Ex.No:6(a) **DISEASE OUTBREAK PREDICTION** **Date:31-Jan-2025**

Aim:-

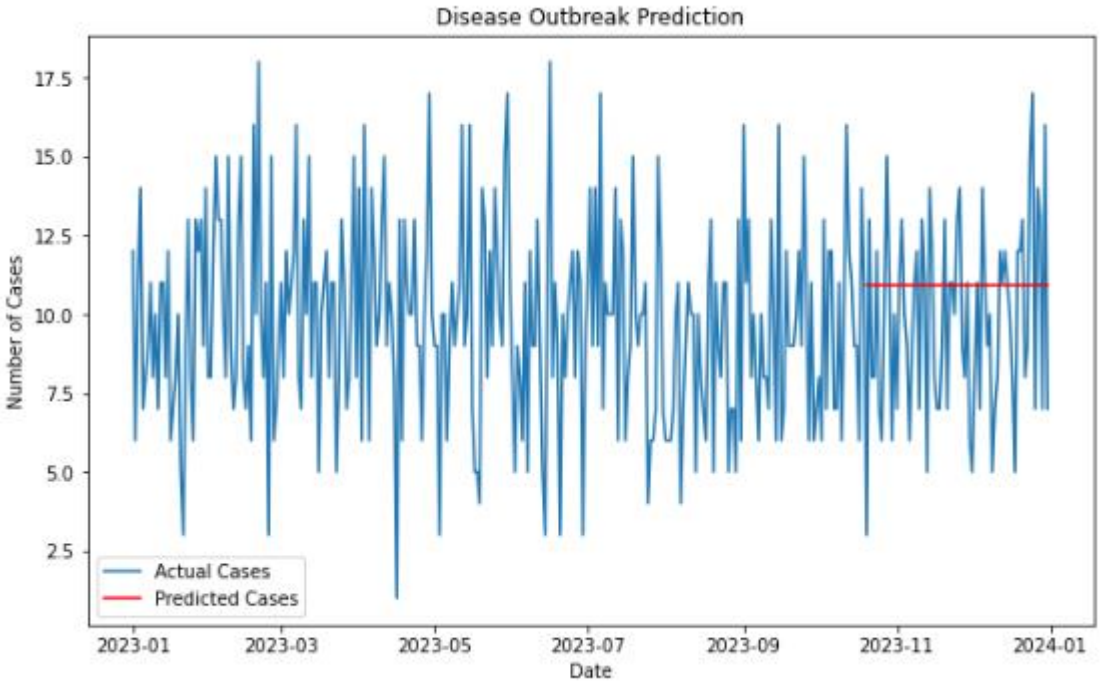
Predict the number of disease cases over time using historical data and basic regression models.

Program Code:-

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

# Simulated data (this would usually be from real historical data)
np.random.seed(42)
dates = pd.date_range('2023-01-01', periods=365, freq='D')
cases = np.random.poisson(lam=10, size=(365,))

# Create a DataFrame
data = pd.DataFrame({'date': dates, 'cases': cases})

# Feature Engineering (using simple time features)
data['day_of_year'] = data['date'].dt.dayofyear
data['month'] = data['date'].dt.month

# Splitting the data
X = data[['day_of_year', 'month']]
y = data['cases']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

# Model
model = RandomForestRegressor(n_estimators=100)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Performance
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error: {mae}")

# Plot the predictions
plt.figure(figsize=(10, 6))
plt.plot(data['date'], data['cases'], label='Actual Cases')
plt.plot(data['date'].iloc[-len(y_test):], y_pred, label='Predicted Cases', color='red')
plt.legend()
plt.title('Disease Outbreak Prediction')
plt.xlabel('Date')
plt.ylabel('Number of Cases')
plt.show()
```

Output:-

```
Mean Absolute Error: 2.592191780821918
```



Result:-

The model forecassts the number of cases with an accuracy measured by Mean Absolute Error (MAE) and visualizes the predicted vs. actual cases on a time series plot.

Ex.No:6(b) **BED OCCUPANCY FORECASTING** **Date:31-Jan-2025**

Aim:-

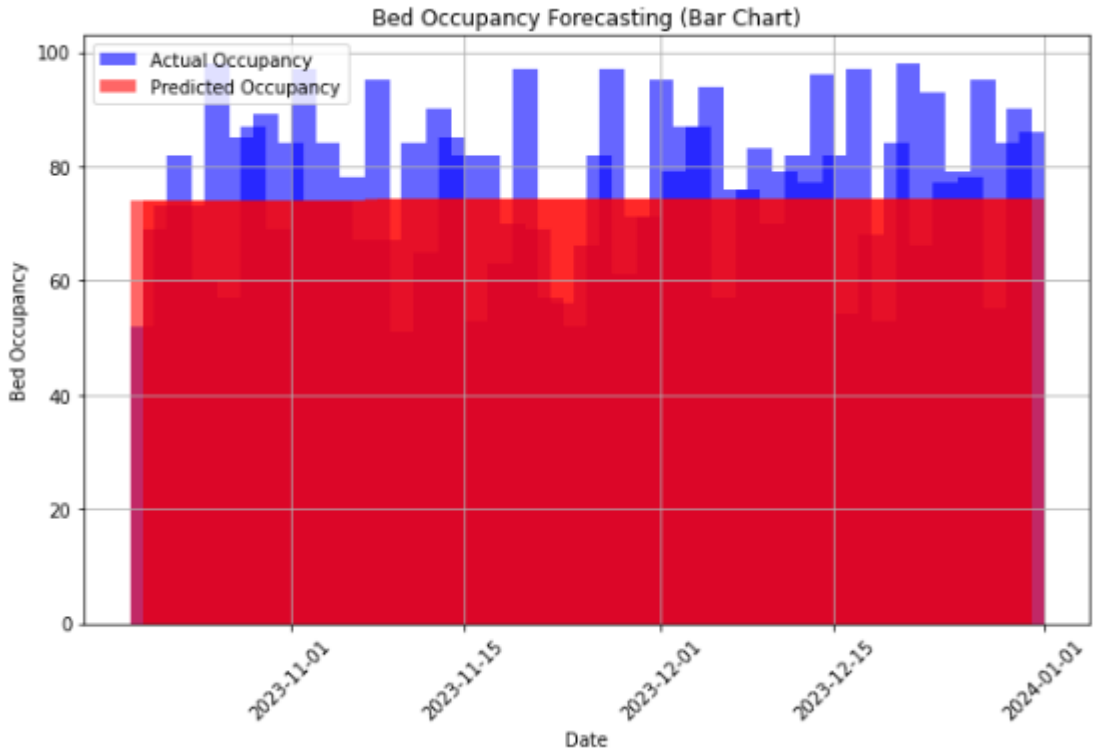
Forecast hospital bed occupancy based on historical usage patterns to aid resource planning.

Program Code:-

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

# Simulating bed occupancy data (daily occupancy between 50 and 100 beds)
np.random.seed(42)
dates = pd.date_range('2023-01-01', periods=365, freq='D')
occupancy = np.random.randint(50, 100, size=(365,))

# Create a DataFrame with the simulated data
data = pd.DataFrame({'date': dates, 'occupancy': occupancy})
# Feature Engineering (use day of year as a feature)
data['day_of_year'] = data['date'].dt.dayofyear

# Splitting the data into train and test sets (80% training, 20% testing)
X = data[['day_of_year']]
y = data['occupancy']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

# Train a Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

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

# Calculate the performance (Mean Absolute Error)
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error: {mae}")

# Plotting the actual vs predicted bed occupancy using a bar chart
plt.figure(figsize=(10, 6))
# Plot actual vs predicted for a specific range of dates
plt.bar(data['date'].iloc[-len(y_test):], y_test, label='Actual Occupancy', width=2, align='center', color='blue',
alpha=0.6)
plt.bar(data['date'].iloc[-len(y_test):], y_pred, label='Predicted Occupancy', width=2, align='center', color='re
d', alpha=0.6)
plt.legend()
plt.title('Bed Occupancy Forecasting (Bar Chart)')
plt.xlabel('Date')
plt.ylabel('Bed Occupancy')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

Output:-

```
Mean Absolute Error: 11.685612010323112
```



Result:-

The model predicts future bed occupancy, showing a comparison of actual vs. predicted values on a line plot, with performance evaluated by MAE.

Ex.No:6(c) MEDICATION EFFECTIVENESS ANALYSIS Date:31-Jan-2025

Aim:-

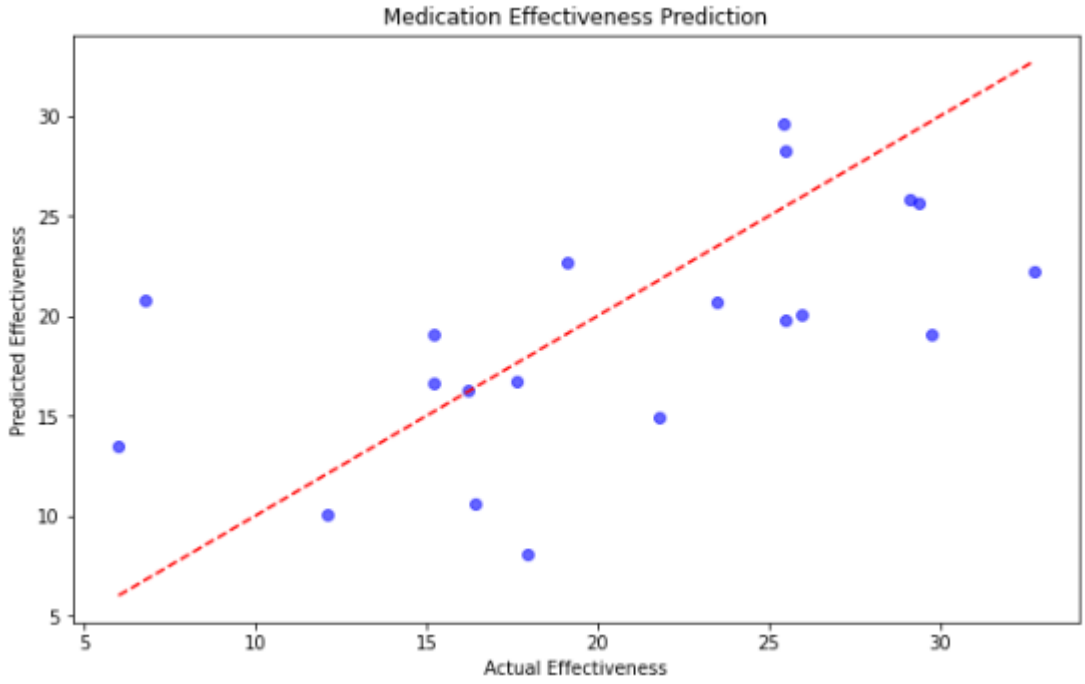
Predict the effectiveness of medication based on treatment duration and patient demographics.

Program Code:-

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# Simulating data for medication effectiveness
np.random.seed(42)
days_treated = np.random.randint(1, 30, size=(100,))
patient_age = np.random.randint(20, 70, size=(100,))
treatment_effectiveness = 0.5 * days_treated + 0.3 * patient_age + np.random.normal(0, 5, 100)

# Create a DataFrame
data = pd.DataFrame({'days_treated': days_treated, 'patient_age': patient_age, 'effectiveness': treatment_effe
ctiveness})

# Feature Engineering (Interaction features)
X = data[['days_treated', 'patient_age']]
y = data['effectiveness']

# Polynomial Feature Transformation for non-linearity
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2)

# Model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Performance
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error: {mae}")

# Plot the predictions
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.6)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
plt.title('Medication Effectiveness Prediction')
plt.xlabel('Actual Effectiveness')
plt.ylabel('Predicted Effectiveness')
plt.show()
```

Output:-

```
Mean Absolute Error: 5.2773016051146415
```



Result:-

The model estimates the medication's effectiveness, visualized by a scatter plot comparing actual vs. predicted effectiveness, with MAE as the evaluation metric.