaler(feature_range=(0,1))
data = history[['Close']].copy()
data_scaled = scaler.fit_transform(data)

# Create sequences
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i+seq_length])
    return np.array(X), np.array(y)

seq_length = 730  # Use last 2 years (730 days) to predict the next
day
X, y = create_sequences(data_scaled, seq_length)
X_train, y_train = X[:-365], y[:-365]
X_test, y_test = X[-365:], y[-365:]

# Build RNN Model
model = Sequential([
```

```
    LSTM(units=50, return_sequences=True, input_shape=(seq_length,
1)),
    LSTM(units=50, return_sequences=False),
    Dense(units=25),
    Dense(units=1)
])

model.compile(optimizer='adam', loss='mean_squared_error')

# Train Model
model.fit(X_train, y_train, batch_size=16, epochs=20)

# Predict
y_pred = model.predict(X_test)
y_pred_rescaled = scaler.inverse_transform(y_pred)
y_test_rescaled = scaler.inverse_transform(y_test)

# Plot results
print(f"\n\n{stock_ticker} : Actual vs Predicted Price using Recurrent
Neural Networks (RNNs) :\n")
plt.figure(figsize=(12,6))
plt.plot(data.index[-365:], y_test_rescaled, label='Actual Price',
color='black')
plt.plot(data.index[-365:], y_pred_rescaled, label='Predicted Price',
color='blue')
plt.title(f"{stock_ticker} Actual vs Predicted Price using Recurrent
Neural Networks (RNNs)")
plt.legend()
plt.show()

# Conclusion
last_predicted = y_pred_rescaled[-1][0]
last_actual = y_test_rescaled[-1][0]
print("\nConclusion :\n")
print(f"Last Actual Price: {last_actual}")
print(f"Last Predicted Price: {last_predicted}")

if last_predicted > last_actual:
    print(f"Prediction suggests an upward trend. Consider BUYING
{stock_ticker}.\n")
    rnn_model = "Buy"
elif last_predicted < last_actual:
    print(f"Prediction suggests a downward trend. Consider SELLING
{stock_ticker}.\n")
    rnn_model = "Sell"
else:
    print(f"Prediction suggests stability. Hold your position in
{stock_ticker}.\n")
    rnn_model = "Hold"
```

```
Epoch 1/20

/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/
rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an
`Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)

89/89 ━━━━━━━━━━━━━━━━━━━━ 65s 676ms/step - loss: 0.0463
Epoch 2/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 94s 810ms/step - loss: 0.0013
Epoch 3/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 58s 544ms/step - loss: 0.0014
Epoch 4/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 50s 558ms/step - loss: 0.0010
Epoch 5/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 48s 545ms/step - loss: 0.0010
Epoch 6/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 81s 539ms/step - loss: 0.0010
Epoch 7/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 83s 546ms/step - loss: 0.0011
Epoch 8/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 49s 549ms/step - loss: 9.2398e-04
Epoch 9/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 82s 553ms/step - loss: 8.1301e-04
Epoch 10/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 82s 558ms/step - loss: 6.9935e-04
Epoch 11/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 81s 553ms/step - loss: 7.2962e-04
Epoch 12/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 49s 548ms/step - loss: 6.0973e-04
Epoch 13/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 82s 541ms/step - loss: 5.6734e-04
Epoch 14/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 48s 545ms/step - loss: 5.3742e-04
Epoch 15/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 50s 557ms/step - loss: 5.3371e-04
Epoch 16/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 81s 548ms/step - loss: 6.0307e-04
Epoch 17/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 82s 552ms/step - loss: 5.4923e-04
Epoch 18/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 82s 557ms/step - loss: 4.5648e-04
Epoch 19/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 80s 537ms/step - loss: 4.2927e-04
Epoch 20/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 83s 548ms/step - loss: 4.0619e-04
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 168ms/step
```

```
RIO : Actual vs Predicted Price using Recurrent Neural Networks (RNNs)
:
```




```
Conclusion :

Last Actual Price: 63.529998779296875
Last Predicted Price: 63.34847640991211
Prediction suggests a downward trend. Consider SELLING RIO.
```

Long Short-Term Memory (LSTM)

```
# Fetch data from previous analysis
data = history[['Close']]
scaler = MinMaxScaler(feature_range=(0,1))
data_scaled = scaler.fit_transform(data)

# Prepare data for LSTM model
def create_sequences(data, seq_length):
    sequences = []
    labels = []
    for i in range(len(data) - seq_length):
        sequences.append(data[i:i + seq_length])
        labels.append(data[i + seq_length])
    return np.array(sequences), np.array(labels)
```

```
seq_length = 50  # Higher number for higher accuracy
X, y = create_sequences(data_scaled, seq_length)
X_train, X_test = X[:int(0.8 * len(X))], X[int(0.8 * len(X)):]
y_train, y_test = y[:int(0.8 * len(y))], y[int(0.8 * len(y)):]

# Build LSTM model
model = Sequential([
    LSTM(units=50, return_sequences=True, input_shape=(seq_length,
1)),
    Dropout(0.2),
    LSTM(units=50, return_sequences=False),
    Dropout(0.2),
    Dense(units=25),
    Dense(units=1)
])

model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=16,
validation_data=(X_test, y_test))

# Make predictions
y_pred = model.predict(X_test)
y_pred_inv = scaler.inverse_transform(y_pred)
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))

# Plot actual vs predicted
print(f"\n\n{stock_ticker} Price Prediction using Long Short-Term
Memory (LSTM) :\n")
plt.figure(figsize=(12,6))
plt.plot(y_test_inv, label='Actual Price', color='blue')
plt.plot(y_pred_inv, label='Predicted Price', color='red')
plt.title(f"{stock_ticker} - Actual vs Predicted Price using Long
Short-Term Memory (LSTM)")
plt.legend()
plt.show()

# Conclusion
latest_pred = y_pred_inv[-1][0]
latest_actual = y_test_inv[-1][0]
print("\nConclusion :\n")
if latest_pred > latest_actual:
    print(f"Buy Signal: {stock_ticker}'s predicted price
({latest_pred:.2f}) is higher than its current price
({latest_actual:.2f}).\n")
    lstm_model = "Buy"
else:
    print(f"Sell Signal: {stock_ticker}'s predicted price
```

```
({latest_pred:.2f}) is lower than its current price
({latest_actual:.2f}).\n")
    lstm_model = "Sell"
```

```
Epoch 1/50
```

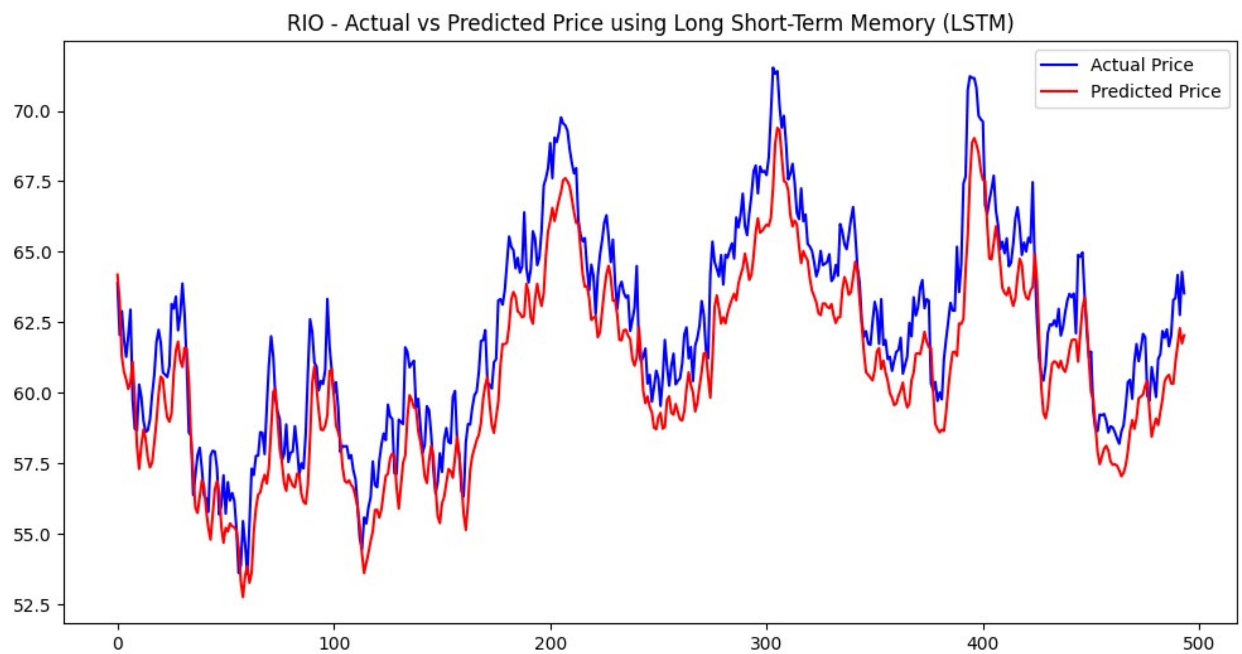
```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/
rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an
`Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

```
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 49ms/step - loss: 0.0352 - val_loss:
0.0018
Epoch 2/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 46ms/step - loss: 0.0030 - val_loss:
0.0036
Epoch 3/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 9s 38ms/step - loss: 0.0031 - val_loss:
0.0015
Epoch 4/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 45ms/step - loss: 0.0020 - val_loss:
0.0012
Epoch 5/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 47ms/step - loss: 0.0018 - val_loss:
0.0010
Epoch 6/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 38ms/step - loss: 0.0018 - val_loss:
8.5525e-04
Epoch 7/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 39ms/step - loss: 0.0015 - val_loss:
0.0031
Epoch 8/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 45ms/step - loss: 0.0015 - val_loss:
0.0014
Epoch 9/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 38ms/step - loss: 0.0015 - val_loss:
9.3028e-04
Epoch 10/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 46ms/step - loss: 0.0013 - val_loss:
6.5709e-04
Epoch 11/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 9s 40ms/step - loss: 0.0012 - val_loss:
6.6365e-04
Epoch 12/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 45ms/step - loss: 0.0014 - val_loss:
6.5637e-04
Epoch 13/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 47ms/step - loss: 0.0011 - val_loss:
6.7125e-04
```

```
Epoch 14/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 37ms/step - loss: 0.0012 - val_loss:
5.9944e-04
Epoch 15/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 41ms/step - loss: 0.0011 - val_loss:
5.8022e-04
Epoch 16/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 42ms/step - loss: 0.0011 - val_loss:
7.8495e-04
Epoch 17/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 37ms/step - loss: 0.0011 - val_loss:
8.4448e-04
Epoch 18/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 47ms/step - loss: 9.6244e-04 -
val_loss: 0.0013
Epoch 19/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 9s 39ms/step - loss: 0.0010 - val_loss:
5.5853e-04
Epoch 20/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 43ms/step - loss: 0.0010 - val_loss:
6.3978e-04
Epoch 21/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 40ms/step - loss: 9.8483e-04 -
val_loss: 6.3507e-04
Epoch 22/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 50ms/step - loss: 9.6077e-04 -
val_loss: 9.9764e-04
Epoch 23/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 9s 38ms/step - loss: 9.3362e-04 -
val_loss: 0.0013
Epoch 24/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 44ms/step - loss: 0.0011 - val_loss:
5.3432e-04
Epoch 25/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 37ms/step - loss: 9.1689e-04 -
val_loss: 4.2190e-04
Epoch 26/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 49ms/step - loss: 8.8602e-04 -
val_loss: 8.5222e-04
Epoch 27/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 9s 37ms/step - loss: 0.0010 - val_loss:
7.4540e-04
Epoch 28/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 44ms/step - loss: 9.7136e-04 -
val_loss: 0.0012
Epoch 29/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 46ms/step - loss: 8.7734e-04 -
val_loss: 6.1902e-04
Epoch 30/50
```

```
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 37ms/step - loss: 9.6597e-04 -
val_loss: 4.0623e-04
Epoch 31/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 40ms/step - loss: 8.7207e-04 -
val_loss: 4.4030e-04
Epoch 32/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 43ms/step - loss: 9.1195e-04 -
val_loss: 4.0814e-04
Epoch 33/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 11s 47ms/step - loss: 0.0010 - val_loss:
4.1644e-04
Epoch 34/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 38ms/step - loss: 7.8846e-04 -
val_loss: 3.5725e-04
Epoch 35/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 43ms/step - loss: 8.8976e-04 -
val_loss: 3.6081e-04
Epoch 36/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 9s 37ms/step - loss: 9.1628e-04 -
val_loss: 4.8396e-04
Epoch 37/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 47ms/step - loss: 8.7129e-04 -
val_loss: 3.5283e-04
Epoch 38/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 44ms/step - loss: 8.3814e-04 -
val_loss: 7.3448e-04
Epoch 39/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 9s 37ms/step - loss: 9.5527e-04 -
val_loss: 3.7980e-04
Epoch 40/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 46ms/step - loss: 8.6275e-04 -
val_loss: 6.9427e-04
Epoch 41/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 37ms/step - loss: 8.3474e-04 -
val_loss: 4.2921e-04
Epoch 42/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 45ms/step - loss: 8.2912e-04 -
val_loss: 3.3031e-04
Epoch 43/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 9s 37ms/step - loss: 7.3417e-04 -
val_loss: 4.2625e-04
Epoch 44/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 47ms/step - loss: 9.7387e-04 -
val_loss: 4.0028e-04
Epoch 45/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 46ms/step - loss: 8.4156e-04 -
val_loss: 5.9916e-04
Epoch 46/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 9s 37ms/step - loss: 7.8755e-04 -
```

```
val_loss: 3.1676e-04
Epoch 47/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 47ms/step - loss: 8.7023e-04 -
val_loss: 3.3422e-04
Epoch 48/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 47ms/step - loss: 0.0010 - val_loss:
4.8654e-04
Epoch 49/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 37ms/step - loss: 8.8504e-04 -
val_loss: 6.3296e-04
Epoch 50/50
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 38ms/step - loss: 8.9217e-04 -
val_loss: 9.2946e-04
16/16 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step


RIO Price Prediction using Long Short-Term Memory (LSTM) :
```




```
Conclusion :

Sell Signal: RIO's predicted price (62.04) is lower than its current
price (63.53).
```

Gated Recurrent Units (GRU)

```
time_steps = 50  # Number of time steps to look back

def create_sequences(data, time_steps):
    sequences, labels = [], []
    for i in range(len(data) - time_steps):
        sequences.append(data[i:i + time_steps])
        labels.append(data[i + time_steps])
    return np.array(sequences), np.array(labels)

# Prepare data for GRU model
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(history[['Close']])
X, y = create_sequences(scaled_data, time_steps)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

# Build GRU Model
model = Sequential([
    GRU(units=50, return_sequences=True, input_shape=(time_steps, 1)),
    Dropout(0.2),
    GRU(units=50, return_sequences=False),
    Dropout(0.2),
    Dense(units=25),
    Dense(units=1)
])

model.compile(optimizer='adam', loss='mean_squared_error')

# Train model
model.fit(X_train, y_train, epochs=20, batch_size=16,
validation_data=(X_test, y_test))

# Make predictions
y_pred = model.predict(X_test)
y_pred_inv = scaler.inverse_transform(y_pred.reshape(-1, 1))
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))

# Plot actual vs predicted values
print(f"\n\n{stock_ticker} Price Prediction using Gated Recurrent
Units (GRU) :\n")
plt.figure(figsize=(12, 6))
plt.plot(y_test_inv, label='Actual Price', color='blue')
plt.plot(y_pred_inv, label='Predicted Price', color='red',
linestyle='dashed')
plt.title(f'{stock_ticker} Stock Price Prediction using Gated
Recurrent Units (GRU)')
plt.legend()
plt.show()
```

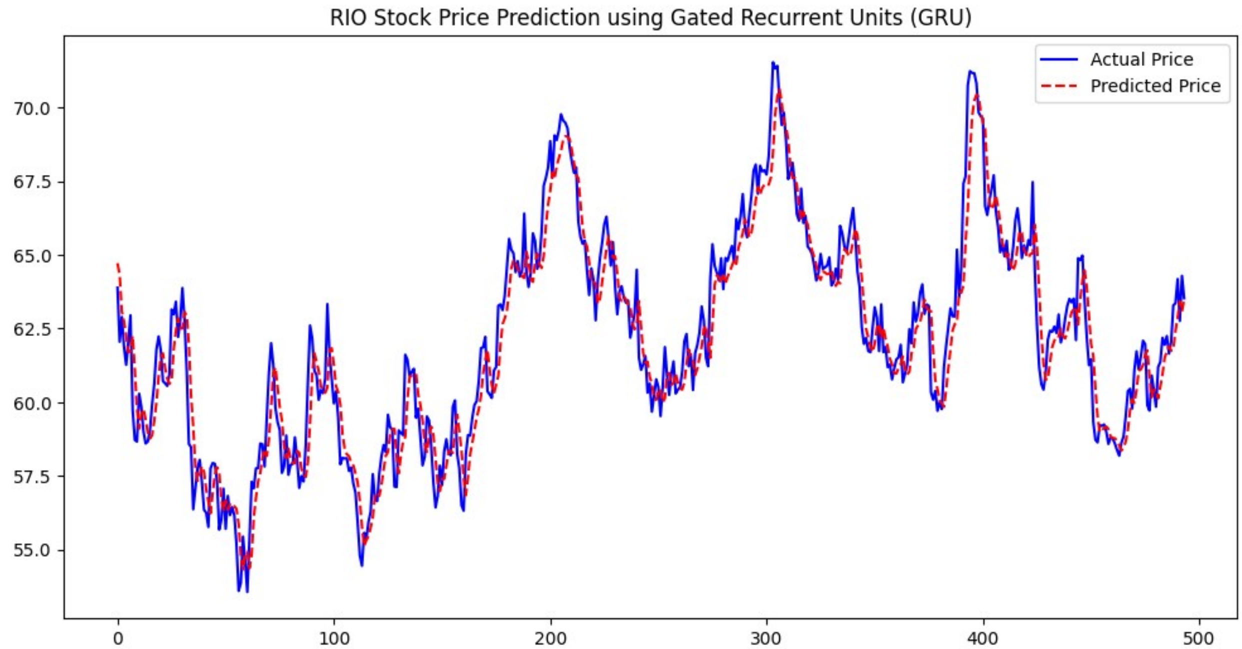
```python
# Conclusion
last_actual = y_test_inv[-1][0]
last_predicted = y_pred_inv[-1][0]
price_change = ((last_predicted - last_actual) / last_actual) * 100

print("\nConclusion :\n")
if price_change > 1:
    print(f"Predicted price increase: {price_change:.2f}%. Suggested
action: BUY")
    gru_model = "Buy"
elif price_change < -1:
    print(f"Predicted price decrease: {price_change:.2f}%. Suggested
action: SELL")
    gru_model = "Sell"
else:
    print(f"Minor price change ({price_change:.2f}%). Suggested
action: HOLD")
    gru_model = "Hold"
```

```
Epoch 1/20
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/
rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an
`Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

```
124/124 ━━━━━━━━━━━━━━━━━━━━ 12s 56ms/step - loss: 0.0375 - val_loss:
0.0028
Epoch 2/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 50ms/step - loss: 0.0027 - val_loss:
7.4166e-04
Epoch 3/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 11s 58ms/step - loss: 0.0028 - val_loss:
0.0030
Epoch 4/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 56ms/step - loss: 0.0016 - val_loss:
0.0012
Epoch 5/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 9s 47ms/step - loss: 0.0018 - val_loss:
7.0656e-04
Epoch 6/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 47ms/step - loss: 0.0014 - val_loss:
4.8225e-04
Epoch 7/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 48ms/step - loss: 0.0014 - val_loss:
0.0010
Epoch 8/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 49ms/step - loss: 0.0012 - val_loss:
```

```
5.3410e-04
Epoch 9/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 7s 54ms/step - loss: 0.0011 - val_loss:
0.0016
Epoch 10/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 49ms/step - loss: 0.0011 - val_loss:
4.2674e-04
Epoch 11/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 11s 56ms/step - loss: 9.2278e-04 -
val_loss: 4.0930e-04
Epoch 12/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 47ms/step - loss: 9.3356e-04 -
val_loss: 6.0204e-04
Epoch 13/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 7s 56ms/step - loss: 8.4006e-04 -
val_loss: 4.2181e-04
Epoch 14/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 48ms/step - loss: 9.9543e-04 -
val_loss: 8.3277e-04
Epoch 15/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 7s 56ms/step - loss: 8.9138e-04 -
val_loss: 3.8068e-04
Epoch 16/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 51ms/step - loss: 9.7232e-04 -
val_loss: 3.7115e-04
Epoch 17/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 7s 54ms/step - loss: 8.4570e-04 -
val_loss: 5.3725e-04
Epoch 18/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 50ms/step - loss: 8.2590e-04 -
val_loss: 9.9947e-04
Epoch 19/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 48ms/step - loss: 9.0698e-04 -
val_loss: 4.9504e-04
Epoch 20/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 48ms/step - loss: 8.9099e-04 -
val_loss: 3.3347e-04
16/16 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step


RIO Price Prediction using Gated Recurrent Units (GRU) :
```




```
Conclusion :

Minor price change (-0.08%). Suggested action: HOLD
```

Transformer Models

```
# Data Preprocessing
scaler = MinMaxScaler()
history['Scaled_Close'] = scaler.fit_transform(history[['Close']])

# Prepare data for Transformer model
def create_sequences(data, seq_length):
    sequences = []
    labels = []
    for i in range(len(data) - seq_length):
        sequences.append(data[i:i+seq_length])
        labels.append(data[i+seq_length])
    return np.array(sequences), np.array(labels)

seq_length = 50
X, y = create_sequences(history['Scaled_Close'].values, seq_length)
X_train, X_test = X[:int(0.8*len(X))], X[int(0.8*len(X)):]
y_train, y_test = y[:int(0.8*len(y))], y[int(0.8*len(y)):]

# Convert to tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.float32)
```

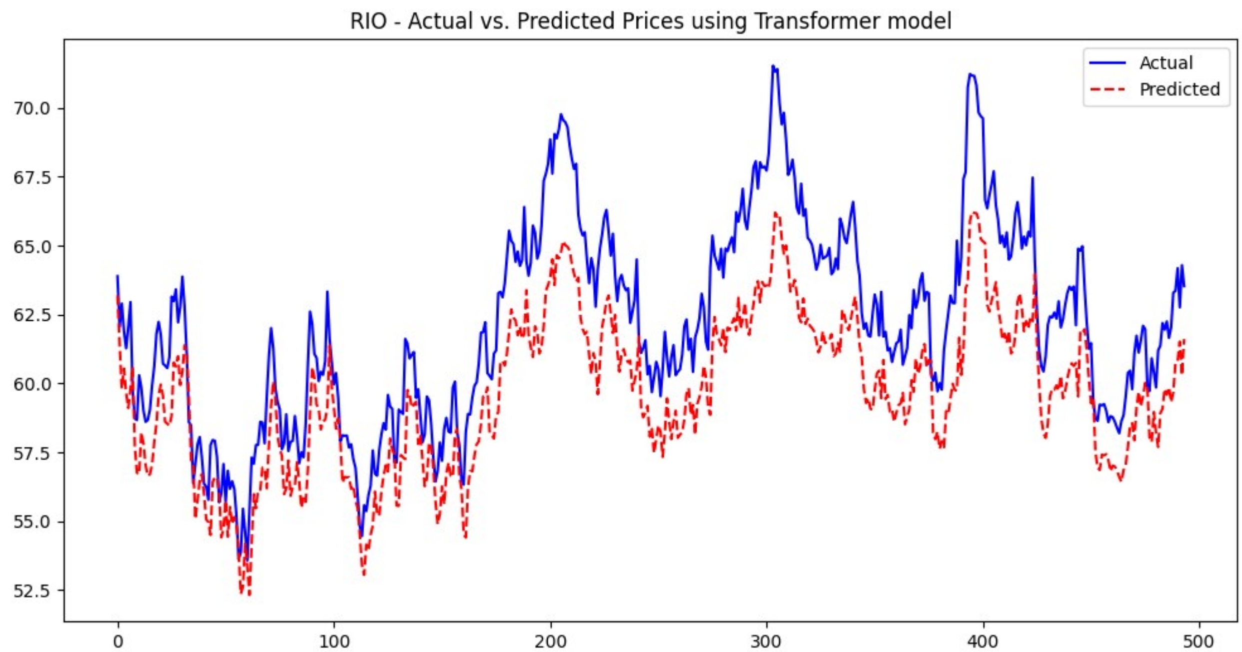
```
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32)

# Create DataLoader
train_dataset = TensorDataset(X_train_tensor, y_train_tensor)
test_dataset = TensorDataset(X_test_tensor, y_test_tensor)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)

# Transformer Model
class TransformerModel(nn.Module):
    def __init__(self, input_dim, embed_dim, num_heads, hidden_dim,
num_layers):
        super(TransformerModel, self).__init__()
        self.embedding = nn.Linear(input_dim, embed_dim)
        self.transformer = nn.TransformerEncoder(
            nn.TransformerEncoderLayer(d_model=embed_dim,
nhead=num_heads, dim_feedforward=hidden_dim),
            num_layers=num_layers
        )
        self.fc = nn.Linear(embed_dim, 1)

    def forward(self, x):
        x = self.embedding(x)
        x = self.transformer(x)
        x = self.fc(x[:, -1, :])
        return x

# Initialize model
model = TransformerModel(input_dim=1, embed_dim=64, num_heads=4,
hidden_dim=128, num_layers=2)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

# Training loop
epochs = 20 # Higher number for higher accuracy
for epoch in range(epochs):
    model.train()
    for X_batch, y_batch in train_loader:
        optimizer.zero_grad()
        y_pred = model(X_batch.unsqueeze(-1))
        loss = criterion(y_pred.squeeze(), y_batch)
        loss.backward()
        optimizer.step()
    print(f"Epoch {epoch+1}/{epochs}, Loss: {loss.item():.4f}")

# Predictions
model.eval()
y_pred_list = []
with torch.no_grad():
```

```
    for X_batch, _ in test_loader:
        y_pred = model(X_batch.unsqueeze(-1))
        y_pred_list.extend(y_pred.squeeze().tolist())

# Inverse scale
y_pred_list = scaler.inverse_transform(np.array(y_pred_list).reshape(-
1, 1))
y_test_actual = scaler.inverse_transform(y_test.reshape(-1, 1))

# Plot Actual vs Predicted
print(f"\n\n{stock_ticker} Price Prediction using Transformer model :\
n")
plt.figure(figsize=(12,6))
plt.plot(y_test_actual, label='Actual', color='blue')
plt.plot(y_pred_list, label='Predicted', color='red',
linestyle='dashed')
plt.title(f"{stock_ticker} - Actual vs. Predicted Prices using
Transformer model")
plt.legend()
plt.show()

# Buy/Sell Recommendation
print("\nConclusion :\n")
if y_pred_list[-1] > y_test_actual[-1]:
    print(f"Predicted price is higher than the current price. Consider
Buying {stock_ticker}.")
    transformer_model = "Buy"
else:
    print(f"Predicted price is lower than the current price. Consider
Selling {stock_ticker}.")
    transformer_model = "Sell"
```

```
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/
transformer.py:379: UserWarning: enable_nested_tensor is True, but
self.use_nested_tensor is False because
encoder_layer.self_attn.batch_first was not True(use batch_first for
better inference performance)
  warnings.warn(
```

```
Epoch 1/20, Loss: 0.0073
Epoch 2/20, Loss: 0.0047
Epoch 3/20, Loss: 0.0034
Epoch 4/20, Loss: 0.0012
Epoch 5/20, Loss: 0.0012
Epoch 6/20, Loss: 0.0011
Epoch 7/20, Loss: 0.0019
Epoch 8/20, Loss: 0.0007
Epoch 9/20, Loss: 0.0012
Epoch 10/20, Loss: 0.0006
Epoch 11/20, Loss: 0.0011
```

```
Epoch 12/20, Loss: 0.0007
Epoch 13/20, Loss: 0.0002
Epoch 14/20, Loss: 0.0023
Epoch 15/20, Loss: 0.0007
Epoch 16/20, Loss: 0.0006
Epoch 17/20, Loss: 0.0006
Epoch 18/20, Loss: 0.0013
Epoch 19/20, Loss: 0.0010
Epoch 20/20, Loss: 0.0007


RIO Price Prediction using Transformer model :
```




```
Conclusion :

Predicted price is lower than the current price. Consider Selling RIO.
```

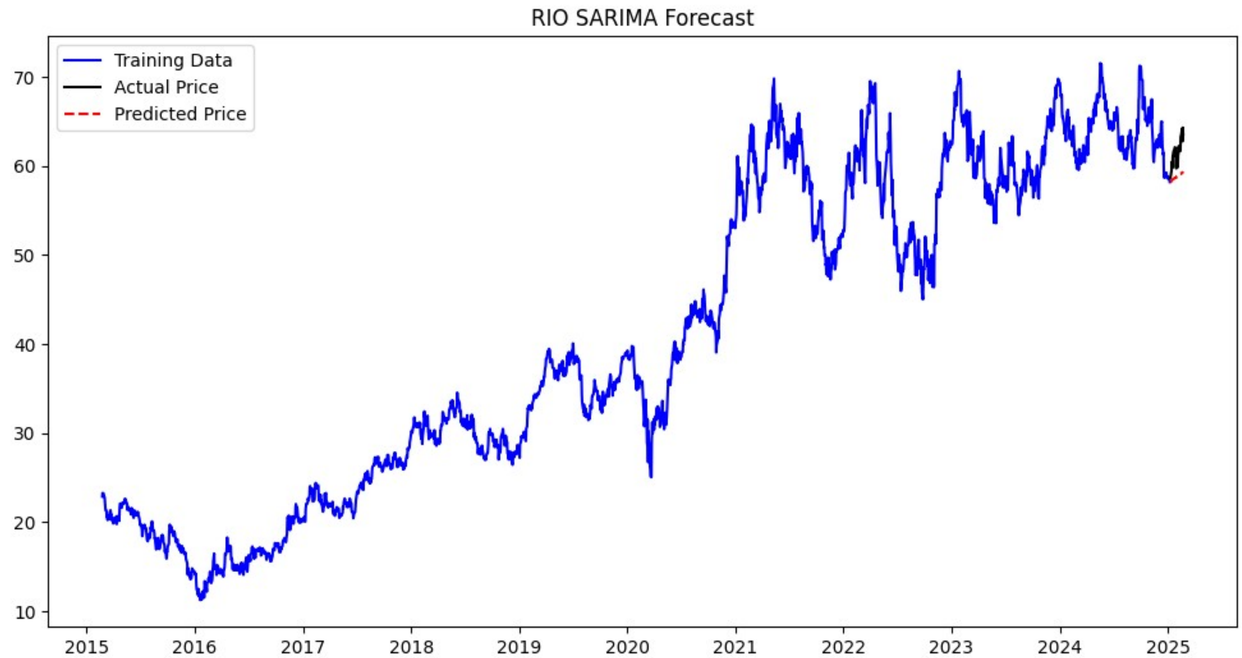
SARIMA Model

```
print("\n SARIMA Model Forecasting :\n")

# Checking stationarity
def check_stationarity(series):
    result = adfuller(series.dropna())
    print("ADF Statistic:", result[0])
```

```
    print("p-value:", result[1])
    if result[1] < 0.05:
        print("The data is stationary.\n")
    else:
        print("The data is not stationary.\n")

# Ensure 'Close' column exists before proceeding
if 'Close' not in history.columns:
    print("\nError: 'Close' column not found in the data. Check the
ticker or data source.\n")
    exit()

check_stationarity(history['Close'])

# Handle missing values
if history['Close'].isna().all():
    print("\nError: 'Close' column has all NaN values. Exiting...\n")
    exit()
history['Close'] = history['Close'].ffill()

# Fit SARIMA model
train = history['Close'][:-30]  # Use all but last 30 days for
training
test = history['Close'][-30:]

model = SARIMAX(train, order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
model_fit = model.fit(disp=False)

# Forecast
forecast = model_fit.predict(start=len(train), end=len(train) +
len(test) - 1, dynamic=False)

# Plot actual vs predicted
print(f"\n\n{stock_ticker} Price Prediction using SARIMA :\n")
plt.figure(figsize=(12,6))
plt.plot(train.index, train, label='Training Data', color='blue')
plt.plot(test.index, test, label='Actual Price', color='black')
plt.plot(test.index, forecast, label='Predicted Price', color='red',
linestyle='dashed')
plt.title(f"{stock_ticker} SARIMA Forecast")
plt.legend()
plt.show()

# Conclusion
print("\nConclusion :\n")
if forecast.iloc[-1] > test.iloc[-1]:
    print(f"Buy Signal: {stock_ticker} is expected to rise based on
SARIMA prediction.")
    sarima_model = "Buy"
else:
```

```
    print(f"Sell Signal: {stock_ticker} is expected to drop based on SARIMA prediction.")
    sarima_model = "Sell"
```

```
 SARIMA Model Forecasting :

ADF Statistic: -1.0954394470615039
p-value: 0.7168871699922149
The data is not stationary.


/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(



RIO Price Prediction using SARIMA :
```




```
Conclusion :

Sell Signal: RIO is expected to drop based on SARIMA prediction.
```

Prophet model

```
# Prepare data for Prophet
history = history.reset_index(drop=True)  # Fix: Prevent duplicate
index columns
print(history.columns)  # Debugging: Print column names
history = history[['ds', 'y']] if 'ds' in history.columns else
history[['Date', 'Close']]
history.columns = ['ds', 'y']  # Prophet requires 'ds' for date and
'y' for values
history['ds'] = history['ds'].dt.tz_localize(None)  # Remove timezone

# Initialize and fit Prophet model
model = Prophet()
model.fit(history)

# Create future dataframe for prediction
future = model.make_future_dataframe(periods=30)
forecast = model.predict(future)

# Plot actual vs predicted
print(f"\n\n{stock_ticker} Price Prediction using Prophet :\n")
plt.figure(figsize=(12, 6))
```

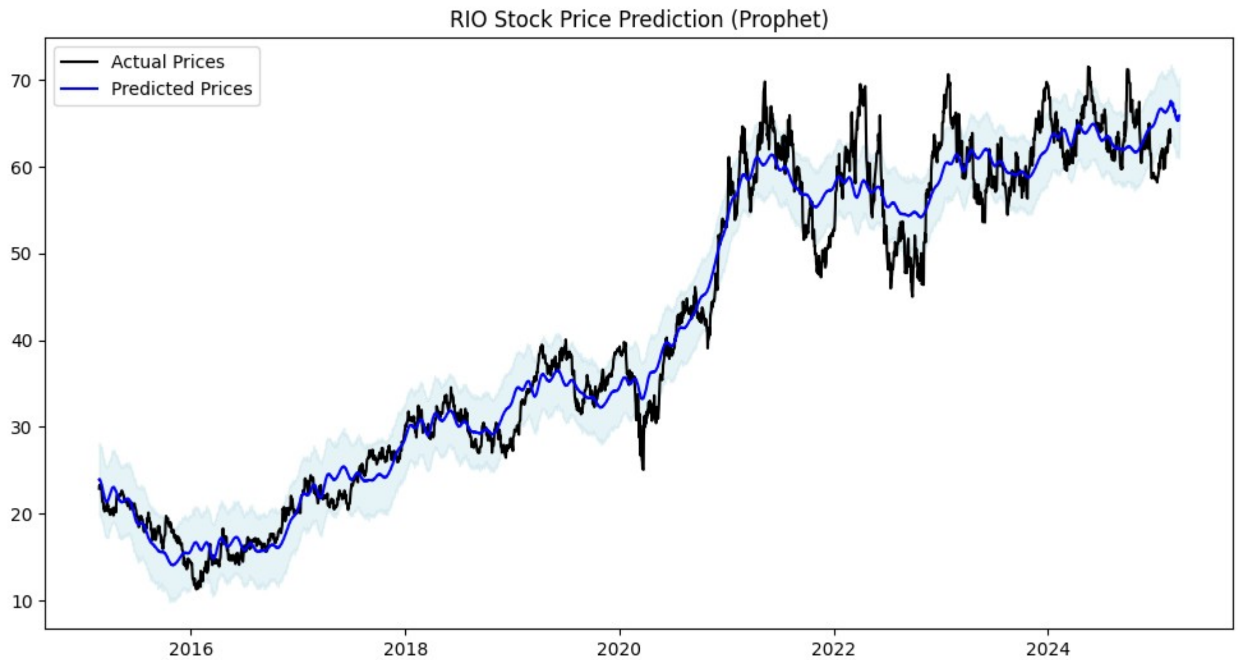
```
plt.plot(history['ds'], history['y'], label='Actual Prices',
color='black')
plt.plot(forecast['ds'], forecast['yhat'], label='Predicted Prices',
color='blue')
plt.fill_between(forecast['ds'], forecast['yhat_lower'],
forecast['yhat_upper'], color='lightblue', alpha=0.3)
plt.title(f"{stock_ticker} Stock Price Prediction (Prophet)")
plt.legend()
plt.show()

# Detailed Buy/Sell Conclusion
predicted_price = forecast['yhat'].iloc[-1]
current_price = history['y'].iloc[-1]

print("\nConclusion :\n")
print(f"Current Price: {current_price:.2f}")
print(f"Predicted Price (Next 30 days): {predicted_price:.2f}")
if predicted_price > current_price * 1.05:
    print("Strong Buy Signal: Price expected to rise significantly.")
    prophet_model = "Buy"
elif predicted_price > current_price:
    print("Buy Signal: Price expected to increase.")
    prophet_model = "Buy"
elif predicted_price < current_price * 0.95:
    print("Strong Sell Signal: Price expected to drop significantly.")
    prophet_model = "Sell"
else:
    print("Sell Signal: Price expected to decrease.")
    prophet_model = "Sell"
```

```
INFO:prophet:Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
```

```
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'Dividends', 'Stock
Splits',
       'Date', 'Ordinal Date', 'Scaled_Close'],
      dtype='object')
```

```
DEBUG:cmdstanpy:input tempfile: /tmp/tmpwxhizq__/3i8b5vv8.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpwxhizq__/b9gokkbf.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.11/dist-
packages/prophet/stan_model/prophet_model.bin', 'random',
'seed=79877', 'data', 'file=/tmp/tmpwxhizq__/3i8b5vv8.json',
'init=/tmp/tmpwxhizq__/b9gokkbf.json', 'output',
'file=/tmp/tmpwxhizq__/prophet_modelqhsdwlh_/prophet_model-
20250223113027.csv', 'method=optimize', 'algorithm=lbfgs',
'iter=10000']
11:30:27 - cmdstanpy - INFO - Chain [1] start processing
```

```
INFO:cmdstanpy:Chain [1] start processing
11:30:28 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing



RIO Price Prediction using Prophet :
```



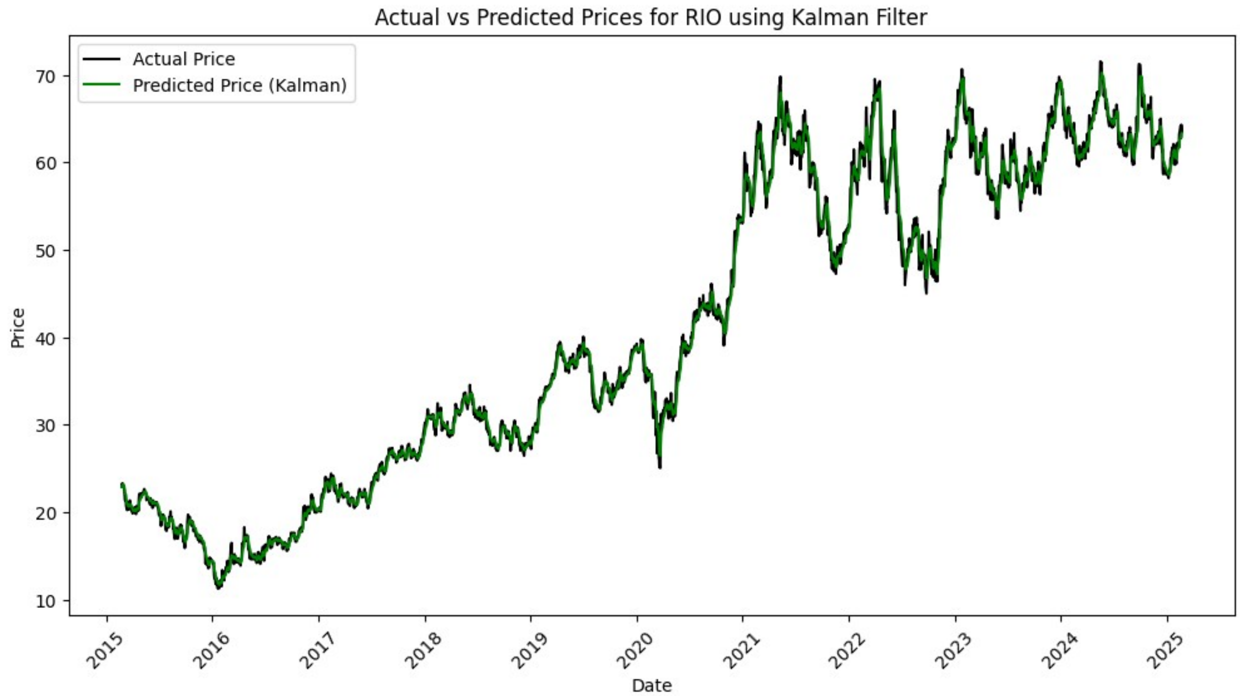

```

Conclusion :

Current Price: 63.53
Predicted Price (Next 30 days): 65.87
Buy Signal: Price expected to increase.
```

Kalman Filter

```
history = history.rename(columns={'y': 'Close', 'ds': 'Date'})
history = history.dropna(subset=['Close'])

# Kalman Filter Implementation
kf = KalmanFilter(initial_state_mean=history['Close'].iloc[0],
                  n_dim_obs=1,
                  n_dim_state=1,
                  transition_matrices=[1],
                  observation_matrices=[1],
```

```python
                    transition_covariance=[0.01],
                    observation_covariance=[0.1])

filtered_state_means, _ = kf.filter(history['Close'].values)

# Plot actual vs predicted
print(f"\n{stock_ticker} Price Prediction using Kalman Filter :\n")
plt.figure(figsize=(12,6))
plt.plot(history['Date'], history['Close'], label='Actual Price',
color='black')
plt.plot(history['Date'], filtered_state_means, label='Predicted Price
(Kalman)', color='green')
plt.title(f"Actual vs Predicted Prices for {stock_ticker} using Kalman
Filter")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.show()

# Buy/Sell Conclusion
print("\nConclusion :\n")
if filtered_state_means[-1] > history['Close'].iloc[-1]:
    print(f"BUY Signal: {stock_ticker} is expected to rise based on
Kalman Filter predictions.")
    kalman_filter_model = "Buy"
elif filtered_state_means[-1] < history['Close'].iloc[-1]:
    print(f"SELL Signal: {stock_ticker} is expected to decline based
on Kalman Filter predictions.")
    kalman_filter_model = "Sell"
else:
    print(f"HOLD: {stock_ticker} is stable, no strong movement
detected.")
    kalman_filter_model = "Hold"
```

```
RIO Price Prediction using Kalman Filter :
```




```
Conclusion :

SELL Signal: RIO is expected to decline based on Kalman Filter
predictions.
```

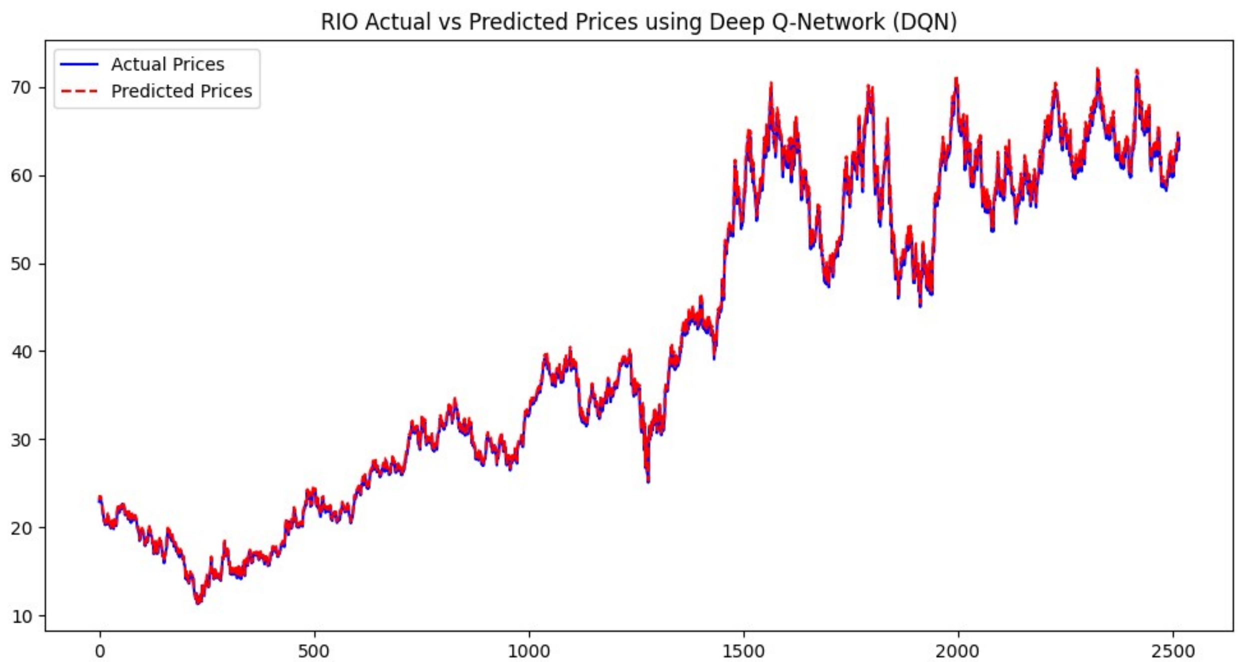
Deep Q-Network (DQN)

```
# Define the Deep Q-Network (DQN)
class DQN(nn.Module):
    def __init__(self, state_size, action_size):
        super(DQN, self).__init__()
        self.fc1 = nn.Linear(state_size, 128)
        self.fc2 = nn.Linear(128, 128)
        self.fc3 = nn.Linear(128, action_size)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        return self.fc3(x)

# Experience Replay Memory
class ReplayMemory:
    def __init__(self, capacity):
        self.memory = deque(maxlen=capacity)
```

```
    def push(self, state, action, reward, next_state, done):
        self.memory.append((state, action, reward, next_state, done))

    def sample(self, batch_size):
        return random.sample(self.memory, batch_size)

    def __len__(self):
        return len(self.memory)

# Training function
def train_dqn(model, memory, optimizer, batch_size, gamma):
    if len(memory) < batch_size:
        return

    transitions = memory.sample(batch_size)
    batch = list(zip(*transitions))

    states = torch.tensor(np.array(batch[0]), dtype=torch.float32)
    actions = torch.tensor(batch[1], dtype=torch.int64).unsqueeze(1)
    rewards = torch.tensor(batch[2], dtype=torch.float32)
    next_states = torch.tensor(np.array(batch[3]),
dtype=torch.float32)
    dones = torch.tensor(batch[4], dtype=torch.float32)

    q_values = model(states).gather(1, actions)
    next_q_values = model(next_states).max(1)[0].detach()
    target_q_values = rewards + (gamma * next_q_values * (1 - dones))

    loss = F.mse_loss(q_values.squeeze(), target_q_values)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

# Initialize environment and model
env = gym.make("CartPole-v1", new_step_api=True)
state_size = env.observation_space.shape[0]
action_size = env.action_space.n
dqn_model = DQN(state_size, action_size)
optimizer = optim.Adam(dqn_model.parameters(), lr=0.001)
memory = ReplayMemory(10000)

episodes = 100
batch_size = 32
gamma = 0.99

def get_stock_data():
    return history[['Close']].pct_change().dropna().values.reshape(-1,
1)  # Ensure 2D shape

stock_data = get_stock_data()
```

```
state_size = stock_data.shape[1]

dqn_model = DQN(state_size, action_size)  # Update model with correct
input size

# Train the model
for episode in range(episodes):
    state = stock_data[0]
    for t in range(len(stock_data) - 1):
        state_tensor = torch.tensor(state,
dtype=torch.float32).view(1, -1)
        action = dqn_model(state_tensor).argmax().item()
        next_state = stock_data[t + 1]
        reward = next_state - state
        done = t == len(stock_data) - 2
        memory.push(state, action, reward, next_state, done)
        state = next_state
        train_dqn(dqn_model, memory, optimizer, batch_size, gamma)

# Predictions vs. Actual Data
actual_prices = history['Close'].values
predicted_prices = []
state = stock_data[0]
for t in range(len(stock_data) - 1):
    state_tensor = torch.tensor(state, dtype=torch.float32).view(1, -
1)
    action = dqn_model(state_tensor).argmax().item()
    predicted_prices.append(actual_prices[t] * (1 + action * 0.01))
    state = stock_data[t + 1]

print(f"\n\n{stock_ticker} Price Prediction using Deep Q-Network (DQN)
:\n")
plt.figure(figsize=(12,6))
plt.plot(actual_prices, label='Actual Prices', color='blue')
plt.plot(predicted_prices, label='Predicted Prices', color='red',
linestyle='dashed')
plt.legend()
plt.title(f"{stock_ticker} Actual vs Predicted Prices using Deep Q-
Network (DQN)")
plt.show()

# Conclusion for Buy/Sell Decision
print("\nConclusion :\n")
if predicted_prices[-1] > actual_prices[-1]:
    print(f"Prediction suggests BUY for {stock_ticker}.")
    dqn_model = "Buy"
else:
    print(f"Prediction suggests SELL for {stock_ticker}.")
    dqn_model = "Sell"
```

```
<ipython-input-20-87549a395662>:38: UserWarning: Creating a tensor
from a list of numpy.ndarrays is extremely slow. Please consider
converting the list to a single numpy.ndarray with numpy.array()
before converting to a tensor. (Triggered internally at
../torch/csrc/utils/tensor_new.cpp:278.)
  rewards = torch.tensor(batch[2], dtype=torch.float32)
<ipython-input-20-87549a395662>:46: UserWarning: Using a target size
(torch.Size([32, 32])) that is different to the input size
(torch.Size([32])). This will likely lead to incorrect results due to
broadcasting. Please ensure they have the same size.
  loss = F.mse_loss(q_values.squeeze(), target_q_values)



RIO Price Prediction using Deep Q-Network (DQN) :
```




```
Conclusion :

Prediction suggests SELL for RIO.
```

Proximal Policy Optimization (PPO)

```
class PolicyNetwork(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(PolicyNetwork, self).__init__()
```

```python
        self.fc1 = nn.Linear(input_dim, 128)
        self.fc2 = nn.Linear(128, 128)
        self.fc3 = nn.Linear(128, output_dim)

    def forward(self, x):
        x = torch.tensor(x, dtype=torch.float32).clone().detach()  #
Ensure proper tensor conversion
        if x.dim() == 1:
            x = x.unsqueeze(0)  # Ensure batch dimension
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        return torch.softmax(self.fc3(x), dim=-1)

class PPOAgent:
    def __init__(self, input_dim, output_dim, lr=1e-3, gamma=0.99,
eps_clip=0.2):
        self.policy = PolicyNetwork(input_dim, output_dim)
        self.optimizer = optim.Adam(self.policy.parameters(), lr=lr)
        self.gamma = gamma
        self.eps_clip = eps_clip
        self.memory = []

    def select_action(self, state):
        state = torch.tensor(state, dtype=torch.float32).unsqueeze(0)
# Ensure batch dimension
        probs = self.policy(state).squeeze(0)  # Remove extra
dimension
        dist = Categorical(probs)
        action = dist.sample()
        return action.item(), dist.log_prob(action)

    def store_transition(self, transition):
        self.memory.append(transition)

    def update(self):
        if not self.memory:
            return

        states, actions, log_probs, rewards = zip(*self.memory)
        states = torch.stack([torch.tensor(s, dtype=torch.float32) for
s in states])
        actions = torch.tensor(actions, dtype=torch.int64)
        log_probs = torch.stack(log_probs)
        rewards = torch.tensor(rewards, dtype=torch.float32)

        returns = []
        discounted_sum = 0
        for reward in reversed(rewards):
            discounted_sum = reward + self.gamma * discounted_sum
            returns.insert(0, discounted_sum)
```

```
        returns = torch.tensor(returns, dtype=torch.float32)

        new_probs = self.policy(states)
        new_dist = Categorical(new_probs)
        new_log_probs = new_dist.log_prob(actions)

        ratios = torch.exp(new_log_probs - log_probs)
        advantages = returns - returns.mean()
        surrogate1 = ratios * advantages
        surrogate2 = torch.clamp(ratios, 1 - self.eps_clip, 1 +
self.eps_clip) * advantages
        loss = -torch.min(surrogate1, surrogate2).mean()

        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()
        self.memory = []

# Running PPO on a sample environment
env = gym.make("CartPole-v1")
agent = PPOAgent(input_dim=4, output_dim=2)

num_episodes = 500 # Higher number for higher accuracy
actual_rewards = []
predicted_rewards = []
for episode in range(num_episodes):
    state = env.reset()
    if isinstance(state, tuple):
        state = state[0]  # Handle environments returning (state,
info)
    state = state.tolist()  # Ensure proper format
    episode_reward = 0

    for _ in range(200):
        action, log_prob = agent.select_action(state)
        next_state, reward, done, _ = env.step(action)  # Corrected
tuple unpacking
        next_state = next_state.tolist()
        agent.store_transition((state, action, log_prob, reward))
        state = next_state
        episode_reward += reward
        if done:
            break

    agent.update()
    actual_rewards.append(episode_reward)
    predicted_rewards.append(agent.policy(torch.tensor(state,
dtype=torch.float32)).detach().numpy().mean())
    print(f"Episode {episode+1}: Reward = {episode_reward}")
```

```python
# Compare actual vs predicted rewards
print(f"\n\n{stock_ticker} Price Prediction using Proximal Policy
Optimization (PPO) :\n")
plt.figure(figsize=(12,6))
plt.plot(actual_rewards, label='Actual Rewards', color='blue')
plt.plot(predicted_rewards, label='Predicted Rewards', color='red',
linestyle='dashed')
plt.title("Actual vs Predicted Rewards")
plt.legend()
plt.show()

# Conclusion on buy or sell
print("\nConclusion :\n")
diff = actual_rewards[-1] - predicted_rewards[-1]
if diff > 0:
    print("Recommendation: BUY - Model predicts an upward trend.")
    ppo_model = "Buy"
elif diff < 0:
    print("Recommendation: SELL - Model predicts a downward trend.")
    ppo_model = "Sell"
else:
    print("Recommendation: HOLD - Model shows stability.")
    ppo_model = "Hold"
```

```
Episode 1: Reward = 37.0
Episode 2: Reward = 21.0
Episode 3: Reward = 9.0
Episode 4: Reward = 11.0
Episode 5: Reward = 33.0
Episode 6: Reward = 12.0
```

```
/usr/local/lib/python3.11/dist-packages/gym/core.py:317:
DeprecationWarning: WARN: Initializing wrapper in old step API which
returns one bool instead of two. It is recommended to set
`new_step_api=True` to use new step API. This will be the default
behaviour in future.
  deprecation(
/usr/local/lib/python3.11/dist-packages/gym/wrappers/step_api_compatib
ility.py:39: DeprecationWarning: WARN: Initializing environment in old
step API which returns one bool instead of two. It is recommended to
set `new_step_api=True` to use new step API. This will be the default
behaviour in future.
  deprecation(
<ipython-input-21-7631c031f636>:9: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  x = torch.tensor(x, dtype=torch.float32).clone().detach()  # Ensure
proper tensor conversion
/usr/local/lib/python3.11/dist-packages/gym/utils/passive_env_checker.
```

```
py:241: DeprecationWarning: `np.bool8` is a deprecated alias for
`np.bool_`.  (Deprecated NumPy 1.24)
  if not isinstance(terminated, (bool, np.bool8)):
```

```
Episode 7: Reward = 18.0
Episode 8: Reward = 17.0
Episode 9: Reward = 23.0
Episode 10: Reward = 25.0
Episode 11: Reward = 15.0
Episode 12: Reward = 26.0
Episode 13: Reward = 24.0
Episode 14: Reward = 23.0
Episode 15: Reward = 10.0
Episode 16: Reward = 23.0
Episode 17: Reward = 15.0
Episode 18: Reward = 23.0
Episode 19: Reward = 14.0
Episode 20: Reward = 31.0
Episode 21: Reward = 30.0
Episode 22: Reward = 17.0
Episode 23: Reward = 23.0
Episode 24: Reward = 15.0
Episode 25: Reward = 21.0
Episode 26: Reward = 38.0
Episode 27: Reward = 26.0
Episode 28: Reward = 29.0
Episode 29: Reward = 10.0
Episode 30: Reward = 18.0
Episode 31: Reward = 14.0
Episode 32: Reward = 8.0
Episode 33: Reward = 12.0
Episode 34: Reward = 16.0
Episode 35: Reward = 18.0
Episode 36: Reward = 23.0
Episode 37: Reward = 19.0
Episode 38: Reward = 20.0
Episode 39: Reward = 12.0
Episode 40: Reward = 16.0
Episode 41: Reward = 11.0
Episode 42: Reward = 15.0
Episode 43: Reward = 18.0
Episode 44: Reward = 17.0
Episode 45: Reward = 56.0
Episode 46: Reward = 41.0
Episode 47: Reward = 25.0
Episode 48: Reward = 11.0
Episode 49: Reward = 18.0
Episode 50: Reward = 11.0
Episode 51: Reward = 14.0
Episode 52: Reward = 29.0
```

```
Episode 53: Reward = 26.0
Episode 54: Reward = 11.0
Episode 55: Reward = 13.0
Episode 56: Reward = 16.0
Episode 57: Reward = 19.0
Episode 58: Reward = 11.0
Episode 59: Reward = 11.0
Episode 60: Reward = 15.0
Episode 61: Reward = 31.0
Episode 62: Reward = 22.0
Episode 63: Reward = 28.0
Episode 64: Reward = 15.0
Episode 65: Reward = 37.0
Episode 66: Reward = 25.0
Episode 67: Reward = 16.0
Episode 68: Reward = 13.0
Episode 69: Reward = 20.0
Episode 70: Reward = 18.0
Episode 71: Reward = 19.0
Episode 72: Reward = 17.0
Episode 73: Reward = 10.0
Episode 74: Reward = 21.0
Episode 75: Reward = 12.0
Episode 76: Reward = 36.0
Episode 77: Reward = 30.0
Episode 78: Reward = 9.0
Episode 79: Reward = 28.0
Episode 80: Reward = 22.0
Episode 81: Reward = 15.0
Episode 82: Reward = 36.0
Episode 83: Reward = 28.0
Episode 84: Reward = 37.0
Episode 85: Reward = 19.0
Episode 86: Reward = 11.0
Episode 87: Reward = 16.0
Episode 88: Reward = 22.0
Episode 89: Reward = 13.0
Episode 90: Reward = 34.0
Episode 91: Reward = 12.0
Episode 92: Reward = 30.0
Episode 93: Reward = 10.0
Episode 94: Reward = 16.0
Episode 95: Reward = 23.0
Episode 96: Reward = 12.0
Episode 97: Reward = 17.0
Episode 98: Reward = 34.0
Episode 99: Reward = 11.0
Episode 100: Reward = 16.0
Episode 101: Reward = 20.0
```

```
Episode 102: Reward = 17.0
Episode 103: Reward = 18.0
Episode 104: Reward = 15.0
Episode 105: Reward = 11.0
Episode 106: Reward = 28.0
Episode 107: Reward = 49.0
Episode 108: Reward = 16.0
Episode 109: Reward = 19.0
Episode 110: Reward = 43.0
Episode 111: Reward = 23.0
Episode 112: Reward = 11.0
Episode 113: Reward = 26.0
Episode 114: Reward = 12.0
Episode 115: Reward = 18.0
Episode 116: Reward = 18.0
Episode 117: Reward = 20.0
Episode 118: Reward = 17.0
Episode 119: Reward = 11.0
Episode 120: Reward = 13.0
Episode 121: Reward = 20.0
Episode 122: Reward = 15.0
Episode 123: Reward = 22.0
Episode 124: Reward = 16.0
Episode 125: Reward = 13.0
Episode 126: Reward = 12.0
Episode 127: Reward = 12.0
Episode 128: Reward = 21.0
Episode 129: Reward = 11.0
Episode 130: Reward = 24.0
Episode 131: Reward = 19.0
Episode 132: Reward = 11.0
Episode 133: Reward = 21.0
Episode 134: Reward = 13.0
Episode 135: Reward = 17.0
Episode 136: Reward = 12.0
Episode 137: Reward = 36.0
Episode 138: Reward = 35.0
Episode 139: Reward = 15.0
Episode 140: Reward = 14.0
Episode 141: Reward = 36.0
Episode 142: Reward = 44.0
Episode 143: Reward = 13.0
Episode 144: Reward = 12.0
Episode 145: Reward = 12.0
Episode 146: Reward = 35.0
Episode 147: Reward = 11.0
Episode 148: Reward = 28.0
Episode 149: Reward = 22.0
Episode 150: Reward = 12.0
```

```
Episode 151: Reward = 13.0
Episode 152: Reward = 13.0
Episode 153: Reward = 27.0
Episode 154: Reward = 63.0
Episode 155: Reward = 9.0
Episode 156: Reward = 38.0
Episode 157: Reward = 31.0
Episode 158: Reward = 15.0
Episode 159: Reward = 15.0
Episode 160: Reward = 17.0
Episode 161: Reward = 17.0
Episode 162: Reward = 21.0
Episode 163: Reward = 17.0
Episode 164: Reward = 18.0
Episode 165: Reward = 21.0
Episode 166: Reward = 24.0
Episode 167: Reward = 13.0
Episode 168: Reward = 16.0
Episode 169: Reward = 15.0
Episode 170: Reward = 13.0
Episode 171: Reward = 63.0
Episode 172: Reward = 27.0
Episode 173: Reward = 19.0
Episode 174: Reward = 13.0
Episode 175: Reward = 11.0
Episode 176: Reward = 20.0
Episode 177: Reward = 32.0
Episode 178: Reward = 11.0
Episode 179: Reward = 16.0
Episode 180: Reward = 23.0
Episode 181: Reward = 40.0
Episode 182: Reward = 21.0
Episode 183: Reward = 16.0
Episode 184: Reward = 14.0
Episode 185: Reward = 36.0
Episode 186: Reward = 11.0
Episode 187: Reward = 10.0
Episode 188: Reward = 9.0
Episode 189: Reward = 48.0
Episode 190: Reward = 11.0
Episode 191: Reward = 21.0
Episode 192: Reward = 9.0
Episode 193: Reward = 43.0
Episode 194: Reward = 20.0
Episode 195: Reward = 19.0
Episode 196: Reward = 33.0
Episode 197: Reward = 19.0
Episode 198: Reward = 44.0
Episode 199: Reward = 21.0
```

```
Episode 200: Reward = 8.0
Episode 201: Reward = 12.0
Episode 202: Reward = 38.0
Episode 203: Reward = 10.0
Episode 204: Reward = 25.0
Episode 205: Reward = 37.0
Episode 206: Reward = 21.0
Episode 207: Reward = 38.0
Episode 208: Reward = 12.0
Episode 209: Reward = 18.0
Episode 210: Reward = 14.0
Episode 211: Reward = 39.0
Episode 212: Reward = 23.0
Episode 213: Reward = 32.0
Episode 214: Reward = 16.0
Episode 215: Reward = 14.0
Episode 216: Reward = 9.0
Episode 217: Reward = 27.0
Episode 218: Reward = 22.0
Episode 219: Reward = 41.0
Episode 220: Reward = 12.0
Episode 221: Reward = 26.0
Episode 222: Reward = 39.0
Episode 223: Reward = 41.0
Episode 224: Reward = 12.0
Episode 225: Reward = 40.0
Episode 226: Reward = 28.0
Episode 227: Reward = 12.0
Episode 228: Reward = 26.0
Episode 229: Reward = 23.0
Episode 230: Reward = 24.0
Episode 231: Reward = 12.0
Episode 232: Reward = 22.0
Episode 233: Reward = 18.0
Episode 234: Reward = 11.0
Episode 235: Reward = 13.0
Episode 236: Reward = 13.0
Episode 237: Reward = 15.0
Episode 238: Reward = 14.0
Episode 239: Reward = 19.0
Episode 240: Reward = 11.0
Episode 241: Reward = 15.0
Episode 242: Reward = 20.0
Episode 243: Reward = 17.0
Episode 244: Reward = 74.0
Episode 245: Reward = 44.0
Episode 246: Reward = 22.0
Episode 247: Reward = 15.0
Episode 248: Reward = 18.0
```

```
Episode 249: Reward = 15.0
Episode 250: Reward = 13.0
Episode 251: Reward = 18.0
Episode 252: Reward = 24.0
Episode 253: Reward = 11.0
Episode 254: Reward = 24.0
Episode 255: Reward = 14.0
Episode 256: Reward = 26.0
Episode 257: Reward = 10.0
Episode 258: Reward = 10.0
Episode 259: Reward = 22.0
Episode 260: Reward = 14.0
Episode 261: Reward = 17.0
Episode 262: Reward = 10.0
Episode 263: Reward = 12.0
Episode 264: Reward = 28.0
Episode 265: Reward = 14.0
Episode 266: Reward = 25.0
Episode 267: Reward = 11.0
Episode 268: Reward = 18.0
Episode 269: Reward = 15.0
Episode 270: Reward = 12.0
Episode 271: Reward = 17.0
Episode 272: Reward = 18.0
Episode 273: Reward = 17.0
Episode 274: Reward = 30.0
Episode 275: Reward = 12.0
Episode 276: Reward = 18.0
Episode 277: Reward = 11.0
Episode 278: Reward = 20.0
Episode 279: Reward = 13.0
Episode 280: Reward = 29.0
Episode 281: Reward = 16.0
Episode 282: Reward = 13.0
Episode 283: Reward = 13.0
Episode 284: Reward = 12.0
Episode 285: Reward = 19.0
Episode 286: Reward = 11.0
Episode 287: Reward = 16.0
Episode 288: Reward = 15.0
Episode 289: Reward = 11.0
Episode 290: Reward = 25.0
Episode 291: Reward = 21.0
Episode 292: Reward = 44.0
Episode 293: Reward = 15.0
Episode 294: Reward = 39.0
Episode 295: Reward = 28.0
Episode 296: Reward = 14.0
Episode 297: Reward = 23.0
```

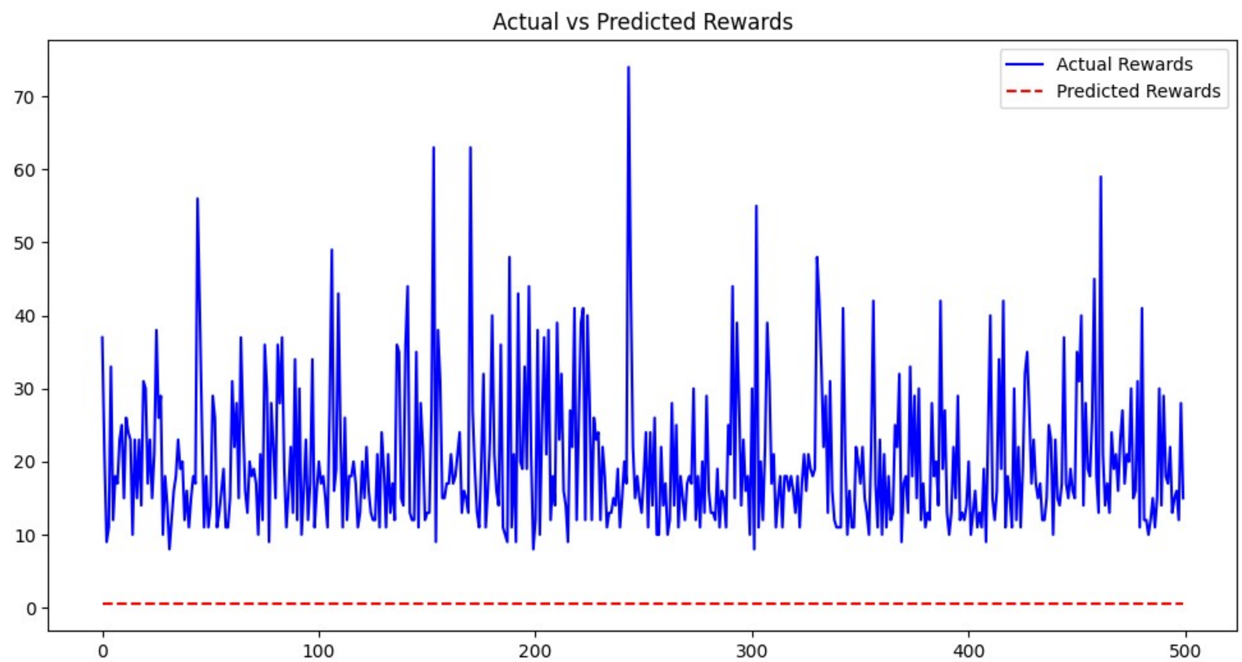
```
Episode 298: Reward = 16.0
Episode 299: Reward = 18.0
Episode 300: Reward = 10.0
Episode 301: Reward = 30.0
Episode 302: Reward = 8.0
Episode 303: Reward = 55.0
Episode 304: Reward = 11.0
Episode 305: Reward = 20.0
Episode 306: Reward = 12.0
Episode 307: Reward = 22.0
Episode 308: Reward = 39.0
Episode 309: Reward = 31.0
Episode 310: Reward = 17.0
Episode 311: Reward = 21.0
Episode 312: Reward = 11.0
Episode 313: Reward = 15.0
Episode 314: Reward = 18.0
Episode 315: Reward = 11.0
Episode 316: Reward = 18.0
Episode 317: Reward = 18.0
Episode 318: Reward = 16.0
Episode 319: Reward = 18.0
Episode 320: Reward = 16.0
Episode 321: Reward = 13.0
Episode 322: Reward = 18.0
Episode 323: Reward = 11.0
Episode 324: Reward = 17.0
Episode 325: Reward = 21.0
Episode 326: Reward = 16.0
Episode 327: Reward = 21.0
Episode 328: Reward = 19.0
Episode 329: Reward = 18.0
Episode 330: Reward = 19.0
Episode 331: Reward = 48.0
Episode 332: Reward = 42.0
Episode 333: Reward = 33.0
Episode 334: Reward = 22.0
Episode 335: Reward = 29.0
Episode 336: Reward = 13.0
Episode 337: Reward = 31.0
Episode 338: Reward = 16.0
Episode 339: Reward = 12.0
Episode 340: Reward = 11.0
Episode 341: Reward = 11.0
Episode 342: Reward = 11.0
Episode 343: Reward = 41.0
Episode 344: Reward = 21.0
Episode 345: Reward = 10.0
Episode 346: Reward = 16.0
```

```
Episode 347: Reward = 11.0
Episode 348: Reward = 11.0
Episode 349: Reward = 22.0
Episode 350: Reward = 20.0
Episode 351: Reward = 17.0
Episode 352: Reward = 22.0
Episode 353: Reward = 15.0
Episode 354: Reward = 13.0
Episode 355: Reward = 10.0
Episode 356: Reward = 22.0
Episode 357: Reward = 42.0
Episode 358: Reward = 17.0
Episode 359: Reward = 11.0
Episode 360: Reward = 23.0
Episode 361: Reward = 10.0
Episode 362: Reward = 21.0
Episode 363: Reward = 11.0
Episode 364: Reward = 18.0
Episode 365: Reward = 12.0
Episode 366: Reward = 13.0
Episode 367: Reward = 25.0
Episode 368: Reward = 22.0
Episode 369: Reward = 32.0
Episode 370: Reward = 9.0
Episode 371: Reward = 17.0
Episode 372: Reward = 18.0
Episode 373: Reward = 13.0
Episode 374: Reward = 33.0
Episode 375: Reward = 18.0
Episode 376: Reward = 29.0
Episode 377: Reward = 15.0
Episode 378: Reward = 30.0
Episode 379: Reward = 12.0
Episode 380: Reward = 17.0
Episode 381: Reward = 11.0
Episode 382: Reward = 13.0
Episode 383: Reward = 12.0
Episode 384: Reward = 28.0
Episode 385: Reward = 18.0
Episode 386: Reward = 20.0
Episode 387: Reward = 14.0
Episode 388: Reward = 42.0
Episode 389: Reward = 19.0
Episode 390: Reward = 27.0
Episode 391: Reward = 13.0
Episode 392: Reward = 10.0
Episode 393: Reward = 13.0
Episode 394: Reward = 22.0
Episode 395: Reward = 15.0
```

```
Episode 396: Reward = 29.0
Episode 397: Reward = 12.0
Episode 398: Reward = 13.0
Episode 399: Reward = 12.0
Episode 400: Reward = 14.0
Episode 401: Reward = 20.0
Episode 402: Reward = 10.0
Episode 403: Reward = 13.0
Episode 404: Reward = 16.0
Episode 405: Reward = 11.0
Episode 406: Reward = 13.0
Episode 407: Reward = 11.0
Episode 408: Reward = 19.0
Episode 409: Reward = 9.0
Episode 410: Reward = 25.0
Episode 411: Reward = 40.0
Episode 412: Reward = 15.0
Episode 413: Reward = 12.0
Episode 414: Reward = 16.0
Episode 415: Reward = 34.0
Episode 416: Reward = 19.0
Episode 417: Reward = 42.0
Episode 418: Reward = 11.0
Episode 419: Reward = 18.0
Episode 420: Reward = 15.0
Episode 421: Reward = 11.0
Episode 422: Reward = 30.0
Episode 423: Reward = 12.0
Episode 424: Reward = 22.0
Episode 425: Reward = 11.0
Episode 426: Reward = 19.0
Episode 427: Reward = 32.0
Episode 428: Reward = 35.0
Episode 429: Reward = 27.0
Episode 430: Reward = 17.0
Episode 431: Reward = 23.0
Episode 432: Reward = 17.0
Episode 433: Reward = 15.0
Episode 434: Reward = 17.0
Episode 435: Reward = 12.0
Episode 436: Reward = 12.0
Episode 437: Reward = 15.0
Episode 438: Reward = 25.0
Episode 439: Reward = 23.0
Episode 440: Reward = 10.0
Episode 441: Reward = 23.0
Episode 442: Reward = 15.0
Episode 443: Reward = 14.0
Episode 444: Reward = 17.0
```

```
Episode 445: Reward = 37.0
Episode 446: Reward = 17.0
Episode 447: Reward = 15.0
Episode 448: Reward = 19.0
Episode 449: Reward = 16.0
Episode 450: Reward = 15.0
Episode 451: Reward = 35.0
Episode 452: Reward = 31.0
Episode 453: Reward = 40.0
Episode 454: Reward = 14.0
Episode 455: Reward = 28.0
Episode 456: Reward = 19.0
Episode 457: Reward = 18.0
Episode 458: Reward = 26.0
Episode 459: Reward = 45.0
Episode 460: Reward = 18.0
Episode 461: Reward = 13.0
Episode 462: Reward = 59.0
Episode 463: Reward = 21.0
Episode 464: Reward = 14.0
Episode 465: Reward = 17.0
Episode 466: Reward = 13.0
Episode 467: Reward = 24.0
Episode 468: Reward = 19.0
Episode 469: Reward = 21.0
Episode 470: Reward = 16.0
Episode 471: Reward = 23.0
Episode 472: Reward = 27.0
Episode 473: Reward = 17.0
Episode 474: Reward = 21.0
Episode 475: Reward = 20.0
Episode 476: Reward = 30.0
Episode 477: Reward = 15.0
Episode 478: Reward = 16.0
Episode 479: Reward = 31.0
Episode 480: Reward = 11.0
Episode 481: Reward = 41.0
Episode 482: Reward = 12.0
Episode 483: Reward = 12.0
Episode 484: Reward = 10.0
Episode 485: Reward = 12.0
Episode 486: Reward = 15.0
Episode 487: Reward = 11.0
Episode 488: Reward = 15.0
Episode 489: Reward = 30.0
Episode 490: Reward = 14.0
Episode 491: Reward = 29.0
Episode 492: Reward = 18.0
Episode 493: Reward = 17.0
```

```
Episode 494: Reward = 22.0
Episode 495: Reward = 13.0
Episode 496: Reward = 15.0
Episode 497: Reward = 16.0
Episode 498: Reward = 12.0
Episode 499: Reward = 28.0
Episode 500: Reward = 15.0


RIO Price Prediction using Proximal Policy Optimization (PPO) :
```




```
Conclusion :

Recommendation: BUY - Model predicts an upward trend.
```

A2C (Advantage Actor-Critic)

```
class ActorCritic(Model):
    def __init__(self, action_size):
        super(ActorCritic, self).__init__()
        self.common = Dense(128, activation='relu')
        self.actor = Dense(action_size, activation='softmax')
        self.critic = Dense(1, activation='linear')

    def call(self, inputs):
```

```
        x = self.common(inputs)
        return self.actor(x), self.critic(x)

def get_action(policy):
    policy = np.squeeze(policy.numpy())  # Remove unnecessary
dimensions
    policy = np.nan_to_num(policy, nan=0.0)  # Replace NaNs with 0
    policy = policy / np.sum(policy)  # Normalize to sum to 1
    return np.random.choice(len(policy), p=policy)

env = gym.make("CartPole-v1")
state_size = env.observation_space.shape[0]
action_size = env.action_space.n
model = ActorCritic(action_size)
optimizer = Adam(learning_rate=0.001)

gamma = 0.99
total_rewards = []
predicted_rewards = []
for episode in range(1000): # Higher number for higher accuracy
    state = env.reset()
    state = np.reshape(state, [1, state_size])
    total_reward = 0
    done = False
    while not done:
        policy, value = model(state)
        action = get_action(policy)
        next_state, reward, done, _ = env.step(action)
        next_state = np.reshape(next_state, [1, state_size])
        _, next_value = model(next_state)
        target = reward + (1 - done) * gamma * next_value
        advantage = target - value

        with tf.GradientTape() as tape:
            policy, value = model(state)
            action_prob = tf.gather(policy[0], action)
            log_prob = tf.math.log(action_prob + 1e-10)
            actor_loss = -log_prob * advantage
            critic_loss = advantage ** 2
            loss = actor_loss + critic_loss

        grads = tape.gradient(loss, model.trainable_variables)
        optimizer.apply_gradients(zip(grads,
model.trainable_variables))

        state = next_state
        total_reward += reward
    total_rewards.append(total_reward)
    predicted_rewards.append(value.numpy()[0][0])
    print(f"Episode {episode + 1}: Total Reward = {total_reward}")
```

```python
env.close()

# Plot rewards
print(f"\n\n{stock_ticker} Price Prediction using A2C (Advantage
Actor-Critic) :\n")
plt.figure(figsize=(10,5))
plt.plot(total_rewards, label='Actual Total Rewards', color='blue')
plt.plot(predicted_rewards, label='Predicted Rewards', color='red',
linestyle='dashed')
plt.xlabel('Episode')
plt.ylabel('Reward')
plt.title('Actual vs Predicted Rewards Comparison')
plt.legend()
plt.show()

# Final Conclusion
print("\nConclusion :\n")
average_reward = np.mean(total_rewards[-50:])
if average_reward > 195:
    print("Conclusion: The model suggests a BUY signal based on
performance.")
    a2c_model = "Buy"
else:
    print("Conclusion: The model suggests a SELL signal based on
performance.")
    a2c_model = "Sell"
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/optimizers/
base_optimizer.py:774: UserWarning: Gradients do not exist for
variables ['actor_critic/dense_11/kernel',
'actor_critic/dense_11/bias'] when minimizing the loss. If using
`model.compile()`, did you forget to provide a `loss` argument?
  warnings.warn(
```

```
Episode 1: Total Reward = 18.0
Episode 2: Total Reward = 12.0
Episode 3: Total Reward = 51.0
Episode 4: Total Reward = 17.0
Episode 5: Total Reward = 15.0
Episode 6: Total Reward = 62.0
Episode 7: Total Reward = 36.0
Episode 8: Total Reward = 20.0
Episode 9: Total Reward = 26.0
Episode 10: Total Reward = 24.0
Episode 11: Total Reward = 38.0
Episode 12: Total Reward = 35.0
Episode 13: Total Reward = 14.0
Episode 14: Total Reward = 26.0
Episode 15: Total Reward = 28.0
```

```
Episode 16: Total Reward = 15.0
Episode 17: Total Reward = 37.0
Episode 18: Total Reward = 11.0
Episode 19: Total Reward = 23.0
Episode 20: Total Reward = 30.0
Episode 21: Total Reward = 20.0
Episode 22: Total Reward = 19.0
Episode 23: Total Reward = 16.0
Episode 24: Total Reward = 45.0
Episode 25: Total Reward = 16.0
Episode 26: Total Reward = 14.0
Episode 27: Total Reward = 15.0
Episode 28: Total Reward = 10.0
Episode 29: Total Reward = 59.0
Episode 30: Total Reward = 12.0
Episode 31: Total Reward = 17.0
Episode 32: Total Reward = 51.0
Episode 33: Total Reward = 24.0
Episode 34: Total Reward = 24.0
Episode 35: Total Reward = 17.0
Episode 36: Total Reward = 29.0
Episode 37: Total Reward = 21.0
Episode 38: Total Reward = 13.0
Episode 39: Total Reward = 20.0
Episode 40: Total Reward = 17.0
Episode 41: Total Reward = 16.0
Episode 42: Total Reward = 16.0
Episode 43: Total Reward = 13.0
Episode 44: Total Reward = 14.0
Episode 45: Total Reward = 56.0
Episode 46: Total Reward = 25.0
Episode 47: Total Reward = 19.0
Episode 48: Total Reward = 13.0
Episode 49: Total Reward = 21.0
Episode 50: Total Reward = 18.0
Episode 51: Total Reward = 21.0
Episode 52: Total Reward = 36.0
Episode 53: Total Reward = 15.0
Episode 54: Total Reward = 12.0
Episode 55: Total Reward = 50.0
Episode 56: Total Reward = 15.0
Episode 57: Total Reward = 16.0
Episode 58: Total Reward = 13.0
Episode 59: Total Reward = 41.0
Episode 60: Total Reward = 22.0
Episode 61: Total Reward = 22.0
Episode 62: Total Reward = 37.0
Episode 63: Total Reward = 12.0
Episode 64: Total Reward = 29.0
```

```
Episode 65: Total Reward = 25.0
Episode 66: Total Reward = 68.0
Episode 67: Total Reward = 27.0
Episode 68: Total Reward = 13.0
Episode 69: Total Reward = 29.0
Episode 70: Total Reward = 41.0
Episode 71: Total Reward = 48.0
Episode 72: Total Reward = 24.0
Episode 73: Total Reward = 17.0
Episode 74: Total Reward = 26.0
Episode 75: Total Reward = 34.0
Episode 76: Total Reward = 14.0
Episode 77: Total Reward = 51.0
Episode 78: Total Reward = 23.0
Episode 79: Total Reward = 15.0
Episode 80: Total Reward = 32.0
Episode 81: Total Reward = 56.0
Episode 82: Total Reward = 37.0
Episode 83: Total Reward = 36.0
Episode 84: Total Reward = 92.0
Episode 85: Total Reward = 32.0
Episode 86: Total Reward = 33.0
Episode 87: Total Reward = 40.0
Episode 88: Total Reward = 54.0
Episode 89: Total Reward = 56.0
Episode 90: Total Reward = 40.0
Episode 91: Total Reward = 60.0
Episode 92: Total Reward = 51.0
Episode 93: Total Reward = 73.0
Episode 94: Total Reward = 33.0
Episode 95: Total Reward = 38.0
Episode 96: Total Reward = 74.0
Episode 97: Total Reward = 60.0
Episode 98: Total Reward = 52.0
Episode 99: Total Reward = 107.0
Episode 100: Total Reward = 36.0
Episode 101: Total Reward = 55.0
Episode 102: Total Reward = 68.0
Episode 103: Total Reward = 40.0
Episode 104: Total Reward = 51.0
Episode 105: Total Reward = 95.0
Episode 106: Total Reward = 37.0
Episode 107: Total Reward = 35.0
Episode 108: Total Reward = 26.0
Episode 109: Total Reward = 23.0
Episode 110: Total Reward = 30.0
Episode 111: Total Reward = 41.0
Episode 112: Total Reward = 32.0
Episode 113: Total Reward = 21.0
```

```
Episode 114: Total Reward = 30.0
Episode 115: Total Reward = 25.0
Episode 116: Total Reward = 29.0
Episode 117: Total Reward = 21.0
Episode 118: Total Reward = 42.0
Episode 119: Total Reward = 14.0
Episode 120: Total Reward = 25.0
Episode 121: Total Reward = 32.0
Episode 122: Total Reward = 27.0
Episode 123: Total Reward = 24.0
Episode 124: Total Reward = 22.0
Episode 125: Total Reward = 28.0
Episode 126: Total Reward = 22.0
Episode 127: Total Reward = 16.0
Episode 128: Total Reward = 34.0
Episode 129: Total Reward = 36.0
Episode 130: Total Reward = 31.0
Episode 131: Total Reward = 40.0
Episode 132: Total Reward = 28.0
Episode 133: Total Reward = 37.0
Episode 134: Total Reward = 36.0
Episode 135: Total Reward = 34.0
Episode 136: Total Reward = 42.0
Episode 137: Total Reward = 27.0
Episode 138: Total Reward = 47.0
Episode 139: Total Reward = 40.0
Episode 140: Total Reward = 41.0
Episode 141: Total Reward = 44.0
Episode 142: Total Reward = 34.0
Episode 143: Total Reward = 53.0
Episode 144: Total Reward = 50.0
Episode 145: Total Reward = 43.0
Episode 146: Total Reward = 25.0
Episode 147: Total Reward = 67.0
Episode 148: Total Reward = 47.0
Episode 149: Total Reward = 49.0
Episode 150: Total Reward = 35.0
Episode 151: Total Reward = 27.0
Episode 152: Total Reward = 38.0
Episode 153: Total Reward = 27.0
Episode 154: Total Reward = 25.0
Episode 155: Total Reward = 17.0
Episode 156: Total Reward = 27.0
Episode 157: Total Reward = 28.0
Episode 158: Total Reward = 28.0
Episode 159: Total Reward = 27.0
Episode 160: Total Reward = 18.0
Episode 161: Total Reward = 28.0
Episode 162: Total Reward = 29.0
```

```
Episode 163: Total Reward = 33.0
Episode 164: Total Reward = 26.0
Episode 165: Total Reward = 26.0
Episode 166: Total Reward = 39.0
Episode 167: Total Reward = 29.0
Episode 168: Total Reward = 33.0
Episode 169: Total Reward = 39.0
Episode 170: Total Reward = 31.0
Episode 171: Total Reward = 24.0
Episode 172: Total Reward = 39.0
Episode 173: Total Reward = 28.0
Episode 174: Total Reward = 22.0
Episode 175: Total Reward = 31.0
Episode 176: Total Reward = 33.0
Episode 177: Total Reward = 24.0
Episode 178: Total Reward = 25.0
Episode 179: Total Reward = 41.0
Episode 180: Total Reward = 28.0
Episode 181: Total Reward = 35.0
Episode 182: Total Reward = 26.0
Episode 183: Total Reward = 23.0
Episode 184: Total Reward = 31.0
Episode 185: Total Reward = 32.0
Episode 186: Total Reward = 27.0
Episode 187: Total Reward = 20.0
Episode 188: Total Reward = 25.0
Episode 189: Total Reward = 13.0
Episode 190: Total Reward = 17.0
Episode 191: Total Reward = 24.0
Episode 192: Total Reward = 24.0
Episode 193: Total Reward = 34.0
Episode 194: Total Reward = 26.0
Episode 195: Total Reward = 17.0
Episode 196: Total Reward = 26.0
Episode 197: Total Reward = 17.0
Episode 198: Total Reward = 22.0
Episode 199: Total Reward = 23.0
Episode 200: Total Reward = 22.0
Episode 201: Total Reward = 19.0
Episode 202: Total Reward = 21.0
Episode 203: Total Reward = 18.0
Episode 204: Total Reward = 14.0
Episode 205: Total Reward = 19.0
Episode 206: Total Reward = 16.0
Episode 207: Total Reward = 16.0
Episode 208: Total Reward = 12.0
Episode 209: Total Reward = 18.0
Episode 210: Total Reward = 14.0
Episode 211: Total Reward = 13.0
```

```
Episode 212: Total Reward = 15.0
Episode 213: Total Reward = 14.0
Episode 214: Total Reward = 16.0
Episode 215: Total Reward = 13.0
Episode 216: Total Reward = 11.0
Episode 217: Total Reward = 11.0
Episode 218: Total Reward = 13.0
Episode 219: Total Reward = 11.0
Episode 220: Total Reward = 13.0
Episode 221: Total Reward = 12.0
Episode 222: Total Reward = 18.0
Episode 223: Total Reward = 16.0
Episode 224: Total Reward = 12.0
Episode 225: Total Reward = 13.0
Episode 226: Total Reward = 16.0
Episode 227: Total Reward = 16.0
Episode 228: Total Reward = 15.0
Episode 229: Total Reward = 17.0
Episode 230: Total Reward = 12.0
Episode 231: Total Reward = 15.0
Episode 232: Total Reward = 17.0
Episode 233: Total Reward = 14.0
Episode 234: Total Reward = 16.0
Episode 235: Total Reward = 13.0
Episode 236: Total Reward = 16.0
Episode 237: Total Reward = 14.0
Episode 238: Total Reward = 12.0
Episode 239: Total Reward = 14.0
Episode 240: Total Reward = 13.0
Episode 241: Total Reward = 14.0
Episode 242: Total Reward = 10.0
Episode 243: Total Reward = 12.0
Episode 244: Total Reward = 10.0
Episode 245: Total Reward = 10.0
Episode 246: Total Reward = 9.0
Episode 247: Total Reward = 14.0
Episode 248: Total Reward = 12.0
Episode 249: Total Reward = 10.0
Episode 250: Total Reward = 14.0
Episode 251: Total Reward = 13.0
Episode 252: Total Reward = 13.0
Episode 253: Total Reward = 13.0
Episode 254: Total Reward = 11.0
Episode 255: Total Reward = 12.0
Episode 256: Total Reward = 10.0
Episode 257: Total Reward = 11.0
Episode 258: Total Reward = 11.0
Episode 259: Total Reward = 10.0
Episode 260: Total Reward = 12.0
```

```
Episode 261: Total Reward = 14.0
Episode 262: Total Reward = 10.0
Episode 263: Total Reward = 11.0
Episode 264: Total Reward = 11.0
Episode 265: Total Reward = 10.0
Episode 266: Total Reward = 12.0
Episode 267: Total Reward = 13.0
Episode 268: Total Reward = 13.0
Episode 269: Total Reward = 13.0
Episode 270: Total Reward = 13.0
Episode 271: Total Reward = 15.0
Episode 272: Total Reward = 13.0
Episode 273: Total Reward = 12.0
Episode 274: Total Reward = 12.0
Episode 275: Total Reward = 9.0
Episode 276: Total Reward = 10.0
Episode 277: Total Reward = 13.0
Episode 278: Total Reward = 12.0
Episode 279: Total Reward = 11.0
Episode 280: Total Reward = 12.0
Episode 281: Total Reward = 16.0
Episode 282: Total Reward = 10.0
Episode 283: Total Reward = 14.0
Episode 284: Total Reward = 10.0
Episode 285: Total Reward = 11.0
Episode 286: Total Reward = 12.0
Episode 287: Total Reward = 13.0
Episode 288: Total Reward = 13.0
Episode 289: Total Reward = 18.0
Episode 290: Total Reward = 16.0
Episode 291: Total Reward = 10.0
Episode 292: Total Reward = 12.0
Episode 293: Total Reward = 13.0
Episode 294: Total Reward = 13.0
Episode 295: Total Reward = 10.0
Episode 296: Total Reward = 10.0
Episode 297: Total Reward = 12.0
Episode 298: Total Reward = 14.0
Episode 299: Total Reward = 12.0
Episode 300: Total Reward = 9.0
Episode 301: Total Reward = 11.0
Episode 302: Total Reward = 11.0
Episode 303: Total Reward = 11.0
Episode 304: Total Reward = 12.0
Episode 305: Total Reward = 11.0
Episode 306: Total Reward = 10.0
Episode 307: Total Reward = 13.0
Episode 308: Total Reward = 10.0
Episode 309: Total Reward = 12.0
```

```
Episode 310: Total Reward = 12.0
Episode 311: Total Reward = 13.0
Episode 312: Total Reward = 10.0
Episode 313: Total Reward = 13.0
Episode 314: Total Reward = 11.0
Episode 315: Total Reward = 13.0
Episode 316: Total Reward = 9.0
Episode 317: Total Reward = 12.0
Episode 318: Total Reward = 11.0
Episode 319: Total Reward = 10.0
Episode 320: Total Reward = 12.0
Episode 321: Total Reward = 12.0
Episode 322: Total Reward = 11.0
Episode 323: Total Reward = 12.0
Episode 324: Total Reward = 10.0
Episode 325: Total Reward = 11.0
Episode 326: Total Reward = 12.0
Episode 327: Total Reward = 11.0
Episode 328: Total Reward = 9.0
Episode 329: Total Reward = 13.0
Episode 330: Total Reward = 11.0
Episode 331: Total Reward = 9.0
Episode 332: Total Reward = 12.0
Episode 333: Total Reward = 11.0
Episode 334: Total Reward = 11.0
Episode 335: Total Reward = 12.0
Episode 336: Total Reward = 11.0
Episode 337: Total Reward = 12.0
Episode 338: Total Reward = 13.0
Episode 339: Total Reward = 13.0
Episode 340: Total Reward = 14.0
Episode 341: Total Reward = 12.0
Episode 342: Total Reward = 11.0
Episode 343: Total Reward = 14.0
Episode 344: Total Reward = 13.0
Episode 345: Total Reward = 13.0
Episode 346: Total Reward = 12.0
Episode 347: Total Reward = 13.0
Episode 348: Total Reward = 13.0
Episode 349: Total Reward = 13.0
Episode 350: Total Reward = 9.0
Episode 351: Total Reward = 10.0
Episode 352: Total Reward = 11.0
Episode 353: Total Reward = 9.0
Episode 354: Total Reward = 12.0
Episode 355: Total Reward = 10.0
Episode 356: Total Reward = 11.0
Episode 357: Total Reward = 10.0
Episode 358: Total Reward = 9.0
```

```
Episode 359: Total Reward = 10.0
Episode 360: Total Reward = 11.0
Episode 361: Total Reward = 10.0
Episode 362: Total Reward = 9.0
Episode 363: Total Reward = 11.0
Episode 364: Total Reward = 10.0
Episode 365: Total Reward = 10.0
Episode 366: Total Reward = 12.0
Episode 367: Total Reward = 9.0
Episode 368: Total Reward = 10.0
Episode 369: Total Reward = 10.0
Episode 370: Total Reward = 9.0
Episode 371: Total Reward = 9.0
Episode 372: Total Reward = 9.0
Episode 373: Total Reward = 10.0
Episode 374: Total Reward = 12.0
Episode 375: Total Reward = 8.0
Episode 376: Total Reward = 10.0
Episode 377: Total Reward = 9.0
Episode 378: Total Reward = 11.0
Episode 379: Total Reward = 9.0
Episode 380: Total Reward = 9.0
Episode 381: Total Reward = 11.0
Episode 382: Total Reward = 10.0
Episode 383: Total Reward = 9.0
Episode 384: Total Reward = 9.0
Episode 385: Total Reward = 10.0
Episode 386: Total Reward = 11.0
Episode 387: Total Reward = 9.0
Episode 388: Total Reward = 11.0
Episode 389: Total Reward = 9.0
Episode 390: Total Reward = 12.0
Episode 391: Total Reward = 10.0
Episode 392: Total Reward = 10.0
Episode 393: Total Reward = 12.0
Episode 394: Total Reward = 11.0
Episode 395: Total Reward = 12.0
Episode 396: Total Reward = 10.0
Episode 397: Total Reward = 11.0
Episode 398: Total Reward = 10.0
Episode 399: Total Reward = 9.0
Episode 400: Total Reward = 10.0
Episode 401: Total Reward = 10.0
Episode 402: Total Reward = 10.0
Episode 403: Total Reward = 11.0
Episode 404: Total Reward = 11.0
Episode 405: Total Reward = 9.0
Episode 406: Total Reward = 9.0
Episode 407: Total Reward = 11.0
```

```
Episode 408: Total Reward = 8.0
Episode 409: Total Reward = 9.0
Episode 410: Total Reward = 9.0
Episode 411: Total Reward = 9.0
Episode 412: Total Reward = 9.0
Episode 413: Total Reward = 9.0
Episode 414: Total Reward = 11.0
Episode 415: Total Reward = 10.0
Episode 416: Total Reward = 9.0
Episode 417: Total Reward = 9.0
Episode 418: Total Reward = 10.0
Episode 419: Total Reward = 9.0
Episode 420: Total Reward = 10.0
Episode 421: Total Reward = 11.0
Episode 422: Total Reward = 10.0
Episode 423: Total Reward = 9.0
Episode 424: Total Reward = 8.0
Episode 425: Total Reward = 9.0
Episode 426: Total Reward = 9.0
Episode 427: Total Reward = 10.0
Episode 428: Total Reward = 9.0
Episode 429: Total Reward = 9.0
Episode 430: Total Reward = 8.0
Episode 431: Total Reward = 8.0
Episode 432: Total Reward = 10.0
Episode 433: Total Reward = 11.0
Episode 434: Total Reward = 11.0
Episode 435: Total Reward = 9.0
Episode 436: Total Reward = 8.0
Episode 437: Total Reward = 10.0
Episode 438: Total Reward = 11.0
Episode 439: Total Reward = 10.0
Episode 440: Total Reward = 10.0
Episode 441: Total Reward = 10.0
Episode 442: Total Reward = 9.0
Episode 443: Total Reward = 8.0
Episode 444: Total Reward = 10.0
Episode 445: Total Reward = 10.0
Episode 446: Total Reward = 9.0
Episode 447: Total Reward = 10.0
Episode 448: Total Reward = 9.0
Episode 449: Total Reward = 10.0
Episode 450: Total Reward = 9.0
Episode 451: Total Reward = 9.0
Episode 452: Total Reward = 11.0
Episode 453: Total Reward = 8.0
Episode 454: Total Reward = 9.0
Episode 455: Total Reward = 10.0
Episode 456: Total Reward = 8.0
```

```
Episode 457: Total Reward = 9.0
Episode 458: Total Reward = 8.0
Episode 459: Total Reward = 10.0
Episode 460: Total Reward = 9.0
Episode 461: Total Reward = 10.0
Episode 462: Total Reward = 9.0
Episode 463: Total Reward = 9.0
Episode 464: Total Reward = 9.0
Episode 465: Total Reward = 10.0
Episode 466: Total Reward = 10.0
Episode 467: Total Reward = 10.0
Episode 468: Total Reward = 9.0
Episode 469: Total Reward = 9.0
Episode 470: Total Reward = 9.0
Episode 471: Total Reward = 10.0
Episode 472: Total Reward = 9.0
Episode 473: Total Reward = 10.0
Episode 474: Total Reward = 9.0
Episode 475: Total Reward = 9.0
Episode 476: Total Reward = 9.0
Episode 477: Total Reward = 8.0
Episode 478: Total Reward = 10.0
Episode 479: Total Reward = 9.0
Episode 480: Total Reward = 9.0
Episode 481: Total Reward = 10.0
Episode 482: Total Reward = 9.0
Episode 483: Total Reward = 9.0
Episode 484: Total Reward = 10.0
Episode 485: Total Reward = 10.0
Episode 486: Total Reward = 8.0
Episode 487: Total Reward = 10.0
Episode 488: Total Reward = 10.0
Episode 489: Total Reward = 10.0
Episode 490: Total Reward = 9.0
Episode 491: Total Reward = 9.0
Episode 492: Total Reward = 9.0
Episode 493: Total Reward = 9.0
Episode 494: Total Reward = 10.0
Episode 495: Total Reward = 10.0
Episode 496: Total Reward = 11.0
Episode 497: Total Reward = 9.0
Episode 498: Total Reward = 8.0
Episode 499: Total Reward = 8.0
Episode 500: Total Reward = 9.0
Episode 501: Total Reward = 9.0
Episode 502: Total Reward = 10.0
Episode 503: Total Reward = 10.0
Episode 504: Total Reward = 10.0
Episode 505: Total Reward = 9.0
Episode 506: Total Reward = 10.0
```

```
Episode 507: Total Reward = 10.0
Episode 508: Total Reward = 10.0
Episode 509: Total Reward = 9.0
Episode 510: Total Reward = 9.0
Episode 511: Total Reward = 9.0
Episode 512: Total Reward = 9.0
Episode 513: Total Reward = 10.0
Episode 514: Total Reward = 9.0
Episode 515: Total Reward = 9.0
Episode 516: Total Reward = 9.0
Episode 517: Total Reward = 10.0
Episode 518: Total Reward = 8.0
Episode 519: Total Reward = 9.0
Episode 520: Total Reward = 8.0
Episode 521: Total Reward = 10.0
Episode 522: Total Reward = 9.0
Episode 523: Total Reward = 8.0
Episode 524: Total Reward = 10.0
Episode 525: Total Reward = 10.0
Episode 526: Total Reward = 9.0
Episode 527: Total Reward = 10.0
Episode 528: Total Reward = 11.0
Episode 529: Total Reward = 10.0
Episode 530: Total Reward = 12.0
Episode 531: Total Reward = 10.0
Episode 532: Total Reward = 10.0
Episode 533: Total Reward = 9.0
Episode 534: Total Reward = 9.0
Episode 535: Total Reward = 8.0
Episode 536: Total Reward = 9.0
Episode 537: Total Reward = 9.0
Episode 538: Total Reward = 9.0
Episode 539: Total Reward = 11.0
Episode 540: Total Reward = 10.0
Episode 541: Total Reward = 8.0
Episode 542: Total Reward = 9.0
Episode 543: Total Reward = 9.0
Episode 544: Total Reward = 10.0
Episode 545: Total Reward = 9.0
Episode 546: Total Reward = 9.0
Episode 547: Total Reward = 8.0
Episode 548: Total Reward = 9.0
Episode 549: Total Reward = 9.0
Episode 550: Total Reward = 9.0
Episode 551: Total Reward = 10.0
Episode 552: Total Reward = 9.0
Episode 553: Total Reward = 9.0
Episode 554: Total Reward = 8.0
Episode 555: Total Reward = 9.0
```

```
Episode 556: Total Reward = 9.0
Episode 557: Total Reward = 9.0
Episode 558: Total Reward = 9.0
Episode 559: Total Reward = 9.0
Episode 560: Total Reward = 8.0
Episode 561: Total Reward = 10.0
Episode 562: Total Reward = 8.0
Episode 563: Total Reward = 10.0
Episode 564: Total Reward = 9.0
Episode 565: Total Reward = 8.0
Episode 566: Total Reward = 10.0
Episode 567: Total Reward = 10.0
Episode 568: Total Reward = 10.0
Episode 569: Total Reward = 9.0
Episode 570: Total Reward = 9.0
Episode 571: Total Reward = 9.0
Episode 572: Total Reward = 10.0
Episode 573: Total Reward = 9.0
Episode 574: Total Reward = 10.0
Episode 575: Total Reward = 9.0
Episode 576: Total Reward = 9.0
Episode 577: Total Reward = 10.0
Episode 578: Total Reward = 10.0
Episode 579: Total Reward = 10.0
Episode 580: Total Reward = 10.0
Episode 581: Total Reward = 10.0
Episode 582: Total Reward = 9.0
Episode 583: Total Reward = 9.0
Episode 584: Total Reward = 9.0
Episode 585: Total Reward = 9.0
Episode 586: Total Reward = 10.0
Episode 587: Total Reward = 10.0
Episode 588: Total Reward = 10.0
Episode 589: Total Reward = 9.0
Episode 590: Total Reward = 10.0
Episode 591: Total Reward = 9.0
Episode 592: Total Reward = 8.0
Episode 593: Total Reward = 8.0
Episode 594: Total Reward = 11.0
Episode 595: Total Reward = 10.0
Episode 596: Total Reward = 10.0
Episode 597: Total Reward = 8.0
Episode 598: Total Reward = 9.0
Episode 599: Total Reward = 10.0
Episode 600: Total Reward = 10.0
Episode 601: Total Reward = 9.0
Episode 602: Total Reward = 10.0
Episode 603: Total Reward = 8.0
Episode 604: Total Reward = 9.0
```

```
Episode 605: Total Reward = 11.0
Episode 606: Total Reward = 9.0
Episode 607: Total Reward = 9.0
Episode 608: Total Reward = 10.0
Episode 609: Total Reward = 9.0
Episode 610: Total Reward = 9.0
Episode 611: Total Reward = 10.0
Episode 612: Total Reward = 9.0
Episode 613: Total Reward = 10.0
Episode 614: Total Reward = 10.0
Episode 615: Total Reward = 9.0
Episode 616: Total Reward = 10.0
Episode 617: Total Reward = 11.0
Episode 618: Total Reward = 10.0
Episode 619: Total Reward = 10.0
Episode 620: Total Reward = 10.0
Episode 621: Total Reward = 10.0
Episode 622: Total Reward = 8.0
Episode 623: Total Reward = 10.0
Episode 624: Total Reward = 8.0
Episode 625: Total Reward = 9.0
Episode 626: Total Reward = 10.0
Episode 627: Total Reward = 8.0
Episode 628: Total Reward = 9.0
Episode 629: Total Reward = 8.0
Episode 630: Total Reward = 10.0
Episode 631: Total Reward = 10.0
Episode 632: Total Reward = 9.0
Episode 633: Total Reward = 10.0
Episode 634: Total Reward = 9.0
Episode 635: Total Reward = 9.0
Episode 636: Total Reward = 8.0
Episode 637: Total Reward = 10.0
Episode 638: Total Reward = 9.0
Episode 639: Total Reward = 10.0
Episode 640: Total Reward = 10.0
Episode 641: Total Reward = 10.0
Episode 642: Total Reward = 10.0
Episode 643: Total Reward = 9.0
Episode 644: Total Reward = 10.0
Episode 645: Total Reward = 9.0
Episode 646: Total Reward = 10.0
Episode 647: Total Reward = 9.0
Episode 648: Total Reward = 10.0
Episode 649: Total Reward = 10.0
Episode 650: Total Reward = 10.0
Episode 651: Total Reward = 9.0
Episode 652: Total Reward = 8.0
Episode 653: Total Reward = 9.0
```

```
Episode 654: Total Reward = 10.0
Episode 655: Total Reward = 9.0
Episode 656: Total Reward = 11.0
Episode 657: Total Reward = 10.0
Episode 658: Total Reward = 9.0
Episode 659: Total Reward = 9.0
Episode 660: Total Reward = 9.0
Episode 661: Total Reward = 9.0
Episode 662: Total Reward = 8.0
Episode 663: Total Reward = 10.0
Episode 664: Total Reward = 9.0
Episode 665: Total Reward = 9.0
Episode 666: Total Reward = 10.0
Episode 667: Total Reward = 10.0
Episode 668: Total Reward = 10.0
Episode 669: Total Reward = 9.0
Episode 670: Total Reward = 9.0
Episode 671: Total Reward = 10.0
Episode 672: Total Reward = 10.0
Episode 673: Total Reward = 9.0
Episode 674: Total Reward = 9.0
Episode 675: Total Reward = 9.0
Episode 676: Total Reward = 8.0
Episode 677: Total Reward = 9.0
Episode 678: Total Reward = 8.0
Episode 679: Total Reward = 10.0
Episode 680: Total Reward = 9.0
Episode 681: Total Reward = 10.0
Episode 682: Total Reward = 8.0
Episode 683: Total Reward = 9.0
Episode 684: Total Reward = 9.0
Episode 685: Total Reward = 10.0
Episode 686: Total Reward = 9.0
Episode 687: Total Reward = 8.0
Episode 688: Total Reward = 10.0
Episode 689: Total Reward = 10.0
Episode 690: Total Reward = 9.0
Episode 691: Total Reward = 10.0
Episode 692: Total Reward = 9.0
Episode 693: Total Reward = 10.0
Episode 694: Total Reward = 10.0
Episode 695: Total Reward = 10.0
Episode 696: Total Reward = 9.0
Episode 697: Total Reward = 10.0
Episode 698: Total Reward = 9.0
Episode 699: Total Reward = 9.0
Episode 700: Total Reward = 9.0
Episode 701: Total Reward = 8.0
Episode 702: Total Reward = 9.0
```

```
Episode 703: Total Reward = 10.0
Episode 704: Total Reward = 10.0
Episode 705: Total Reward = 10.0
Episode 706: Total Reward = 10.0
Episode 707: Total Reward = 9.0
Episode 708: Total Reward = 10.0
Episode 709: Total Reward = 8.0
Episode 710: Total Reward = 8.0
Episode 711: Total Reward = 9.0
Episode 712: Total Reward = 9.0
Episode 713: Total Reward = 10.0
Episode 714: Total Reward = 9.0
Episode 715: Total Reward = 9.0
Episode 716: Total Reward = 8.0
Episode 717: Total Reward = 10.0
Episode 718: Total Reward = 10.0
Episode 719: Total Reward = 10.0
Episode 720: Total Reward = 9.0
Episode 721: Total Reward = 8.0
Episode 722: Total Reward = 9.0
Episode 723: Total Reward = 9.0
Episode 724: Total Reward = 10.0
Episode 725: Total Reward = 9.0
Episode 726: Total Reward = 10.0
Episode 727: Total Reward = 9.0
Episode 728: Total Reward = 10.0
Episode 729: Total Reward = 10.0
Episode 730: Total Reward = 10.0
Episode 731: Total Reward = 9.0
Episode 732: Total Reward = 9.0
Episode 733: Total Reward = 9.0
Episode 734: Total Reward = 10.0
Episode 735: Total Reward = 9.0
Episode 736: Total Reward = 9.0
Episode 737: Total Reward = 11.0
Episode 738: Total Reward = 10.0
Episode 739: Total Reward = 9.0
Episode 740: Total Reward = 10.0
Episode 741: Total Reward = 8.0
Episode 742: Total Reward = 9.0
Episode 743: Total Reward = 9.0
Episode 744: Total Reward = 10.0
Episode 745: Total Reward = 9.0
Episode 746: Total Reward = 10.0
Episode 747: Total Reward = 9.0
Episode 748: Total Reward = 10.0
Episode 749: Total Reward = 9.0
Episode 750: Total Reward = 9.0
Episode 751: Total Reward = 10.0
```

```
Episode 752: Total Reward = 11.0
Episode 753: Total Reward = 10.0
Episode 754: Total Reward = 10.0
Episode 755: Total Reward = 10.0
Episode 756: Total Reward = 9.0
Episode 757: Total Reward = 10.0
Episode 758: Total Reward = 10.0
Episode 759: Total Reward = 9.0
Episode 760: Total Reward = 9.0
Episode 761: Total Reward = 9.0
Episode 762: Total Reward = 11.0
Episode 763: Total Reward = 9.0
Episode 764: Total Reward = 10.0
Episode 765: Total Reward = 10.0
Episode 766: Total Reward = 9.0
Episode 767: Total Reward = 9.0
Episode 768: Total Reward = 9.0
Episode 769: Total Reward = 9.0
Episode 770: Total Reward = 10.0
Episode 771: Total Reward = 10.0
Episode 772: Total Reward = 9.0
Episode 773: Total Reward = 10.0
Episode 774: Total Reward = 10.0
Episode 775: Total Reward = 9.0
Episode 776: Total Reward = 10.0
Episode 777: Total Reward = 8.0
Episode 778: Total Reward = 10.0
Episode 779: Total Reward = 10.0
Episode 780: Total Reward = 10.0
Episode 781: Total Reward = 8.0
Episode 782: Total Reward = 10.0
Episode 783: Total Reward = 10.0
Episode 784: Total Reward = 9.0
Episode 785: Total Reward = 10.0
Episode 786: Total Reward = 9.0
Episode 787: Total Reward = 8.0
Episode 788: Total Reward = 9.0
Episode 789: Total Reward = 10.0
Episode 790: Total Reward = 9.0
Episode 791: Total Reward = 9.0
Episode 792: Total Reward = 10.0
Episode 793: Total Reward = 10.0
Episode 794: Total Reward = 10.0
Episode 795: Total Reward = 9.0
Episode 796: Total Reward = 11.0
Episode 797: Total Reward = 9.0
Episode 798: Total Reward = 9.0
Episode 799: Total Reward = 9.0
Episode 800: Total Reward = 10.0
```

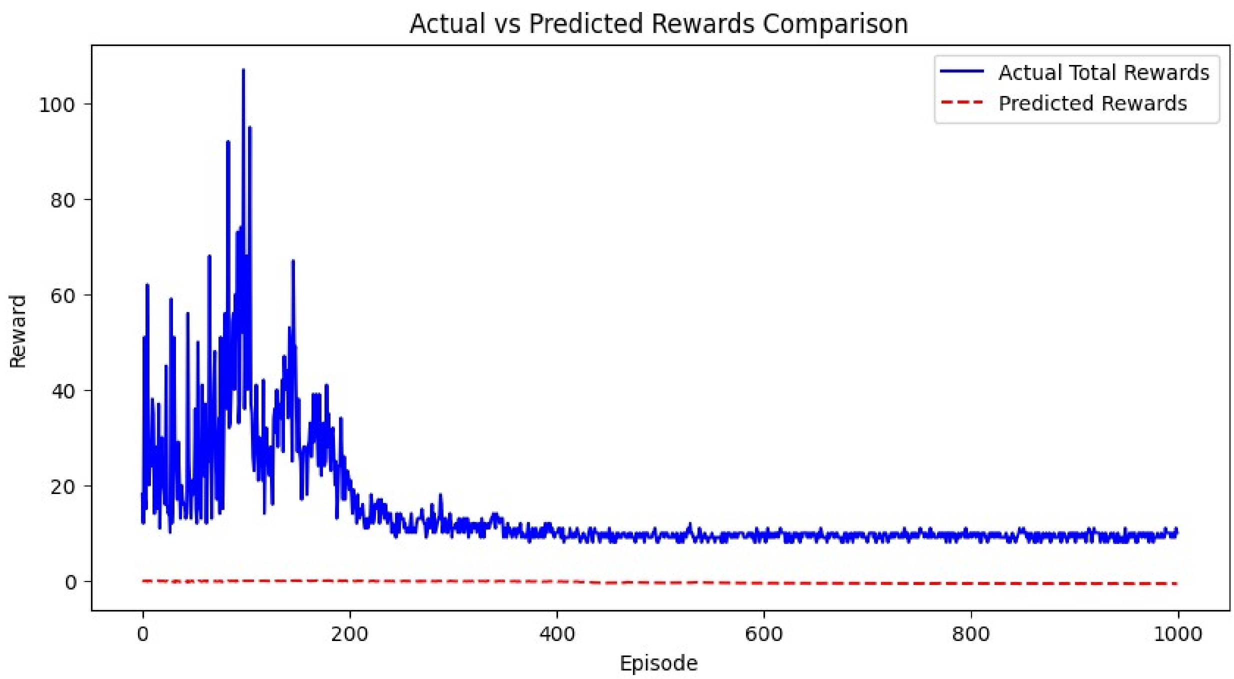
```
Episode 801: Total Reward = 10.0
Episode 802: Total Reward = 9.0
Episode 803: Total Reward = 10.0
Episode 804: Total Reward = 10.0
Episode 805: Total Reward = 10.0
Episode 806: Total Reward = 9.0
Episode 807: Total Reward = 10.0
Episode 808: Total Reward = 10.0
Episode 809: Total Reward = 10.0
Episode 810: Total Reward = 8.0
Episode 811: Total Reward = 8.0
Episode 812: Total Reward = 10.0
Episode 813: Total Reward = 10.0
Episode 814: Total Reward = 10.0
Episode 815: Total Reward = 9.0
Episode 816: Total Reward = 9.0
Episode 817: Total Reward = 8.0
Episode 818: Total Reward = 9.0
Episode 819: Total Reward = 10.0
Episode 820: Total Reward = 9.0
Episode 821: Total Reward = 10.0
Episode 822: Total Reward = 10.0
Episode 823: Total Reward = 9.0
Episode 824: Total Reward = 10.0
Episode 825: Total Reward = 9.0
Episode 826: Total Reward = 10.0
Episode 827: Total Reward = 10.0
Episode 828: Total Reward = 9.0
Episode 829: Total Reward = 10.0
Episode 830: Total Reward = 9.0
Episode 831: Total Reward = 9.0
Episode 832: Total Reward = 9.0
Episode 833: Total Reward = 9.0
Episode 834: Total Reward = 9.0
Episode 835: Total Reward = 10.0
Episode 836: Total Reward = 9.0
Episode 837: Total Reward = 8.0
Episode 838: Total Reward = 9.0
Episode 839: Total Reward = 8.0
Episode 840: Total Reward = 9.0
Episode 841: Total Reward = 10.0
Episode 842: Total Reward = 9.0
Episode 843: Total Reward = 9.0
Episode 844: Total Reward = 8.0
Episode 845: Total Reward = 9.0
Episode 846: Total Reward = 9.0
Episode 847: Total Reward = 10.0
Episode 848: Total Reward = 11.0
Episode 849: Total Reward = 10.0
```

```
Episode 850: Total Reward = 10.0
Episode 851: Total Reward = 10.0
Episode 852: Total Reward = 11.0
Episode 853: Total Reward = 10.0
Episode 854: Total Reward = 10.0
Episode 855: Total Reward = 9.0
Episode 856: Total Reward = 10.0
Episode 857: Total Reward = 8.0
Episode 858: Total Reward = 9.0
Episode 859: Total Reward = 10.0
Episode 860: Total Reward = 9.0
Episode 861: Total Reward = 9.0
Episode 862: Total Reward = 8.0
Episode 863: Total Reward = 10.0
Episode 864: Total Reward = 8.0
Episode 865: Total Reward = 10.0
Episode 866: Total Reward = 10.0
Episode 867: Total Reward = 10.0
Episode 868: Total Reward = 8.0
Episode 869: Total Reward = 8.0
Episode 870: Total Reward = 9.0
Episode 871: Total Reward = 10.0
Episode 872: Total Reward = 9.0
Episode 873: Total Reward = 9.0
Episode 874: Total Reward = 10.0
Episode 875: Total Reward = 9.0
Episode 876: Total Reward = 9.0
Episode 877: Total Reward = 10.0
Episode 878: Total Reward = 8.0
Episode 879: Total Reward = 9.0
Episode 880: Total Reward = 9.0
Episode 881: Total Reward = 10.0
Episode 882: Total Reward = 9.0
Episode 883: Total Reward = 9.0
Episode 884: Total Reward = 9.0
Episode 885: Total Reward = 9.0
Episode 886: Total Reward = 10.0
Episode 887: Total Reward = 10.0
Episode 888: Total Reward = 8.0
Episode 889: Total Reward = 10.0
Episode 890: Total Reward = 9.0
Episode 891: Total Reward = 10.0
Episode 892: Total Reward = 9.0
Episode 893: Total Reward = 9.0
Episode 894: Total Reward = 8.0
Episode 895: Total Reward = 9.0
Episode 896: Total Reward = 10.0
Episode 897: Total Reward = 9.0
Episode 898: Total Reward = 10.0
```

```
Episode 899: Total Reward = 10.0
Episode 900: Total Reward = 9.0
Episode 901: Total Reward = 10.0
Episode 902: Total Reward = 8.0
Episode 903: Total Reward = 10.0
Episode 904: Total Reward = 10.0
Episode 905: Total Reward = 9.0
Episode 906: Total Reward = 9.0
Episode 907: Total Reward = 9.0
Episode 908: Total Reward = 10.0
Episode 909: Total Reward = 9.0
Episode 910: Total Reward = 9.0
Episode 911: Total Reward = 10.0
Episode 912: Total Reward = 8.0
Episode 913: Total Reward = 10.0
Episode 914: Total Reward = 10.0
Episode 915: Total Reward = 11.0
Episode 916: Total Reward = 10.0
Episode 917: Total Reward = 10.0
Episode 918: Total Reward = 8.0
Episode 919: Total Reward = 10.0
Episode 920: Total Reward = 11.0
Episode 921: Total Reward = 10.0
Episode 922: Total Reward = 10.0
Episode 923: Total Reward = 10.0
Episode 924: Total Reward = 10.0
Episode 925: Total Reward = 10.0
Episode 926: Total Reward = 8.0
Episode 927: Total Reward = 9.0
Episode 928: Total Reward = 10.0
Episode 929: Total Reward = 9.0
Episode 930: Total Reward = 8.0
Episode 931: Total Reward = 10.0
Episode 932: Total Reward = 9.0
Episode 933: Total Reward = 8.0
Episode 934: Total Reward = 9.0
Episode 935: Total Reward = 9.0
Episode 936: Total Reward = 10.0
Episode 937: Total Reward = 9.0
Episode 938: Total Reward = 9.0
Episode 939: Total Reward = 10.0
Episode 940: Total Reward = 8.0
Episode 941: Total Reward = 9.0
Episode 942: Total Reward = 8.0
Episode 943: Total Reward = 10.0
Episode 944: Total Reward = 8.0
Episode 945: Total Reward = 9.0
Episode 946: Total Reward = 9.0
Episode 947: Total Reward = 8.0
```

```
Episode 948: Total Reward = 9.0
Episode 949: Total Reward = 8.0
Episode 950: Total Reward = 11.0
Episode 951: Total Reward = 8.0
Episode 952: Total Reward = 8.0
Episode 953: Total Reward = 9.0
Episode 954: Total Reward = 10.0
Episode 955: Total Reward = 9.0
Episode 956: Total Reward = 10.0
Episode 957: Total Reward = 9.0
Episode 958: Total Reward = 10.0
Episode 959: Total Reward = 10.0
Episode 960: Total Reward = 8.0
Episode 961: Total Reward = 9.0
Episode 962: Total Reward = 8.0
Episode 963: Total Reward = 10.0
Episode 964: Total Reward = 10.0
Episode 965: Total Reward = 10.0
Episode 966: Total Reward = 9.0
Episode 967: Total Reward = 10.0
Episode 968: Total Reward = 10.0
Episode 969: Total Reward = 9.0
Episode 970: Total Reward = 10.0
Episode 971: Total Reward = 9.0
Episode 972: Total Reward = 10.0
Episode 973: Total Reward = 8.0
Episode 974: Total Reward = 10.0
Episode 975: Total Reward = 9.0
Episode 976: Total Reward = 10.0
Episode 977: Total Reward = 8.0
Episode 978: Total Reward = 9.0
Episode 979: Total Reward = 9.0
Episode 980: Total Reward = 9.0
Episode 981: Total Reward = 10.0
Episode 982: Total Reward = 9.0
Episode 983: Total Reward = 9.0
Episode 984: Total Reward = 9.0
Episode 985: Total Reward = 10.0
Episode 986: Total Reward = 9.0
Episode 987: Total Reward = 9.0
Episode 988: Total Reward = 9.0
Episode 989: Total Reward = 11.0
Episode 990: Total Reward = 10.0
Episode 991: Total Reward = 10.0
Episode 992: Total Reward = 10.0
Episode 993: Total Reward = 10.0
Episode 994: Total Reward = 9.0
Episode 995: Total Reward = 10.0
Episode 996: Total Reward = 9.0
```

```
Episode 997: Total Reward = 10.0
Episode 998: Total Reward = 9.0
Episode 999: Total Reward = 11.0
Episode 1000: Total Reward = 10.0


RIO Price Prediction using A2C (Advantage Actor-Critic) :
```

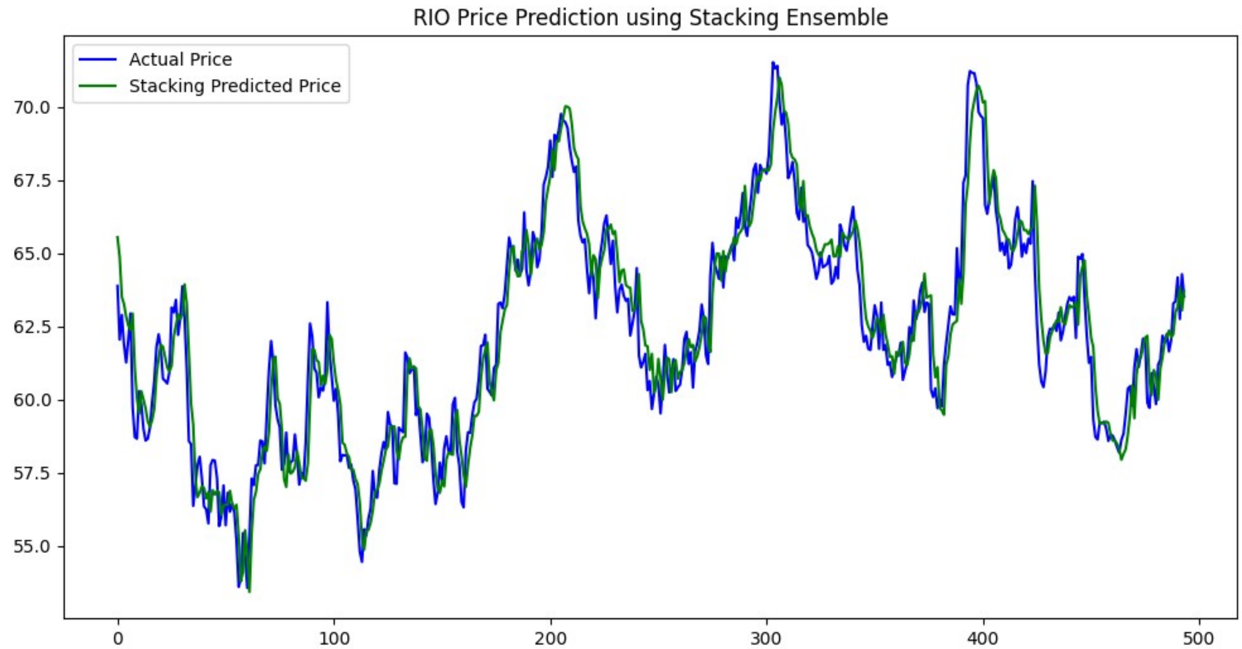



```
Conclusion :

Conclusion: The model suggests a SELL signal based on performance.
```

Stacking (Ensemble of ML models)

```
# Define base models
base_models = [
    ('ridge', Ridge()),
    ('decision_tree', DecisionTreeRegressor()),
    ('random_forest', RandomForestRegressor(n_estimators=100)),
    ('svr', SVR())
]

# Define meta model
meta_model = Ridge()
```

```
# Create stacking model
stacking_model = StackingRegressor(estimators=base_models,
final_estimator=meta_model)

# Train stacking model
stacking_model.fit(X_train.reshape(X_train.shape[0], -1), y_train)

# Predict using stacking model
y_pred_stacking =
stacking_model.predict(X_test.reshape(X_test.shape[0], -1))

# Calculate error
mse_stacking = mean_squared_error(y_test, y_pred_stacking)
print(f"\n\nStacking Model MSE: {mse_stacking}")

# Plot predictions
print(f"\n{stock_ticker} Price Prediction using Stacking Ensemble :\
n")
plt.figure(figsize=(12,6))
plt.plot(y_test_actual, label='Actual Price', color='blue')
plt.plot(scaler.inverse_transform(y_pred_stacking.reshape(-1, 1)),
label='Stacking Predicted Price', color='green')
plt.title(f"{stock_ticker} Price Prediction using Stacking Ensemble")
plt.legend()
plt.show()

# Conclusion
print("\nConclusion:\n")
if y_pred_stacking[-1] > y_test[-1]:
    print(f"Bullish Signal: The stacking model predicts an upward
trend in {stock_ticker}'s stock price. This suggests positive momentum
and potential growth, making it a favorable opportunity for
investors.")
    stacking_model = "Buy"
else:
    print(f"Bearish Signal: The stacking model predicts a downward
trend in {stock_ticker}'s stock price. This indicates possible market
corrections or a decline, urging investors to be cautious.")
    stacking_model = "Sell"



Stacking Model MSE: 0.00032006976044387316

RIO Price Prediction using Stacking Ensemble :
```




```
Conclusion:

Bullish Signal: The stacking model predicts an upward trend in RIO's
stock price. This suggests positive momentum and potential growth,
making it a favorable opportunity for investors.
```

CNN + LSTM model

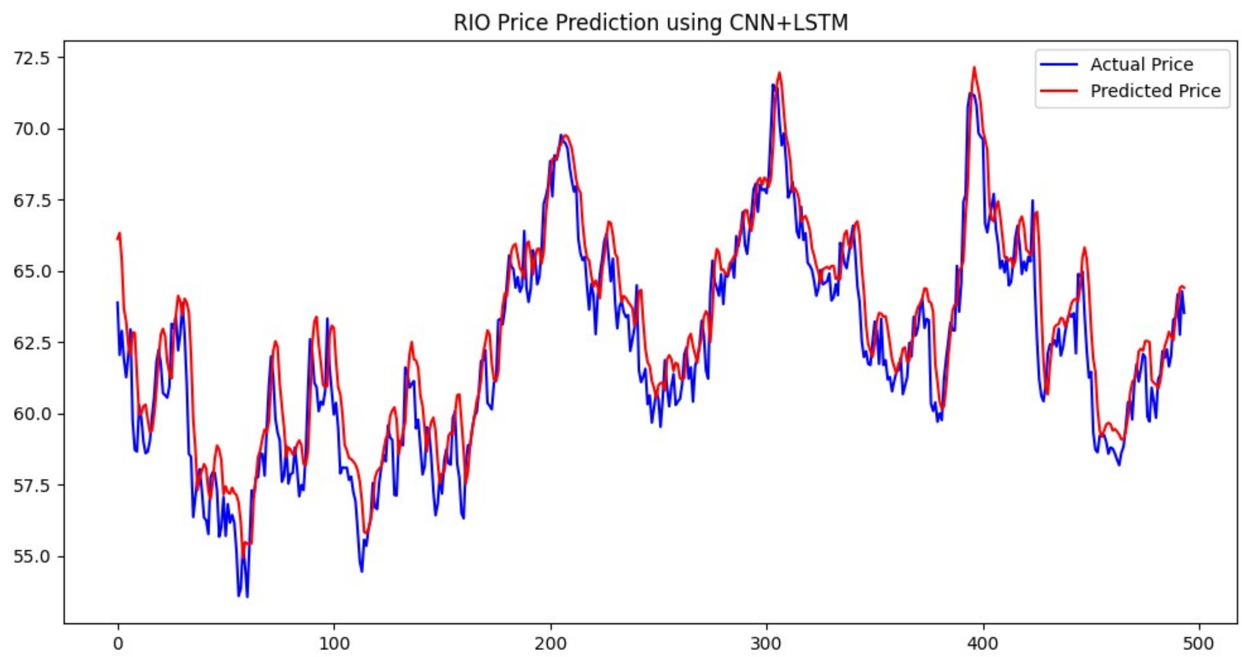
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense

print("\nCNN + LSTM :\n")

# Prepare data for CNN+LSTM model
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(history[['Close']].dropna())

# Function to create sequences for time series forecasting
def create_sequences(data, seq_length):
    sequences, labels = [], []
    for i in range(len(data) - seq_length):
        sequences.append(data[i:i+seq_length])
```

```python
        labels.append(data[i+seq_length])
    return np.array(sequences), np.array(labels)

seq_length = 50  # Sequence length for LSTM input
X, y = create_sequences(scaled_data, seq_length)
X = np.expand_dims(X, axis=2)  # Reshape for Conv1D

# Split data into training and testing sets
split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

# Define CNN + LSTM Model
model = Sequential([
    Conv1D(filters=64, kernel_size=3, activation='relu',
input_shape=(seq_length, 1)),
    MaxPooling1D(pool_size=2),
    LSTM(50, return_sequences=True),
    LSTM(50),
    Dense(25, activation='relu'),
    Dense(1)
])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train, y_train, epochs=20, batch_size=16,
validation_data=(X_test, y_test)) # Higher number for higher accuracy

# Make predictions
predictions = model.predict(X_test)
predictions = scaler.inverse_transform(predictions)  # Convert back to
original scale
y_test_actual = scaler.inverse_transform(y_test.reshape(-1, 1))

print(f"\n{stock_ticker} Price Prediction using CNN+LSTM\n")
# Plot actual vs predicted prices
plt.figure(figsize=(12,6))
plt.plot(y_test_actual, label='Actual Price', color='blue')
plt.plot(predictions, label='Predicted Price', color='red')
plt.title(f"{stock_ticker} Price Prediction using CNN+LSTM")
plt.legend()
plt.show()

# Calculate RMSE for performance evaluation
print("\nConclusion :\n")
rmse = np.sqrt(mean_squared_error(y_test_actual, predictions))
print(f"Root Mean Squared Error: {rmse}")
```

```python
# Buy/Sell Decision based on trend
if predictions[-1] > y_test_actual[-1]:
    decision = "BUY"
else:
    decision = "SELL"

print(f"Predicted trend suggests: {decision}\n\n")
cnn_lstm_model = decision
```

```
CNN + LSTM :

Epoch 1/20

/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

124/124 ━━━━━━━━━━━━━━━━━━━━ 10s 33ms/step - loss: 0.0889 - val_loss:
0.0035
Epoch 2/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 43ms/step - loss: 0.0023 - val_loss:
0.0019
Epoch 3/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 7s 52ms/step - loss: 0.0020 - val_loss:
0.0014
Epoch 4/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 7s 24ms/step - loss: 0.0015 - val_loss:
0.0021
Epoch 5/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 3s 23ms/step - loss: 0.0014 - val_loss:
0.0022
Epoch 6/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 23ms/step - loss: 0.0011 - val_loss:
0.0010
Epoch 7/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 24ms/step - loss: 0.0011 - val_loss:
0.0011
Epoch 8/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 28ms/step - loss: 0.0010 - val_loss:
0.0010
Epoch 9/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 23ms/step - loss: 9.1713e-04 -
val_loss: 8.0519e-04
Epoch 10/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 3s 23ms/step - loss: 8.0849e-04 -
```

```
val_loss: 8.9803e-04
Epoch 11/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 4s 33ms/step - loss: 6.3834e-04 -
val_loss: 0.0036
Epoch 12/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 3s 23ms/step - loss: 9.5491e-04 -
val_loss: 9.7588e-04
Epoch 13/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 3s 23ms/step - loss: 6.3372e-04 -
val_loss: 5.5615e-04
Epoch 14/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 6s 29ms/step - loss: 5.8022e-04 -
val_loss: 0.0020
Epoch 15/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 3s 25ms/step - loss: 6.2206e-04 -
val_loss: 0.0013
Epoch 16/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 3s 23ms/step - loss: 4.6952e-04 -
val_loss: 6.0878e-04
Epoch 17/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 3s 23ms/step - loss: 4.8739e-04 -
val_loss: 6.5485e-04
Epoch 18/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 3s 27ms/step - loss: 3.9659e-04 -
val_loss: 6.2789e-04
Epoch 19/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 23ms/step - loss: 4.4380e-04 -
val_loss: 7.2709e-04
Epoch 20/20
124/124 ━━━━━━━━━━━━━━━━━━━━ 3s 23ms/step - loss: 4.5337e-04 -
val_loss: 5.3946e-04
16/16 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step

RIO Price Prediction using CNN+LSTM
```




```
Conclusion :

Root Mean Squared Error: 1.399748250481022
Predicted trend suggests: BUY
```

Final Output

```
print(f"Final Decision for {stock_ticker} based on all analysis :\n")
print(f"Fundamental Analysis : {fundamental_analysis.split()
[0].upper()}")
print(f"Technical Analysis : {technical_analysis.upper()}")
print(f"Time Series Forecasting : {time_series_forecasting.upper()}")
print(f"Linear Regression : {linear_regression_model.upper()}")
print(f"Logistic Regression : {logistic_regression_model.upper()}")
print(f"Random Forest Regressor & Classifier :
{random_forest.upper()}")
print(f"Support Vector Machine (SVM) : {svm_model.upper()}")
print(f"XGBoost & LightGBM : {xgboost_lightgbm_model.upper()}")
print(f"Artificial Neural Networks (ANN) : {ann_model.upper()}")
print(f"Recurrent Neural Networks (RNN) : {rnn_model.upper()}")
print(f"Long Short-Term Memory (LSTM) : {lstm_model.upper()}")
print(f"Gated Recurrent Units (GRU) : {gru_model.upper()}")
print(f"Transformer Models : {transformer_model.upper()}")
print(f"SARIMA Model : {sarima_model.upper()}")
print(f"Prophet : {prophet_model.upper()}")
```

```
print(f"Kalman Filter : {kalman_filter_model.upper()}")
print(f"Deep Q-Network (DQN) : {dqn_model.upper()}")
print(f"Proximal Policy Optimization (PPO) : {ppo_model.upper()}")
print(f"A2C (Advantage Actor-Critic) : {a2c_model.upper()}")
print(f"Stacking Ensemble : {stacking_model.upper()}")
print(f"CNN + LSTM : {cnn_lstm_model.upper()}")
```

```
Final Decision for RIO based on all analysis :

Fundamental Analysis : BUY
Technical Analysis : SELL
Time Series Forecasting : SELL
Linear Regression : BUY
Logistic Regression : BUY
Random Forest Regressor & Classifier : SELL
Support Vector Machine (SVM) : BUY
XGBoost & LightGBM : SELL
Artificial Neural Networks (ANN) : SELL
Recurrent Neural Networks (RNN) : SELL
Long Short-Term Memory (LSTM) : SELL
Gated Recurrent Units (GRU) : HOLD
Transformer Models : SELL
SARIMA Model : SELL
Prophet : BUY
Kalman Filter : SELL
Deep Q-Network (DQN) : SELL
Proximal Policy Optimization (PPO) : BUY
A2C (Advantage Actor-Critic) : SELL
Stacking Ensemble : BUY
CNN + LSTM : BUY
```

```
# Store all models' predictions in a list
model_predictions = [
    fundamental_analysis.split()[0].upper(),
technical_analysis.upper(), time_series_forecasting.upper(),
    linear_regression_model.upper(),
logistic_regression_model.upper(), random_forest.upper(),
    svm_model.upper(), xgboost_lightgbm_model.upper(),
ann_model.upper(), rnn_model.upper(), lstm_model.upper(),
    gru_model.upper(), transformer_model.upper(),
sarima_model.upper(), prophet_model.upper(),
    kalman_filter_model.upper(), dqn_model.upper(), ppo_model.upper(),
a2c_model.upper(),
    stacking_model.upper(), cnn_lstm_model.upper()
]

# Count occurrences of BUY, SELL, and HOLD
buy_count = sum(1 for model in model_predictions if model.upper() ==
"BUY")
sell_count = sum(1 for model in model_predictions if model.upper() ==
```

```
"SELL")
hold_count = sum(1 for model in model_predictions if model.upper() ==
"HOLD")

# Calculate total models used
total_models = len(model_predictions)

# Calculate probabilities
buy_prob = (buy_count / total_models) * 100
sell_prob = (sell_count / total_models) * 100
hold_prob = (hold_count / total_models) * 100

# Determine final decision
if buy_prob > sell_prob and buy_prob > hold_prob:
    final_decision = "BUY"
elif sell_prob > buy_prob and sell_prob > hold_prob:
    final_decision = "SELL"
else:
    final_decision = "HOLD"

# Print final decision with probability breakdown
print(f"Final Decision for {stock_ticker} based on all analysis:\n")
print(f"Fundamental Analysis : {fundamental_analysis.split()
[0].upper()}")
print(f"Technical Analysis : {technical_analysis.upper()}")
print(f"Time Series Forecasting : {time_series_forecasting.upper()}")
print(f"Linear Regression : {linear_regression_model.upper()}")
print(f"Logistic Regression : {logistic_regression_model.upper()}")
print(f"Random Forest Regressor & Classifier :
{random_forest.upper()}")
print(f"Support Vector Machine (SVM) : {svm_model.upper()}")
print(f"XGBoost & LightGBM : {xgboost_lightgbm_model.upper()}")
print(f"Artificial Neural Networks (ANN) : {ann_model.upper()}")
print(f"Recurrent Neural Networks (RNN) : {rnn_model.upper()}")
print(f"Long Short-Term Memory (LSTM) : {lstm_model.upper()}")
print(f"Gated Recurrent Units (GRU) : {gru_model.upper()}")
print(f"Transformer Models : {transformer_model.upper()}")
print(f"SARIMA Model : {sarima_model.upper()}")
print(f"Prophet : {prophet_model.upper()}")
print(f"Kalman Filter : {kalman_filter_model.upper()}")
print(f"Deep Q-Network (DQN) : {dqn_model.upper()}")
print(f"Proximal Policy Optimization (PPO) : {ppo_model.upper()}")
print(f"A2C (Advantage Actor-Critic) : {a2c_model.upper()}")
print(f"Stacking Ensemble : {stacking_model.upper()}")
print(f"CNN + LSTM : {cnn_lstm_model.upper()}")

print("\nDecision Summary :\n")
print(f"Total BUY recommendations: {buy_count}")
print(f"Total SELL recommendations: {sell_count}")
print(f"Total HOLD recommendations: {hold_count}")
```

```
print(f"Total Models used: {total_models}\n")
print(f"\nProbability of BUY : {buy_prob:.2f}%")
print(f"Probability of SELL : {sell_prob:.2f}%")
print(f"Probability of HOLD : {hold_prob:.2f}%\n")
print(f"Final Conclusion: The overall recommendation is to
{final_decision} the stock.")

print("\nWarning: This is not financial advice. Please conduct your
own research before making investment decisions.")
```

```
Final Decision for RIO based on all analysis:

Fundamental Analysis : BUY
Technical Analysis : SELL
Time Series Forecasting : SELL
Linear Regression : BUY
Logistic Regression : BUY
Random Forest Regressor & Classifier : SELL
Support Vector Machine (SVM) : BUY
XGBoost & LightGBM : SELL
Artificial Neural Networks (ANN) : SELL
Recurrent Neural Networks (RNN) : SELL
Long Short-Term Memory (LSTM) : SELL
Gated Recurrent Units (GRU) : HOLD
Transformer Models : SELL
SARIMA Model : SELL
Prophet : BUY
Kalman Filter : SELL
Deep Q-Network (DQN) : SELL
Proximal Policy Optimization (PPO) : BUY
A2C (Advantage Actor-Critic) : SELL
Stacking Ensemble : BUY
CNN + LSTM : BUY

Decision Summary :

Total BUY recommendations: 8
Total SELL recommendations: 12
Total HOLD recommendations: 1
Total Models used: 21


Probability of BUY : 38.10%
Probability of SELL : 57.14%
Probability of HOLD : 4.76%

Final Conclusion: The overall recommendation is to SELL the stock.

Warning: This is not financial advice. Please conduct your own
research before making investment decisions.
```