Lab Programs:

Week 8: Heart Disease Program

Aim:To analyze a heart disease dataset and visualize relevant insights.

Programs:

1. Program 1: Loading Heart Disease Dataset
   - Procedure: Load the heart disease dataset and display its structure.
   - Code:

```
import pandas as pd
df = pd.read_csv('heart_disease.csv')
print(df.head())
```

   - Expected Result: A preview of the heart disease dataset.

2. Program 2: Summary Statistics
   - Procedure: Generate summary statistics of the dataset.
   - Code:

```
print(df.describe())
```

   - Expected Result: Summary statistics including count, mean, std, min, and max values.

3. Program 3: Correlation Matrix
   - Procedure: Create a correlation matrix and visualize it using a heatmap.
   - Code:

```
import seaborn as sns
import matplotlib.pyplot as plt
correlation = df.corr()
sns.heatmap(correlation, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix for Heart Disease Dataset")
plt.show()
```

   - Expected Result: A heatmap displaying the correlation matrix for the features in the dataset.

4. Program 4: Visualizing Age Distribution

- Procedure: Visualize the age distribution of patients with heart disease.
- Code:

```
sns.histplot(df['age'], bins=20, kde=True)
plt.title("Age Distribution of Heart Disease Patients")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```

- Expected Result: A histogram with a KDE curve showing the age distribution.

5. Program 5: Classification of Heart Disease

- Procedure: Train a simple logistic regression model to classify heart disease.
- Code:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
X = df.drop('target', axis=1)
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

- Expected Result: An accuracy score indicating the model's performance.

Week 9: Agriculture Program

Aim:To analyze agricultural data and extract meaningful insights.

Programs:

1. Program 1: Loading Agriculture Dataset

   - Procedure: Load the agricultural dataset and display its structure.

   - Code:

     df = pd.read_csv('agriculture_data.csv')

     print(df.head())

   - Expected Result: A preview of the agriculture dataset.

2. Program 2: Summary Statistics

   - Procedure: Generate summary statistics of the dataset.

   - Code:

     print(df.describe())

   - Expected Result: Summary statistics including count, mean, std, min, and max values.

3. Program 3: Crop Yield Visualization

   - Procedure: Visualize crop yields based on different factors.

   - Code:

     sns.boxplot(x='crop_type', y='yield', data=df)

     plt.title("Crop Yield Distribution by Type")

     plt.xlabel("Crop Type")

     plt.ylabel("Yield")

     plt.show()

   - Expected Result: A box plot showing the distribution of yields for different crop types.

4. Program 4: Relationship Between Fertilizer Use and Yield

   - Procedure: Analyze the relationship between fertilizer use and crop yield.

   - Code:

     sns.scatterplot(x='fertilizer_use', y='yield', data=df)

     plt.title("Fertilizer Use vs Crop Yield")

     plt.xlabel("Fertilizer Use")

     plt.ylabel("Crop Yield")

plt.show()

- Expected Result: A scatter plot showing the relationship between fertilizer use and crop yield.

5. Program 5: Predicting Crop Yield

- Procedure: Build a simple linear regression model to predict crop yield.

- Code:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
X = df[['fertilizer_use', 'rainfall']]
y = df['yield']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
```

- Expected Result: A mean squared error indicating the prediction error of the model.

Week 10: Marketing Program

Aim: To analyze marketing data and derive actionable insights.

Programs:

1. Program 1: Loading Marketing Dataset

- Procedure: Load the marketing dataset and display its structure.

- Code:

```
df = pd.read_csv('marketing_data.csv')
print(df.head())
```

- Expected Result: A preview of the marketing dataset.

2. Program 2: Summary Statistics

- Procedure: Generate summary statistics of the dataset.
- Code:

```
print(df.describe())
```

- Expected Result: Summary statistics including count, mean, std, min, and max values.

3. Program 3: Customer Segmentation
- Procedure: Segment customers based on their spending behavior.
- Code:

```
from sklearn.cluster import KMeans
X = df[['annual_spending', 'frequency']]
kmeans = KMeans(n_clusters=3)
df['segment'] = kmeans.fit_predict(X)
sns.scatterplot(x='annual_spending', y='frequency', hue='segment', data=df, palette='Set2')
plt.title("Customer Segmentation")
plt.xlabel("Annual Spending")
plt.ylabel("Frequency")
plt.show()
```

- Expected Result: A scatter plot showing different customer segments based on spending behavior.

4. Program 4: Analyzing Marketing Campaign Success
- Procedure: Analyze the success rate of a marketing campaign.
- Code:

```
campaign_success_rate = df['campaign_success'].value_counts(normalize=True) * 100
campaign_success_rate.plot(kind='bar', color='green')
plt.title("Marketing Campaign Success Rate")
plt.xlabel("Success")
plt.ylabel("Percentage")
plt.show()
```

- Expected Result: A bar chart displaying the percentage of successful versus unsuccessful marketing campaigns.

5. Program 5: Predicting Customer Churn

- Procedure: Build a logistic regression model to predict customer churn.

- Code:

```
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

X = df.drop('churn', axis=1)  # Assuming 'churn' is the target column

y = df['churn']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LogisticRegression(max_iter=1000)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy of Churn Prediction Model:", accuracy_score(y_te

st, y_pred))
```

- Expected Result: An accuracy score indicating the model's performance in predicting customer churn.