```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.tsa.statespace.sarimax import SARIMAX

# Step 1: Load and Clean the Data
path = "C:/Users/james/Downloads/"
stock = pd.read_csv(path + "Microsoft_Stock-1.csv", parse_dates=['Date'],
index_col='Date')
print("Data Overview:")
print(stock.head())
print("\nMissing Values:\n", stock.isnull().sum())

# Drop any missing values
stock = stock.dropna()

# Step 2: Explore the Data
# Plot Closing Price
plt.figure(figsize=(12, 6))
stock['Close'].plot(title="Microsoft Closing Stock Prices", ylabel="Closing Price")
plt.show()

# Plot Volume
plt.figure(figsize=(12, 6))
stock['Volume'].plot(title="Microsoft Trading Volume", ylabel="Volume")
plt.show()

print("Interpretation: The closing prices show a general upward trend, while the
trading volume exhibits fluctuations.")

# Step 3: Train/Test Split (Last 6 months as test set)
train = stock[:'2020-09-29']
test = stock['2020-09-30':'2021-03-31']

print(f"Training data: {train.shape}, Testing data: {test.shape}")

# Step 4: Linear Regression
X_train = np.array(range(len(train))).reshape(-1, 1)
y_train = train['Close'].values
X_test = np.array(range(len(train), len(stock))).reshape(-1, 1)

# Fit Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
LR_preds = lr_model.predict(X_test)

# Linear Regression Metrics
mae_lr = mean_absolute_error(test['Close'], LR_preds[-len(test):])
rmse_lr = np.sqrt(mean_squared_error(test['Close'], LR_preds[-len(test):]))

print("\nLinear Regression Metrics:")
print(f"MAE: {mae_lr:.4f}, RMSE: {rmse_lr:.4f}")
```

```
# Plot Linear Regression Results
plt.figure(figsize=(12, 6))
plt.plot(train.index, train['Close'], label="Training Data")
plt.plot(test.index, test['Close'], label="Testing Data")
plt.plot(test.index, LR_preds[-len(test):], label="Linear Regression Predictions")
plt.legend()
plt.title("Linear Regression Results")
plt.xlabel("Date")
plt.ylabel("Closing Price")
plt.show()

# Step 5: Stationarity Check and Differencing
print("\nADF Test on Original Data:")
result = adfuller(stock['Close'])
print(f"ADF Statistic: {result[0]}, p-value: {result[1]}")

# Differencing the Series
stock['Close_diff'] = stock['Close'].diff()
print("\nADF Test on Differenced Data:")
result_diff = adfuller(stock['Close_diff'].dropna())
print(f"ADF Statistic: {result_diff[0]}, p-value: {result_diff[1]}")

# Plot ACF and PACF
fig, axes = plt.subplots(1, 2, figsize=(14, 5))
plot_acf(stock['Close_diff'].dropna(), ax=axes[0], title="ACF of Differenced Data")
plot_pacf(stock['Close_diff'].dropna(), ax=axes[1], title="PACF of Differenced
Data")
plt.show()

# Step 6: AutoReg Model
# Choose lag based on ACF/PACF or experiment with different values
lag = 1  # Choose an appropriate lag value, or experiment with ACF/PACF

auto_reg_model = AutoReg(train['Close'], lags=lag)
auto_reg_fitted = auto_reg_model.fit()

# Make predictions
TSModel1_preds = auto_reg_fitted.predict(start=len(train), end=len(stock)-1,
dynamic=False)

# AutoReg Metrics
mae_ar = mean_absolute_error(test['Close'], TSModel1_preds)
rmse_ar = np.sqrt(mean_squared_error(test['Close'], TSModel1_preds))

print("\nAutoReg Model Metrics:")
print(f"MAE: {mae_ar:.4f}, RMSE: {rmse_ar:.4f}")

# Plot AutoReg Results
plt.figure(figsize=(12, 6))
plt.plot(train.index, train['Close'], label="Training Data")
plt.plot(test.index, test['Close'], label="Testing Data")
plt.plot(test.index, TSModel1_preds, label="AutoReg Predictions")
plt.legend()
plt.title("AutoReg Results")
plt.show()

# Step 7: SARIMA Model (optional, can be kept if needed)
sarima_model = SARIMAX(
    train['Close'],
```

```
    order=(1, 1, 1),
    seasonal_order=(1, 1, 1, 12)
)
sarima_fitted = sarima_model.fit(disp=False)
TSModel2_preds = sarima_fitted.forecast(steps=len(test))

# SARIMA Metrics
mae_sarima = mean_absolute_error(test['Close'], TSModel2_preds)
rmse_sarima = np.sqrt(mean_squared_error(test['Close'], TSModel2_preds))

print("\nSARIMA Model Metrics:")
print(f"MAE: {mae_sarima:.4f}, RMSE: {rmse_sarima:.4f}")

# Plot SARIMA Results (optional)
plt.figure(figsize=(12, 6))
plt.plot(train.index, train['Close'], label="Training Data")
plt.plot(test.index, test['Close'], label="Testing Data")
plt.plot(test.index, TSModel2_preds, label="SARIMA Predictions")
plt.legend()
plt.title("SARIMA Results")
plt.show()

# Step 8: Compare All Models
plt.figure(figsize=(12, 6))
plt.plot(train.index, train['Close'], label="Training Data")
plt.plot(test.index, test['Close'], label="Testing Data")
plt.plot(test.index, LR_preds[-len(test):], label="Linear Regression")
plt.plot(test.index, TSModel1_preds, label="AutoReg")
plt.plot(test.index, TSModel2_preds, label="SARIMA")
plt.legend()
plt.title("Model Comparison")
plt.xlabel("Date")
plt.ylabel("Closing Price")
plt.show()

# Step 9: Summary
print("\nModel Comparison Summary:")
print(f"Linear Regression - MAE: {mae_lr:.4f}, RMSE: {rmse_lr:.4f}")
print(f"AutoReg - MAE: {mae_ar:.4f}, RMSE: {rmse_ar:.4f}")
print(f"SARIMA - MAE: {mae_sarima:.4f}, RMSE: {rmse_sarima:.4f}")
print("\nAutoReg provides a simpler alternative and may perform well depending on
the time series structure.")
```