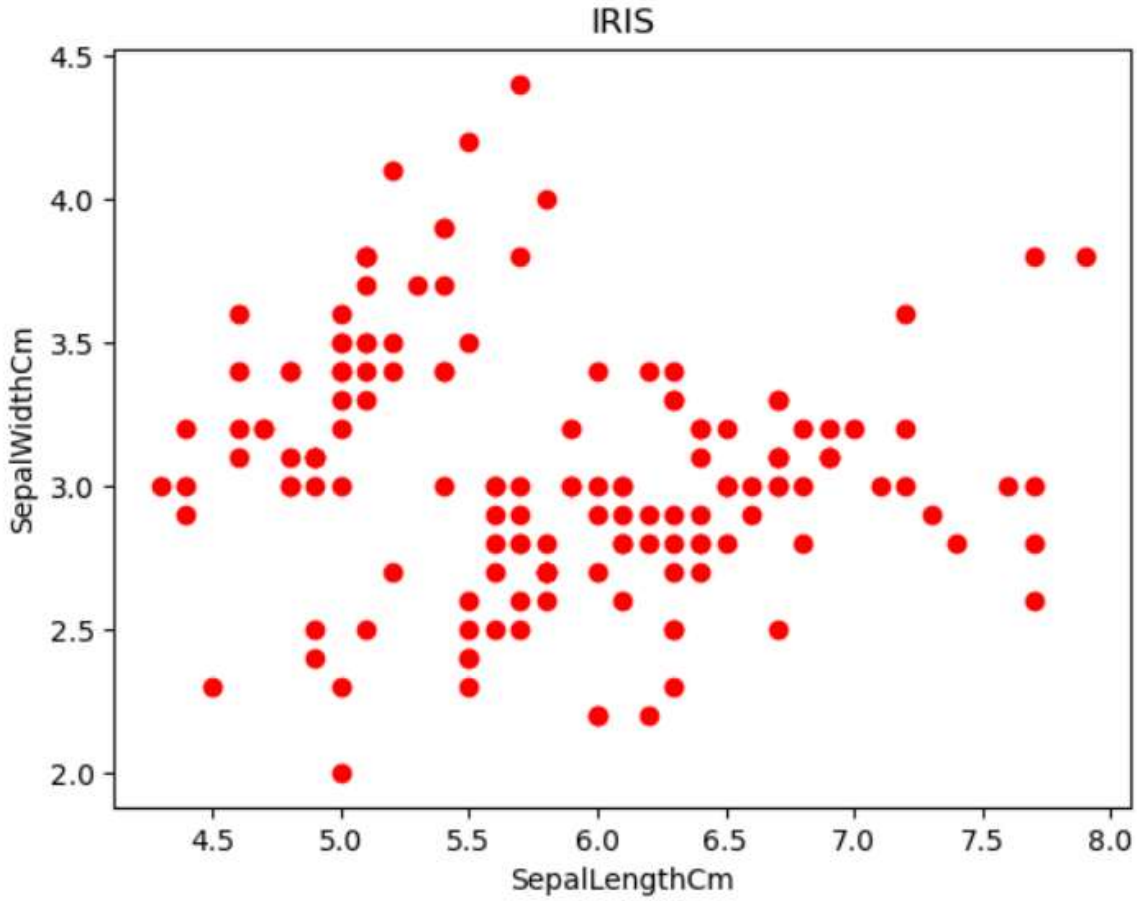
1. Write a python program to Prepare Scatter Plot (Use Forge Dataset / Iris Dataset).

```
import pandas as pd
import matplotlib.pyplot as plt

data=pd.read_csv("iris.csv")

x=data["SepalLengthCm"]
y=data["SepalWidthCm"]

plt.scatter(x,y,c="red")
plt.title("IRIS")
plt.xlabel("SepalLengthCm")
plt.ylabel("SepalWidthCm")
plt.show()
```

Output:



2. Write a python program to find all null values in a given data set and remove them.

```
import pandas as pd

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

print("Null values before removing:")
print(df.isnull().sum())
df_cleaned = df.dropna()

print("\nNull values after removing:")
print(df_cleaned.isnull().sum())
```

Output:

```
Null values before removing:
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64

Null values after removing:
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

3. Write a python program the Categorical values in numeric format for a given dataset.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

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

print("Original dataset:")
print(df.head())

label_encoder = LabelEncoder()

for column in df.select_dtypes(include=['object']):
    df[column] = label_encoder.fit_transform(df[column].astype(str))

print("\nDataset after label encoding:")
print(df.head())
```

Output:

```
Dataset after label encoding:
   PassengerId  Survived  Pclass  Name  Sex   Age  SibSp  Parch  Ticket  \
0            1         0       3   108    1  22.0      1      0     523
1            2         1       1   190    0  38.0      1      0     596
2            3         1       3   353    0  26.0      0      0     669
3            4         1       1   272    0  35.0      1      0      49
4            5         0       3    15    1  35.0      0      0     472

      Fare  Cabin  Embarked
0   7.2500    147         2
1  71.2833     81         0
2   7.9250    147         2
3  53.1000     55         2
4   8.0500    147         2
```

4. Write a python program to implement simple Linear Regression for predicting houseprice.

```
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_squared_error, r2_score

data = pd.read_csv('boston-housing-dataset.csv')

X = data.drop('MEDV', axis=1)
y = data['MEDV']

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)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean Squared Error:', mse)
print('R^2 Score:', r2)

plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual Prices vs Predicted Prices")
plt.show()
```

Output:

```
Mean Squared Error: 24.497819777630113
R^2 Score: 0.6659408703343074
```

5. Write a python program to implement multiple Linear Regression for a given dataset.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
df = pd.read_csv('Advertising.csv')
print("Dataset:")
print(df.head())
X = df[['TV', 'Radio', 'Newspaper']]
y = df['Sales']
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)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('\nMean Squared Error:', mse)
print('R^2 Score:', r2)
print('\nCoefficients:')
for i, feature in enumerate(['TV', 'radio', 'newspaper']):
    print(f'{feature}: {model.coef_[i]}')

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

Output:

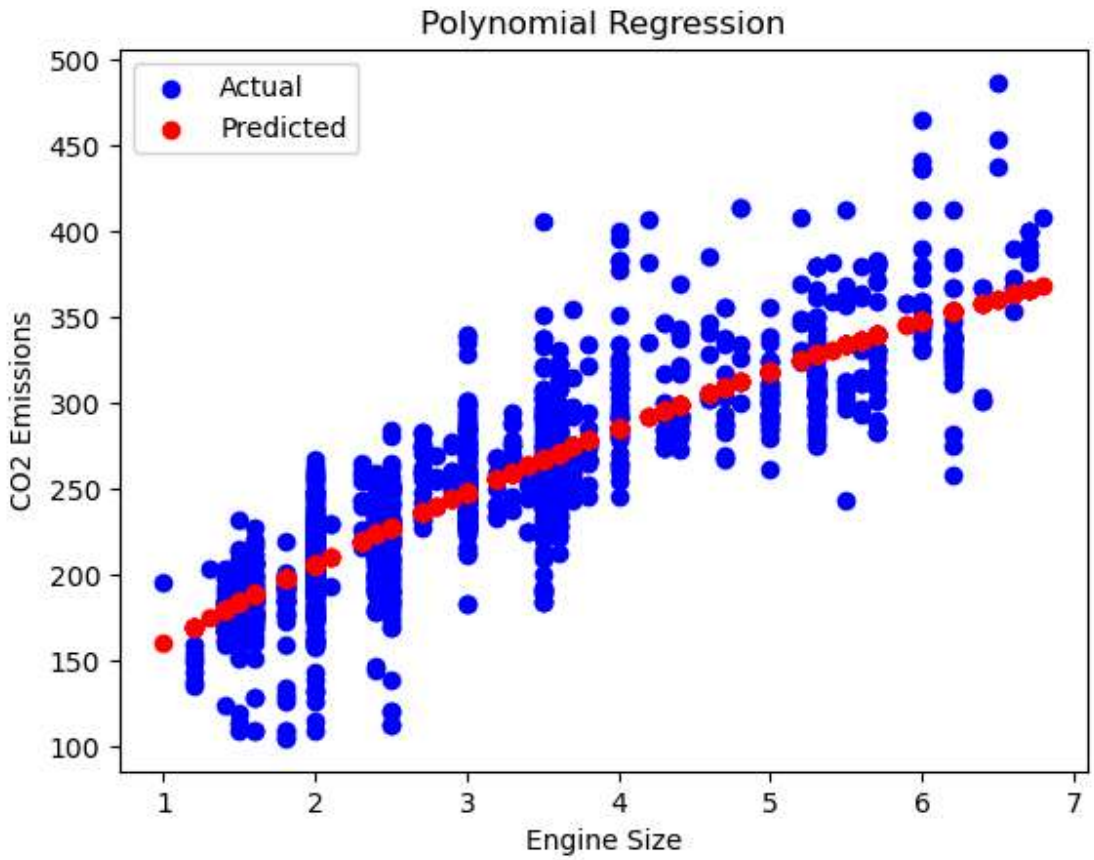
```
Dataset:
   Unnamed: 0     TV  Radio  Newspaper  Sales
0           1  230.1   37.8       69.2   22.1
1           2   44.5   39.3       45.1   10.4
2           3   17.2   45.9       69.3    9.3
3           4  151.5   41.3       58.5   18.5
4           5  180.8   10.8       58.4   12.9

Mean Squared Error: 3.1740973539761046
R^2 Score: 0.899438024100912

Coefficients:
TV: 0.04472951746871633
radio: 0.18919505423437658
newspaper: 0.0027611143413671796
Intercept: 2.9790673381226274
```

6. Write a python program to implement Polynomial Regression for given dataset.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
df = pd.read_csv('CO2_Emissions_Canada.csv')
print("Dataset:")
print(df.head())
X = df[["Engine Size(L)"]]
y = df["CO2 Emissions(g/km)"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
poly_degree = 2  # Degree of the polynomial
poly_features = PolynomialFeatures(degree=poly_degree)
X_train_poly = poly_features.fit_transform(X_train)
X_test_poly = poly_features.transform(X_test)
model = LinearRegression()
model.fit(X_train_poly, y_train)
y_train_pred = model.predict(X_train_poly)
y_test_pred = model.predict(X_test_poly)
train_rmse = np.sqrt(mean_squared_error(y_train, y_train_pred))
test_rmse = np.sqrt(mean_squared_error(y_test, y_test_pred))
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)
print('\nTraining RMSE:', train_rmse)
print('Testing RMSE:', test_rmse)
print('Training R^2 Score:', train_r2)
print('Testing R^2 Score:', test_r2)
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.scatter(X_test, y_test_pred, color='red', label='Predicted')
plt.title('Polynomial Regression')
plt.xlabel('Engine Size')
plt.ylabel('CO2 Emissions')
plt.legend()
plt.show()
```

Output:

Training RMSE: 30.377401378335648
Testing RMSE: 30.389153365664814
Training R^2 Score: 0.7300650161339413
Testing R^2 Score: 0.7315114310405249



7. Write a python program to Implement Naïve Bayes.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

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

print("Dataset:")
print(df.head())

X = df.drop('Species', axis=1)
y = df['Species']

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

model = GaussianNB()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print('\nAccuracy:', accuracy)

print('\nClassification Report:')
print(classification_report(y_test, y_pred))

print('\nConfusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

Output:

Dataset:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Accuracy: 1.0

Classification Report:

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	10
Iris-versicolor	1.00	1.00	1.00	9
Iris-virginica	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

Confusion Matrix:

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

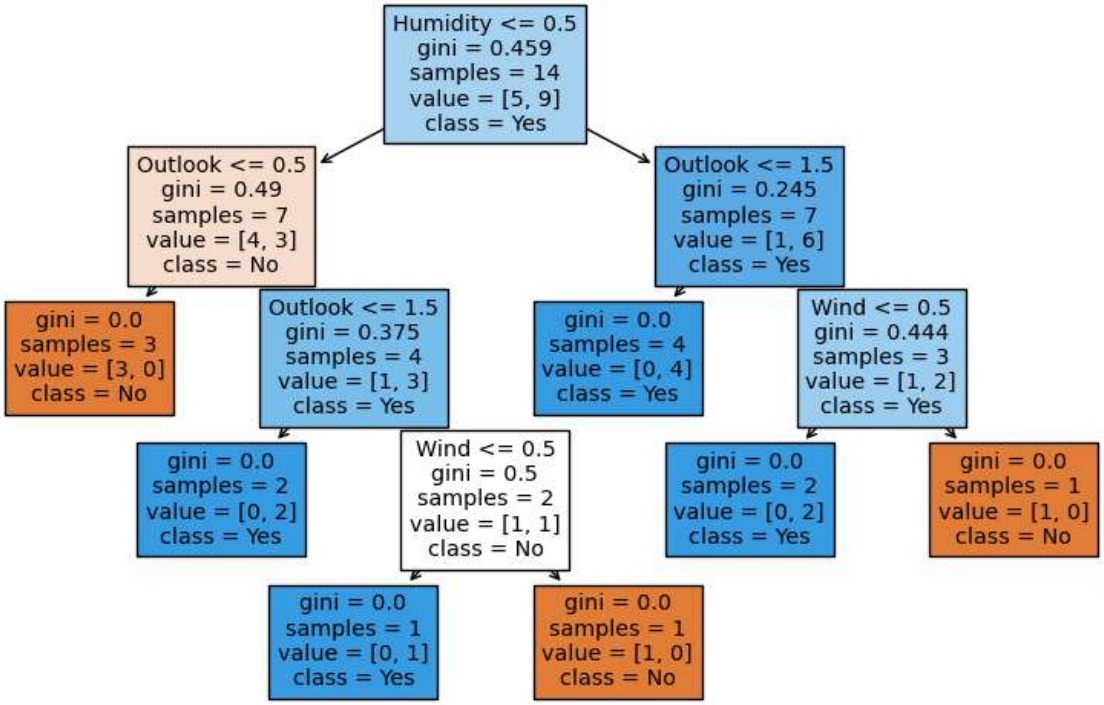
8. Write a python program to Implement Decision Tree whether or not to play tennis.

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

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

print("Dataset:")
print(df.head())

df['Outlook'] = df['Outlook'].map({'Sunny': 0, 'Overcast': 1, 'Rain': 2})
df['Temperature'] = df['Temperature'].map({'Hot': 0, 'Mild': 1, 'Cool': 2})
df['Humidity'] = df['Humidity'].map({'High': 0, 'Normal': 1})
df['Wind'] = df['Wind'].map({'Weak': 0, 'Strong': 1})
df['PlayTennis'] = df['PlayTennis'].map({'No': 0, 'Yes': 1})

x = df.drop('PlayTennis', axis=1)
y = df['PlayTennis']

clf = DecisionTreeClassifier()

clf.fit(x, y)

plt.figure(figsize=(10, 6))
plot_tree(clf, feature_names=['Outlook', 'Temperature', 'Humidity', 'Wind'], class_names=['No',
'Yes'], filled=True)
plt.show()
```

Output:

Dataset:

	Outlook	Temperature	Humidity	Wind	PlayTennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes



9. Write a python program to implement linear SVM.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

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

print("Dataset:")
print(df.head())

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 = SVC(kernel='linear')

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print('\nAccuracy:', accuracy)

print('\nClassification Report:')
print(classification_report(y_test, y_pred))

print('\nConfusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

Output:

Dataset:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	\
0	52	1	0	125	212	0	1	168	0	1.0	2	
1	53	1	0	140	203	1	0	155	1	3.1	0	
2	70	1	0	145	174	0	1	125	1	2.6	0	
3	61	1	0	148	203	0	1	161	0	0.0	2	
4	62	0	0	138	294	1	1	106	0	1.9	1	

	ca	thal	target
0	2	3	0
1	0	3	0
2	0	3	0
3	1	3	0
4	3	2	0

Accuracy: 0.8048780487804879

Classification Report:

	precision	recall	f1-score	support
0	0.88	0.71	0.78	102
1	0.76	0.90	0.82	103
accuracy			0.80	205
macro avg	0.82	0.80	0.80	205
weighted avg	0.82	0.80	0.80	205

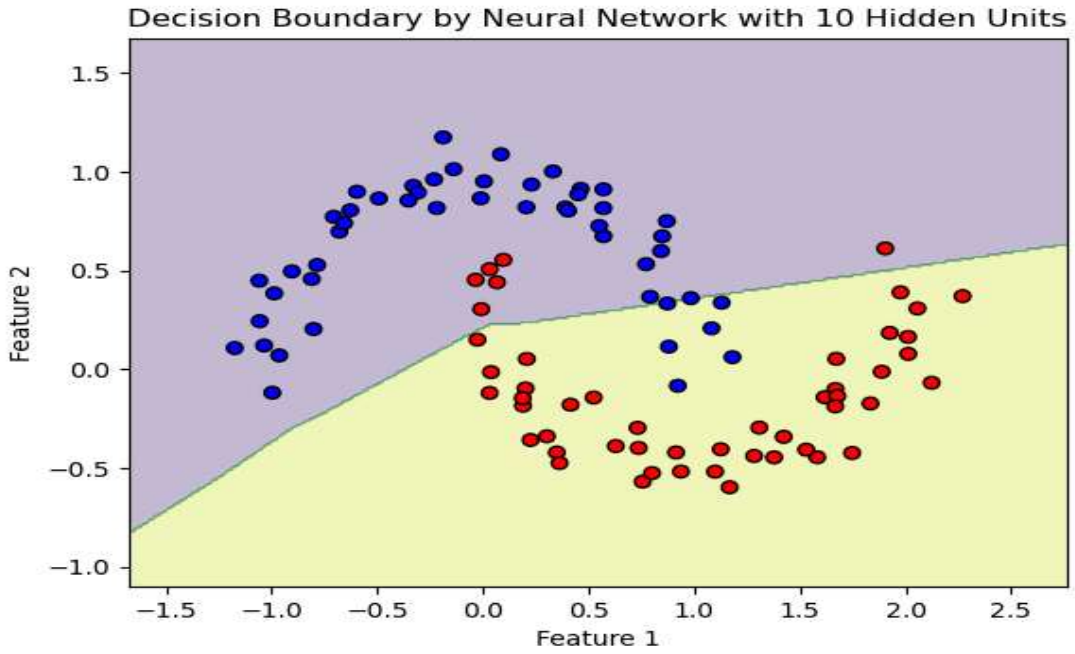
Confusion Matrix:

```
[[72 30]
 [10 93]]
```

10. Write a python program to find Decision boundary by using a neural network with 10 hidden units on two moons dataset.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
from sklearn.neural_network import MLPClassifier
X, y = make_moons(n_samples=100, noise=0.1, random_state=42)
model = MLPClassifier(hidden_layer_sizes=(10,), activation='relu', solver='adam',
max_iter=1000, random_state=42)
model.fit(X, y)
x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max, 0.01))
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.3)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.bwr, edgecolors='k')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Decision Boundary by Neural Network with 10 Hidden Units')
plt.show()
```

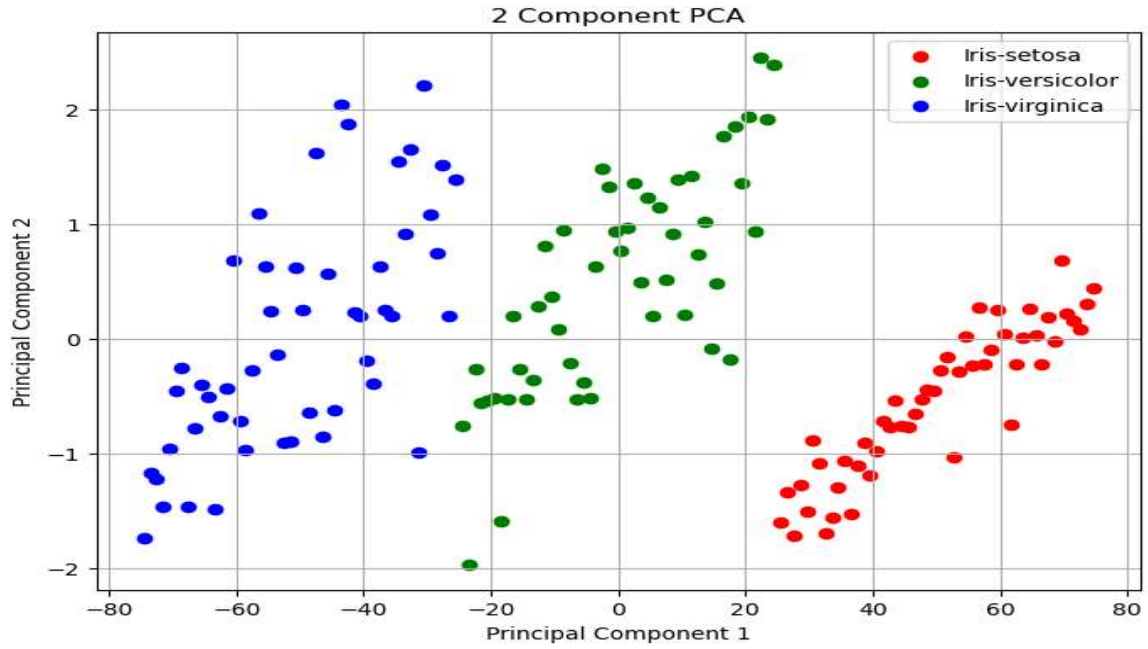
Output:



11. Write a python program to transform data with Principal Component Analysis (PCA).

```
import pandas as pd
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
data = pd.read_csv("Iris.csv")
X = data.drop('Species', axis=1)
y = data['Species']
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
plt.figure(figsize=(8, 6))
targets = data['Species'].unique()
colors = ['r', 'g', 'b']
for target, color in zip(targets, colors):
    indicesToKeep = y == target
    plt.scatter(X_pca[indicesToKeep, 0], X_pca[indicesToKeep, 1], c=color, label=target)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('2 Component PCA')
plt.legend(targets)
plt.grid()
plt.show()
```

Output:

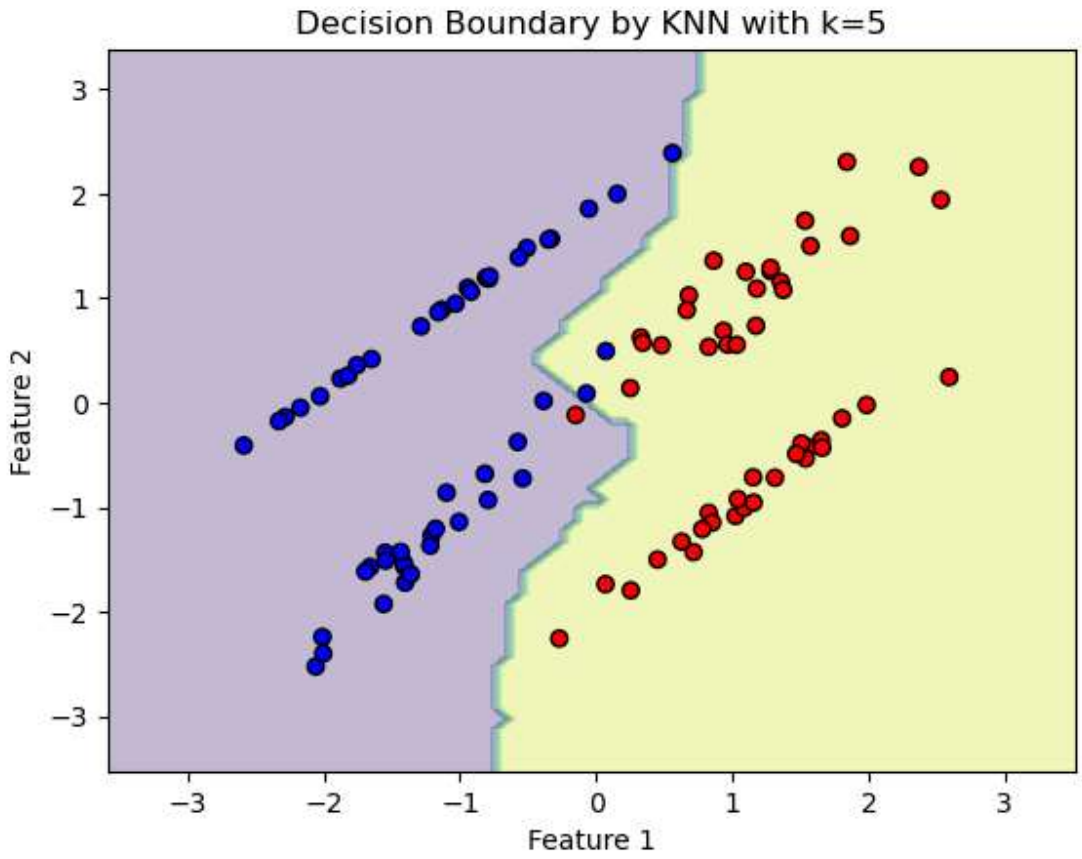


12. Write a python program to implement k-nearest Neighbors ML algorithm to build prediction model (Use Forge Dataset) .

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

X, y = make_classification(n_samples=100, n_features=2, n_informative=2, n_redundant=0,
random_state=42)

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

knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

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, 0.1), np.arange(y_min, y_max, 0.1))
Z = knn.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.3)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.bwr, edgecolors='k')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Decision Boundary by KNN with k=5')
plt.show()
```

Output:

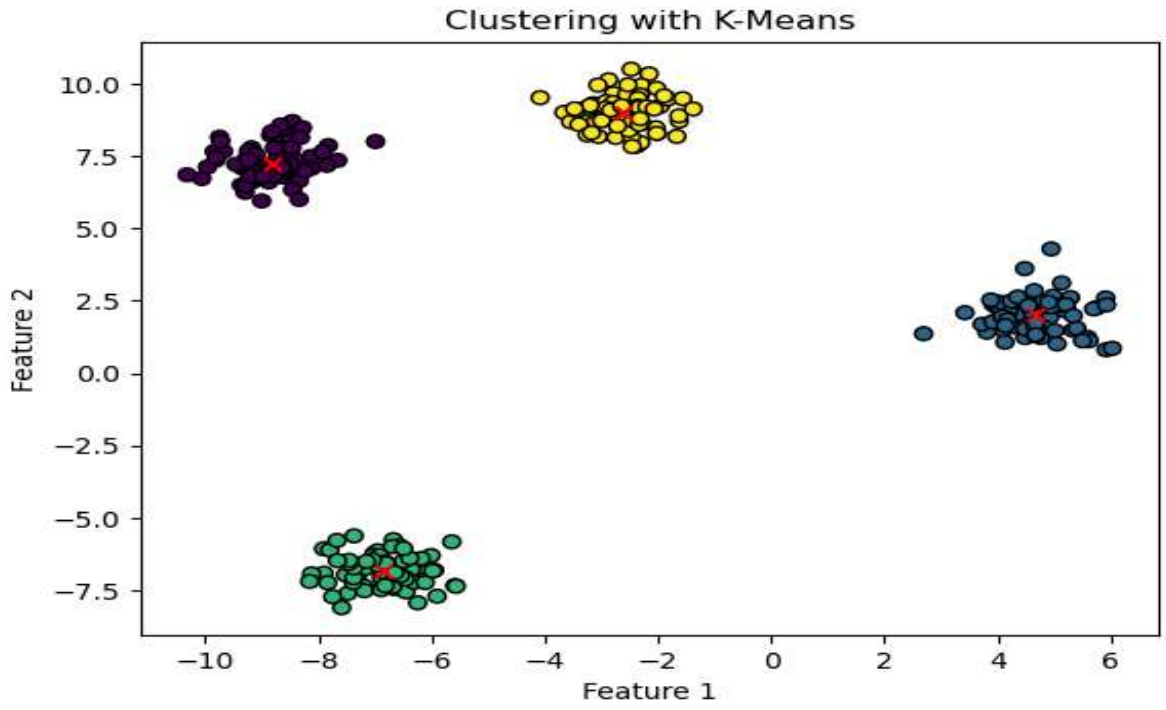
Accuracy: 1.0



13. Write a python program to implement k-means algorithm on a synthetic dataset.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans

data, labels = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=42)

kmeans = KMeans(n_clusters=4)

kmeans.fit(data)

centers = kmeans.cluster_centers_
predicted_labels = kmeans.labels_

plt.scatter(data[:, 0], data[:, 1], c=predicted_labels, cmap='viridis', edgecolors='k')
plt.scatter(centers[:, 0], centers[:, 1], c='red', marker='x')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Clustering with K-Means')
plt.show()
```

Output:



14. Write a python program to implement Agglomerative clustering on a synthetic dataset.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import AgglomerativeClustering

data, true_labels = make_blobs(n_samples=300, centers=4, cluster_std=0.60,
random_state=42)

agglomerative = AgglomerativeClustering(n_clusters=4)

agglomerative.fit(data)

predicted_labels = agglomerative.labels_

plt.scatter(data[:, 0], data[:, 1], c=predicted_labels, cmap='viridis', edgecolors='k')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Agglomerative Clustering')
plt.show()
```

Output: