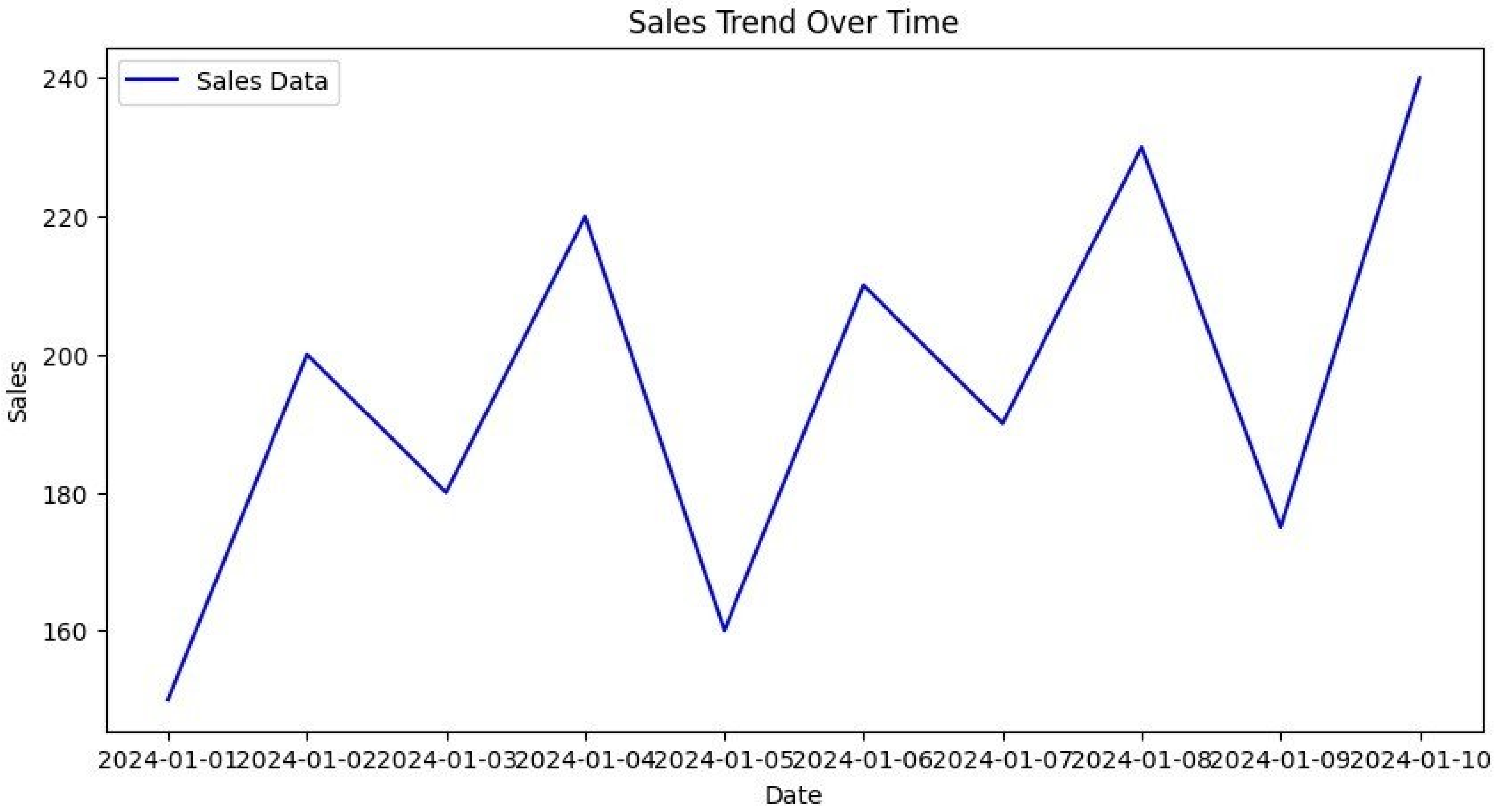
[2]:
```python
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
import warnings
warnings.filterwarnings("ignore")

#load and inspect dataset
df=pd.read_csv("sales_data_.csv",parse_dates=['Date'],index_col='Date')
df = df.asfreq('D')

#display first few rows
print(df.head())

#check for missing values
print("\nMissing Values:")
print(df.isnull().sum())

#visualize sales trends
#plot sales over time
plt.figure(figsize=(10,5))
plt.plot(df.index,df['Sales'],label="Sales Data",color='blue')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales Trend Over Time')
plt.legend()
plt.show()
```

```python
#Check Stationarity (Dickey-Fuller Test)
from statsmodels.tsa.stattools import adfuller

#Perform Dickey-Fuller Test
result=adfuller(df['Sales'])
print(f"ADF Statistic: {result[0]}")
print(f"P-Value: {result[1]}")

if result[1]>0.05:
    print("The data is non-stationary. Differencing is needed.")
else:
    print("The data is stationary.")

#build ARIMA Model
#Fit ARIMA Model (assuming p=2,d=1,q=2 based on ACF/PACF)
model=ARIMA(df['Sales'],order=(2,1,2))
model_fit=model.fit()

#forecast next 10 periods
forecast=model_fit.forecast(steps=10)
print("Forecasted Sales for next 10 periods:\n",forecast)

#visualize forecast
#plot original and forecasted sales
plt.figure(figsize=(10,5))
plt.plot(df.index,df['Sales'],label="Actual Sales",color='blue')
plt.plot(pd.date_range(start=df.index[-1],periods=11,freq='D')[1:],forecast,label="Forecast",color='red')
plt.xlabel("Date")
plt.ylabel("Sales")
plt.title("Sales Forecast using ARIMA")
plt.legend()
plt.show()
```

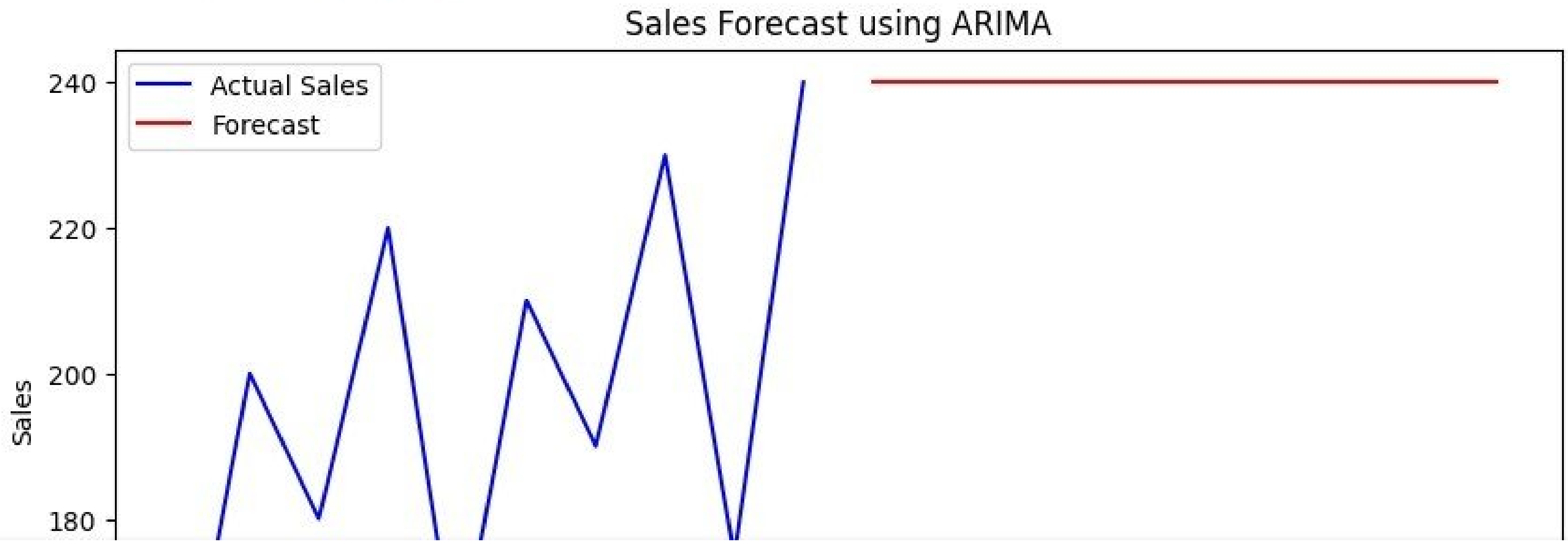
```
#project 2
#importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

#load and inspect dataset
df=pd.read_csv("heart_disease.csv")

#display first few rows
print(df.head())

#check for missing values
print("\nMissing Values:")
print(df.isnull().sum())

#preprocess data
#define features (X) and target variable (y)
X=df.drop(columns=['Heart Disease'])
y=df['Heart Disease']
X_encoded = pd.get_dummies(X, drop_first=True)

#standardize numerical features
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X_encoded)

#split into train and test sets
```

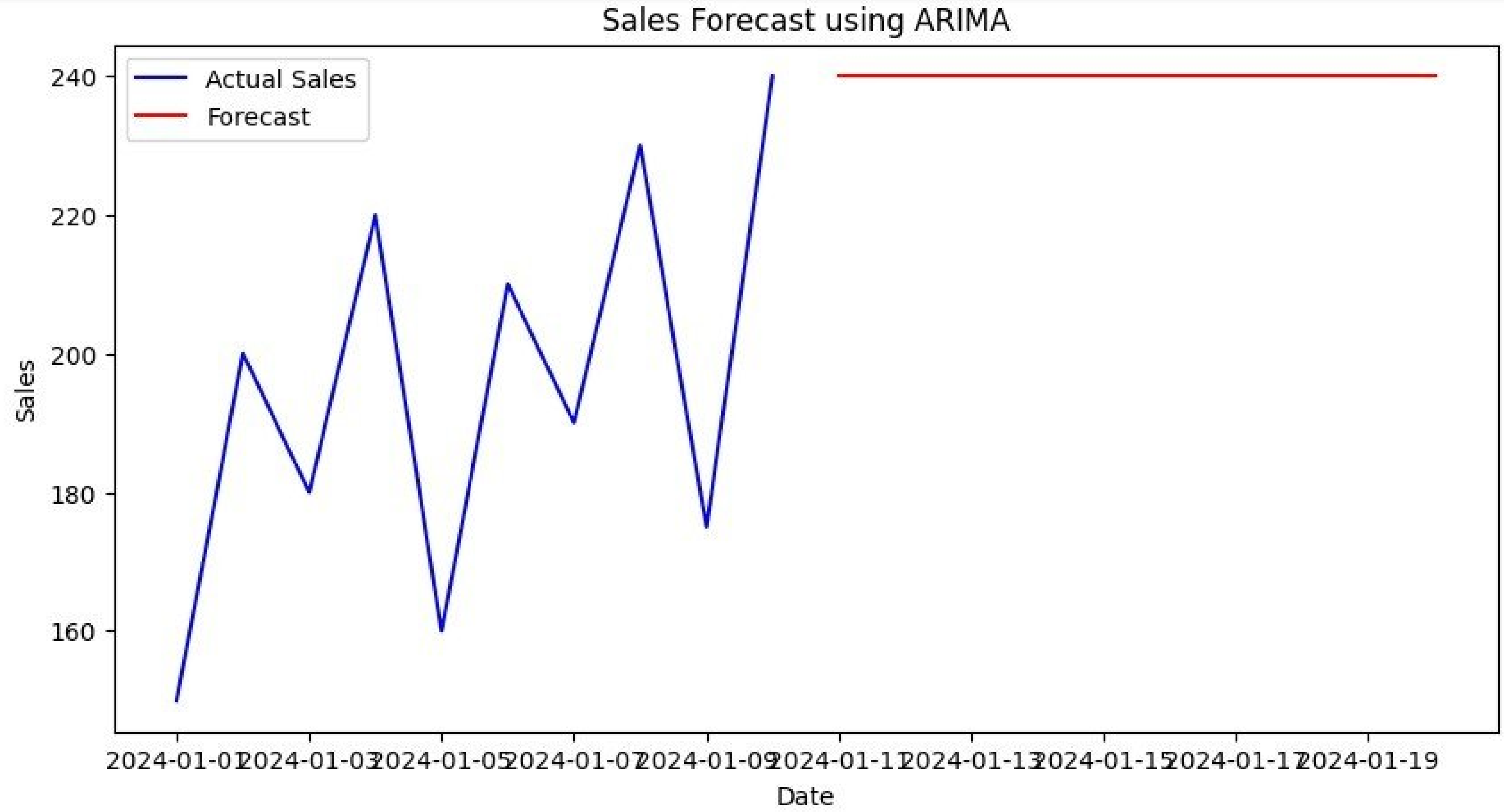
```python
#split into train and test sets
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.2,random_state=42)
print(f"Training samples: {X_train.shape[0]},Testing samples:{X_test.shape[0]}")

#train logistic regression model
model=LogisticRegression()
model.fit(X_train,y_train)

#predict on test data
y_pred=model.predict(X_test)

#print accuracy
accuracy=accuracy_score(y_test,y_pred)
print(f"ModelAccuracy:{accuracy:.2f}")

#evaluate model performance
#confusion matrix
cm=confusion_matrix(y_test,y_pred)
print("\nConfusion Matrix:")
print(cm)

#classification report
print("\nClassification Report:")
print(classification_report(y_test,y_pred))

#plot confusion matrix
plt.figure(figsize=(6,4))
sns.heatmap(cm,annot=True,fmt="d",cmap="Blues",xticklabels=["No Disease","Disease"],yticklabels=["No Disease","Disease"])
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
            Sales Product Region
Date
2024-01-01    150       A  North
2024-01-02    200       B  South
2024-01-03    180       A  North
2024-01-04    220       C   East
2024-01-05    160       A   West

Missing Values:
Sales      0
Product    0
Region     0
dtype: int64
```


Sales Trend Over Time
Sales Data
240
220
200
180
160
Sales
2024-01-01
2024-01-02
2024-01-03
2024-01-04
2024-01-05
2024-01-06
2024-01-07
2024-01-08
2024-01-09
2024-01-10
Date

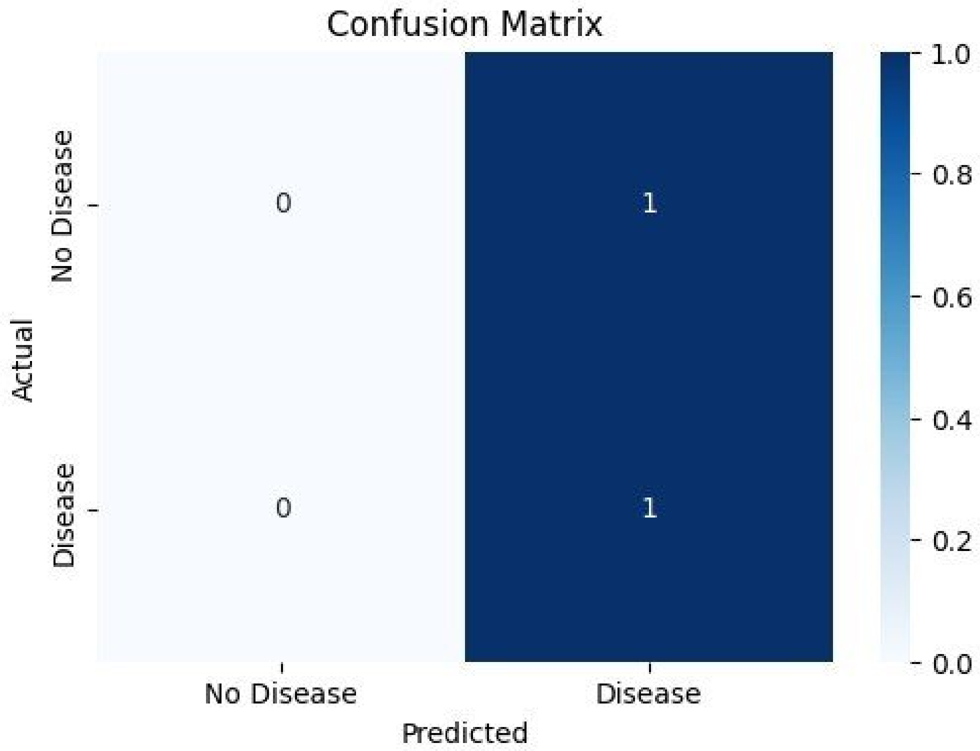
```
ADF Statistic: 1.896571585241519
P-Value: 0.9985224831326911
The data is non-stationary. Differencing is needed.
Forecasted Sales for next 10 periods:
 2024-01-11    240.000047
2024-01-12    240.000038
2024-01-13    240.000038
2024-01-14    240.000038
2024-01-15    240.000038
2024-01-16    240.000038
2024-01-17    240.000038
2024-01-18    240.000038
2024-01-19    240.000038
2024-01-20    240.000038
Freq: D, Name: predicted_mean, dtype: float64
```





	Age	Gender	Cholesterol	Blood Pressure	Heart Disease
0	52	Male	245	130/85	1
1	58	Female	210	140/90	0
2	46	Male	190	120/80	0
3	59	Female	225	135/88	1
4	63	Male	250	150/95	1

```
Missing Values:
Age               0
Gender            0
Cholesterol       0
Blood Pressure    0
Heart Disease     0
dtype: int64
Training samples: 8,Testing samples:2
ModelAccuracy:0.50

Confusion Matrix:
[[0 1]
 [0 1]]

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.50      1.00      0.67         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2
```


Confusion Matrix
No Disease
Disease
Actual
0
1
0
1
No Disease
Disease
Predicted
1.0
0.8
0.6
0.4
0.2
0.0