LAB PROGRAMS

# Week 8: Heart Disease Program

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

Programs: import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

# Program 1: Loading Heart Disease Dataset

# Load the dataset

df = pd.read\_csv('heart\_.csv') # Ensure the file is in your working directory

print("Preview of the dataset:")

print(df.head())

# Program 2: Summary Statistics

# Generate summary statistics

print("\nSummary Statistics:")

print(df.describe())

# Program 3: Correlation Matrix

# Create a correlation matrix and visualize it using a heatmap

correlation = df.corr()

plt.figure(figsize=(10, 8))

sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt=".2f")

plt.title("Correlation Matrix for Heart Disease Dataset")

plt.show()

# Program 4: Visualizing Age Distribution

# Visualize the age distribution of patients with heart disease

plt.figure(figsize=(8, 6))

sns.histplot(df['age'], bins=20, kde=True, color='blue')

plt.title("Age Distribution of Heart Disease Patients")

plt.xlabel("Age")

plt.ylabel("Frequency")

plt.show()

# Program 5: Classification of Heart Disease

# Train a simple logistic regression model to classify heart disease

X = df.drop('target', axis=1) # Features

y = df['target'] # Target variable

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize and train the logistic regression model

model = LogisticRegression(max\_iter=1000)

model.fit(X\_train, y\_train)

# Predict and evaluate the model

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print("\nModel Accuracy:", accuracy)

# output:

Preview of the dataset:

age sex cp trestbps chol fbs restecg thalach exang oldpeak slope \

0 63 1 3 145 233 1 0 150 0 2.3 0

1 37 1 2 130 250 0 1 187 0 3.5 0

2 41 0 1 130 204 0 0 172 0 1.4 2

3 56 1 1 120 236 0 1 178 0 0.8 2

4 57 0 0 120 354 0 1 163 1 0.6 2

ca thal target

0 0 1 1

1 0 2 1

2 0 2 1

3 0 2 1

4 0 2 1

Summary Statistics:

age sex cp trestbps chol fbs \

count 14.000000 14.000000 14.000000 14.000000 14.000000 14.000000

mean 52.214286 0.642857 1.142857 135.357143 242.357143 0.142857

std 7.816930 0.497245 0.949262 14.547573 48.154570 0.363137

min 37.000000 0.000000 0.000000 120.000000 168.000000 0.000000

25% 46.250000 0.000000 0.250000 122.500000 210.500000 0.000000

50% 55.000000 1.000000 1.000000 134.000000 234.500000 0.000000

75% 57.000000 1.000000 2.000000 140.000000 259.750000 0.000000

max 63.000000 1.000000 3.000000 172.000000 354.000000 1.000000

restecg thalach exang oldpeak slope ca \

count 14.000000 14.000000 14.000000 14.000000 14.000000 14.000000

mean 0.714286 163.214286 0.142857 1.250000 1.357143 0.142857

std 0.468807 12.084183 0.363137 0.951719 0.744946 0.363137

min 0.000000 148.000000 0.000000 0.000000 0.000000 0.000000

25% 0.250000 153.500000 0.000000 0.525000 1.000000 0.000000

50% 1.000000 162.500000 0.000000 1.200000 1.500000 0.000000

75% 1.000000 172.750000 0.000000 1.750000 2.000000 0.000000

max 1.000000 187.000000 1.000000 3.500000 2.000000 1.000000

thal target

count 14.000000 14.000000

mean 2.214286 0.571429

std 0.699293 0.513553

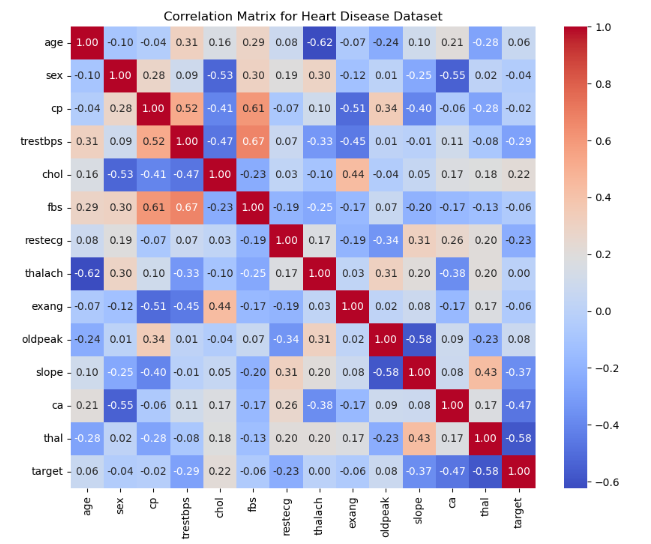
min 1.000000 0.000000

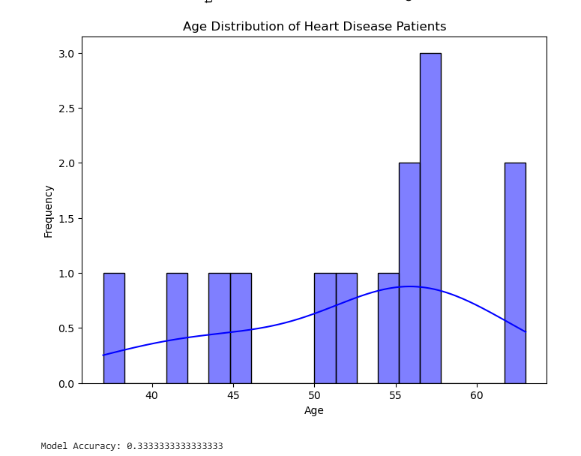
25% 2.000000 0.000000

50% 2.000000 1.000000

75% 3.000000 1.000000

max 3.000000 1.000000





# Week 9: Agriculture Program

Aim:To analyze agricultural data and extract meaningful insights.

Programs:

# Import required libraries

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegressiona

from sklearn.metrics import mean\_squared\_error

# 1. Loading the Dataset

print("Program 1: Loading Agriculture Dataset")

df = pd.read\_csv('agriculture.csv')

print(df.head())

# 2. Summary Statistics

print("\nProgram 2: Summary Statistics")

print(df.describe())

# 3. Crop Yield Visualization

print("\nProgram 3: Crop Yield Visualization")

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()

# 4. Relationship Between Fertilizer Use and Yield

print("\nProgram 4: Fertilizer Use vs Crop Yield")

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()

# 5. Predicting Crop Yield

print("\nProgram 5: Predicting Crop Yield")

# Prepare data for the model

X = df[['fertilizer\_use', 'rainfall']]

y = df['yield']

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train the model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Predict and evaluate

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

print("Mean Squared Error:", mse)

# output:

Program 1: Loading Agriculture Dataset

crop\_type yield fertilizer\_use rainfall region

0 Wheat 2.5 50 800 North

1 Rice 3.2 60 1200 South

2 Corn 4.1 55 900 East

3 Wheat 2.8 52 850 North

4 Rice 3.5 62 1300 South

Program 2: Summary Statistics

yield fertilizer\_use rainfall

count 9.000000 9.000000 9.000000

mean 3.377778 56.111111 990.000000

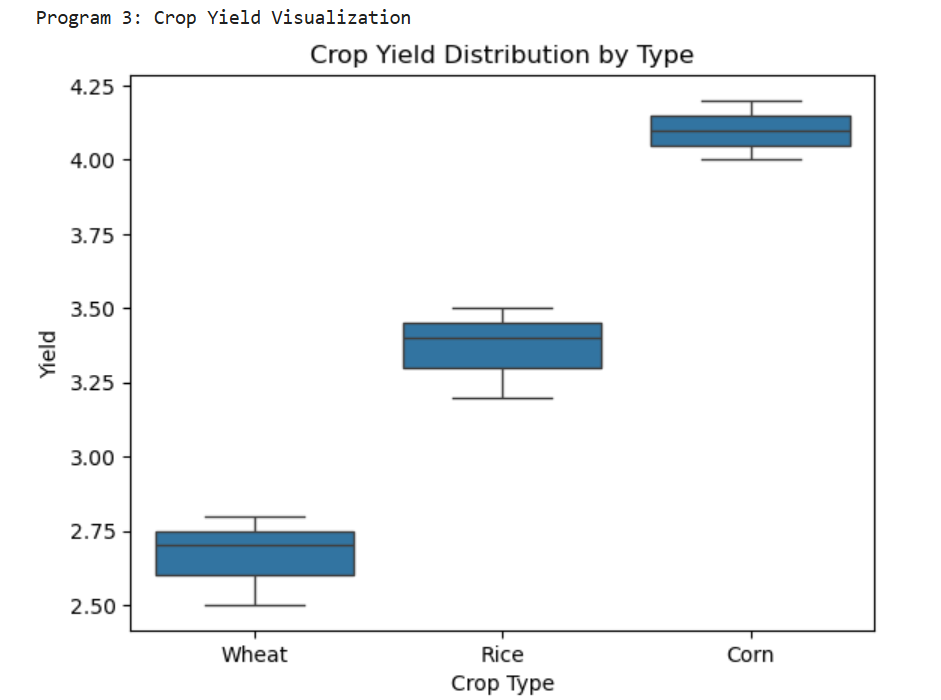
std 0.632016 4.456581 200.312256

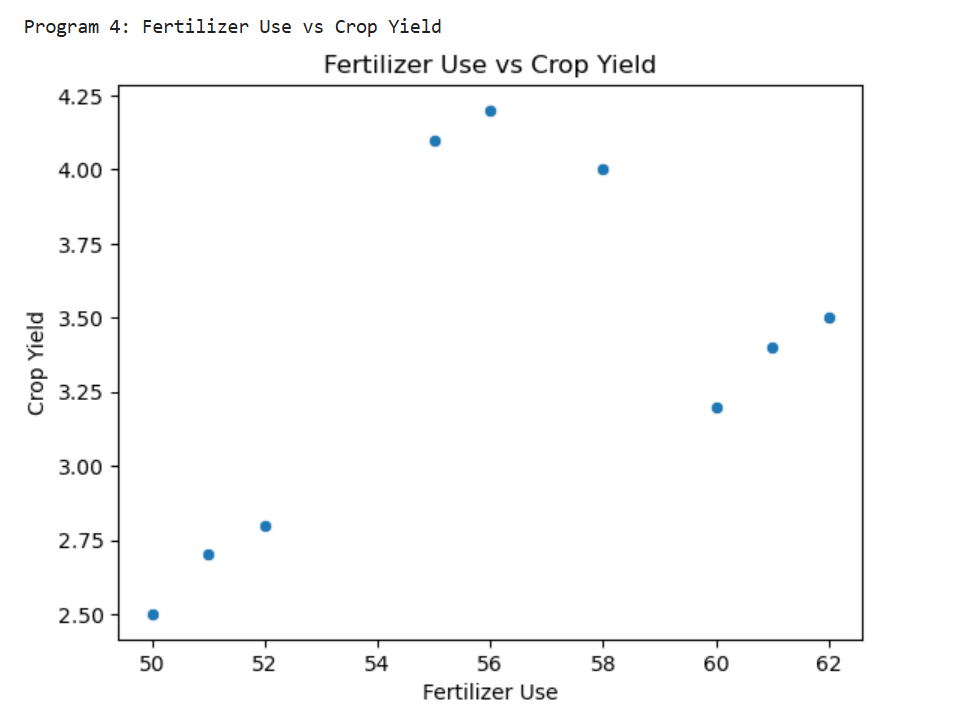
min 2.500000 50.000000 800.000000

25% 2.800000 52.000000 850.000000

50% 3.400000 56.000000 900.000000

75% 4.000000 60.000000 1200.000000

max 4.200000 62.000000 1300.000000



Program 5: Predicting Crop Yield

Mean Squared Error: 0.07784903861650244

# Week 10: Marketing Program

Aim: To analyze marketing data and derive actionable insights.

Programs:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.cluster import KMeans

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

# Program 1: Loading Marketing Dataset

df = pd.read\_csv('marketing\_data.csv') # Assuming the file is in the same directory

print(df.head()) # Displaying the first few rows of the dataset

# Program 2: Summary Statistics

print(df.describe()) # Display summary statistics

# Program 3: Customer Segmentation

# Assuming columns 'annual\_spending' and 'frequency' are numeric

X = df[['annual\_spending', 'frequency']]

kmeans = KMeans(n\_clusters=3)

df['segment'] = kmeans.fit\_predict(X)

# Plotting the segments

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()

# Program 4: Analyzing Marketing Campaign Success

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()

# Program 5: Predicting Customer Churn

X = df.drop('churn', axis=1) # Dropping 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)

# Logistic Regression Model

model = LogisticRegression(max\_iter=1000)

model.fit(X\_train, y\_train)

# Predicting and Evaluating the Model

y\_pred = model.predict(X\_test)

print("Accuracy of Churn Prediction Model:", accuracy\_score(y\_test, y\_pred))

# output:

customer\_id annual\_spending frequency campaign\_success churn

0 1 2500 5 1 0

1 2 1200 3 0 1

2 3 3500 8 1 0

3 4 1900 4 0 1

4 5 3000 6 1 0

customer\_id annual\_spending frequency campaign\_success churn

count 10.00000 10.00000 10.00000 10.000000 10.000000

mean 5.50000 2370.00000 4.80000 0.600000 0.500000

std 3.02765 1155.71046 2.65832 0.516398 0.527046

min 1.00000 500.00000 1.00000 0.000000 0.000000

25% 3.25000 1750.00000 3.00000 0.000000 0.000000

50% 5.50000 2300.00000 4.50000 1.000000 0.500000

75% 7.75000 2950.00000 6.75000 1.000000 1.000000

max 10.00000 4500.00000 9.00000 1.000000 1.000000

