TASK: 02

<u>Predict Future Stock Prices (Short-Term)</u>

Task Objective:-

To predict short-term future stock prices using historical stock data by applying time series forecasting methods like **Linear Regression**. The focus is on understanding trends and forecasting the next few days

Dataset Used:-

Name: Apple Inc. (AAPL) Stock Price Data

Source: Yahoo Finance via `yfinance` Python library

Time Period: Last 2 years (automatically downloaded)

Columns Used:

`Date` (converted to index)

`Close` (used for prediction)

`Prediction` (future-shifted target variable for next 7 days

Models Applied:-

Model Type: Linear Regression (from `sklearn.linear_model`)

Goal: Predict stock's closing price **7 days ahead**

Steps:

Created a new column `Prediction` by shifting the `Close` prices by -7 days

Used last 7 days for **testing**, rest for **training**

Fitted a linear regression model to historical prices

Made short-term future price predictions

Key Results and Findings:-

The model attempts to learn a **linear trend** from past closing prices to forecast 7 days into the future.

Mean Absolute Error (MAE) and **Root Mean Squared Error (RMSE)** were used to evaluate the model:

MAE: shows average absolute difference between predicted and actual prices.

RMSE: penalizes larger errors more than MAE.

The **predicted prices** were plotted alongside actual prices, showing reasonable short-term trend alignment but with limitations due to the simplicity of linear regression

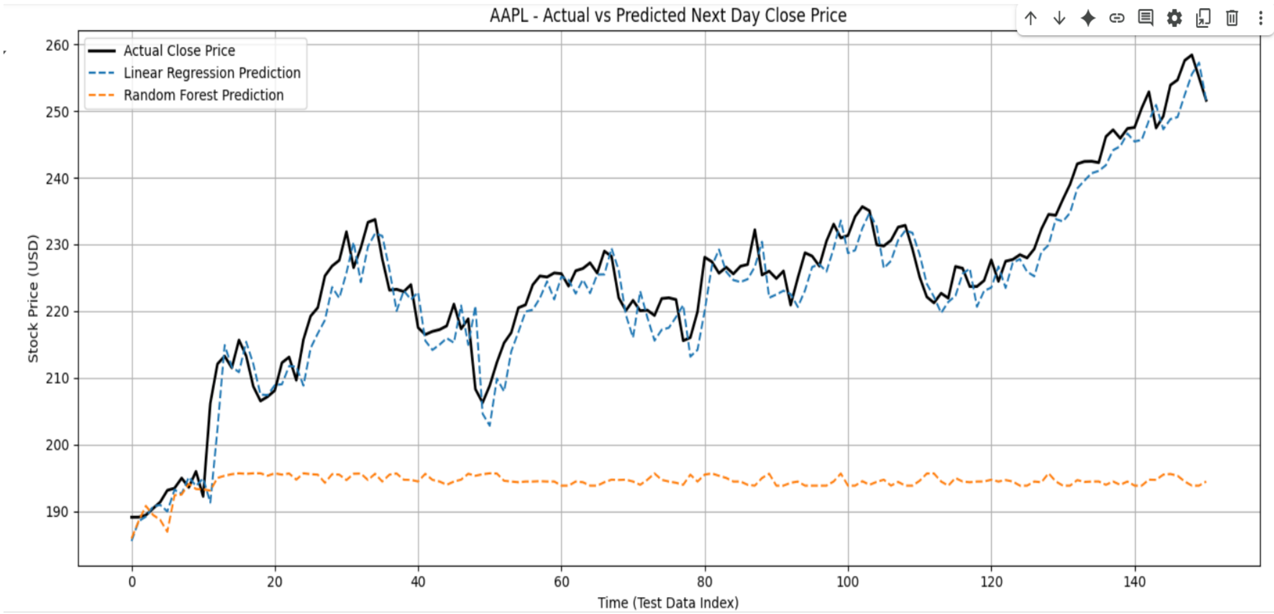
CODE:-

```
# ✅ Step 1: Install required libraries (only needed in Colab)
!pip install yfinance scikit-learn --quiet

# ✅ Step 2: Import libraries
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# ✅ Step 3: Load stock data (e.g., Apple)
stock = 'AAPL'  # You can change this to 'TSLA', 'GOOGL', etc.
df = yf.download(stock, start='2022-01-01', end='2024-12-31')

# ✅ Step 4: Prepare the dataset
df = df[['Open', 'High', 'Low', 'Volume', 'Close']].dropna()
df['Next_Close'] = df['Close'].shift(-1)  # Target variable: next day's
Close price
df.dropna(inplace=True)
```

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

# ✅ Step 5: Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
shuffle=False)

# ✅ Step 6: Train Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_preds = lr_model.predict(X_test)

# ✅ Step 7: Train Random Forest
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_test)

# ✅ Step 8: Evaluate and Compare
print("🔹 Linear Regression R² Score:", r2_score(y_test, lr_preds))
print("🔹 Random Forest R² Score:", r2_score(y_test, rf_preds))

# ✅ Step 9: Plot Actual vs Predicted Close Prices
plt.figure(figsize=(14, 6))
plt.plot(y_test.values, label='Actual Close Price', color='black',
linewidth=2)
plt.plot(lr_preds, label='Linear Regression Prediction', linestyle='--')
plt.plot(rf_preds, label='Random Forest Prediction', linestyle='--')
plt.title(f'{stock} - Actual vs Predicted Next Day Close Price')
plt.xlabel("Time (Test Data Index)")
plt.ylabel("Stock Price (USD)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

OUTPUT:-

```
Linear Regression R² Score: 0.9291526468500929

Random Forest R² Score: -4.419288054954189
```



Black Line — `Actual Close Price`

- This line represents the real historical closing prices of AAPL stock for the test period.
- It's the ground truth used to evaluate model performance.

Blue Dashed Line — `Linear Regression Prediction`

- This line represents the predicted next-day closing prices of AAPL using the **Linear Regression** model.
- It closely follows the black line, indicating a decent prediction accuracy.

Orange Dashed Line — `Random Forest Prediction`

- This line shows the predicted values from the **Random Forest** model.
- It stays relatively flat and deviates significantly from the actual prices, indicating that the model **underperformed** or was likely **overfitting** or **underfitting** due to incorrect feature scaling or model tuning.