1. 1) Data Source: Utilize a dataset containing historical sales data, including features like date, product ID, store ID, and sales quantity.

import pandas as pd

import random

from datetime import datetime, timedelta

num\_records = 1000

dates = []

product\_ids = []

store\_ids = []

sales\_quantities = []

for \_ in range(num\_records):

start\_date = datetime(2020, 1, 1)

end\_date = datetime(2021, 12, 31)

random\_date = start\_date + timedelta(days=random.randint(0, (end\_date - start\_date).days)

product\_id = random.randint(1, 100)

store\_id = random.randint(1, 10)

sales\_quantity = random.randint(1, 100)

dates.append(random\_date)

product\_ids.append(product\_id)

store\_ids.append(store\_id)

sales\_quantities.append(sales\_quantity)

sales\_data = pd.DataFrame({

'Date': dates,

'Product\_ID': product\_ids,

'Store\_ID': store\_ids,

'Sales\_Quantity': sales\_quantities

})

print(sales\_data.head())

sales\_data.to\_csv('sales\_data.csv', index=False)

2) Data Preprocessing: Clean and preprocess the data, handle missing values, and convert categorical features into numerical representations.

missing\_values = sales\_data.isnull().sum()

sales\_data = sales\_data.dropna()

sales\_data['Sales\_Quantity'].fillna(sales\_data['Sales\_Quantity'].mean(), inplace=True)

sales\_data['Product\_ID'].fillna('Unknown', inplace=True)

sales\_data = pd.get\_dummies(sales\_data, columns=['Product\_ID', 'Store\_ID'])

from sklearn.preprocessing import LabelEncoder

label\_encoder = LabelEncoder()

sales\_data['Product\_ID'] = label\_encoder.fit\_transform(sales\_data['Product\_ID'])

sales\_data['Store\_ID'] = label\_encoder.fit\_transform(sales\_data['Store\_ID'])

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

sales\_data['Sales\_Quantity'] = scaler.fit\_transform(sales\_data[['Sales\_Quantity']])

3) Feature Engineering: Create additional features that could enhance the predictive power of the model, such as time-based features (e.g., day of the week, month).

import pandas as pd

sales\_data['Date'] = pd.to\_datetime(sales\_data['Date']

sales\_data['Year'] = sales\_data['Date'].dt.year

sales\_data['Month'] = sales\_data['Date'].dt.month

sales\_data['Day'] = sales\_data['Date'].dt.day

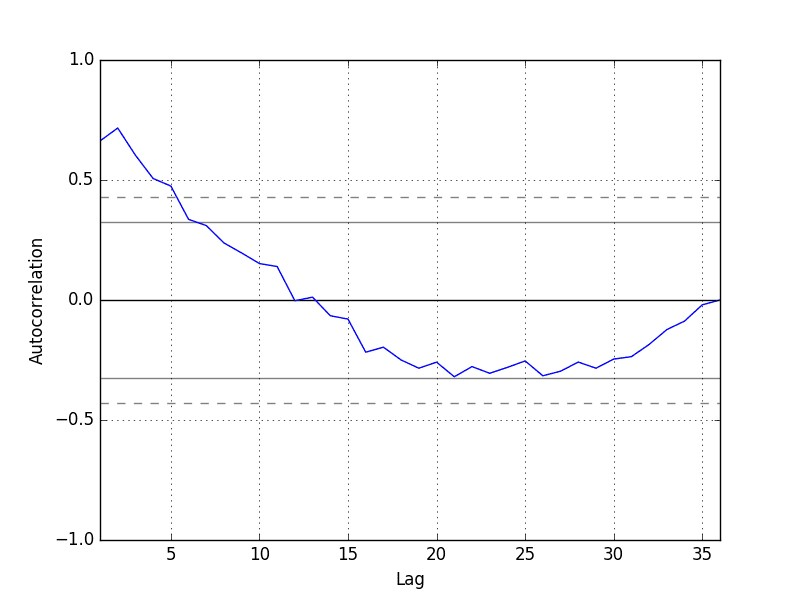
sales\_data['Day\_of\_Week'] = sales\_data['Date'].dt.dayofweek # Monday is 0, Sunday is 6

sales\_data['Week\_of\_Year'] = sales\_data['Date'].dt.weekofyear

sales\_data['Quarter'] = sales\_data['Date'].dt.quarter

sales\_data['Is\_Weekend'] = sales\_data['Day\_of\_Week'].apply(lambda x: 1 if x >= 5 else 0)

1. 4) Model Selection: Choose suitable time series forecasting algorithms (e.g., ARIMA, Exponential Smoothing) for predicting future sales.



from pandas import read\_csv

from pandas import datetime

from matplotlib import pyplot

from pandas.plotting import autocorrelation\_plot

def parser(x):

return datetime.strptime('190'+x, '%Y-%m')

series = read\_csv('shampoo-sales.csv', header=0, parse\_dates=[0], index\_col=0, squeeze=True, date\_parser=parser)

autocorrelation\_plot(series)

pyplot.show()

5

import numpy as np

actual = np.array([actual\_values]) # Replace with your actual values

forecast = np.array([forecast\_values]) # Replace with your forecasted values

mae = np.mean(np.abs(actual - forecast))

rmse = np.sqrt(np.mean((actual - forecast) \*\* 2))

print("MAE:", mae)

print("RMSE:", rmse)