Task 2: Predict Future Stock Prices

Import Necessary Libraries

```
# Install yfinance
!pip install yfinance

# Import necessary libraries
import pandas as pd
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns

# For machine learning
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

```
Requirement already satisfied: yfinance in /usr/local/lib/python3.11/dist-packages (0.2.63)
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.2.2)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.0.2)
Requirement already satisfied: requests>=2.31 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.32.3)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.11/dist-packages (from yfinance) (0.0.11)
Requirement already satisfied: platformdirs>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from yfinance) (4.3.8)
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2025.2)
Requirement already satisfied: frozendict>=2.3.4 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.4.6)
Requirement already satisfied: peewee>=3.16.2 in /usr/local/lib/python3.11/dist-packages (from yfinance) (3.18.1)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /usr/local/lib/python3.11/dist-packages (from yfinance) (4.13
Requirement already satisfied: curl_cffi>=0.7 in /usr/local/lib/python3.11/dist-packages (from yfinance) (0.11.3)
Requirement already satisfied: protobuf>=3.19.0 in /usr/local/lib/python3.11/dist-packages (from yfinance) (5.29.5)
Requirement already satisfied: websockets>=13.0 in /usr/local/lib/python3.11/dist-packages (from yfinance) (15.0.1)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4>=4.11.1->
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup
Requirement already satisfied: cffi>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from curl_cffi>=0.7->yfinance)
Requirement already satisfied: certifi>=2024.2.2 in /usr/local/lib/python3.11/dist-packages (from curl_cffi>=0.7->yfir
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.3.0->
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.3.0->yfinance
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.3
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31->yfinance)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31->yfi
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (from cffi>=1.12.0->curl_cffi>=0.7
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->panda
```

Data Download

```
# Downloading Apple stock data for last 2 years
ticker = 'AAPL'
df = yf.download(ticker, period='2y')
print(df.head())
```

```
/tmp/ipython-input-2-558416272.py:3: FutureWarning: YF.download() has changed argument auto_adjust default to True
  df = yf.download(ticker, period='2y')
[*********************100%***********************]  1 of 1 completedPrice            Close        High         Low
Ticker            AAPL        AAPL        AAPL        AAPL      AAPL
Date
2023-06-21  182.136414  183.572037  180.779984  183.067083  49515700
2023-06-22  185.146255  185.195763  181.849264  181.918578  51245300
2023-06-23  184.829422  185.700703  183.175979  183.710634  53079300
2023-06-26  183.433411  186.185851  183.393799  184.977944  48088700
2023-06-27  186.195755  186.522486  183.829448  184.047268  50730800
```

Data Preparation

```
# Select relevant columns
df = df[['Open', 'High', 'Low', 'Close', 'Volume']]

# Target: next day's Close
df['Next_Close'] = df['Close'].shift(-1)

# Remove last row (target is NaN)
df = df.dropna()
print(df.head())
```

```
Price             Open        High         Low       Close    Volume  \
Ticker            AAPL        AAPL        AAPL        AAPL      AAPL
Date
2023-06-21  183.067083  183.572037  180.779984  182.136414  49515700
2023-06-22  181.918578  185.195763  181.849264  185.146255  51245300
2023-06-23  183.710634  185.700703  183.175979  184.829422  53079300
2023-06-26  184.977944  186.185851  183.393799  183.433411  48088700
2023-06-27  184.047268  186.522486  183.829448  186.195755  50730800

Price       Next_Close
Ticker
Date
2023-06-21  185.146255
2023-06-22  184.829422
2023-06-23  183.433411
2023-06-26  186.195755
2023-06-27  187.373962
/tmp/ipython-input-3-2354445125.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
  df['Next_Close'] = df['Close'].shift(-1)
```

Train/Test Split

```
# Features and target
X = df[['Open', 'High', 'Low', 'Volume']]
y = df['Next_Close']

# Split: 80% train, 20% test (chronological order for time series)
split = int(0.8 * len(df))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]
```

Train Model (Linear Regression & Random Forest)

Linear Regression:

```
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# Predict
y_pred_lr = lr_model.predict(X_test)
```

Random Forest

```
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

```
# Predict
y_pred_rf = rf_model.predict(X_test)
```

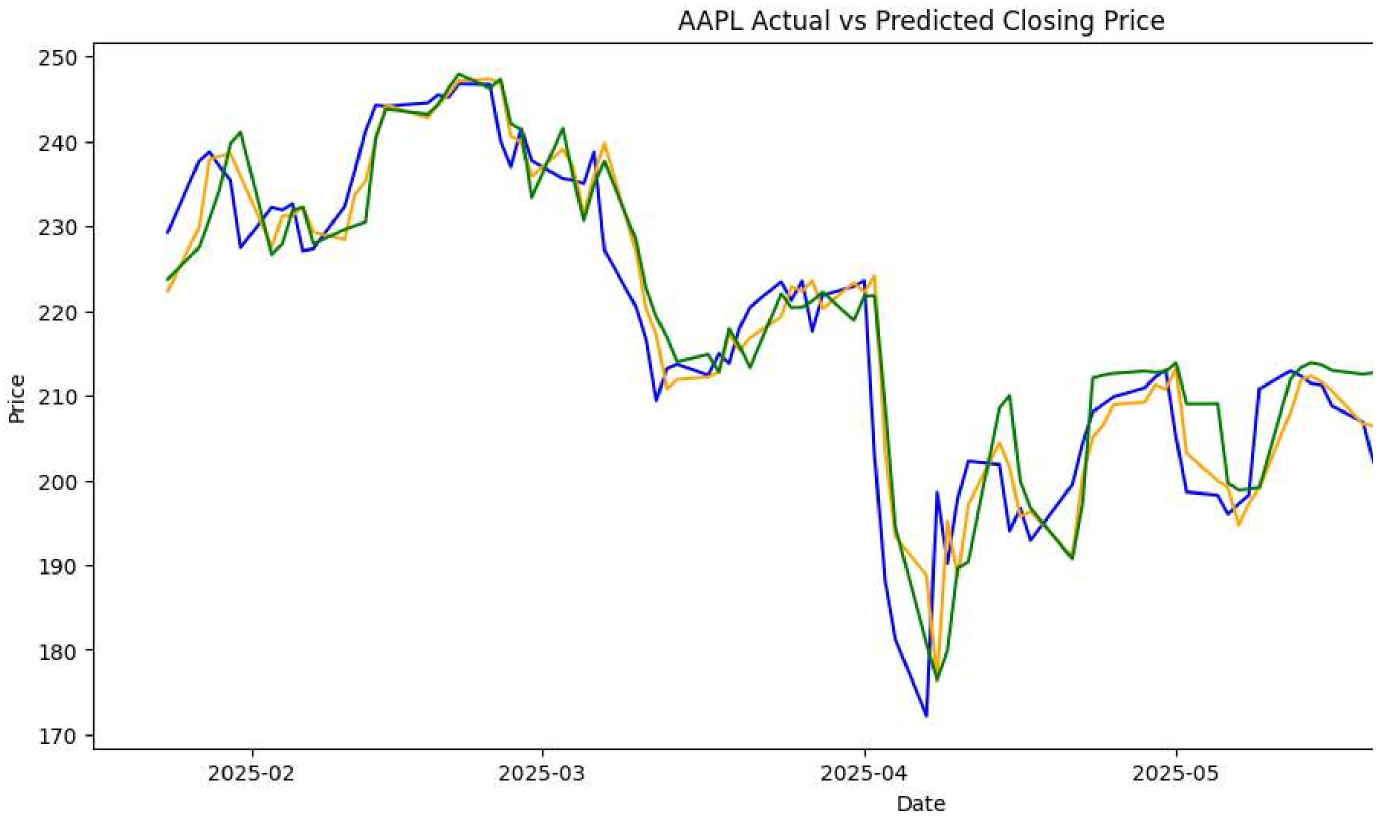
Evaluate Models

```
def print_metrics(y_true, y_pred, model_name):
    print(f"\nResults for {model_name}:")
    print("MSE:", mean_squared_error(y_true, y_pred))
    print("R2 Score:", r2_score(y_true, y_pred))

print_metrics(y_test, y_pred_lr, "Linear Regression")
print_metrics(y_test, y_pred_rf, "Random Forest")
```

```
Results for Linear Regression:
MSE: 31.55040567196701
R2 Score: 0.8931884457816126

Results for Random Forest:
MSE: 55.52636424029065
R2 Score: 0.8120196194538511
```

Plot Actual vs Predicted Prices

```
plt.figure(figsize=(14,6))
plt.plot(y_test.index, y_test, label='Actual Close', color='blue')
plt.plot(y_test.index, y_pred_lr, label='Predicted Close (LR)', color='orange')
plt.plot(y_test.index, y_pred_rf, label='Predicted Close (RF)', color='green')
plt.title(f"{ticker} Actual vs Predicted Closing Price")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
plt.show()
```



Explanation of Results and Final Insights:

By downloading historical stock data and building regression models (Linear Regression and Random Forest), we attempted to predict the next day's closing price using features like Open, High, Low, and Volume. The predicted prices were compared to the actual prices using line plots. The results show that the models can capture the overall trend of the stock's movement, but there is still some deviation, especially during volatile periods. The accuracy metrics (MSE and R2-score) give a quantitative measure of model performance. In summary, while basic models can provide reasonable short-term predictions, real-world forecasting remains challenging due to the noisy and unpredictable nature of financial markets.