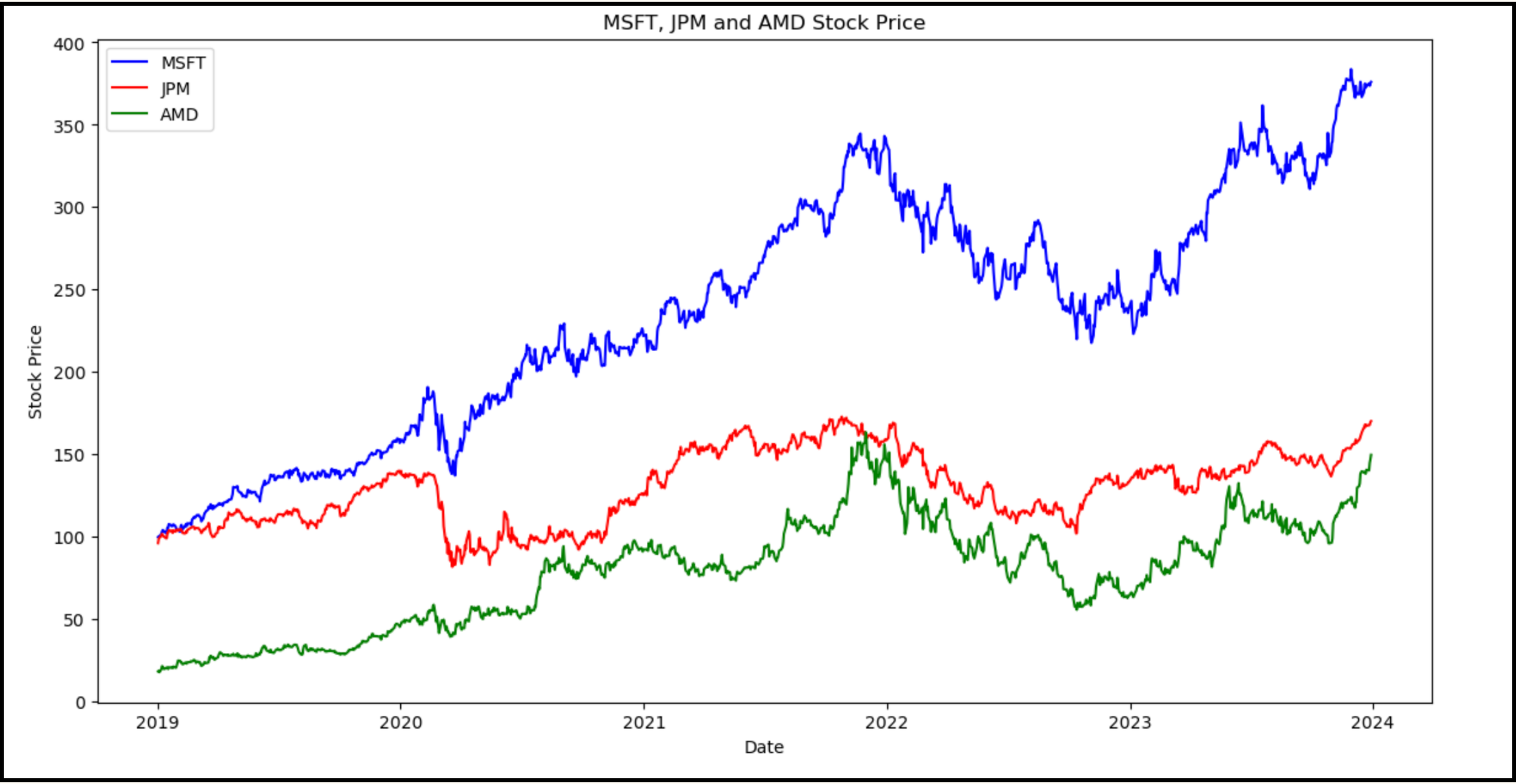
SỬ DỤNG RNN DỰ ĐOÁN GIÁ CỔ PHIẾU

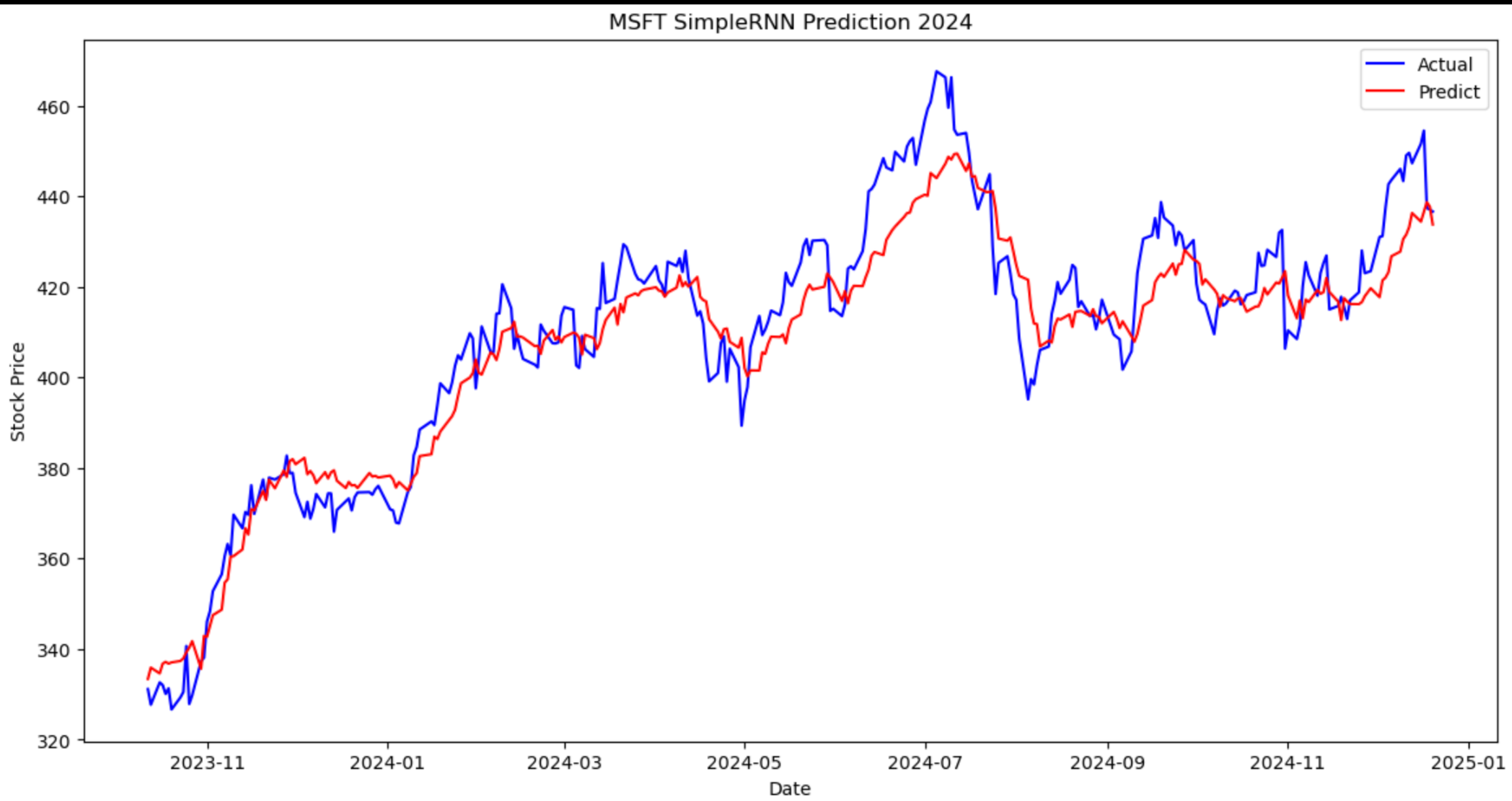

J.P.Morgan


AMD
AMD

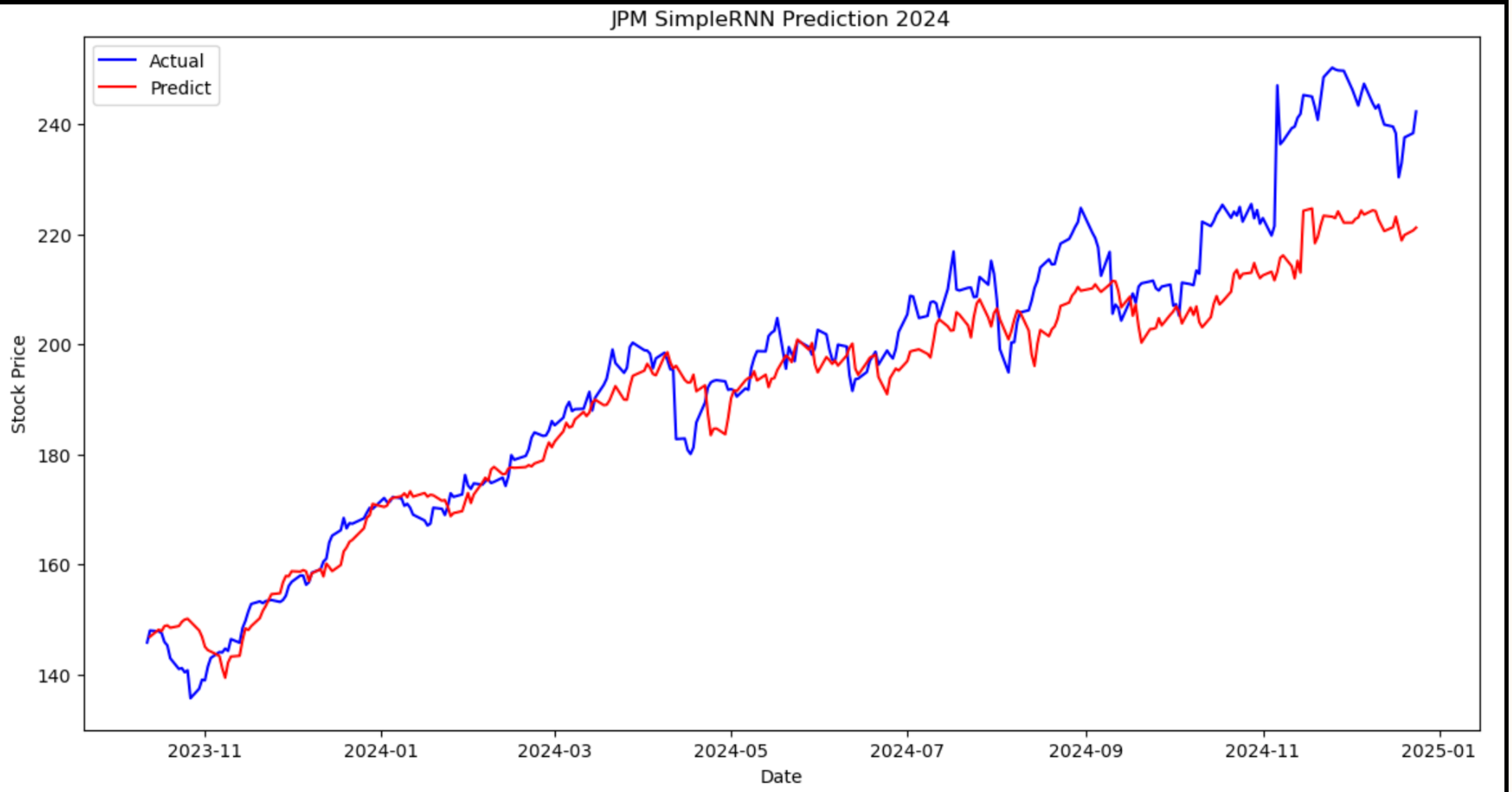

MSFT, JPM and AMD Stock Price
MSFT
JPM
AMD
Stock Price
Date
0
50
100
150
200
250
300
350
400
2019
2020
2021
2022
2023
2024

```
data = yf.download('MSFT', start='2019-01-01', end='2023-12-31')['Close']
scaler = MinMaxScaler(feature_range=(0, 1))
data_scaled = scaler.fit_transform(data.values.reshape(-1, 1))

def create_dataset(dataset, time_step=900):
    X, y = [], []
    for i in range(len(dataset) - time_step):
        X.append(dataset[i:i + time_step, 0])
        y.append(dataset[i + time_step, 0])
    return np.array(X), np.array(y)

time_step = 900
train_size = int(len(data_scaled) * 0.8)
X_train, y_train = create_dataset(data_scaled[:train_size], time_step)
X_test, y_test = create_dataset(data_scaled[train_size - time_step:], time_step)
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

model = Sequential()
model.add(SimpleRNN(50, return_sequences=True, input_shape=(time_step, 1)))
model.add(SimpleRNN(50))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=1)

train_pred = scaler.inverse_transform(model.predict(X_train))
test_pred = scaler.inverse_transform(model.predict(X_test))
y_train_actual = scaler.inverse_transform(y_train.reshape(-1, 1))
y_test_actual = scaler.inverse_transform(y_test.reshape(-1, 1))

future_days = 365
future_predictions = []
last_900_days = data_scaled[-time_step:]
for _ in range(future_days):
    pred = model.predict(last_60_days.reshape(1, -1, 1))[0, 0]
    future_predictions.append(pred)
    last_900_days = np.append(last_900_days[1:], pred)

future_predictions = scaler.inverse_transform(np.array(future_predictions).reshape(-1, 1))
future_dates = [datetime(2024, 1, 1) + timedelta(days=i) for i in range(future_days)]
```

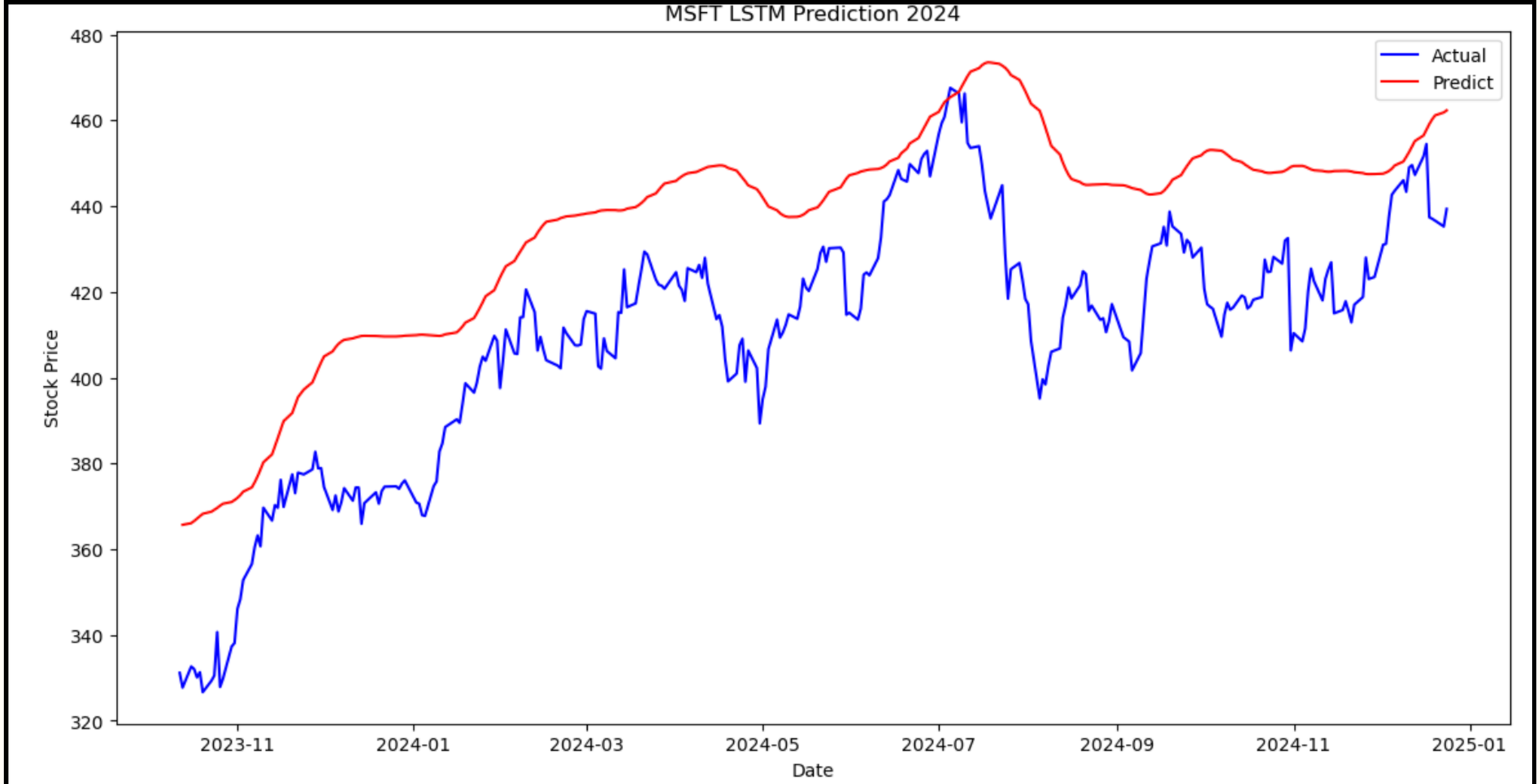
	Date	Actual	Predicted
0	2023-10-12	331.1600036621094	330.89014
1	2023-10-13	327.7300109863281	332.12347
2	2023-10-16	332.6400146484375	330.53775
3	2023-10-17	332.05999755859375	332.6437
4	2023-10-18	330.10998535156244	334.4453
5	2023-10-19	331.32000732421875	333.49127
6	2023-10-20	326.6700134277344	332.8676
7	2023-10-23	329.32000732421875	331.85278
8	2023-10-24	330.5299987792969	333.08197
9	2023-10-25	340.6700134277343	332.88623
10	2023-10-26	327.89001464843744	335.469
11	2023-10-27	329.80999755859375	332.5684
12	2023-10-30	337.3099975585937	334.07642
13	2023-10-31	338.10998535156244	334.4458
14	2023-11-01	346.0700073242187	337.2557
15	2023-11-02	348.32000732421875	342.36346
16	2023-11-03	352.7999877929687	344.2895
17	2023-11-06	356.5299987792969	346.41403
18	2023-11-07	360.5299987792968	351.73578
19	2023-11-08	363.2000122070312	355.39615
280	2024-11-21	412.8699951171875	417.02945
281	2024-11-22	417.00000000000006	412.34613
282	2024-11-25	418.7900085449219	412.49527
283	2024-11-26	427.989990234375	413.084
284	2024-11-27	422.989990234375	417.30798
285	2024-11-29	423.4599914550781	416.01352
286	2024-12-02	430.9800109863281	416.61044
287	2024-12-03	431.2000122070312	420.1315
288	2024-12-04	437.42001342773443	422.83984
289	2024-12-05	442.61999511718756	424.61578
290	2024-12-06	443.57000732421875	429.73215
291	2024-12-09	446.0199890136719	433.03714
292	2024-12-10	443.3299865722656	433.40927
293	2024-12-11	448.989990234375	433.68805
294	2024-12-12	449.5599975585938	435.85788
295	2024-12-13	447.26998901367193	438.75858
296	2024-12-16	451.5899963378906	438.84396
297	2024-12-17	454.4599914550781	439.76096
298	2024-12-18	437.39001464843744	441.61795
299	2024-12-19	437.0299987792969	438.56128
300	2024-12-20	441.5199890136719	434.86093



MAE: 7.041071083854601
MSE: 74.88445688472551
RMSE: 8.653580581743347
R^2: 0.9117124009808832
MAPE: 1.7016945418727436%
Adjusted R^2: 0.9114171247299832



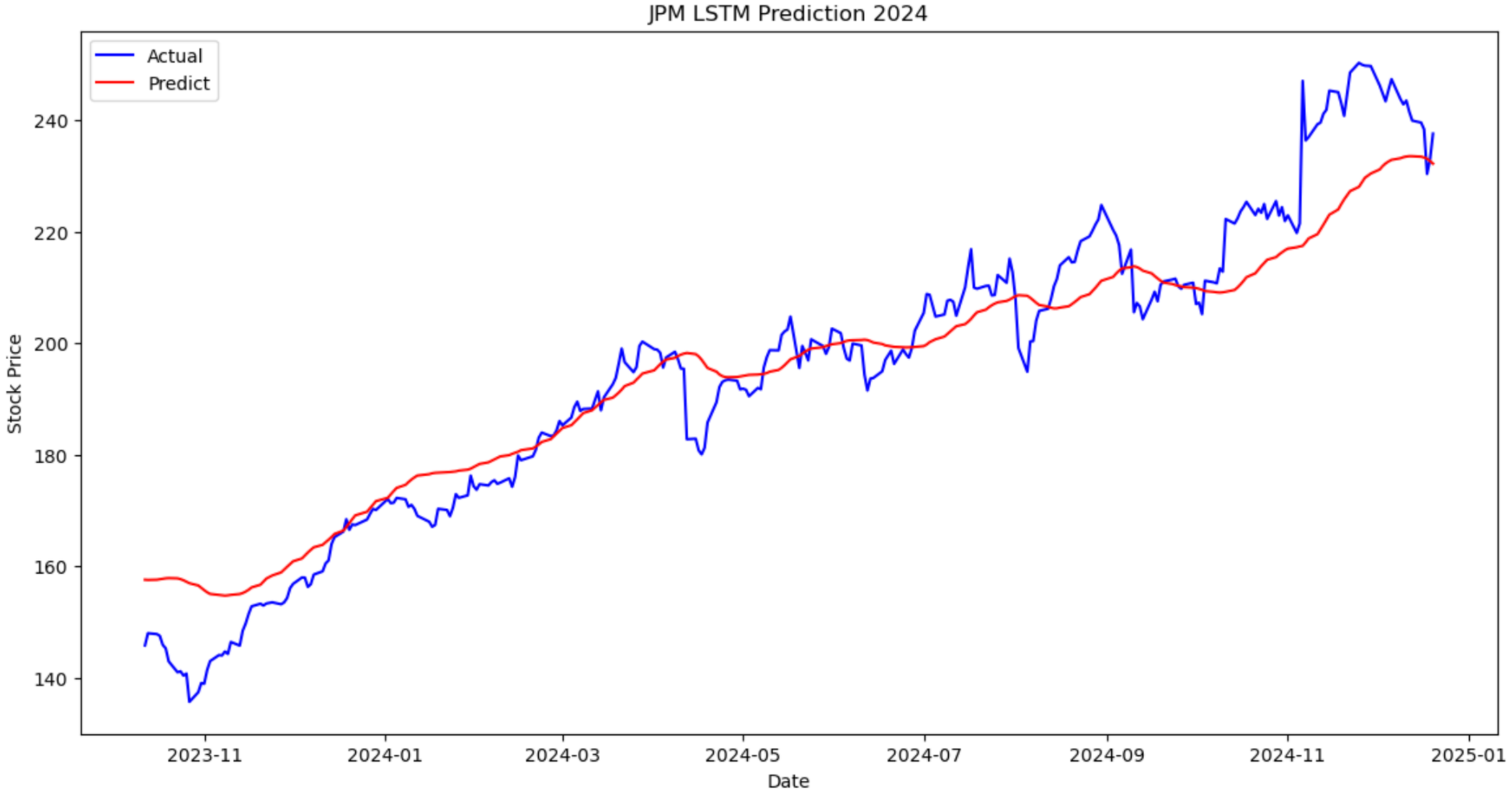
MAE: 6.968082049035079
MSE: 95.01233989432674
RMSE: 9.747427347476192
R^2: 0.8771391629985064
MAPE: 3.326556098397191%
Adjusted R^2: 0.8767296268751681



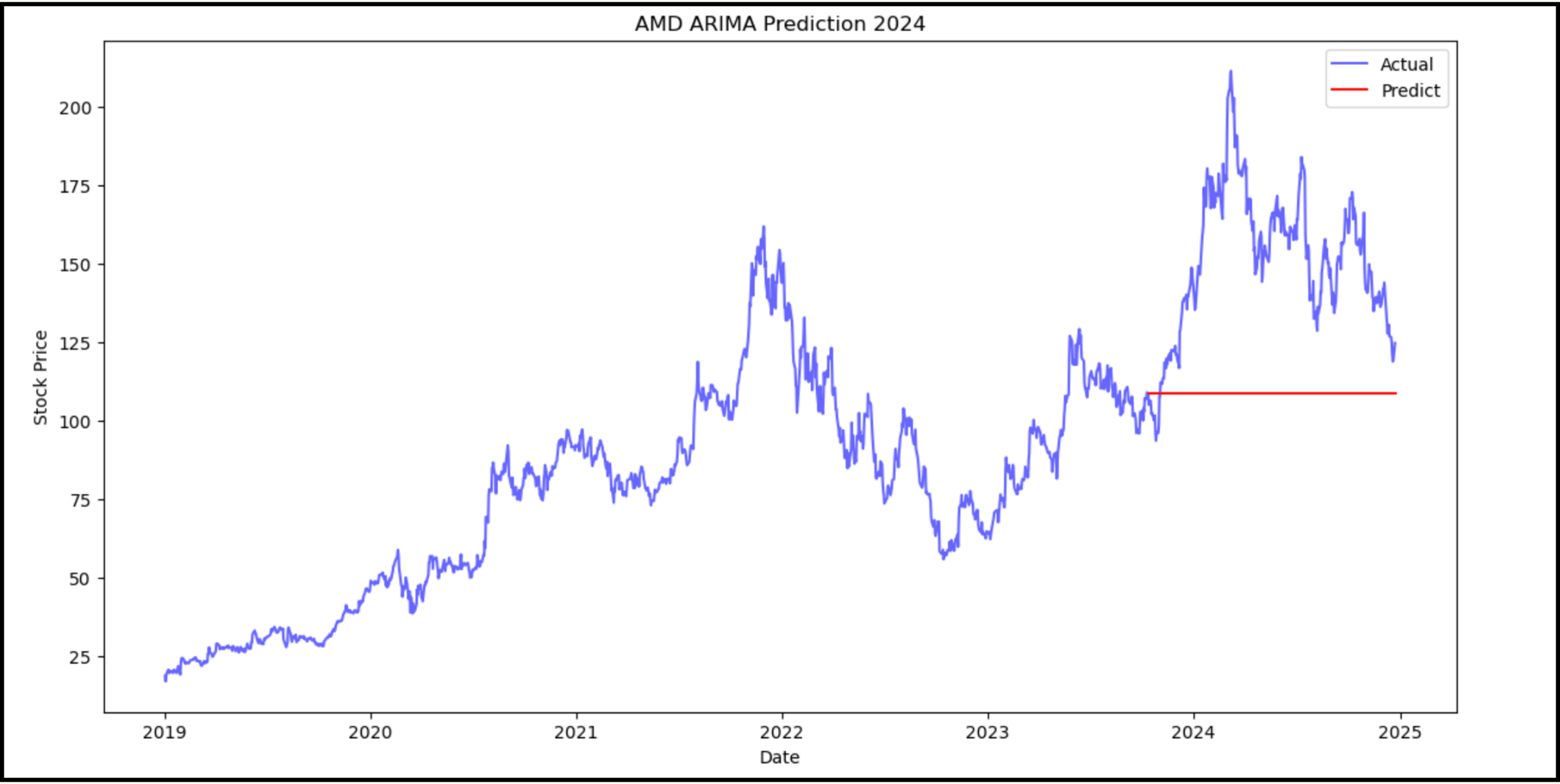
MAE: 26.081211014299207
MSE: 826.2470255698424
RMSE: 28.74451296456147
R^2: 0.004871041882598792
MAPE: 6.500049553809328%
Adjusted R^2: 0.0015539453555407645

```
data = yf.download('JPM', start='2019-01-01', end='2024-12-31')['Close']
scaler = MinMaxScaler(feature_range=(0, 1))
data_scaled = scaler.fit_transform(data.values.reshape(-1, 1))

def create_dataset(dataset, time_step=1200):
    X, y = [], []
    for i in range(len(dataset) - time_step):
        X.append(dataset[i:i + time_step, 0])
        y.append(dataset[i + time_step, 0])
    return np.array(X), np.array(y)

time_step = 1200
train_size = int(len(data_scaled) * 0.8)
X_train, y_train = create_dataset(data_scaled[:train_size], time_step)
X_test, y_test = create_dataset(data_scaled[train_size - time_step:], time_step)
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(time_step, 1)))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=1)

train_pred = scaler.inverse_transform(model.predict(X_train))
test_pred = scaler.inverse_transform(model.predict(X_test))
y_train_actual = scaler.inverse_transform(y_train.reshape(-1, 1))
y_test_actual = scaler.inverse_transform(y_test.reshape(-1, 1))

future_days = 365
future_predictions = []
last_900_days = data_scaled[-time_step:]
for _ in range(future_days):
    pred = model.predict(last_900_days.reshape(1, -1, 1), verbose=0)[0, 0]
    future_predictions.append(pred)
    last_900_days = np.append(last_900_days[1:], pred)

future_predictions = scaler.inverse_transform(np.array(future_predictions).reshape(-1, 1))
future_dates = [datetime(2024, 1, 1) + timedelta(days=i) for i in range(future_days)]
```

	Date	Actual	Predict
0	2023-10-12	145.80999755859375	161.38556
1	2023-10-13	148.0	161.31401
2	2023-10-16	147.85000610351562	161.30019
3	2023-10-17	147.52999877929688	161.33582
4	2023-10-18	145.91000366210938	161.40619
5	2023-10-19	145.2899932861328	161.47768
6	2023-10-20	142.9499969482422	161.53322
7	2023-10-23	141.0	161.53622
8	2023-10-24	141.1699981689453	161.4603
9	2023-10-25	140.39999389648438	161.31673
10	2023-10-26	140.75999450683594	161.10928
11	2023-10-27	135.69000244140625	160.85887
12	2023-10-30	137.4199981689453	160.50818
13	2023-10-31	139.05999755859375	160.0971
14	2023-11-01	138.94000244140625	159.67183
15	2023-11-02	141.4199981689453	159.24971
16	2023-11-03	143.0	158.87857
17	2023-11-06	144.0800018310547	158.58815
18	2023-11-07	144.00999450683594	158.39209
19	2023-11-08	144.72000122070312	158.27983
280	2024-11-21	244.75999450683594	230.49118
281	2024-11-22	248.5500030517578	231.2199
282	2024-11-25	250.2899932861328	231.95049
283	2024-11-26	249.97000122070312	232.6952
284	2024-11-27	249.78999328613278	233.43768
285	2024-11-29	249.72000122070312	234.16167
286	2024-12-02	246.25000000000003	234.8539
287	2024-12-03	244.82000732421875	235.46658
288	2024-12-04	243.39999389648438	235.9817
289	2024-12-05	245.47999572753906	236.38838
290	2024-12-06	247.36000061035153	236.72139
291	2024-12-09	243.80999755859375	237.014
292	2024-12-10	242.86000061035156	237.23596
293	2024-12-11	243.52999877929688	237.3857
294	2024-12-12	241.52999877929685	237.48103
295	2024-12-13	239.94000244140622	237.51125
296	2024-12-16	239.58000183105466	237.47108
297	2024-12-17	238.36000061035156	237.37029
298	2024-12-18	230.3699951171875	237.2102
299	2024-12-19	232.9600067138672	236.91663
300	2024-12-20	237.60000610351565	236.5408

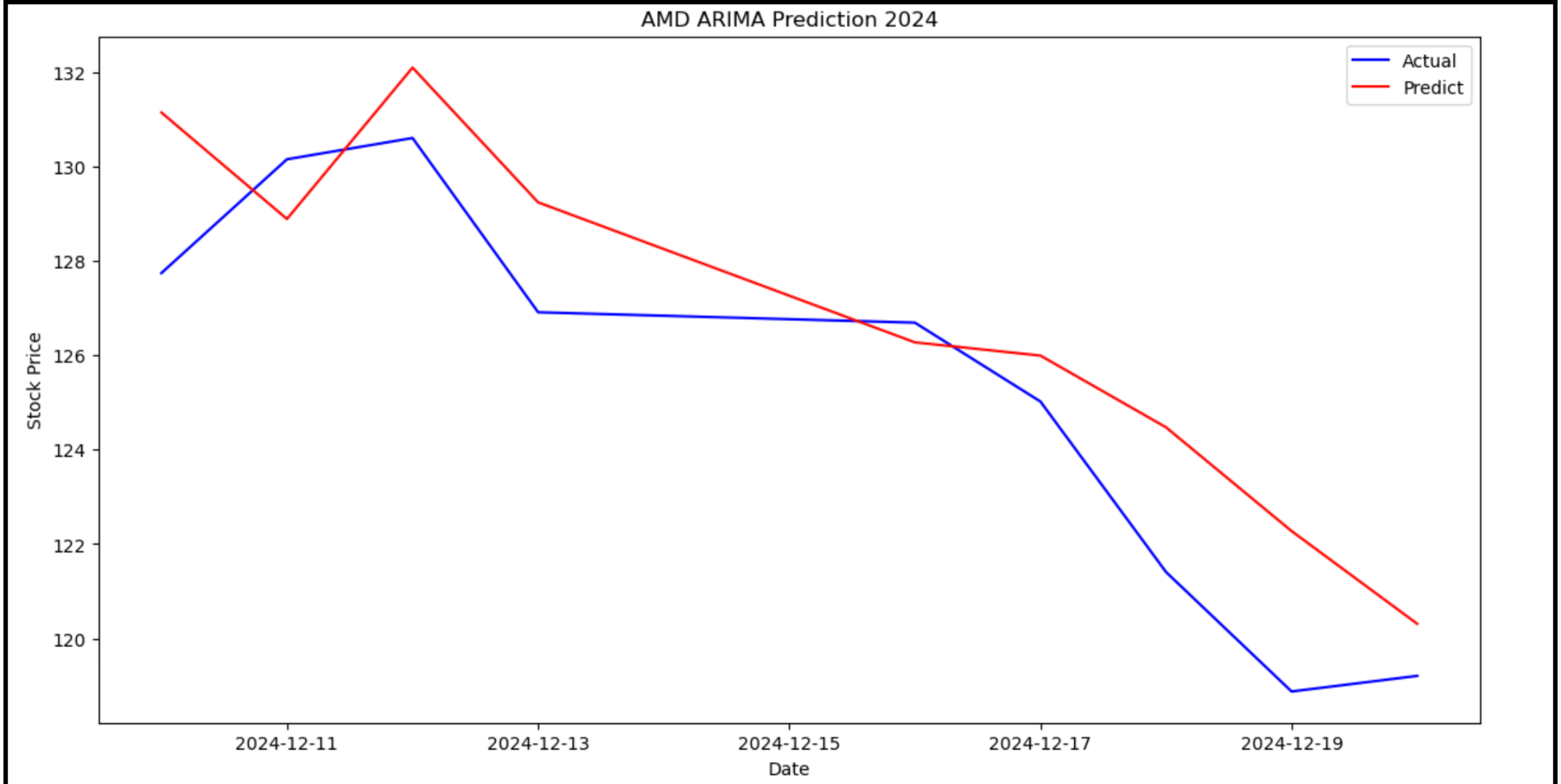


MAE: 6.2688037454091825
MSE: 70.12878951271826
RMSE: 8.3742933739342
R^2: 0.9089657525683575
MAPE: 3.2530765151316277%
Adjusted R^2: 0.9086612902023654


AMD ARIMA Prediction 2024
Actual
Predict
Stock Price
200
175
150
125
100
75
50
25
2019
2020
2021
2022
2023
2024
2025
Date

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import MinMaxScaler
import yfinance as yf
from datetime import timedelta, datetime

# Load data
stock_symbol = 'AMD'
data = yf.download(stock_symbol, start='2024-10-20', end='2024-12-20')['Close']
data_forecast = yf.download(stock_symbol, start='2024-12-10', end=datetime.now().strftime('%Y-%m-%d'))['Close']
data_combined = pd.concat([data, data_forecast])

# Normalize data
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data_combined.values.reshape(-1, 1))
data_train = data_scaled[:len(data)]
data_test = data_scaled[len(data):]

# Fit ARIMA model
order = (30, 2, 10)
model = ARIMA(data_train, order=order)
model_fit = model.fit()

# Forecast
future_steps = len(data_test)
forecast_scaled = model_fit.forecast(steps=future_steps)
forecast = scaler.inverse_transform(forecast_scaled.reshape(-1, 1))

# Inverse transform for actual values
data_test_actual = scaler.inverse_transform(data_test.reshape(-1, 1))

# Save comparison dataframe
comparison_df = pd.DataFrame({
    'Date': data_forecast.index,
    'Actual': data_test_actual.flatten(),
    'Predict': forecast.flatten()})
print(comparison_df.head(10))
comparison_df.to_csv('AMD.csv', index=False)
```

Date	Actual	Predict
2024-12-10	127.73999786376953	131.14028565084732
2024-12-11	130.14999389648438	128.88542334669546
2024-12-12	130.60000610351562	132.09084921584562
2024-12-13	126.91000366210938	129.23763646828988
2024-12-16	126.69000244140625	126.27305563699103
2024-12-17	125.0199966430664	125.9926951938466
2024-12-18	121.41000366210936	124.47402241582873
2024-12-19	118.87999725341797	122.27670953339113
2024-12-20	119.20999908447266	120.31393738653675



MAE: 1.937516549592137
MSE: 4.896235055062649
RMSE: 2.2127437843235827
R^2: 0.7165679951405626
MAPE: 1.558140743277945%
Adjusted R^2: 0.6760777087320715


Cảm ơn
vì đã lắng nghe