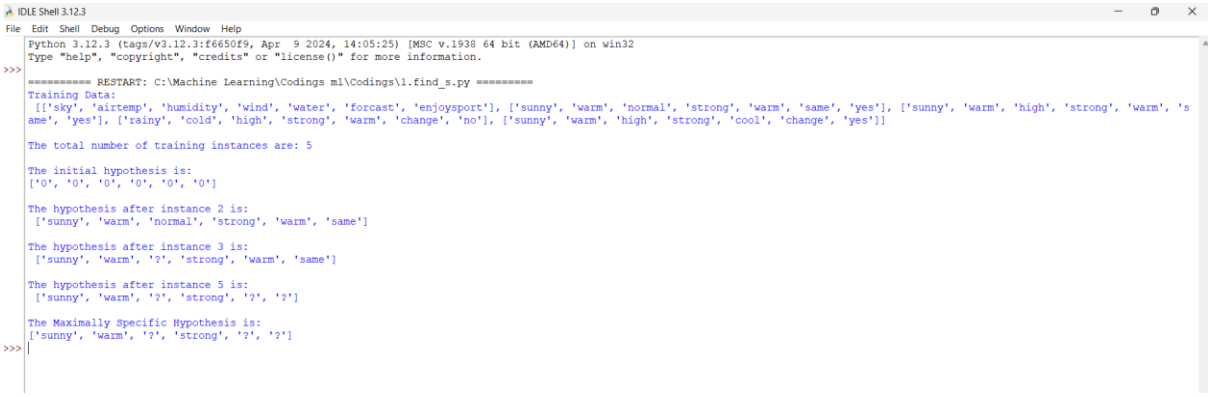
EXPERIMENT 1:

1.find_s.py - C:\Machine Learning\Codings ml\Codings\1.find_s.py (3.12.3)

File Edit Format Run Options Window Help

```
import csv

# Store the data
a = []

# Open the CSV file
with open(r'C:\Machine Learning\Dataset ml\New folder\enjoysport.csv', 'r') as csvfile:
    for row in csv.reader(csvfile):
        a.append([cell.strip() for cell in row])  # Strip whitespace from each cell

# Display the dataset
print("Training Data:\n", a)

print("\nThe total number of training instances are:", len(a))

# Number of attributes (last column is the label)
num_attribute = len(a[0]) - 1

# Initialize the hypothesis with '0'
print("\nThe initial hypothesis is:")
hypothesis = ['0'] * num_attribute
print(hypothesis)

# Skip header if needed (uncomment next line if first row is header)
# a = a[1:]

# Find-S Algorithm
for i in range(len(a)):
    if a[i][num_attribute].lower() == 'yes':
        for j in range(num_attribute):
            if hypothesis[j] == '0' or hypothesis[j] == a[i][j]:
                hypothesis[j] = a[i][j]
            else:
                hypothesis[j] = '?'
        print("\nThe hypothesis after instance {} is:\n".format(i + 1), hypothesis)

# Final hypothesis
print("\nThe Maximally Specific Hypothesis is:")
print(hypothesis)
```

OUTPUT:



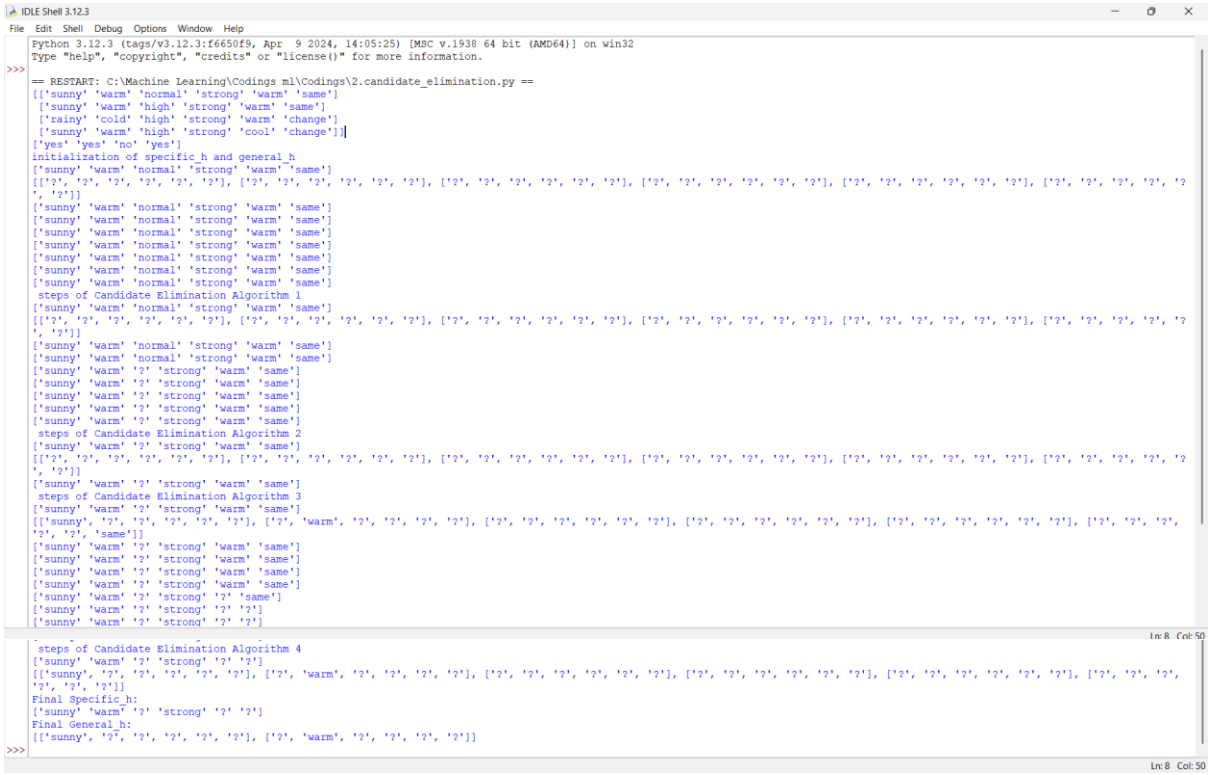
EXPERIMENT 2:

2.candidate_elimination.py - C:\Machine Learning\Codings ml\Codings\2.candidate_elimination.py (3.12.3)

File Edit Format Run Options Window Help

```
import numpy as np
import pandas as pd
data = pd.DataFrame(data=pd.read_csv(r'C:\Machine Learning\Dataset ml\New folder\enjoysport.csv'))
concepts = np.array(data.iloc[:,0:-1])
print(concepts)
target = np.array(data.iloc[:,-1])
print(target)
def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("initialization of specific_h and general_h")
    print(specific_h)
    general_h = [["?" for i in range(len(specific_h))] for i in
range(len(specific_h))]
    print(general_h)
    for i, h in enumerate(concepts):
        if target[i] == "yes":
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'

                print(specific_h)
        print(specific_h)
        if target[i] == "no":
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'
        print(" steps of Candidate Elimination Algorithm",i+1)
        print(specific_h)
        print(general_h)
    indices = [i for i, val in enumerate(general_h) if val ==['?', '?', '?', '?', '?', '?']]
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    return specific_h, general_h
s_final, g_final = learn(concepts, target)
print("Final Specific_h:", s_final, sep="\n")
print("Final General_h:", g_final, sep="\n")
```

OUTPUT:


IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:\Machine Learning\Codings ml\Codings\2.candidate_elimination.py ==
[['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'high' 'strong' 'warm' 'same']
['rainy' 'cold' 'high' 'strong' 'warm' 'change']
['sunny' 'warm' 'high' 'strong' 'cool' 'change']]
['yes' 'yes' 'no' 'yes']
initialization of specific_h and general_h
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
steps of Candidate Elimination Algorithm 1
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
steps of Candidate Elimination Algorithm 2
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
['sunny' 'warm' '?' 'strong' 'warm' 'same']
steps of Candidate Elimination Algorithm 3
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'same']]
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' '?' 'same']
['sunny' 'warm' '?' 'strong' '?' '?']
['sunny' 'warm' '?' 'strong' '?' '?']
Ln: 8 Col: 50
steps of Candidate Elimination Algorithm 4
['sunny' 'warm' '?' 'strong' '?' '?']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
Final Specific_h:
['sunny' 'warm' '?' 'strong' '?' '?']
Final General_h:
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
>>>
Ln: 8 Col: 50

EXPERIMENT 3:

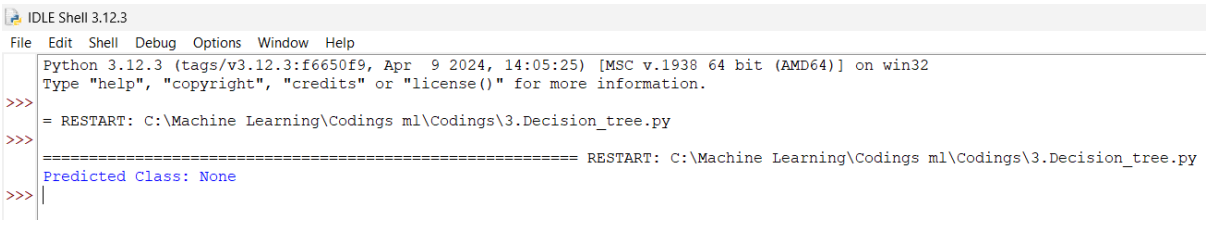
3.Decision_tree.py - C:\Machine Learning\Codings ml\Codings\3.Decision_tree.py (3.12.3)

File Edit Format Run Options Window Help

```
import pandas as pd
import numpy as np
import math

# Define a class for the decision tree node
class DecisionTreeNode:
    def __init__(self, attribute=None, label=None, branches={}):
        self.attribute = attribute  # the attribute used to split the data
        self.label = label  # the label assigned to this node
        self.branches = branches  # the branches of the decision tree

# Define a function to calculate the entropy of a dataset
def entropy(data):
    target = data['target']
    n = len(target)
    unique, counts = np.unique(target, return_counts=True)
    entropy = 0
    for i in range(len(unique)):
        p = counts[i] / n
        entropy -= p * math.log2(p)
    return entropy

# Define a function to calculate the information gain of an attribute
def information_gain(data, attribute):
    n = len(data)
    values = data[attribute].unique()
    entropy_s = entropy(data)
    entropy_attr = 0
    for value in values:
        subset = data[data[attribute] == value]
        subset_n = len(subset)
        subset_entropy = entropy(subset)
        entropy_attr += subset_n / n * subset_entropy
    return entropy_s - entropy_attr

# Define the ID3 algorithm
def id3(data, attributes):
    target = data['target']
    # If all the examples have the same target value, return a leaf node with that value
    if len(target.unique()) == 1:
        return DecisionTreeNode(label=target.iloc[0])
    # If there are no attributes left to split on, return a leaf node with the most common target value
    if len(attributes) == 0:
        return DecisionTreeNode(label=target.value_counts().idxmax())
    # Otherwise, select the attribute with the highest information gain
    gains = {attr: information_gain(data, attr) for attr in attributes}
    best_attribute = max(gains, key=gains.get)
```

```
    # Create a new decision tree node with the selected attribute
    node = DecisionTreeNode(attribute=best_attribute)
    # Split the data based on the selected attribute and recursively build the tree
    for value in data[best_attribute].unique():
        subset = data[data[best_attribute] == value].drop(best_attribute, axis=1)
        if len(subset) == 0:
            node.branches[value] = DecisionTreeNode(label=target.value_counts().idxmax())
        else:
            new_attributes = attributes.copy()
            new_attributes.remove(best_attribute)
            node.branches[value] = id3(subset, new_attributes)
    return node

# Load the dataset
data = pd.read_csv(r'C:\Machine Learning\Dataset ml\New folder\play_tennis.csv')
# Split the dataset into attributes and target variable
attributes = data.columns[:-1].tolist()
# Build the decision tree using ID3 algorithm
root = id3(data, attributes)

# Define a function to classify a new sample using the decision tree
def classify(sample, tree):
    if tree.label is not None:
        return tree.label
    attribute = tree.attribute
    value = sample[attribute]
    if value not in tree.branches:
        return tree.branches[max(tree.branches.keys(), key=int)]
    subtree = tree
# Example test case
test_sample = {
    'Outlook': 'Sunny',
    'Temperature': 'Cool',
    'Humidity': 'High',
    'Wind': 'Strong'
}

# Predict the class for the sample
result = classify(test_sample, root)
print("Predicted Class:", result)
```

OUTPUT:



EXPERIMENT 4:

4.back_propogation.py - C:\Machine Learning\Codings ml\Codings\4.back_propogation.py (3.12.3)

File Edit Format Run Options Window Help

```
import numpy as np

X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)
y = np.array(([92], [86], [89]), dtype=float)
X = X/np.amax(X,axis=0) #maximum of X array longitudinally
y = y/100

#Sigmoid Function
def sigmoid (x):
    return 1/(1 + np.exp(-x))

#Derivative of Sigmoid Function
def derivatives_sigmoid(x):
    return x * (1 - x)

#Variable initialization
epoch=5 #Setting training iterations
lr=0.1 #Setting learning rate

inputlayer_neurons = 2 #number of features in data set
hiddenlayer_neurons = 3 #number of hidden layers neurons
output_neurons = 1 #number of neurons at output layer
#weight and bias initialization

wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bh=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))

#draws a random range of numbers uniformly of dim x*y
for i in range(epoch):
    #Forward Propogation
    hinp1=np.dot(X,wh)
    hinp=hinp1 + bh
    hlayer_act = sigmoid(hinp)
    outinp1=np.dot(hlayer_act,wout)
    outinp= outinp1+bout
    output = sigmoid(outinp)

    #Backpropagation
    EO = y-output
    outgrad = derivatives_sigmoid(output)
    d_output = EO * outgrad
    EH = d_output.dot(wout.T)
    hiddengrad = derivatives_sigmoid(hlayer_act)#how much hidden layer wts contributed to error
    d_hiddenlayer = EH * hiddengrad

    wout += hlayer_act.T.dot(d_output) *lr   # dotproduct of nextlayererror and currentlayerop
    wh += X.T.dot(d_hiddenlayer) *lr

    print ("-----------Epoch-", i+1, "Starts----------")
    print("Input: \n" + str(X))
    print("Actual Output: \n" + str(y))
    print("Predicted Output: \n" ,output)
    print ("-----------Epoch-", i+1, "Ends----------\n")

print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n" ,output)
```

OUTPUT:

```
>>>
========== RESTART: C:\Machine Learning\Codings ml\Codings\4.back_propogation.py ========
----------Epoch- 1 Starts----------
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.77733085]
 [0.76178367]
 [0.77488148]]
----------Epoch- 1 Ends----------

----------Epoch- 2 Starts----------
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.77941107]
 [0.7637776 ]
 [0.77693901]]
----------Epoch- 2 Ends----------

----------Epoch- 3 Starts----------
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.7814299 ]
 [0.76571427]
 [0.77893621]]
```

IDLE Shell 3.12.3

File Edit Shell Debug Options Window Help

```
 [0.76571427]
 [0.77893621]]
----------Epoch- 3 Ends----------

----------Epoch- 4 Starts----------
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.78338998]
 [0.76759607]
 [0.78087564]]
----------Epoch- 4 Ends----------

----------Epoch- 5 Starts----------
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.78529377]
 [0.76942529]
 [0.78275974]]
----------Epoch- 5 Ends----------

Input:
[[0.66666667 1.        ]
```

IDLE Shell 3.12.3

File Edit Shell Debug Options Window Help

```
 [0.76571427]
 [0.77893621]]
-----------Epoch- 3 Ends----------

-----------Epoch- 4 Starts----------
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.78338998]
 [0.76759607]
 [0.78087564]]
-----------Epoch- 4 Ends----------

-----------Epoch- 5 Starts----------
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.78529377]
 [0.76942529]
 [0.78275974]]
-----------Epoch- 5 Ends----------

Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.78529377]
 [0.76942529]
 [0.78275974]]
>>>
```

EXPERIMENT 5:

exp_5.py - C:\Machine Learning\Codings ml\Codings\exp_5.py (3.12.3)

File Edit Format Run Options Window Help

```
from math import sqrt
from statistics import mode
l=[[33.6,50,1],[26.6,30,0],[23.4,40,0],[43.1,67,0],[35.3,23,1],[35.9,67,1],[36.7,45,1],[25.7,46,0],[23.3,29,0],[31,56,1]]
n=[43.6,40]
k=3
m=[]
x=[]
for i in l:
    a=0
    for j in range(len(n)-1):
        a+= (i[j]-n[j])*(i[j]-n[j])
    m.append(sqrt(a))
a=sorted(m)
for i in range(k):
    x.append(m.index(a[i]))
y=[]
for i in x:
    print(l[i])
    y.append(l[i][-1])
print()
print("result -->",mode(y))
```

OUTPUT:

```
>>>
= RESTART: C:\Machine Learning\Codings ml\Codings\exp_5.py
[43.1, 67, 0]
[36.7, 45, 1]
[35.9, 67, 1]

result --> 1
>>>
```

EXPERIMENT 6:

exp_6.py - C:\Machine Learning\Codings ml\Codings\exp_6.py (3.12.3)

File Edit Format Run Options Window Help

```
# import required libraries
from sklearn.datasets import load_iris
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score

# load iris dataset
iris = load_iris()

# split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.3, random_state=0)

# create Naive Bayes classifier
classifier = GaussianNB()

# train the classifier using the training data
classifier.fit(X_train, y_train)

# predict the target values for the testing data
y_pred = classifier.predict(X_test)

# display confusion matrix and accuracy score
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)

acc = accuracy_score(y_test, y_pred)
print("Accuracy Score:", acc)
```

OUTPUT:

```
>>>
    = RESTART: C:\Machine Learning\Codings ml\Codings\exp_6.py
    Confusion Matrix:
    [[16  0  0]
     [ 0 18  0]
     [ 0  0 11]]
    Accuracy Score: 1.0
>>>
```

EXPERIMENT 7:

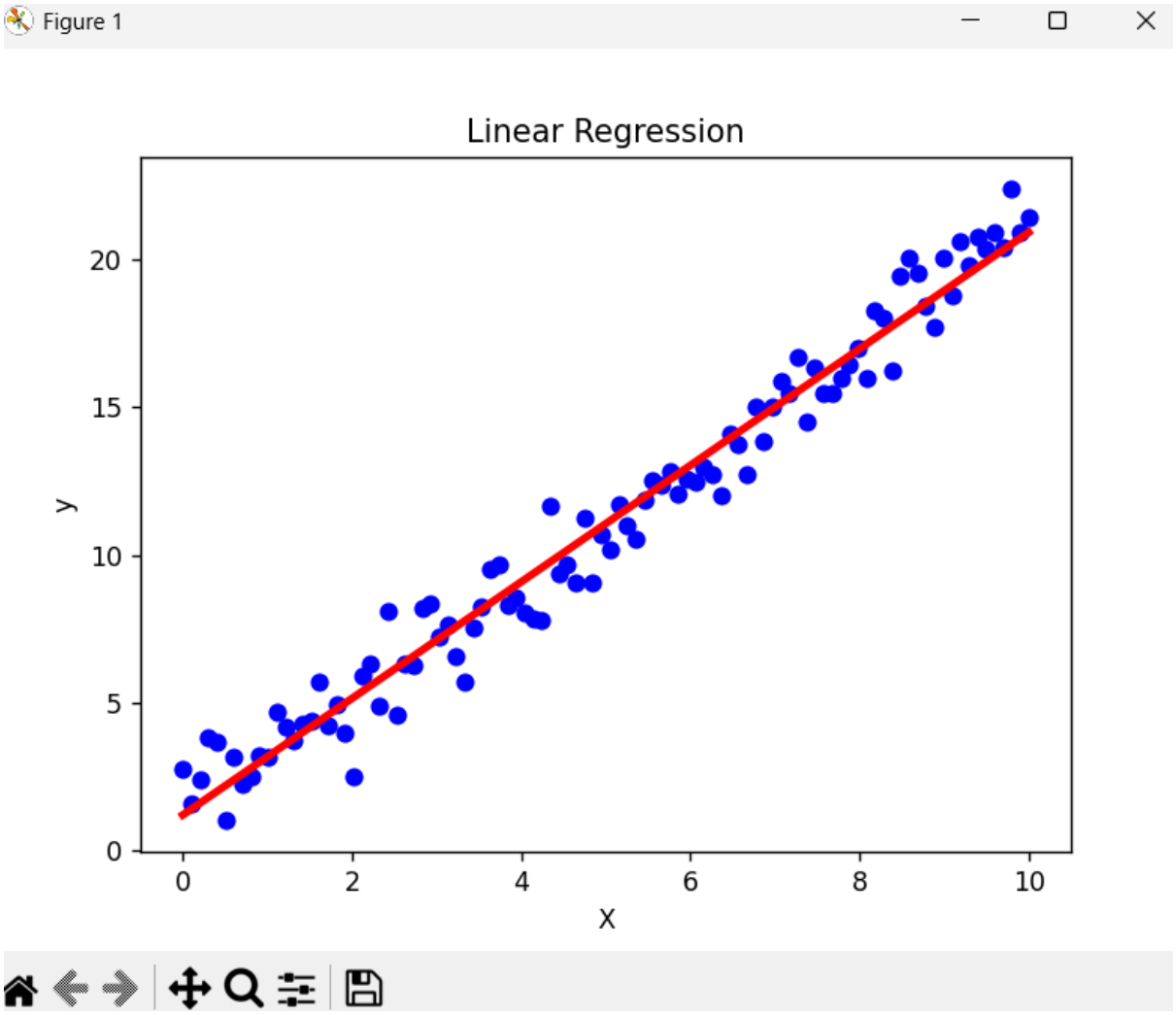
exp_7.py - C:\Machine Learning\Codings ml\Codings\exp_7.py (3.12.3)

File Edit Format Run Options Window Help

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Generate sample data
np.random.seed(0)
X = np.linspace(0, 10, 100).reshape(-1, 1)
y = 2 * X + 1 + np.random.randn(100, 1)

# Create linear regression object
lr_model = LinearRegression()

# Train the model using the training sets
lr_model.fit(X, y)

# Print the coefficients
print('Coefficients: ', lr_model.coef_)
print('Intercept: ', lr_model.intercept_)

# Plot the data and the linear regression line
plt.scatter(X, y, color='blue')
plt.plot(X, lr_model.predict(X), color='red', linewidth=3)
plt.title('Linear Regression')
plt.xlabel('X')
plt.ylabel('y')
plt.show()
```

OUTPUT:

```
========== RESTART: C:\Machine Learning\Codings ml\Codings\exp_7.py ==========
Coefficients:  [[1.97026731]]
Intercept:  [1.20847145]
>>>
```



EXPERIMENT 8:

exp_8.py - C:\Machine Learning\Codings ml\Codings\exp_8.py (3.12.3)

File Edit Format Run Options Window Help

```
# Step 1: Import the required modules
from sklearn.datasets import make_classification
from matplotlib import pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import pandas as pd

# Step 2: Generate the dataset
x, y = make_classification(
    n_samples=100,
    n_features=1,
    n_classes=2,
    n_clusters_per_class=1,
    flip_y=0.03,
    n_informative=1,
    n_redundant=0,
    n_repeated=0
)
print(y)

# Step 3: visualize the data
plt.scatter(x, y, c=y, cmap='rainbow')
plt.title('Scatter Plot of Logistic Regression')
plt.show()

# Step 4: Split the dataset
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1)

x_train.shape

# Step 4: Perform Logistic Regression
log_reg = LogisticRegression()
log_reg.fit(x_train, y_train)

# Step 5: Make prediction using the model
y_pred = log_reg.predict(x_test)

# Step 6: Display the Confusion Matrix
confusion_matrix(y_test, y_pred)
```

OUTPUT:

```
>>>
    = RESTART: C:\Machine Learning\Codings ml\Codings\exp_8.py
    [1 0 1 1 0 0 0 1 0 0 1 0 1 0 1 1 0 0 1 0 0 1 1 1 0 1 0 1 0 1 0 1 1 0 0 0 1
     0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 1 1 1 1 1 1
     1 1 1 0 0 0 1 1 1 1 0 1 0 0 1 0 0 0 1 1 0 0 0 0 1 1]
>>>
```

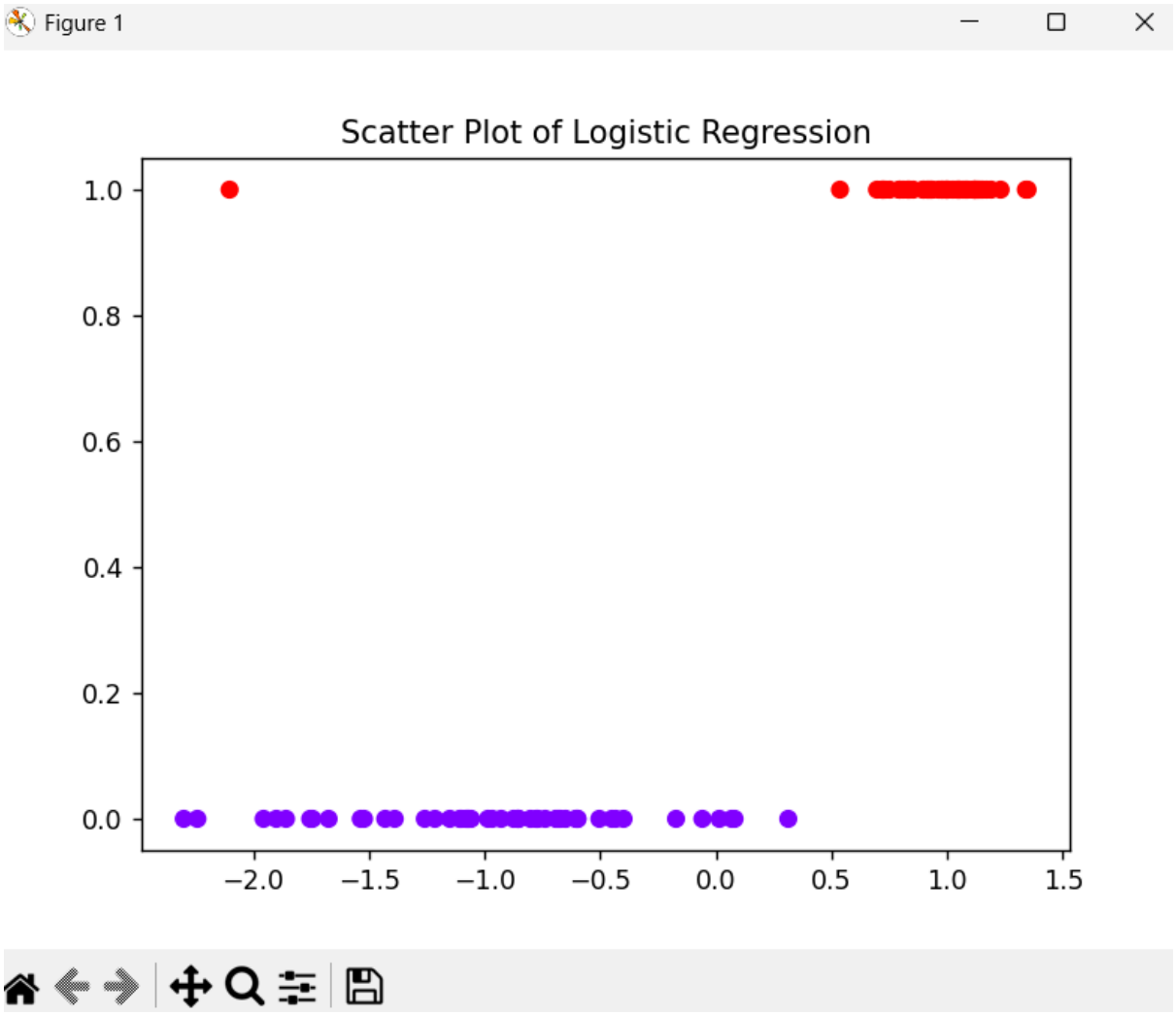


Figure 1
Scatter Plot of Logistic Regression
1.0
0.8
0.6
0.4
0.2
0.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

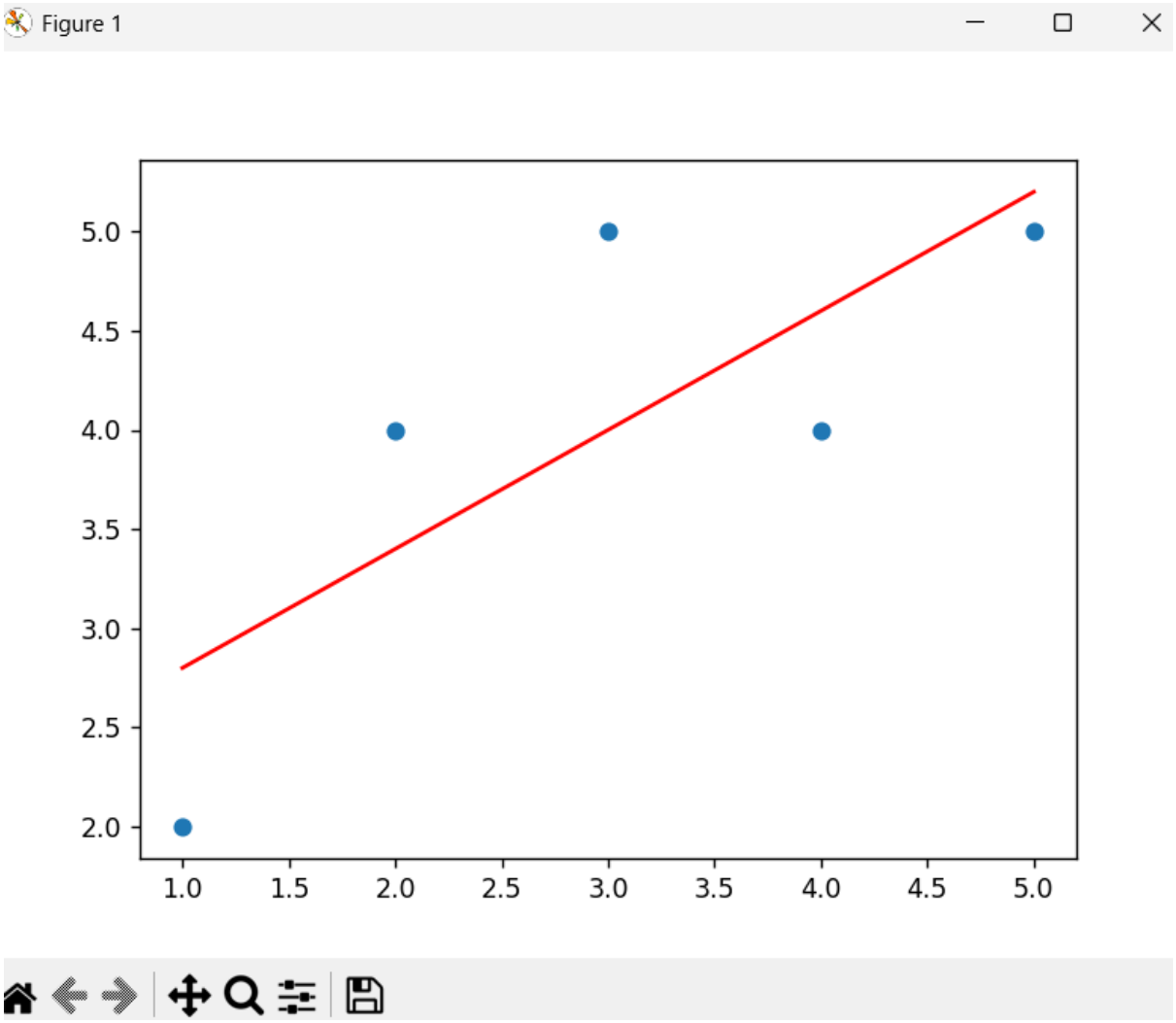
EXPERIMENT 9:

exp_9.py - C:\Machine Learning\Codings ml\Codings\exp_9.py (3.12.3)

File Edit Format Run Options Window Help

```
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Create some sample data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
y = np.array([2, 4, 5, 4, 5]).reshape(-1, 1)

# Create a linear regression object and fit the data
reg = LinearRegression().fit(X, y)

# Predict new values
X_new = np.array([6]).reshape(-1, 1)
y_pred = reg.predict(X_new)

# Plot the data and the linear regression line
plt.scatter(X, y)
plt.plot(X, reg.predict(X), color='red')
plt.show()

import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt

# Create some sample data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
y = np.array([2, 4, 5, 4, 5]).reshape(-1, 1)

# Transform the data to include another axis
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Create a polynomial regression object and fit the data
reg = LinearRegression().fit(X_poly, y)

# Predict new values
X_new = np.array([6]).reshape(-1, 1)
X_new_poly = poly.transform(X_new)
y_pred = reg.predict(X_new_poly)

# Plot the data and the polynomial regression curve
plt.scatter(X, y)
plt.plot(X, reg.predict(X_poly), color='red')
plt.show()
```

OUTPUT:



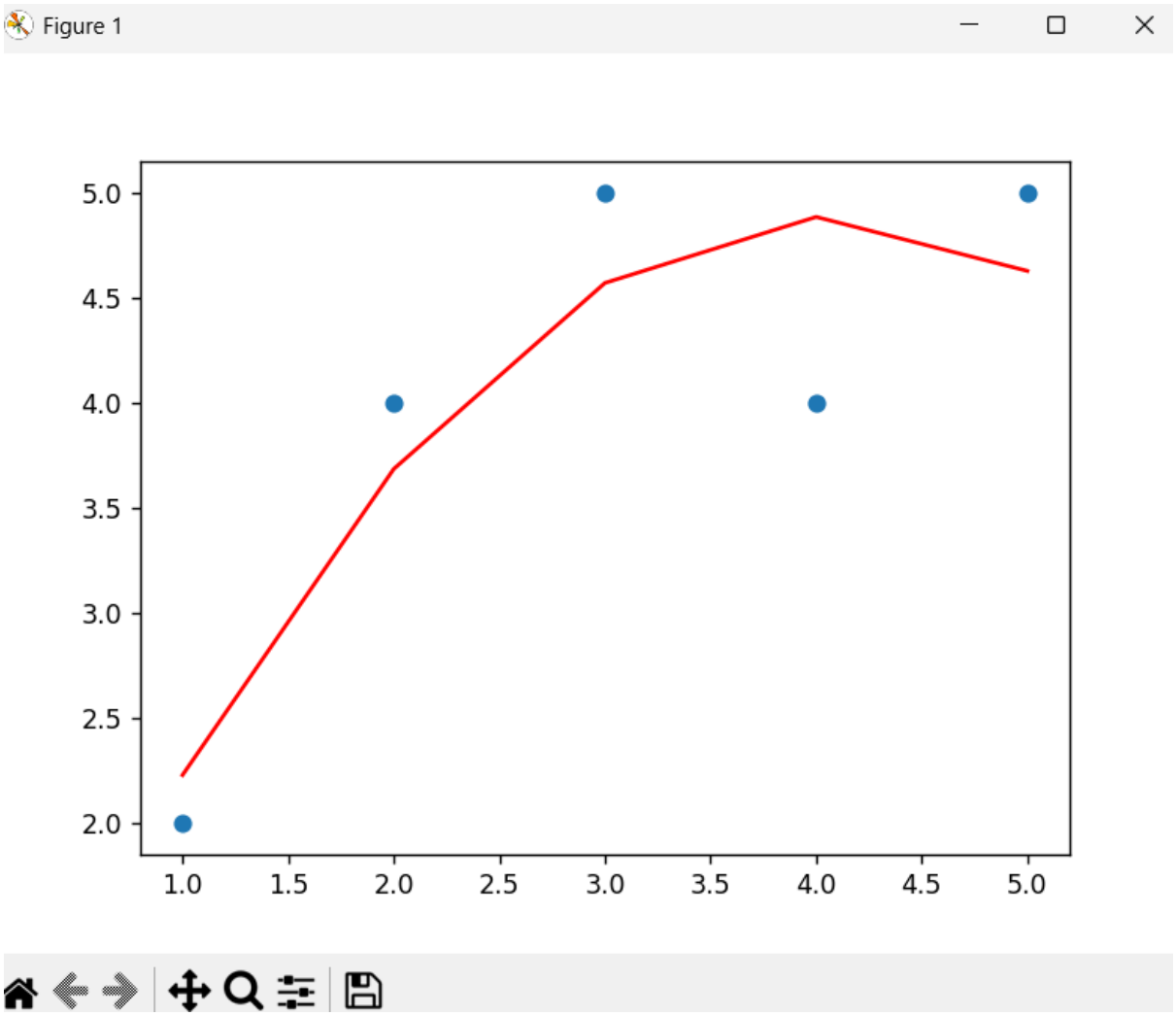


Figure 1
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

EXPERIMENT 10:

exp_10.py - C:\Machine Learning\Codings ml\Codings\exp_10.py (3.12.3)

File Edit Format Run Options Window Help

```
import numpy as np
from scipy.stats import norm

# Define the data
data = np.array([1.2, 2.3, 0.7, 1.6, 1.1, 1.8, 0.9, 2.2])

# Initialize the parameters
mu1 = 0
mu2 = 1
sigma1 = 1
sigma2 = 1
p1 = 0.5
p2 = 0.5

# Run the EM algorithm
for i in range(10):
    # E-step
    likelihood1 = norm.pdf(data, mu1, sigma1)
    likelihood2 = norm.pdf(data, mu2, sigma2)
    weight1 = p1 * likelihood1 / (p1 * likelihood1 + p2 * likelihood2)
    weight2 = p2 * likelihood2 / (p1 * likelihood1 + p2 * likelihood2)

    # M-step
    mu1 = np.sum(weight1 * data) / np.sum(weight1)
    mu2 = np.sum(weight2 * data) / np.sum(weight2)
    sigma1 = np.sqrt(np.sum(weight1 * (data - mu1)**2) / np.sum(weight1))
    sigma2 = np.sqrt(np.sum(weight2 * (data - mu2)**2) / np.sum(weight2))
    p1 = np.mean(weight1)
    p2 = np.mean(weight2)

# Print the final estimates of the parameters
print("mu1:", mu1)
print("mu2:", mu2)
print("sigma1:", sigma1)
print("sigma2:", sigma2)
print("p1:", p1)
print("p2:", p2)
```

OUTPUT:

IDLE Shell 3.12.3

File Edit Shell Debug Options Window Help

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bi
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Machine Learning\Codings ml\Codings\exp_10.py
mu1: 0.9545902456963998
mu2: 1.7595212637782114
sigma1: 0.19986282179149245
sigma2: 0.47713642731204714
p1: 0.3534728534331289
p2: 0.6465271465668712
>>>
```

EXPERIMENT 11:

Experiment 11.py - C:\Machine Learning\Codings ml\Codings\Experiment 11.py (3.12.3)

File Edit Format Run Options Window Help

```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
pio.templates.default = "plotly_white"

data = pd.read_csv(r"C:\Machine Learning\Dataset ml\New folder\CREDITSCORE.csv")
print(data.head())

print(data.info())

from sklearn.model_selection import train_test_split
x = np.array(data[["Annual_Income", "Monthly_Inhand_Salary",
                   "Num_Bank_Accounts", "Num_Credit_Card",
                   "Interest_Rate", "Num_of_Loan",
                   "Delay_from_due_date", "Num_of_Delayed_Payment",
                   "Credit_Mix", "Outstanding_Debt",
                   "Credit_History_Age", "Monthly_Balance"]])
y = np.array(data[["Credit_Score"]])

xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.33,
                                                random_state=42)
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(xtrain, ytrain)

print("Credit Score Prediction : ")
a = float(input("Annual Income: "))
b = float(input("Monthly Inhand Salary: "))
c = float(input("Number of Bank Accounts: "))
d = float(input("Number of Credit cards: "))
e = float(input("Interest rate: "))
f = float(input("Number of Loans: "))
g = float(input("Average number of days delayed by the person: "))
h = float(input("Number of delayed payments: "))
i = input("Credit Mix (Bad: 0, Standard: 1, Good: 3) : ")
j = float(input("Outstanding Debt: "))
k = float(input("Credit History Age: "))
l = float(input("Monthly Balance: "))

features = np.array([[a, b, c, d, e, f, g, h, i, j, k, l]])
print("Predicted Credit Score = ", model.predict(features))
```

OUTPUT:

```
>>>
= RESTART: C:\Machine Learning\Codings ml\Codings\Experiment 11.py
     ID  Customer_ID  ...  Monthly_Balance Credit_Score
0  5634         3392  ...       312.494089         Good
1  5635         3392  ...       284.629162         Good
2  5636         3392  ...       331.209863         Good
3  5637         3392  ...       223.451310         Good
4  5638         3392  ...       341.489231         Good

[5 rows x 28 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 28 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   ID                        100000 non-null  int64
 1   Customer_ID               100000 non-null  int64
 2   Month                     100000 non-null  int64
 3   Name                      100000 non-null  object
 4   Age                       100000 non-null  float64
 5   SSN                       100000 non-null  float64
 6   Occupation                100000 non-null  object
 7   Annual_Income             100000 non-null  float64
 8   Monthly_Inhand_Salary     100000 non-null  float64
 9   Num_Bank_Accounts         100000 non-null  float64
 10  Num_Credit_Card           100000 non-null  float64
 11  Interest_Rate             100000 non-null  float64
 12  Num_of_Loan               100000 non-null  float64
 13  Type_of_Loan              100000 non-null  object
 14  Delay_from_due_date       100000 non-null  float64
 15  Num_of_Delayed_Payment    100000 non-null  float64
 16  Changed_Credit_Limit      100000 non-null  float64
 17  Num_Credit_Inquiries      100000 non-null  float64
 18  Credit_Mix                100000 non-null  object
 19  Outstanding_Debt          100000 non-null  float64
 20  Credit_Utilization_Ratio  100000 non-null  float64
 21  Credit_History_Age        100000 non-null  float64
 22  Payment_of_Min_Amount     100000 non-null  object
 23  Total_EMI_per_month       100000 non-null  float64
 24  Amount_invested_monthly   100000 non-null  float64
 25  Payment_Behaviour         100000 non-null  object
 26  Monthly_Balance           100000 non-null  float64
 27  Credit_Score              100000 non-null  object
dtypes: float64(18), int64(3), object(7)
memory usage: 21.4+ MB
None
```

EXPERIMENT 12:

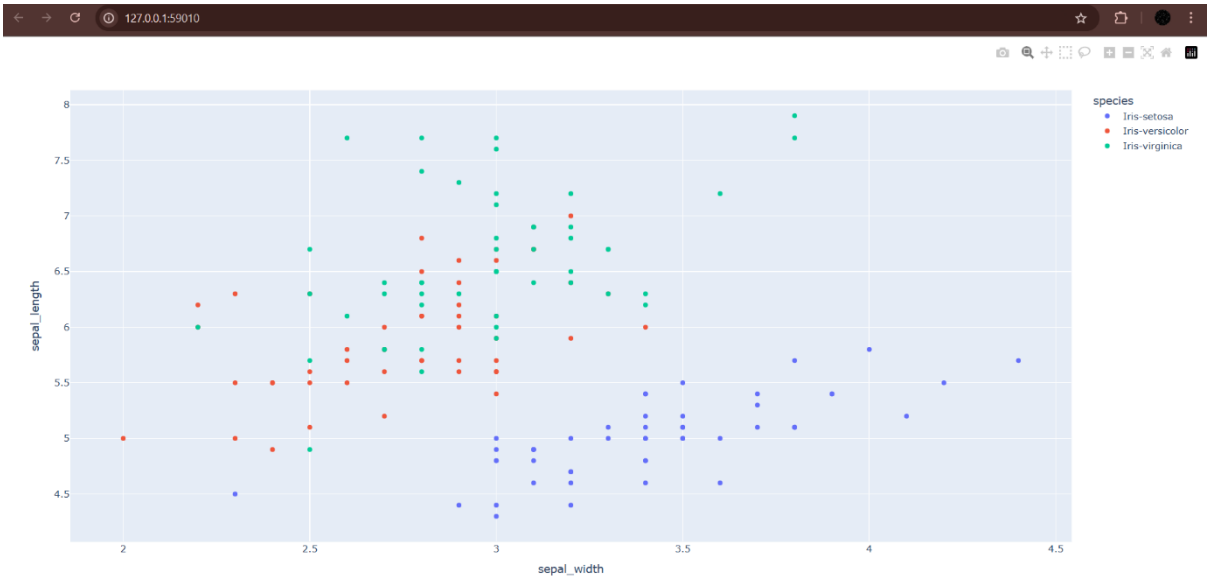
Experiment 12.py - C:\Machine Learning\Codings ml\Codings\Experiment 12.py (3.12.3)

File Edit Format Run Options Window Help

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
iris = pd.read_csv(r"C:\Machine Learning\Dataset ml\New folder\IRIS.csv")
print(iris.head())
print()
print(iris.describe())
print("Target Labels", iris["species"].unique())

import plotly.io as io
import plotly.express as px
fig = px.scatter(iris, x="sepal_width", y="sepal_length", color="species")
fig.show()
x = iris.drop("species", axis=1)
y = iris["species"]
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,test_size=0.2,random_state=0)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(x_train, y_train)

x_new = np.array([[6, 2.9, 1, 0.2]])
prediction = knn.predict(x_new)
print("Prediction: {}".format(prediction))
```

OUTPUT:

```
======= RESTART: C:\Machine Learning\Codings ml\Codings\Experiment 12.py =======
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa

       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
Target Labels ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
UserWarning: X does not have valid feature nam
Prediction: ['Iris-setosa']
>>>
```


127.0.0.1:59010
species
Iris-setosa
Iris-versicolor
Iris-virginica
sepal_length
sepal_width
8
7.5
7
6.5
6
5.5
5
4.5
2
2.5
3
3.5
4
4.5

EXPERIMENT 13:

Experiment 13.py - C:\Machine Learning\Codings ml\Codings\Experiment 13.py (3.12.3)

File Edit Format Run Options Window Help

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor

#Importing the dataset
data = pd.read_csv(r"C:\Machine Learning\Dataset ml\New folder\CarPrice.csv")

#Data Exploration
data.head()
data.shape
data.isnull().sum() #Checking if the dataset has NULL Values
data.info()
data.describe()
data.CarName.unique()

#Analysing correlations & using heatmap
print(data.corr())
plt.figure(figsize=(20, 15))
correlations = data.corr()
sns.heatmap(correlations, cmap="coolwarm", annot=True)
plt.show()

#Training a Car Price Prediction Model
predict = "price"
data = data[["symboling", "wheelbase", "carlength",
             "carwidth", "carheight", "curbweight",
             "enginesize", "boreratio", "stroke",
             "compressionratio", "horsepower", "peakrpm",
             "citympg", "highwaympg", "price"]]
x = np.array(data.drop([predict], 1))
y = np.array(data[predict])

from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2)

from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()
model.fit(xtrain, ytrain)
predictions = model.predict(xtest)

from sklearn.metrics import mean_absolute_error
model.score(xtest, predictions)
```

OUTPUT:

```
========== RESTART: C:\Machine Learning\Codings ml\Codings\Experiment 13.py =========
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

EXPERIMENT 14:

Experiment 14.py - C:\Machine Learning\Codings ml\Codings\Experiment 14.py (3.12.3)

File Edit Format Run Options Window Help

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
#Importing Dataset
dataset = pd.read_csv(r"C:\Machine Learning\Dataset ml\New folder\HousePricePrediction.csv")
#Exploring dataset
print(dataset.head(5))
dataset.shape
obj = (dataset.dtypes == 'object')
object_cols = list(obj[obj].index)
print("Categorical variables:",len(object_cols))

int_ = (dataset.dtypes == 'int')
num_cols = list(int_[int_].index)
print("Integer variables:",len(num_cols))

fl = (dataset.dtypes == 'float')
fl_cols = list(fl[fl].index)
print("Float variables:",len(fl_cols))

plt.figure(figsize=(12, 6))
sns.heatmap(dataset.corr(),
            cmap = 'BrBG',
            fmt = '.2f',
            linewidths = 2,
            annot = True)

unique_values = []
for col in object_cols:
    unique_values.append(dataset[col].unique().size)
    plt.figure(figsize=(10,6))
    plt.title('No. Unique values of Categorical Features')
    plt.xticks(rotation=90)
sns.barplot(x=object_cols,y=unique_values)

plt.figure(figsize=(18, 36))
plt.title('Categorical Features: Distribution')
plt.xticks(rotation=90)
index = 1

for col in object_cols:
        y = dataset[col].value_counts()
        plt.subplot(11, 4, index)
        plt.xticks(rotation=90)
        sns.barplot(x=list(y.index), y=y)
        index += 1
dataset.drop(['Id'],axis=1,inplace=True)
```

```
dataset['SalePrice'] = dataset['SalePrice'].fillna(dataset['SalePrice'].mean())
new_dataset = dataset.dropna()
new_dataset.isnull().sum()

from sklearn.preprocessing import OneHotEncoder
s = (new_dataset.dtypes == 'object')
object_cols = list(s[s].index)
print("Categorical variables:")
print(object_cols)
print('No. of. categorical features: ',len(object_cols))

OH_encoder = OneHotEncoder(sparse=False)
OH_cols = pd.DataFrame(OH_encoder.fit_transform(new_dataset[object_cols]))
OH_cols.index = new_dataset.index
OH_cols.columns = OH_encoder.get_feature_names()
df_final = new_dataset.drop(object_cols, axis=1)
df_final = pd.concat([df_final, OH_cols], axis=1)
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split

X = df_final.drop(['SalePrice'], axis=1)
Y = df_final['SalePrice']

X_train, X_valid, Y_train, Y_valid = train_test_split(X, Y, train_size=0.8, test_size=0.2, random_state=0)

from sklearn import svm
from sklearn.svm import SVC
from sklearn.metrics import mean_absolute_percentage_error

model_SVR = svm.SVR()
model_SVR.fit(X_train,Y_train)
Y_pred = model_SVR.predict(X_valid)

print(mean_absolute_percentage_error(Y_valid, Y_pred))

#LinearRegression
from sklearn.linear_model import LinearRegression

model_LR = LinearRegression()
model_LR.fit(X_train, Y_train)
Y_pred = model_LR.predict(X_valid)

print(mean_absolute_percentage_error(Y_valid, Y_pred))
```

OUTPUT:

```
>>>
= RESTART: C:\Machine Learning\Codings ml\Codings\Experiment 14.py
   Id  MSSubClass MSZoning  ...  BsmtFinSF2 TotalBsmtSF SalePrice
0   0          60       RL  ...         0.0       856.0  208500.0
1   1          20       RL  ...         0.0      1262.0  181500.0
2   2          60       RL  ...         0.0       920.0  223500.0
3   3          70       RL  ...         0.0       756.0  140000.0
4   4          60       RL  ...         0.0      1145.0  250000.0

[5 rows x 13 columns]
Categorical variables: 4
Integer variables: 6
Float variables: 3
```

EXPERIMENT 15:

Experiment 15.py - C:\Machine Learning\Codings ml\Codings\Experiment 15.py (3.12.3)

File Edit Format Run Options Window Help

```
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn import datasets
from sklearn.metrics import confusion_matrix
#Load the iris dataset
iris = datasets.load_iris()
#GaussianNB and MultinomialNB Models
gnb = GaussianNB()
mnb = MultinomialNB()
#Train both GaussianNB and MultinomialNB Models and print their confusion matrices
y_pred_gnb = gnb.fit(iris.data, iris.target).predict(iris.data)
cnf_matrix_gnb = confusion_matrix(iris.target, y_pred_gnb)
print("Confusion Matrix of GNB \n",cnf_matrix_gnb)

y_pred_mnb = mnb.fit(iris.data, iris.target).predict(iris.data)
cnf_matrix_mnb = confusion_matrix(iris.target, y_pred_mnb)
print("Confusion Matrix of MNB \n",cnf_matrix_mnb)
```

OUTPUT:

```
>>>
= RESTART: C:\Machine Learning\Codings ml\Codings\Experiment 15.py
Confusion Matrix of GNB
 [[50  0  0]
 [ 0 47  3]
 [ 0  3 47]]
Confusion Matrix of MNB
 [[50  0  0]
 [ 0 46  4]
 [ 0  3 47]]
>>>
```

EXPERIMENT 16:

Experiment 16.py - C:\Machine Learning\Codings ml\Codings\Experiment 16.py (3.12.3)

File Edit Format Run Options Window Help

```
import numpy
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import BernoulliNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.metrics import classification_report

iris= pd.read_csv(r"C:\Machine Learning\Dataset ml\New folder\IRIS.csv")
print(iris.head())

x = iris.drop("species", axis=1)
y = iris["species"]
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,test_size=0,random_state=42)

#x = np.array(data[["Age", "EstimatedSalary"]])
#y = np.array(data[["Purchased"]])

#xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.10, random_state=42)
decisiontree = DecisionTreeClassifier()
logisticregression = LogisticRegression()
knearestclassifier = KNeighborsClassifier()
#svm_classifier = SVC()
bernoulli_naiveBayes = BernoulliNB()
passiveAggressive = PassiveAggressiveClassifier()

knearestclassifier.fit(x_train, y_train)
decisiontree.fit(x_train, y_train)
logisticregression.fit(x_train, y_train)
passiveAggressive.fit(x_train, y_train)

data1 = {"Classification Algorithms": ["KNN Classifier", "Decision Tree Classifier",
                                       "Logistic Regression", "Passive Aggressive Classifier"],
      "Score": [knearestclassifier.score(x,y), decisiontree.score(x, y),
                logisticregression.score(x, y), passiveAggressive.score(x,y) ]}
score = pd.DataFrame(data1)
score
```

OUTPUT:

```
>>>
= RESTART: C:\Machine Learning\Codings ml\Codings\Experiment 16.py
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
```

EXPERIMENT 17:

Experiment 17.py - C:\Machine Learning\Codings ml\Codings\Experiment 17.py (3.12.3)

File Edit Format Run Options Window Help

```
import numpy as np
import pandas as pd
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
#importing dataset
data = pd.read_csv(r"C:\Machine Learning\Dataset ml\New folder\mobile_prices.csv")
print(data.head())
plt.figure(figsize=(12, 10))
sns.heatmap(data.corr(), annot=True, cmap="coolwarm", linecolor='white', linewidths=1)

#data preparation
x = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
x = StandardScaler().fit_transform(x)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=0)

# Logistic Regression algorithm provided by Scikit-learn:
from sklearn.linear_model import LogisticRegression
lreg = LogisticRegression()
lreg.fit(x_train, y_train)
y_pred = lreg.predict(x_test)
 #accuracy of the model:
accuracy = accuracy_score(y_test, y_pred) * 100
print("Accuracy of the Logistic Regression Model: ",accuracy)

#predictions made by the model:
print(y_pred)

(unique, counts) = np.unique(y_pred, return_counts=True)
price_range = np.asarray((unique, counts)).T
print(price_range)
```

OUTPUT:

```
>>>
= RESTART: C:\Machine Learning\Codings ml\Codings\Experiment 17.py
   battery_power  blue  clock_speed  ...  touch_screen  wifi  price_range
0            842     0          2.2  ...             0     1            1
1           1021     1          0.5  ...             1     0            2
2            563     1          0.5  ...             1     0            2
3            615     1          2.5  ...             0     0            2
4           1821     1          1.2  ...             1     0            1

[5 rows x 21 columns]
Accuracy of the Logistic Regression Model:  95.5
[3 0 2 2 3 0 0 3 3 1 1 3 0 2 3 0 3 2 2 1 0 0 3 1 2 2 3 1 3 1 1 0 2 0 2 3 0
 0 3 3 3 1 3 3 1 3 0 1 3 1 1 3 0 3 0 2 2 2 0 3 3 1 3 2 1 2 3 2 2 2 3 2 1 0
 1 3 2 2 1 2 3 3 3 0 0 0 2 1 2 3 1 2 2 1 0 3 3 3 0 3 1 1 3 1 3 2 2 3 2 3 3
 0 0 1 3 3 0 0 1 0 0 3 2 2 1 2 1 1 0 2 1 3 3 3 3 3 3 2 0 1 1 2 1 3 0 3 0 0
 2 0 1 1 1 1 3 0 0 3 1 3 2 1 3 1 2 3 3 2 1 0 3 1 2 3 3 0 2 2 3 1 2 1 0 1 2
 2 2 0 3 3 1 1 0 2 3 0 1 2 2 0 3 3 3 1 2 3 3 3 0 0 0 2 3 3 0 0 1 3 2 3 3 3
 0 0 2 3 3 1 0 2 0 0 0 3 2 1 2 2 2 1 1 0 2 3 3 0 0 1 3 3 1 3 0 3 1 1 0 2 3 3
 2 0 0 1 2 3 2 2 3 2 1 0 3 3 2 1 3 2 2 2 1 0 2 2 1 0 0 2 2 2 3 0 1 3 0 2 2
 3 0 2 0 1 1 3 0 0 2 3 1 2 0 2 0 3 0 3 3 2 3 1 2 2 1 1 1 0 1 0 3 1 0 3 0 0
 1 3 0 3 1 1 0 1 3 0 2 1 1 2 1 1 0 2 0 0 3 1 2 3 2 2 0 3 2 2 1 3 2 3 3 3 0
 2 0 3 0 1 1 2 3 1 3 1 2 0 1 2 3 0 0 1 3 0 3 0 2 2 1 1 0 2 0]
[[  0  95]
 [  1  90]
 [  2  97]
 [  3 118]]
>>>
```

EXPERIMENT 18:

Experiment 18.py - C:\Machine Learning\Codings ml\Codings\Experiment 18.py (3.12.3)

File Edit Format Run Options Window Help

```
from sklearn import datasets
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

iris = datasets.load_iris()
X = iris.data[:, [2, 3]]
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(
 X, y, test_size=0.3, random_state=1, stratify=y)
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

ppn = Perceptron(eta0=0.1, random_state=1)
ppn.fit(X_train_std, y_train)
y_pred = ppn.predict(X_test_std)

print('Accuracy: %.3f' % accuracy_score(y_test, y_pred))
print('Accuracy: %.3f' % ppn.score(X_test_std, y_test))
```

OUTPUT:

```
>>
   = RESTART: C:\Machine Learning\Codings ml\Codings\Experiment 18.py
   Accuracy: 0.978
   Accuracy: 0.978
>>
```

EXPERIMENT 19:

Experiment 19.py - C:\Machine Learning\Codings ml\Codings\Experiment 19.py (3.12.3)

File Edit Format Run Options Window Help

```
import numpy as np
import pandas as pd

dataset = pd.read_csv(r"C:\Machine Learning\Dataset ml\New folder\breastcancer.csv")
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
GaussianNB(priors=None, var_smoothing=1e-09)

from sklearn.metrics import confusion_matrix, accuracy_score
y_pred = classifier.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

OUTPUT:

```
= RESTART: C:\Machine Learning\Codings ml\Codings\Experiment 19.py
[[99  8]
 [ 2 62]]
>>>
```

EXPERIMENT 20:

Experiment 20.py - C:\Machine Learning\Codings ml\Codings\Experiment 20.py (3.12.3)

File Edit Format Run Options Window Help

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import plotly.io as io
io.renderers.default='browser'

data = pd.read_csv(r"C:\Machine Learning\Dataset ml\New folder\futuresale prediction.csv")
print(data.head())
print(data.sample(5))
print(data.isnull().sum())

import plotly.express as px
import plotly.graph_objects as go
figure = px.scatter(data_frame = data, x="Sales",
                    y="TV", size="TV", trendline="ols")
figure.show()

figure = px.scatter(data_frame = data, x="Sales",
                    y="Newspaper", size="Newspaper", trendline="ols")
figure.show()

figure = px.scatter(data_frame = data, x="Sales",
                    y="Radio", size="Radio", trendline="ols")
figure.show()

correlation = data.corr()
print(correlation["Sales"].sort_values(ascending=False))
x = np.array(data.drop(["Sales"], 1))
y = np.array(data["Sales"])



xtrain, xtest, ytrain, ytest = train_test_split(x, y,   test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(xtrain, ytrain)
print(model.score(xtest, ytest))
features = [[TV, Radio, Newspaper]]
features = np.array([[230.1, 37.8, 69.2]])
print(model.predict(features))
```

OUTPUT:

```
>>>
========== RESTART: C:\Machine Learning\Codings ml\Codings\Experiment 20.py ==========
      TV  Radio  Newspaper  Sales
0  230.1   37.8       69.2   22.1
1   44.5   39.3       45.1   10.4
2   17.2   45.9       69.3   12.0
3  151.5   41.3       58.5   16.5
4  180.8   10.8       58.4   17.9
        TV  Radio  Newspaper  Sales
186  139.5    2.1       26.6   10.3
69   216.8   43.9       27.2   22.3
75    16.9   43.7       89.4    8.7
151  121.0    8.4       48.7   11.6
179  165.6   10.0       17.6   17.6
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```