BCA 507(C) Lab on Machine Learning using Python

1. Write a python program to find mean, mode, median.

Ass: Mean:

```
import numpy

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

x = numpy.mean(speed)

print(x)
```

Output: 89.76923076923077

Median:

```
import numpy

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

x = numpy.median(speed)

print(x)
```

Output: 87.0

Mode:

```
from scipy import stats

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

x = stats.mode(speed)

print(x)
```

Output: ModeResult(mode=array([86]), count=array([3]))

2. Write a python program to typical normal data distribution.

Ass: from numpy import random

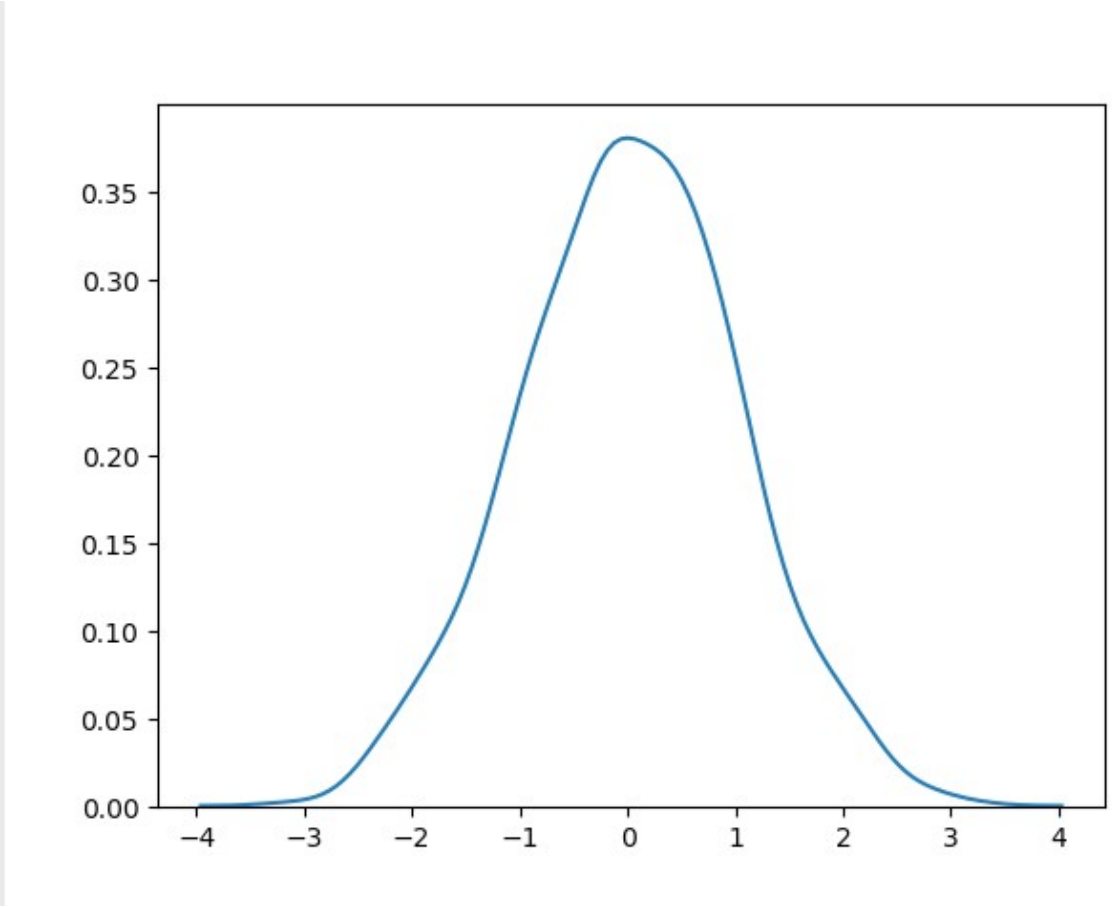
import matplotlib.pyplot as plt

import seaborn as sns

sns.distplot(random.normal(size=1000), hist=False)

plt.show()

Output:

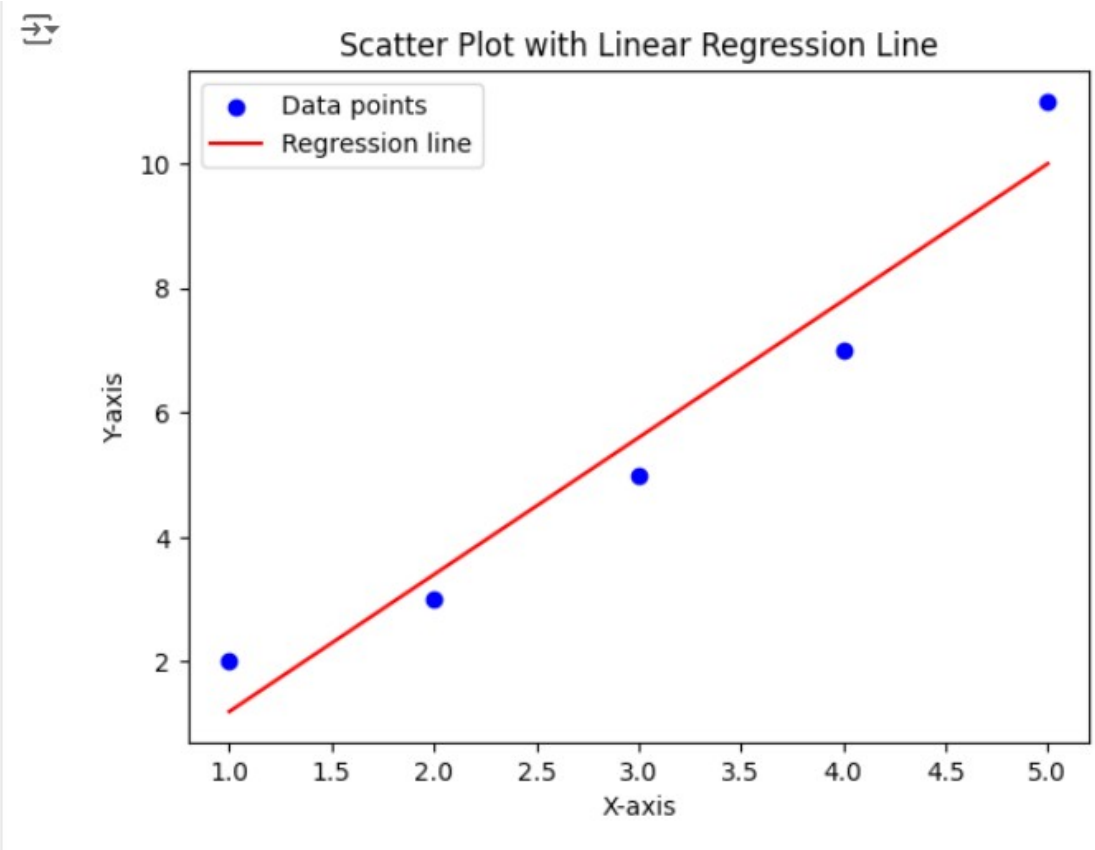


3. Write a python program to draw scatter plot of linear regression.

Ass: import numpy as np

```
import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4, 5])

y = np.array([2, 3, 5, 7, 11])

plt.scatter(x, y, color='blue', label='Data points')

m, b = np.polyfit(x, y, 1)

plt.plot(x, m*x + b, color='red', label='Regression line')

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.title('Scatter Plot with Linear Regression Line')

plt.legend()

plt.show()
```

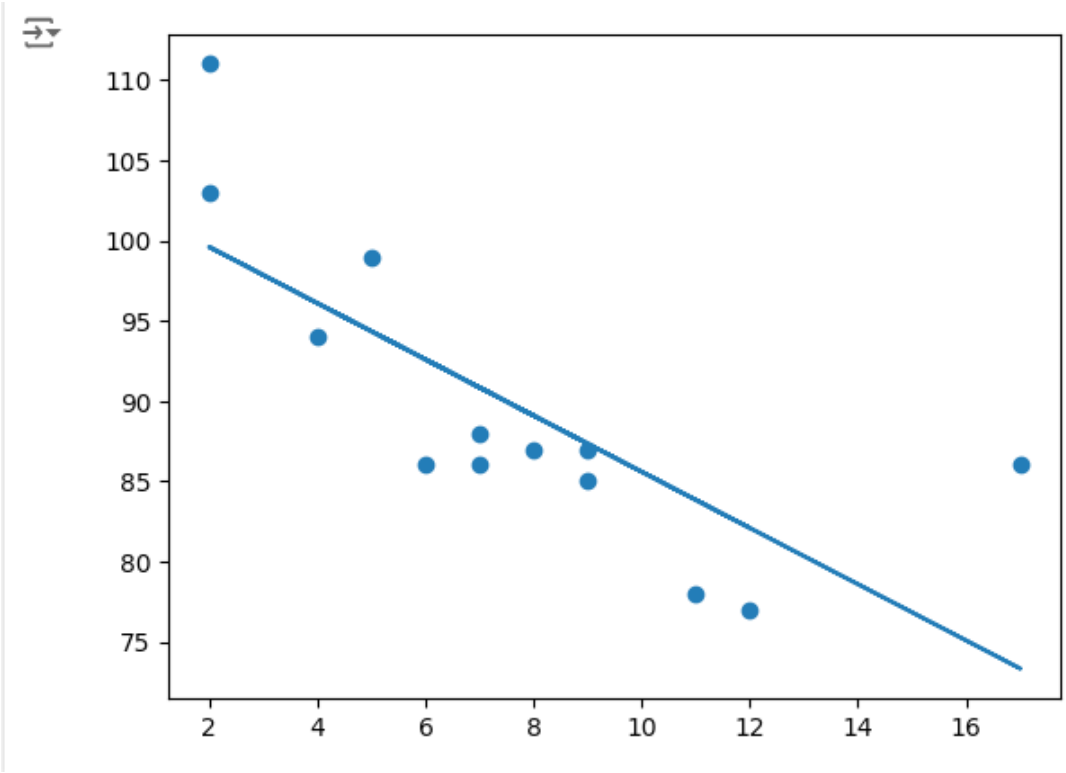
Output:



4. Write a python program to draw the line of Linear Regression.

Ass:
```
import matplotlib.pyplot as plt
from scipy import stats
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]
slope, intercept, r, p, std_err = stats.linregress(x, y)
def myfunc(x):
  return slope * x + intercept
mymodel = list(map(myfunc, x))
plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```

Output:

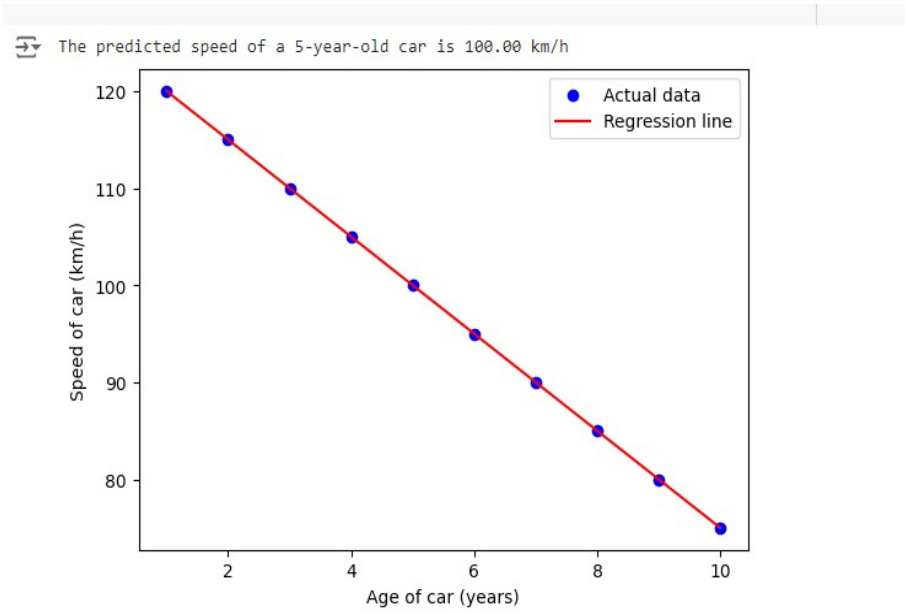


5. Write a python program to predict the speed of a 5 years old car.

Ass: import numpy as np

```
from scipy import stats
import matplotlib.pyplot as plt
# Sample data: ages of cars (in years) and their speeds (in km/h)
ages = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
speeds = np.array([120, 115, 110, 105, 100, 95, 90, 85, 80, 75])
slope, intercept, r_value, p_value, std_err = stats.linregress(ages, speeds)
# Function to predict speed based on age
def predict_speed(age):
    return slope * age + intercept
predicted_speed = predict_speed(5)
print(f"The predicted speed of a 5-year-old car is {predicted_speed:.2f} km/h")
plt.scatter(ages, speeds, color='blue', label='Actual data')
plt.plot(ages, predict_speed(ages), color='red', label='Regression line')
plt.xlabel('Age of car (years)')
plt.ylabel('Speed of car (km/h)')
plt.legend()
plt.show()
```

Output:



6. Write a python program to print the coefficient values of the regression object.

Ass: import numpy as np

```
from sklearn.linear_model import LinearRegression

# Sample data

X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])

y = np.dot(X, np.array([1, 2])) + 3

# Create and fit the model

model = LinearRegression().fit(X, y)

# Print the coefficients

print("Coefficients:", model.coef_)

print("Intercept:", model.intercept_)
```

Output:

```
Coefficients: [1. 2.]
Intercept: 3.0000000000000018
```

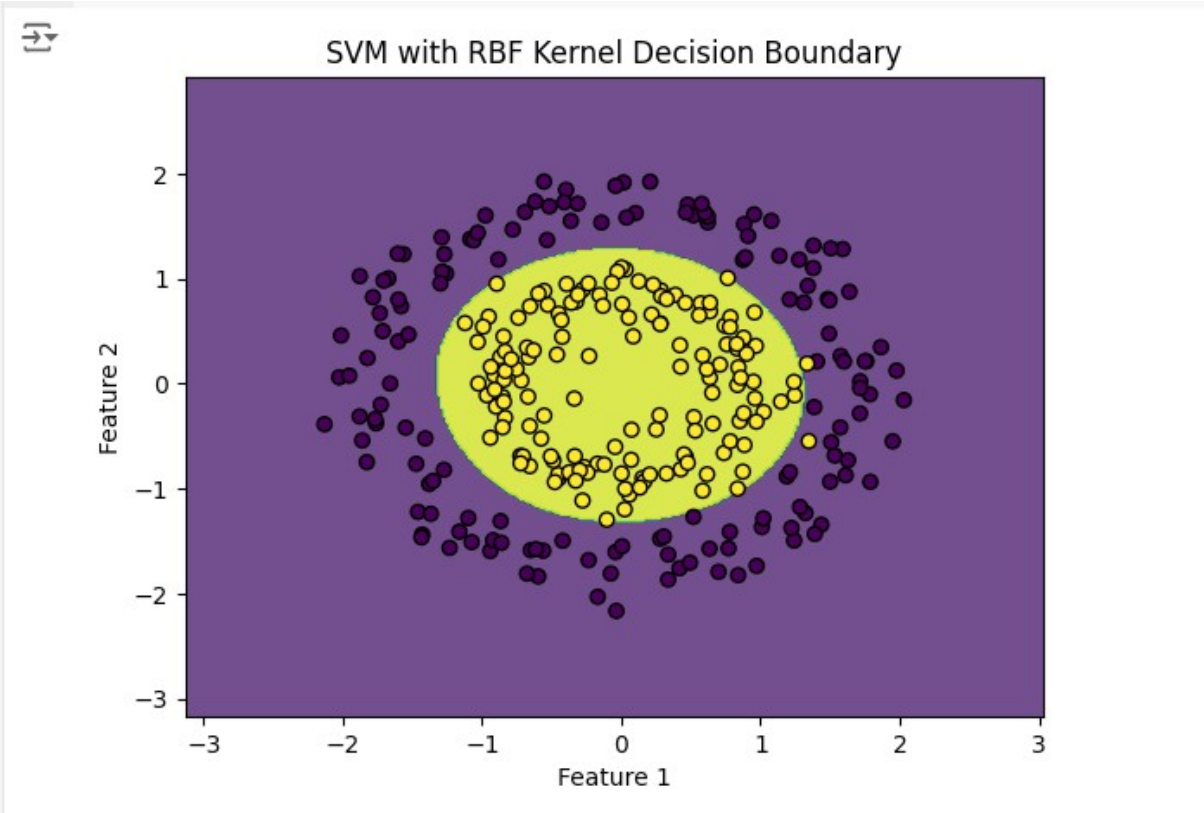
7. Write a python program to 2d binary classification data generated by make_circles() have a spherical decision boundary.

Ass:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
# Generate 2D binary classification data
X, y = make_circles(n_samples=300, factor=0.5, noise=0.1, random_state=42)
# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Fit the SVM model with RBF kernel
svm = SVC(kernel='rbf', C=1.0, gamma='auto')
svm.fit(X_scaled, y)
# Plot the decision boundary
def plot_decision_boundary(model, X, y):
    h = .02  # step size in the mesh
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.8)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
```

```
    plt.title('SVM with RBF Kernel Decision Boundary')
    plt.show()
plot_decision_boundary(svm, X_scaled, y)
```

Output:

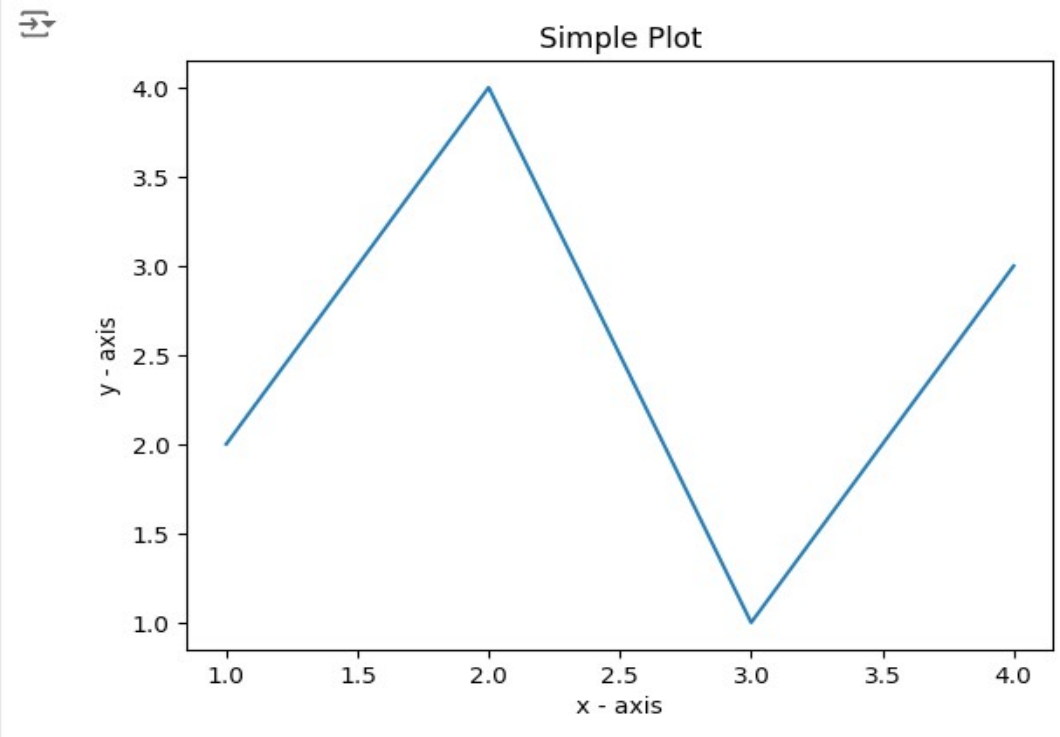


8. Write a python program to display the plot we can use the functions plot() and show() from pyplot.

Ass: import matplotlib.pyplot as plt

```
# Data for plotting
x = [1, 2, 3, 4]
y = [2, 4, 1, 3]
# Creating the plot
plt.plot(x, y)
# Adding labels and title
plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.title('Simple Plot')
# Displaying the plot
plt.show()
```

Output:



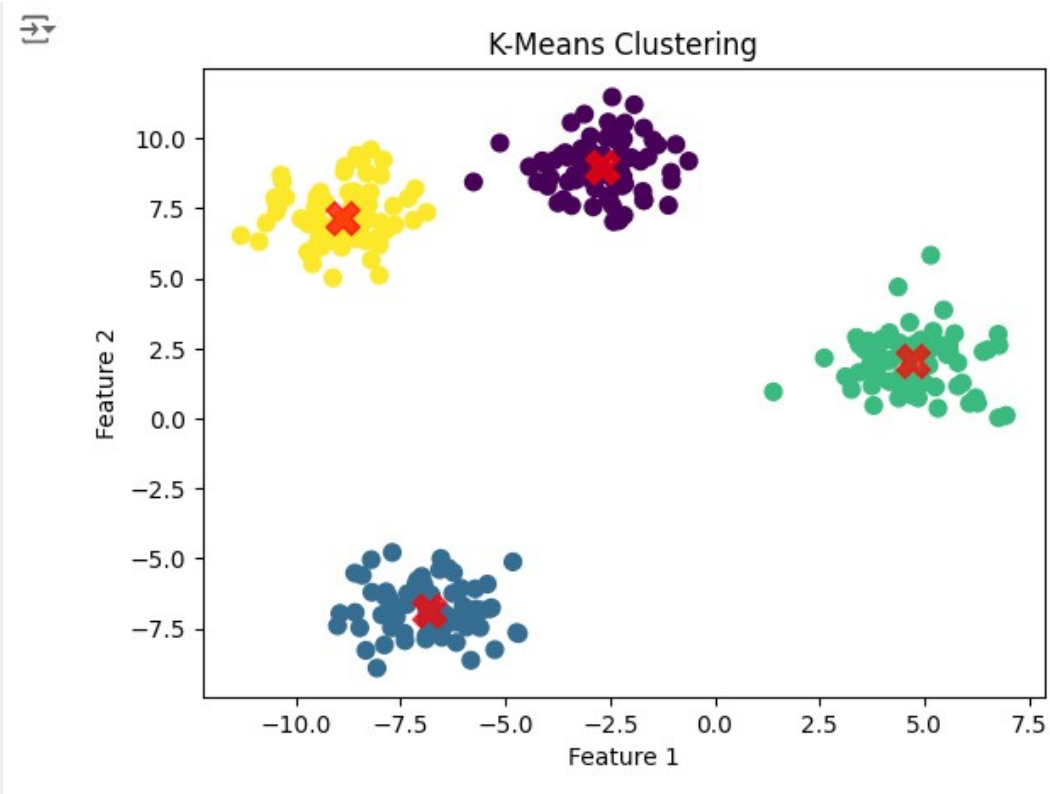
9. Write a python program to data generated by the function make_blobs() are blobs that can be utilized for clustering.

Ass: import matplotlib.pyplot as plt

```
from sklearn.datasets import make_blobs
```

```
from sklearn.cluster import KMeans

# Generate synthetic data

X, y = make_blobs(n_samples=300, centers=4, n_features=2, cluster_std=1.0, random_state=42)

# Apply K-Means clustering

kmeans = KMeans(n_clusters=4, random_state=42)

kmeans.fit(X)

y_kmeans = kmeans.predict(X)

# Plot the clusters

plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster_centers_

plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('K-Means Clustering')

plt.show()
```

Output:



10. Write a python program to random multi-label classification data is created by the function make make_multilabel_classification().

Ass: from sklearn.datasets import make_multilabel_classification

import matplotlib.pyplot as plt

Generate random multi-label classification data

X, y = make_multilabel_classification(n_samples=100, n_features=20, n_classes=5, n_labels=3, random_state=42)

Print the shape of the features and labels

print("Features shape:", X.shape)

print("Labels shape:", y.shape)

Plot the first two features

plt.scatter(X[:, 0], X[:, 1], marker='o', c=y[:, 0], edgecolor='k')

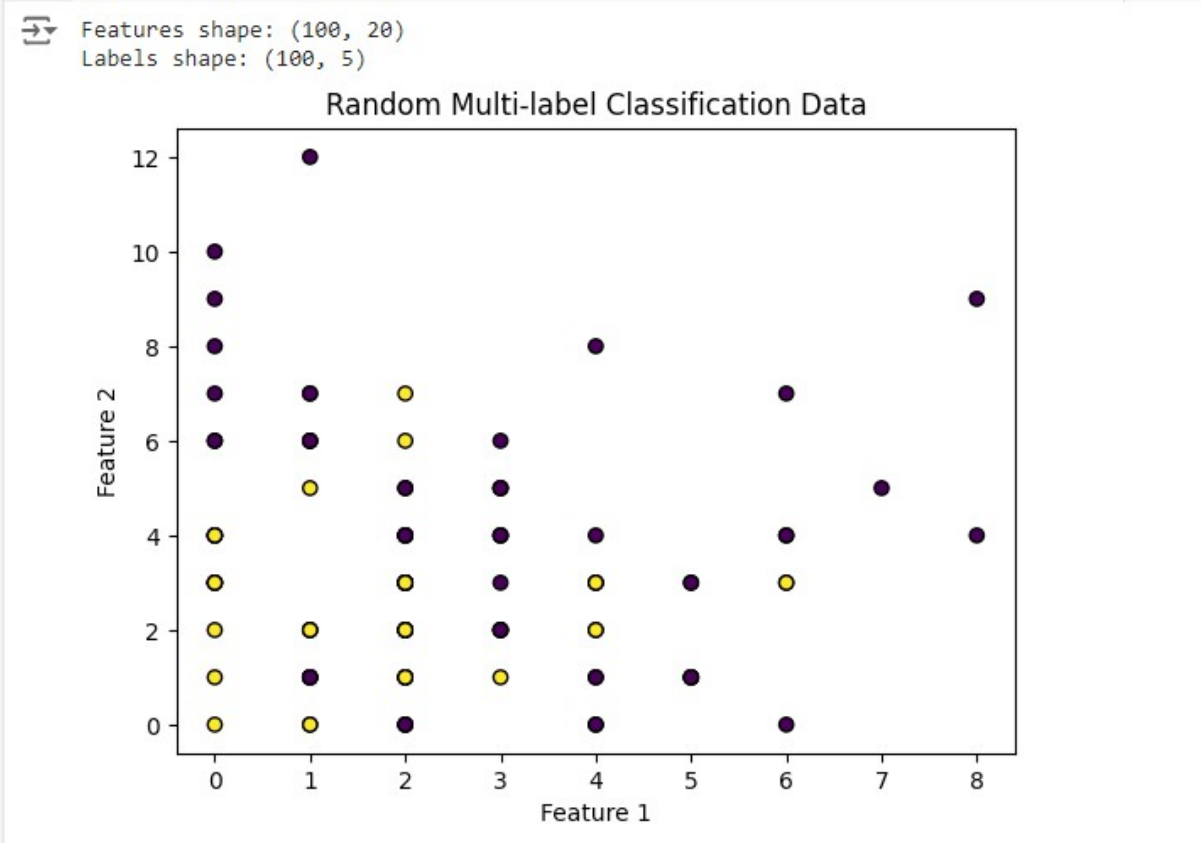
plt.title("Random Multi-label Classification Data")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.show()

Output:



11. Write a python program to implement the KNN algorithm.

Ass: import numpy as np

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# Create the KNN classifier
knn = KNeighborsClassifier(n_neighbors=3)
# Fit the classifier to the training data
knn.fit(X_train, y_train)
# Predict the labels for the test set
y_pred = knn.predict(X_test)
# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

Output:

```
Accuracy: 100.00%
```

12. Write a python program to creating a dataframe to implement one hot encoding from CSV file.

Ass: (Using Data frame)

```
 import pandas as pd

# Load the CSV file into a DataFrame

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

# Identify categorical columns

categorical_columns = df.select_dtypes(include=['object']).columns

# Perform one-hot encoding

df_encoded = pd.get_dummies(df, columns=categorical_columns)

# Print the encoded

print(df_encoded)
```

Output:

```
10190                  False                                  False
10191                  False                                  False
10192                  False                                  False
10193                  False                                   True

       Product Name_Wonka Bar - Milk Chocolate  \
0                                        False
1                                        False
2                                        False
3                                         True
4                                         True
...                                        ...
10189                                    False
10190                                    False
10191                                    False
10192                                    False
10193                                    False

       Product Name_Wonka Bar - Nutty Crunch Surprise  \
0                                               False
1                                               False
2                                               False
3                                               False
4                                               False
...                                               ...
10189                                           False
10190                                            True
10191                                           False
10192                                           False
10193                                           False

       Product Name_Wonka Bar - Triple Dazzle Caramel  \
0                                               True
1                                              False
2                                              False
3                                              False
4                                              False
...                                              ...
10189                                          False
10190                                          False
10191                                           True
10192                                           True
10193                                          False

       Product Name_Wonka Bar -Scrumdiddlyumptious  Product Name_Wonka Gum
0                                            False                  False
1                                             True                  False
2                                            False                  False
3                                            False                  False
4                                            False                  False
...                                            ...                    ...
10189                                         True                  False
10190                                        False                  False
10191                                        False                  False
10192                                        False                  False
10193                                        False                  False

[10194 rows x 12433 columns]
```

Ass: (Without Using Data frame)

```
# Program for demonstration of one hot encoding

# import libraries

import numpy as np

import pandas as pd

# import the data required

data = pd.read_csv('Candy_Sales.csv')

print(data.head())
```

Output:

```
    Row ID                     Order ID  Order Date   Ship Date  \
0     282  US-2021-128055-CHO-TRI-54000  2021-03-31  2026-09-26
1     288  US-2021-128055-CHO-SCR-58000  2021-03-31  2026-09-26
2    1132  US-2021-138100-CHO-FUD-51000  2021-09-15  2027-03-13
3    1133  US-2021-138100-CHO-MIL-31000  2021-09-15  2027-03-13
4    3396  US-2022-121391-CHO-MIL-31000  2022-10-04  2028-03-29

        Ship Mode  Customer ID Country/Region           City State/Province  \
0  Standard Class       128055  United States  San Francisco     California
1  Standard Class       128055  United States  San Francisco     California
2  Standard Class       138100  United States  New York City       New York
3  Standard Class       138100  United States  New York City       New York
4     First Class       121391  United States  San Francisco     California

  Postal Code   Division    Region     Product ID  \
0       94122  Chocolate   Pacific  CHO-TRI-54000
1       94122  Chocolate   Pacific  CHO-SCR-58000
2       10011  Chocolate  Atlantic  CHO-FUD-51000
3       10011  Chocolate  Atlantic  CHO-MIL-31000
4       94109  Chocolate   Pacific  CHO-MIL-31000

                       Product Name  Sales  Units  Gross Profit  Cost
0  Wonka Bar - Triple Dazzle Caramel   7.50      2          4.90  2.60
1     Wonka Bar -Scrumdiddlyumptious   7.20      2          5.00  2.20
2          Wonka Bar - Fudge Mallows   7.20      2          4.80  2.40
3         Wonka Bar - Milk Chocolate   9.75      3          6.33  3.42
4         Wonka Bar - Milk Chocolate   6.50      2          4.22  2.28
```