Experiment 7:

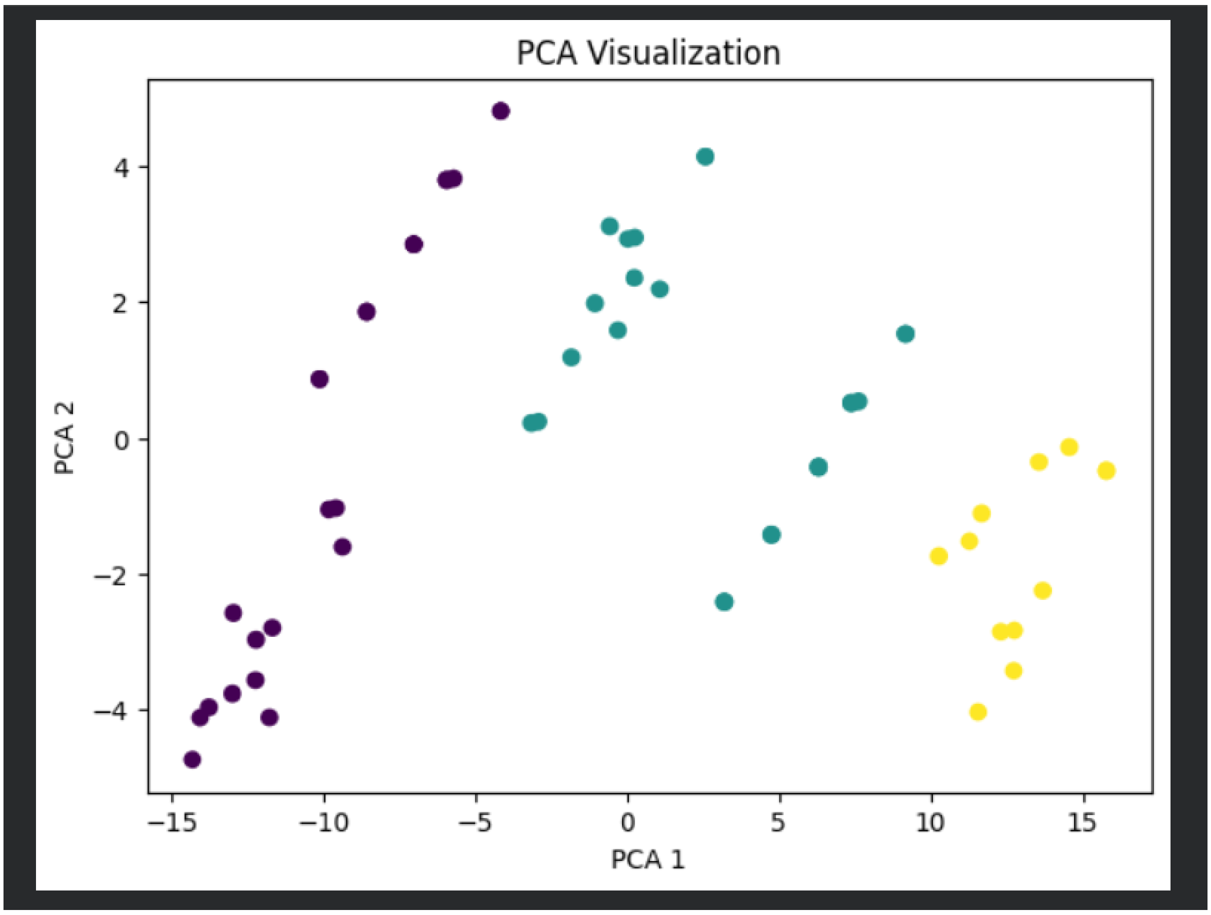
Creating a dataset:

```
import pandas as pd

import numpy as np


data = {

   "feat1": [5,6,4,5,7,8,5,6,4,5,7,8, 10,11,9,12,13,11,10,12,9,13,12,11,

            20,21,19,18,22,23,19,20,21,22,18,23, 6,7,5,8,9,7,6,8,5,9,7,8,

            14,15,13,16,17,15,14,16,13,17,15,16],


   "feat2": [10,11,12,9,8,12,11,10,9,8,13,12, 20,22,19,21,23,20,22,21,19,23,22,21,

            30,29,28,31,32,30,29,31,28,32,30,29, 15,16,14,17,18,16,15,17,14,18,16,17,

            25,26,24,27,28,26,25,27,24,28,26,27],


   "feat3": [2,3,1,2,4,5,2,3,1,2,4,5, 6,7,5,8,9,7,6,8,5,9,7,8,

            1,2,3,1,4,5,3,2,1,4,3,5, 7,8,6,9,10,8,7,9,6,10,8,9,

            3,4,2,5,6,4,3,5,2,6,4,5],


   "feat4": [1,1,2,1,3,2,1,2,1,3,2,1, 4,4,5,4,6,5,4,5,4,6,5,4,

            7,8,6,7,9,8,7,8,6,9,8,7, 2,3,1,2,4,3,2,3,1,4,3,2,

            5,6,4,5,7,6,5,6,4,7,6,5],

}


labels = [0]*12 + [1]*12 + [2]*12 + [0]*12 + [1]*12  # 60 labels total


df = pd.DataFrame(data)

df["label"] = labels


X = df[["feat1", "feat2", "feat3", "feat4"]]

y = df["label"]
```

```
df.head()
```

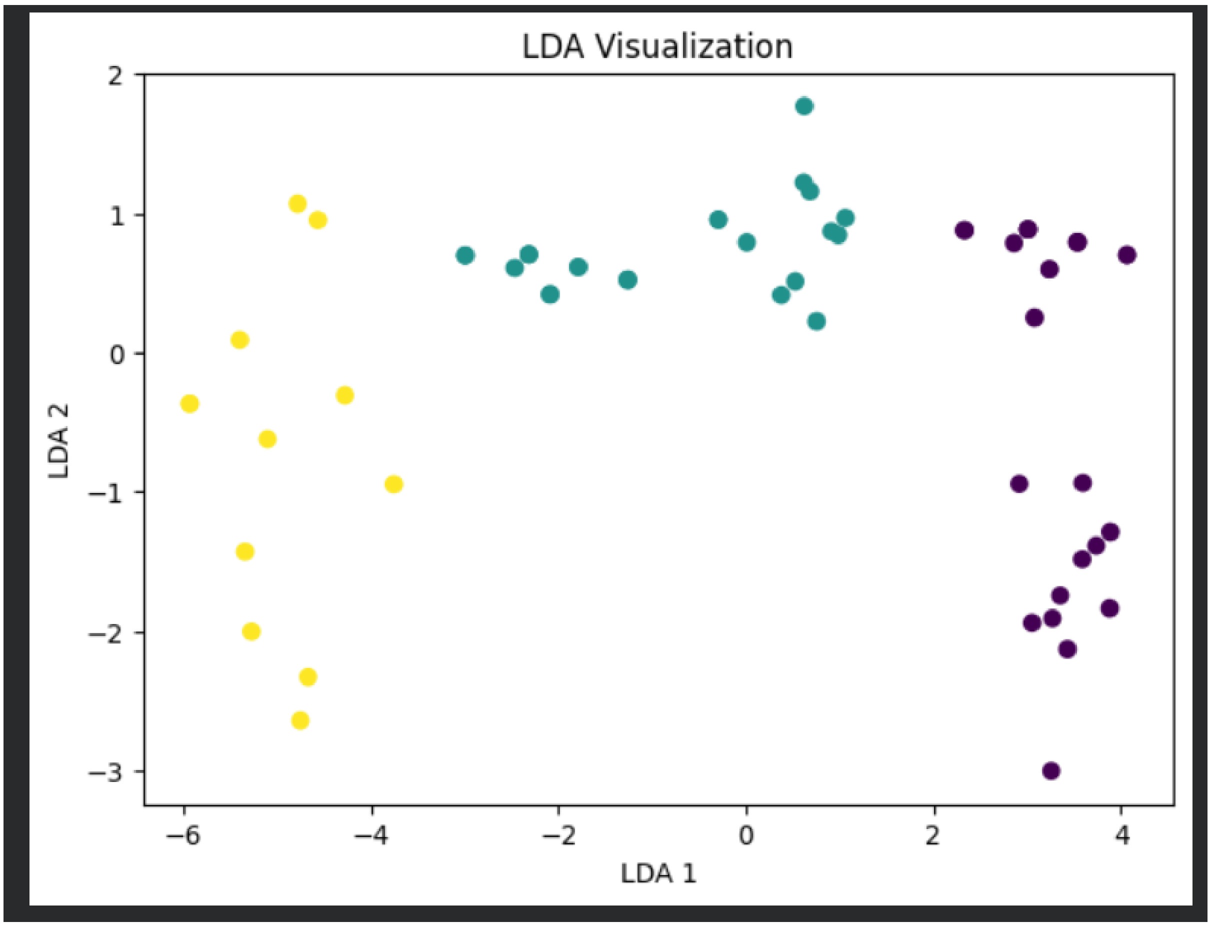
PCA:

```
from sklearn.decomposition import PCA

import matplotlib.pyplot as plt


pca = PCA(n_components=2)

X_pca = pca.fit_transform(X)


plt.figure(figsize=(7,5))

plt.scatter(X_pca[:,0], X_pca[:,1], c=y)

plt.title("PCA Visualization")

plt.xlabel("PCA 1")

plt.ylabel("PCA 2")

plt.show()
```



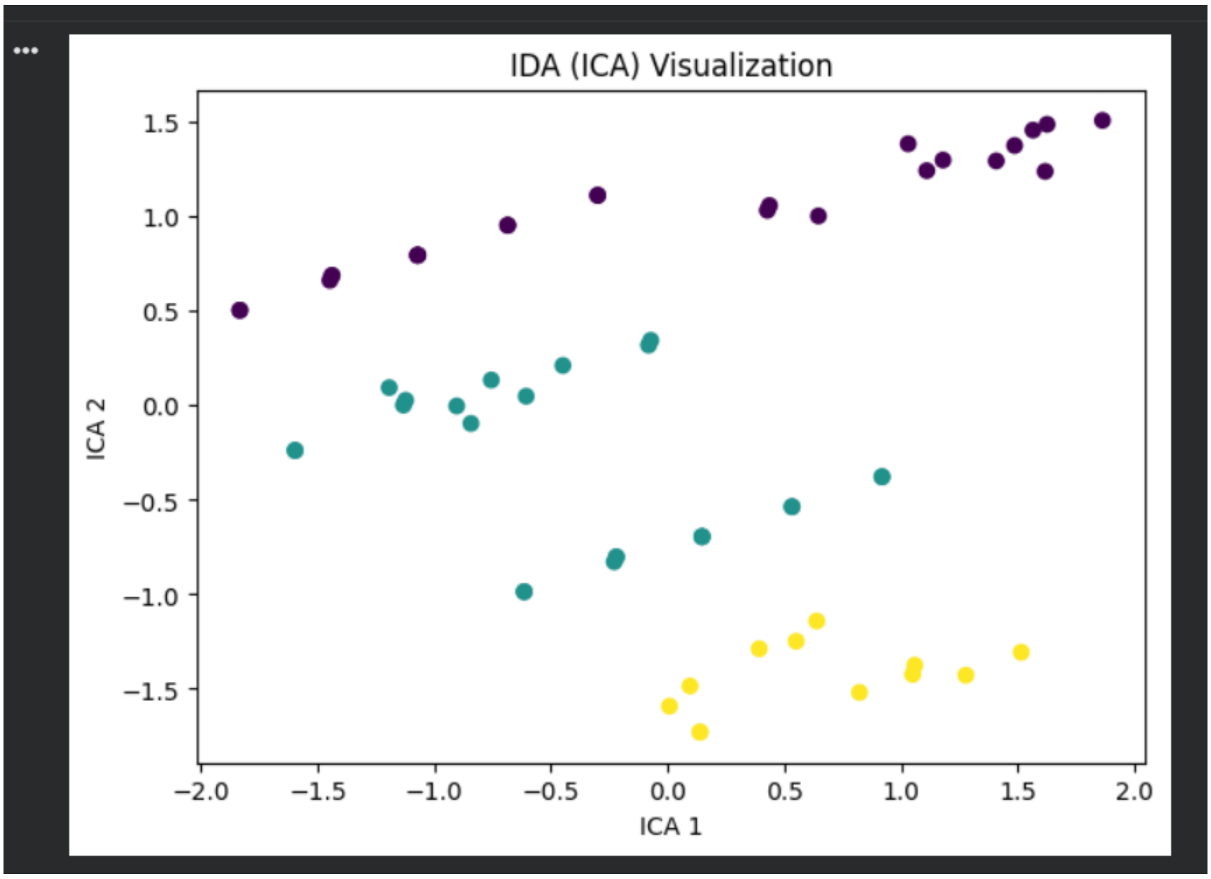
LDA:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis


lda = LinearDiscriminantAnalysis(n_components=2)

X_lda = lda.fit_transform(X, y)


plt.figure(figsize=(7,5))

plt.scatter(X_lda[:,0], X_lda[:,1], c=y)

plt.title("LDA Visualization")

plt.xlabel("LDA 1")

plt.ylabel("LDA 2")

plt.show()
```



ICA:

```
from sklearn.decomposition import FastICA


ica = FastICA(n_components=2, max_iter=500)

X_ica = ica.fit_transform(X)


plt.figure(figsize=(7,5))

plt.scatter(X_ica[:,0], X_ica[:,1], c=y)

plt.title("IDA (ICA) Visualization")

plt.xlabel("ICA 1")

plt.ylabel("ICA 2")

plt.show()
```



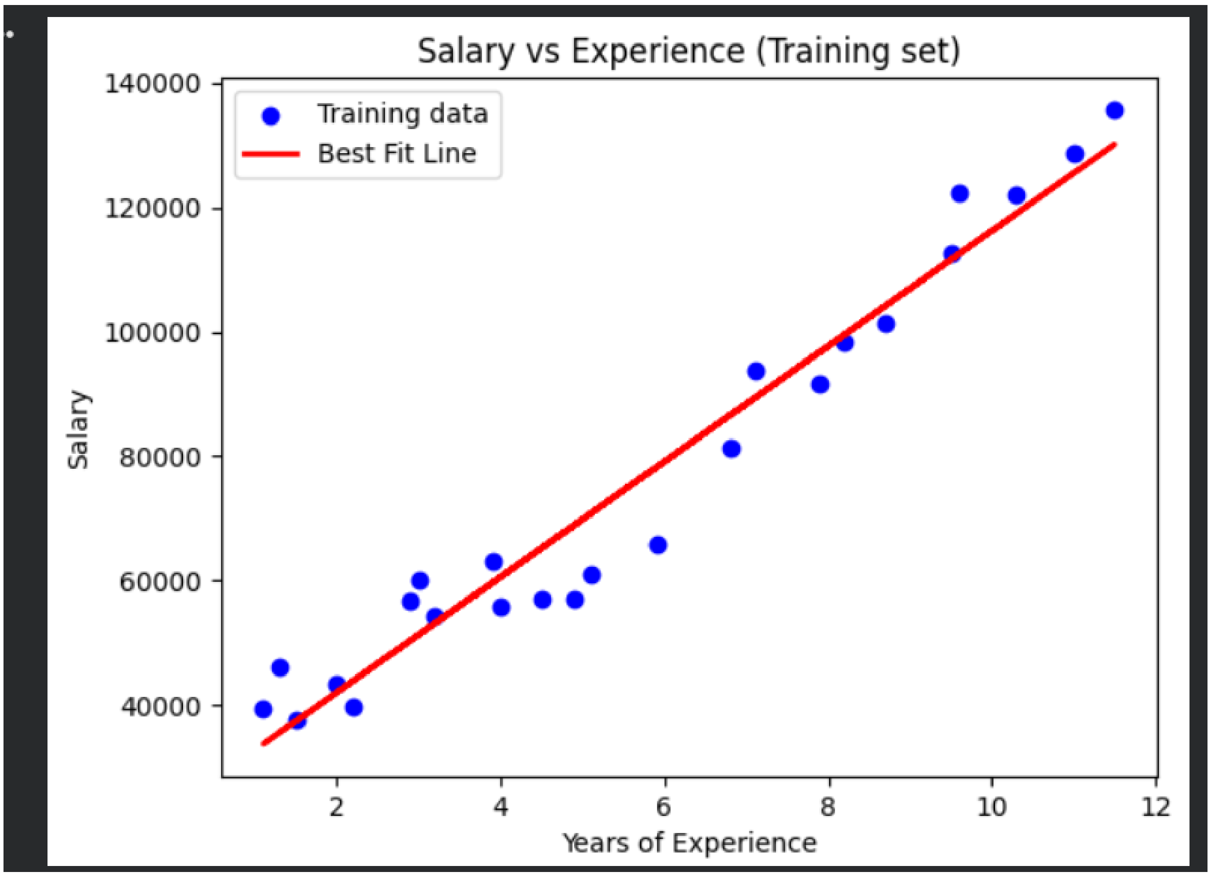
Experiment 8:

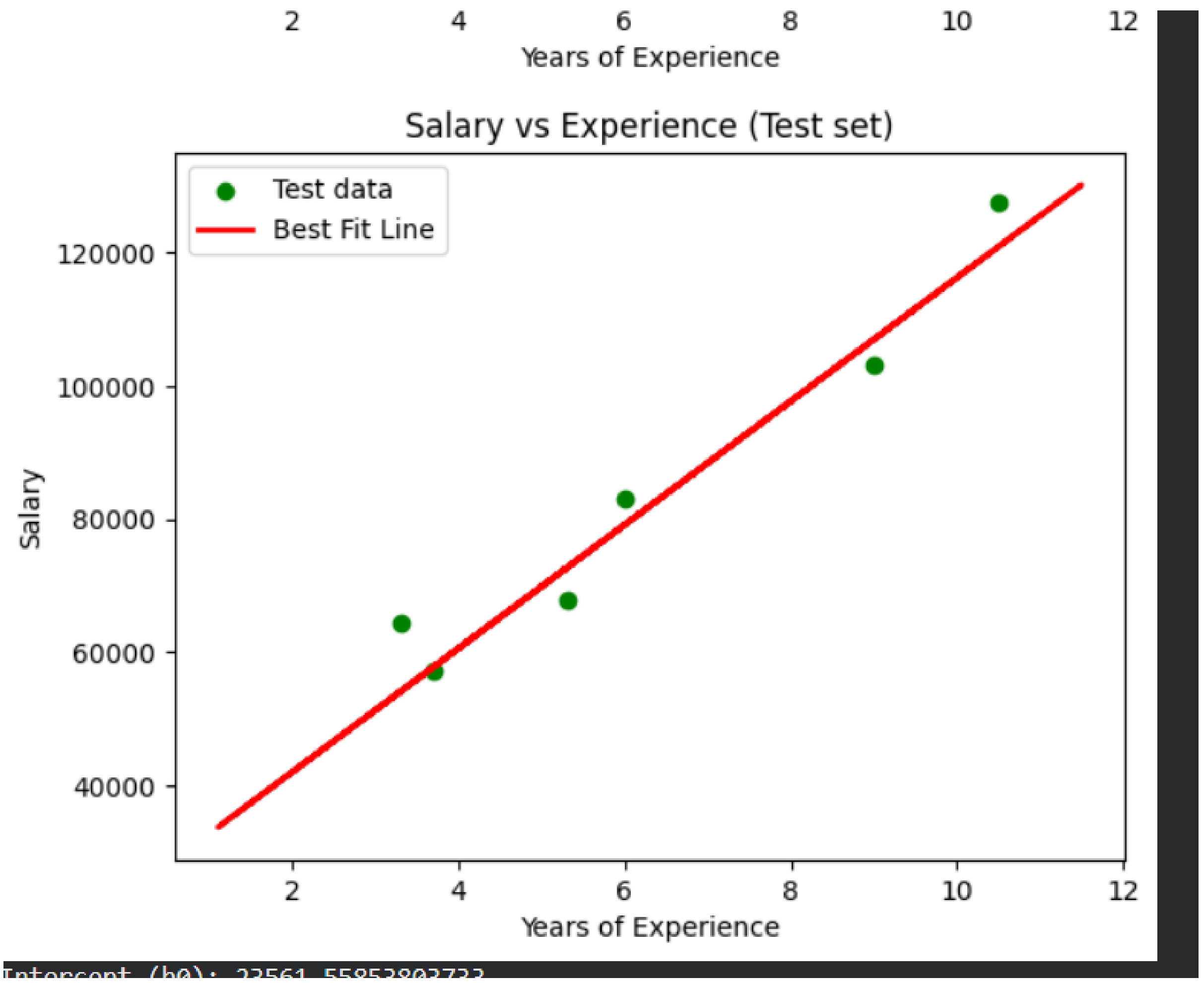
Simple Linear Regression implementation:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split


data = {

   'Experience': [1.1, 1.3, 1.5, 2.0, 2.2, 2.9, 3.0, 3.2, 3.3, 3.7,

                  3.9, 4.0, 4.5, 4.9, 5.1, 5.3, 5.9, 6.0, 6.8, 7.1,

                  7.9, 8.2, 8.7, 9.0, 9.5, 9.6, 10.3, 10.5, 11.0, 11.5],

   'Salary': [39343, 46205, 37731, 43525, 39891, 56642, 60150, 54445,

              64445, 57189, 63218, 55794, 56957, 57081, 61111, 67938,

              66029, 83088, 81363, 93940, 91738, 98273, 101302, 103150,

              112636, 122391, 121872, 127345, 128765, 135675]

}


df = pd.DataFrame(data)


X = df[['Experience']]

y = df['Salary']


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)
```

```
plt.scatter(X_train, y_train, color='blue', label='Training data')

plt.plot(X_train, model.predict(X_train), color='red', linewidth=2, label='Best Fit Line')

plt.title('Salary vs Experience (Training set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.legend()

plt.show()


plt.scatter(X_test, y_test, color='green', label='Test data')

plt.plot(X_train, model.predict(X_train), color='red', linewidth=2, label='Best Fit Line')

plt.title('Salary vs Experience (Test set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.legend()

plt.show()


print("Intercept (b0):", model.intercept_)

print("Coefficient (b1):", model.coef_[0])

print("\nEquation of line: Salary = {:.2f} + {:.2f} * Experience".format(model.intercept_,
model.coef_[0]))


exp = 5

predicted_salary = model.predict([[exp]])
```

```
print(f"\nPredicted Salary for {exp} years of experience: ₹{predicted_salary[0]:.2f}")
```



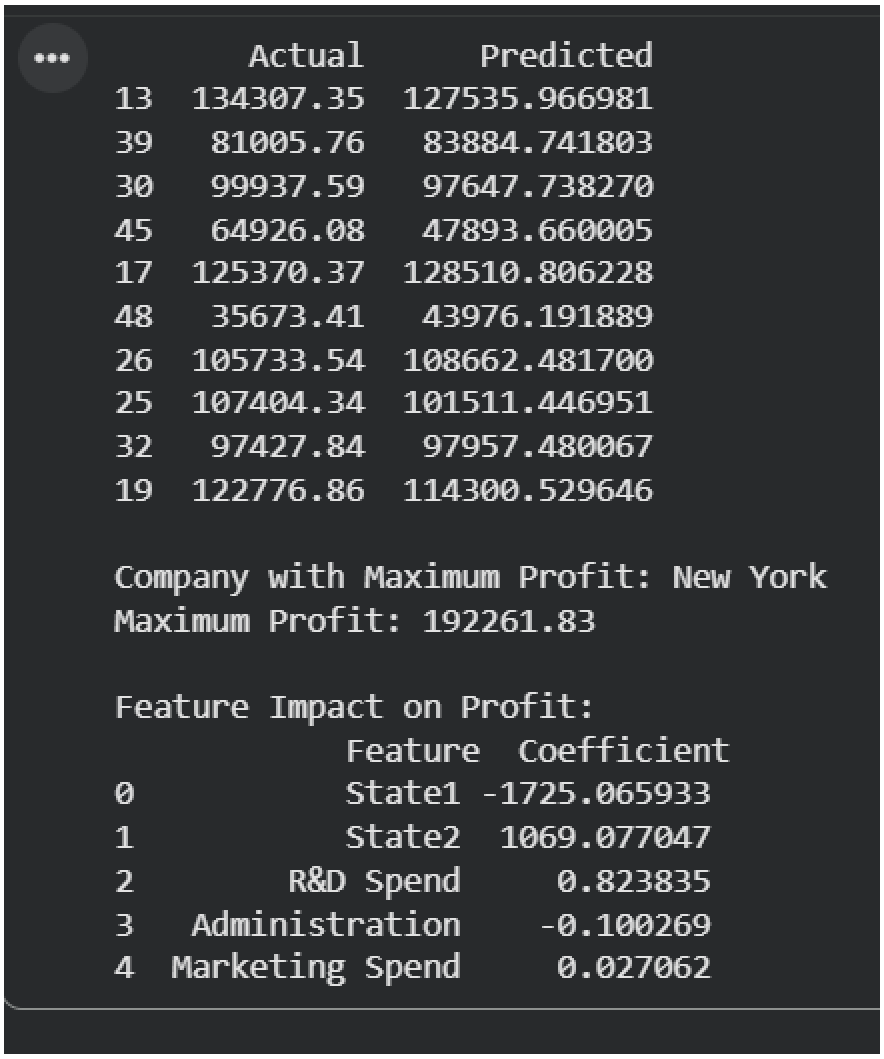


```
Intercept (b0): 23561.55853803733
```

Multiple Linear regression implementation:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer


data = {

    'R&D Spend': [165349.2, 162597.7, 153441.51, 144372.41, 142107.34, 131876.9, 134615.46,
130298.13, 120542.52, 123334.88,

                  101913.08, 100671.96, 93863.75, 91992.39, 119943.24, 114523.61, 78013.11, 94657.16,
91749.16, 86419.7,

                  76253.86, 78389.47, 73994.56, 67532.53, 77044.01, 64664.71, 75328.87, 72107.6,
66051.52, 65605.48,

                  61994.48, 61136.38, 63408.86, 55493.95, 46426.07, 46014.02, 28663.76, 44069.95,
20229.59, 38558.51,

                  28754.33, 27892.92, 23640.93, 15505.73, 22177.74, 1000.23, 1315.46, 0.0, 542.05, 0.0],

    'Administration': [136897.8, 151377.59, 101145.55, 118671.85, 91391.77, 99814.71, 147198.87,
145530.06, 148718.95, 108679.17,

                       110594.11, 91790.61, 127320.38, 135495.07, 156547.42, 122616.84, 121597.55,
145077.58, 114175.79, 153514.11,

                       113867.3, 153773.43, 122782.75, 105751.03, 99281.34, 139553.16, 144135.98,
127864.55, 182645.56, 153032.06,

                       115641.28, 152701.92, 129219.61, 103057.49, 157693.92, 85047.44, 127056.21,
51283.14, 65947.93, 82982.09,

                       118546.05, 84710.77, 96189.63, 127382.3, 154806.14, 124153.04, 115816.21,
135426.92, 154699.78, 127382.3],

    'Marketing Spend': [471784.1, 443898.53, 407934.54, 383199.62, 366168.42, 362861.36,
127716.82, 323876.68, 311613.29, 304981.62,

                        229160.95, 249744.55, 249839.44, 252664.93, 256512.92, 261776.23, 264346.06,
282574.31, 294919.57, 0.0,

                        298664.47, 299737.29, 303319.26, 304768.73, 140574.81, 137962.62, 134050.07,
353183.81, 118148.2, 107138.38,

                        91131.24, 88218.23, 46085.25, 214634.81, 210797.67, 205517.64, 201126.82,
197029.42, 185265.1, 174999.3,
                    172795.67, 164470.71, 157293.19, 155622.91, 152701.92, 140964.55, 134361.69,
130298.13, 123334.88, 120542.52],

    'State': ['New York', 'California', 'Florida', 'New York', 'Florida', 'California', 'Florida', 'New York',
'California', 'Florida',

            'New York', 'California', 'Florida', 'New York', 'California', 'Florida', 'New York', 'California',
'Florida', 'New York',

            'California', 'Florida', 'New York', 'California', 'Florida', 'New York', 'California', 'Florida', 'New
York', 'California',

            'Florida', 'New York', 'California', 'Florida', 'New York', 'California', 'Florida', 'New York',
'California', 'Florida',

            'New York', 'California', 'Florida', 'New York', 'California', 'Florida', 'New York', 'California',
'Florida', 'New York'],

    'Profit': [192261.83, 191792.06, 191050.39, 182901.99, 166187.94, 156991.12, 156122.51,
155752.6, 152211.77, 149759.96,

             146121.95, 144259.4, 141585.52, 134307.35, 132602.65, 129917.04, 126992.93, 125370.37,
124266.9, 122776.86,

             118474.03, 111313.02, 110352.25, 108733.99, 108552.04, 107404.34, 105733.54,
105008.31, 103282.38, 101004.64,

             99937.59, 97483.56, 97427.84, 96778.92, 96712.8, 96479.51, 90708.19, 89949.14,
81229.06, 81005.76,

             78239.91, 77798.83, 71498.49, 69758.98, 65200.33, 64926.08, 49490.75, 42559.73,
35673.41, 14681.4]

}


df = pd.DataFrame(data)


X = df.iloc[:, :-1]

y = df.iloc[:, -1]


ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(drop='first'), [3])],
remainder='passthrough')

X = np.array(ct.fit_transform(X))


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)


results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

print(results)

print("\nCompany with Maximum Profit:", df.loc[df['Profit'].idxmax(), 'State'])

print("Maximum Profit:", df['Profit'].max())


coef_df = pd.DataFrame({'Feature': ['State1','State2','R&D Spend','Administration','Marketing Spend'], 'Coefficient': model.coef_})

print("\nFeature Impact on Profit:\n", coef_df)
```



```
       Actual      Predicted
13  134307.35  127535.966981
39   81005.76   83884.741803
30   99937.59   97647.738270
45   64926.08   47893.660005
17  125370.37  128510.806228
48   35673.41   43976.191889
26  105733.54  108662.481700
25  107404.34  101511.446951
32   97427.84   97957.480067
19  122776.86  114300.529646

Company with Maximum Profit: New York
Maximum Profit: 192261.83

Feature Impact on Profit:
            Feature  Coefficient
0           State1 -1725.065933
1           State2  1069.077047
2        R&D Spend     0.823835
3   Administration    -0.100269
4  Marketing Spend     0.027062
```

Experiment 9:

Simple decision tree:

Importing the dataset:

```
import pandas as pd


data_set = pd.read_csv('/content/drive/MyDrive/Datasets/user_data.csv')

print(data_set)
```

Decision Tree algorithm for data related to salary:

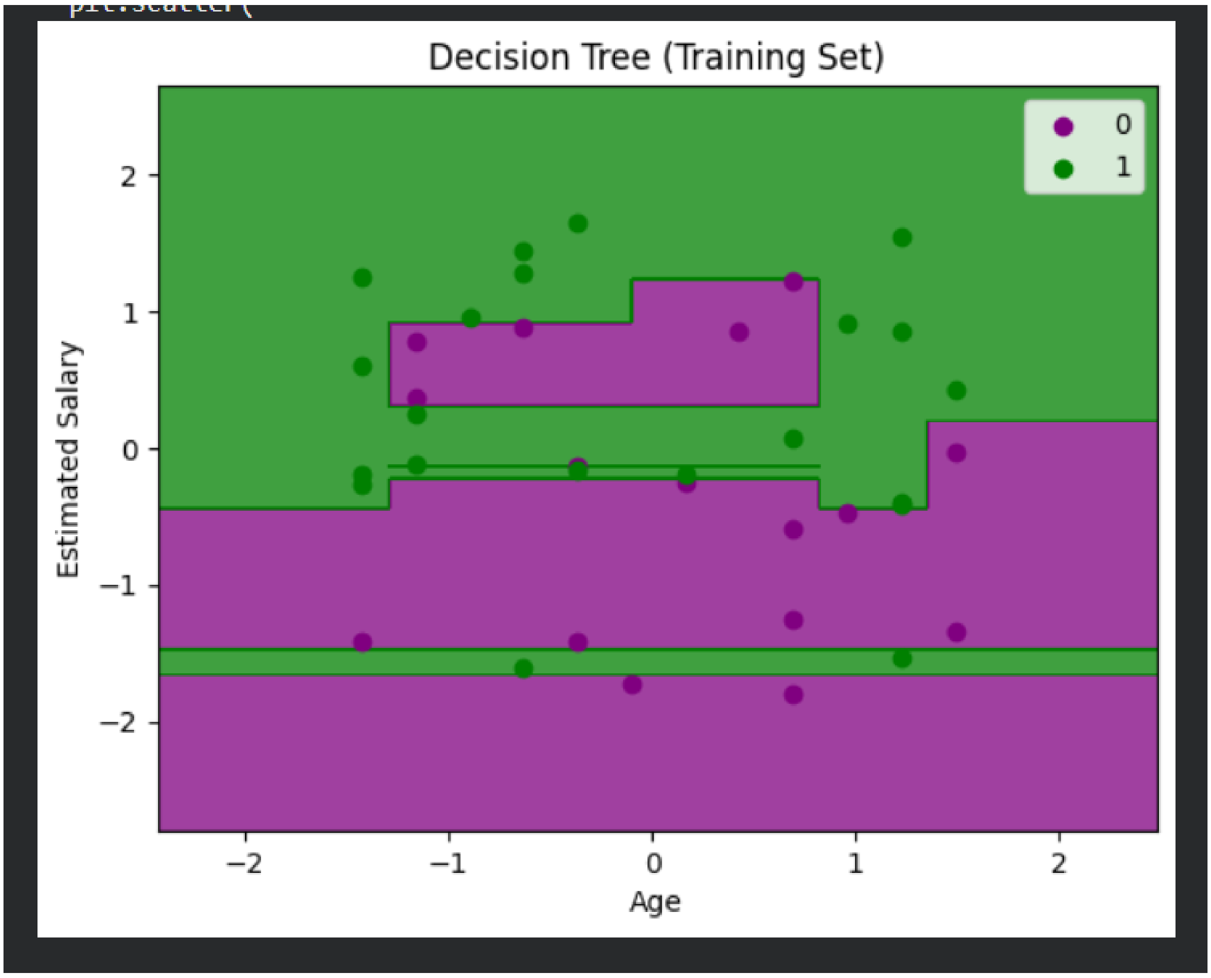
```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd


data_set = pd.read_csv('/content/drive/MyDrive/Datasets/user_data.csv')


x = data_set.iloc[:, [3, 4]].values

y = data_set.iloc[:, 5].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.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(criterion='entropy', random_state=0)

classifier.fit(x_train, y_train)
```

Prediction and confusion matrix:

```
y_pred = classifier.predict(x_test)

print("Predicted:\n", y_pred)

print("\nActual:\n", y_test)


from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

print("\nConfusion Matrix:\n", cm)
```

Training Set visualization:

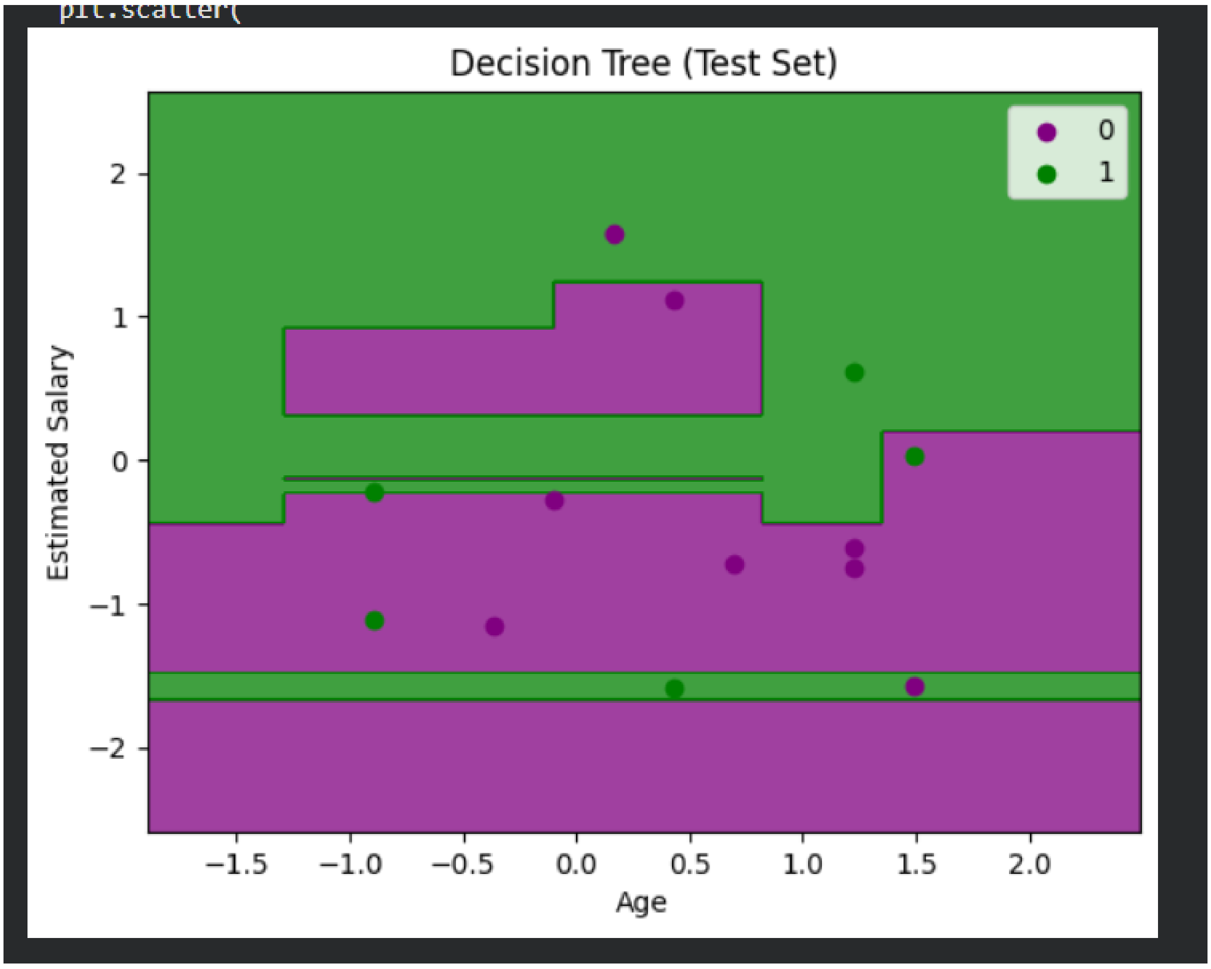
```
from matplotlib.colors import ListedColormap


x_set, y_set = x_train, y_train

x1, x2 = np.meshgrid(

    np.arange(x_set[:, 0].min() - 1, x_set[:, 0].max() + 1, 0.01),

    np.arange(x_set[:, 1].min() - 1, x_set[:, 1].max() + 1, 0.01)

)


plt.contourf(

    x1, x2,

    classifier.predict(np.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),

    alpha=0.75,

    cmap=ListedColormap(('purple', 'green'))

)


plt.xlim(x1.min(), x1.max())

plt.ylim(x2.min(), x2.max())


for i, j in enumerate(np.unique(y_set)):
    plt.scatter(
        x_set[y_set == j, 0],
        x_set[y_set == j, 1],
        c=ListedColormap(('purple', 'green'))(i),
        label=j
    )

plt.title('Decision Tree (Training Set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



Testing set visualization:

```
from matplotlib.colors import ListedColormap


x_set, y_set = x_test, y_test

x1, x2 = np.meshgrid(

    np.arange(x_set[:, 0].min() - 1, x_set[:, 0].max() + 1, 0.01),

    np.arange(x_set[:, 1].min() - 1, x_set[:, 1].max() + 1, 0.01)

)


plt.contourf(

    x1, x2,

    classifier.predict(np.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),

    alpha=0.75,

    cmap=ListedColormap(('purple', 'green'))

)


plt.xlim(x1.min(), x1.max())

plt.ylim(x2.min(), x2.max())


for i, j in enumerate(np.unique(y_set)):

    plt.scatter(

        x_set[y_set == j, 0],

        x_set[y_set == j, 1],

        c=ListedColormap(('purple', 'green'))(i),

        label=j

    )


plt.title('Decision Tree (Test Set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()
```

plt.show()


```
plt.scatter(
```

Experiment 10:

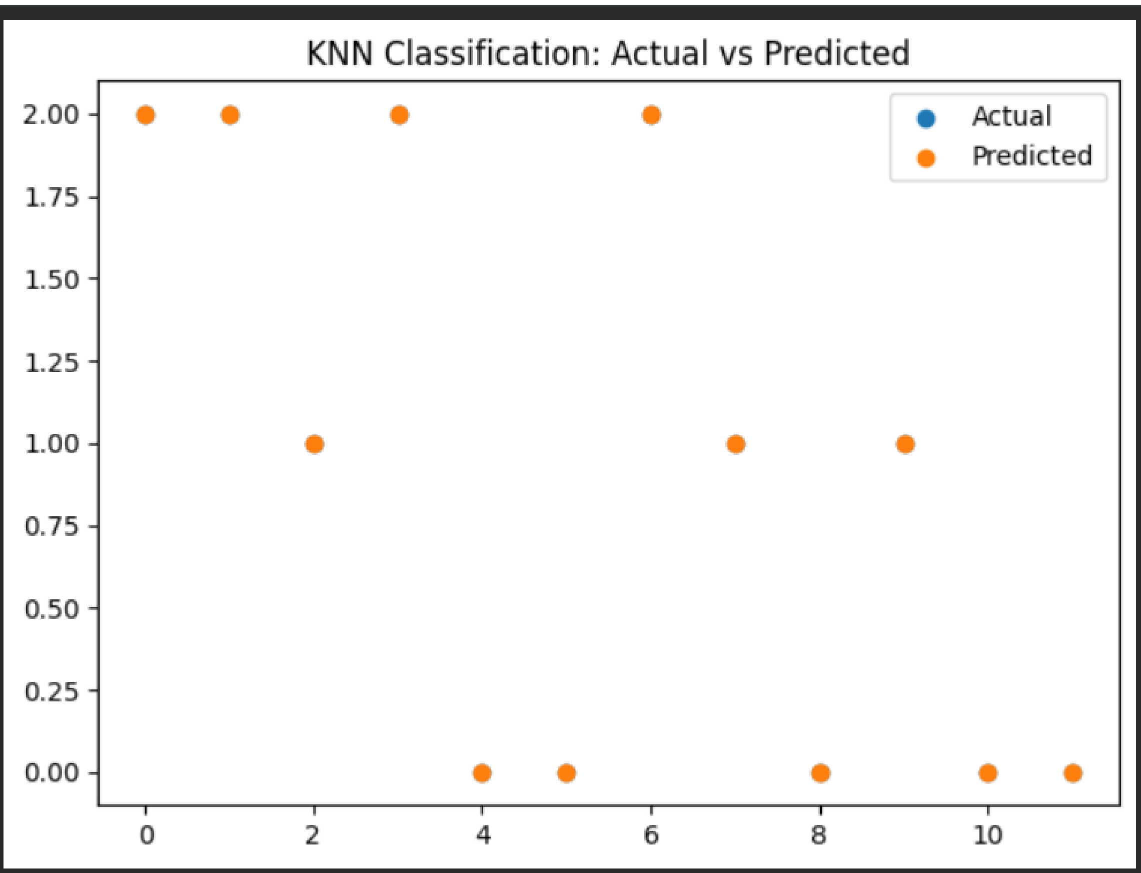
Creating dataset:

```
import pandas as pd

import numpy as np


data = {
    "feat1": [5,6,4,5,7,8,5,6,4,5,7,8, 10,11,9,12,13,11,10,12,9,13,12,11,
              20,21,19,18,22,23,19,20,21,22,18,23, 6,7,5,8,9,7,6,8,5,9,7,8,
              14,15,13,16,17,15,14,16,13,17,15,16],

    "feat2": [10,11,12,9,8,12,11,10,9,8,13,12, 20,22,19,21,23,20,22,21,19,23,22,21,
              30,29,28,31,32,30,29,31,28,32,30,29, 15,16,14,17,18,16,15,17,14,18,16,17,
              25,26,24,27,28,26,25,27,24,28,26,27],

    "feat3": [2,3,1,2,4,5,2,3,1,2,4,5, 6,7,5,8,9,7,6,8,5,9,7,8,
              1,2,3,1,4,5,3,2,1,4,3,5, 7,8,6,9,10,8,7,9,6,10,8,9,
              3,4,2,5,6,4,3,5,2,6,4,5],

    "feat4": [1,1,2,1,3,2,1,2,1,3,2,1, 4,4,5,4,6,5,4,5,4,6,5,4,
              7,8,6,7,9,8,7,8,6,9,8,7, 2,3,1,2,4,3,2,3,1,4,3,2,
              5,6,4,5,7,6,5,6,4,7,6,5],
}


labels = [0]*12 + [1]*12 + [2]*12 + [0]*12 + [1]*12  # 60 labels total


df = pd.DataFrame(data)

df["label"] = labels


X = df[["feat1", "feat2", "feat3", "feat4"]]

y = df["label"]
```

```
df.head()
```

KNN Classification:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

plt.figure(figsize=(7,5))
plt.scatter(range(len(y_test)), y_test, label="Actual")
plt.scatter(range(len(y_pred)), y_pred, label="Predicted")
plt.title("KNN Classification: Actual vs Predicted")
plt.legend()
plt.show()
```


KNN Classification: Actual vs Predicted
Actual
Predicted
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
10

Experiment 11:

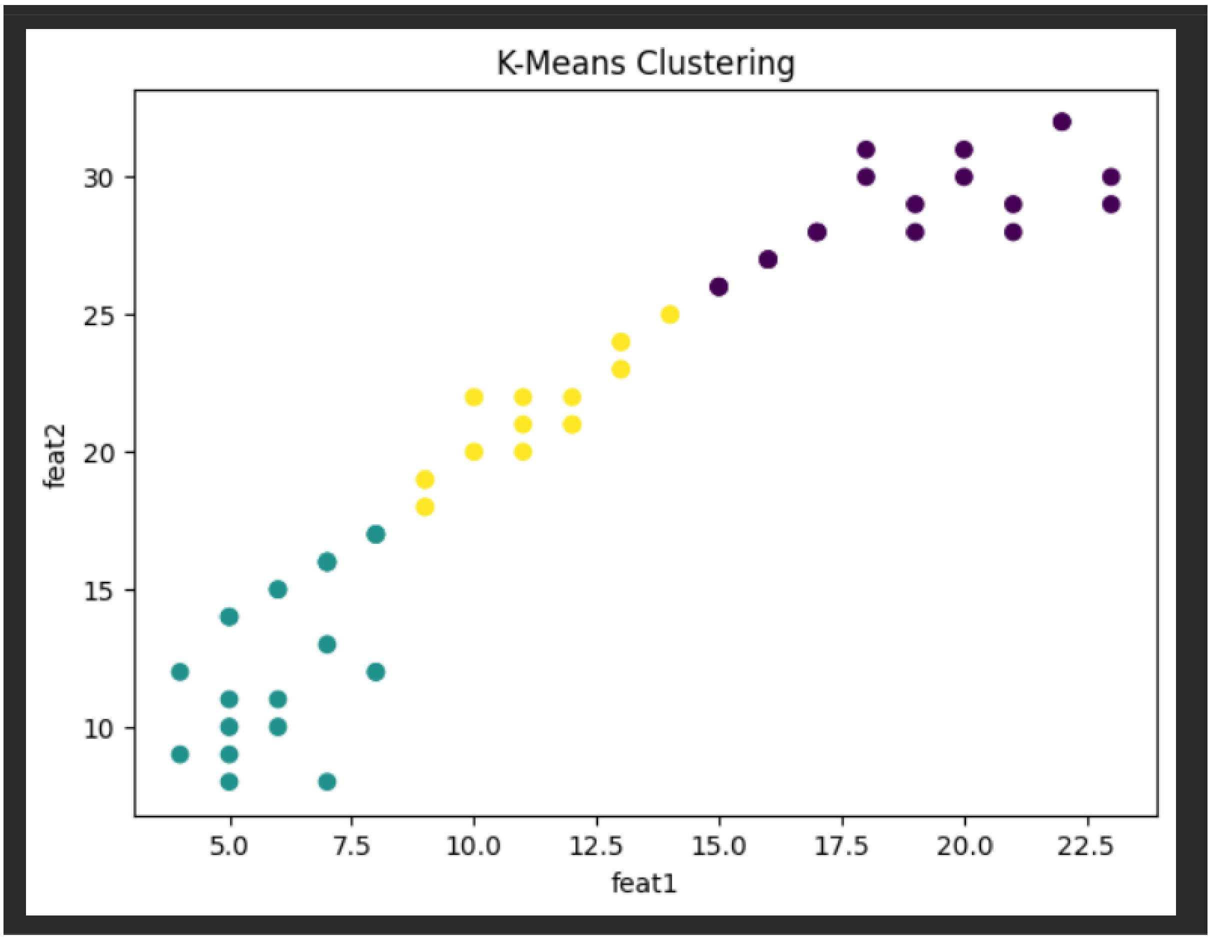
Creating dataset:

```
import pandas as pd

import numpy as np


data = {

    "feat1": [5,6,4,5,7,8,5,6,4,5,7,8, 10,11,9,12,13,11,10,12,9,13,12,11,

              20,21,19,18,22,23,19,20,21,22,18,23, 6,7,5,8,9,7,6,8,5,9,7,8,

              14,15,13,16,17,15,14,16,13,17,15,16],


    "feat2": [10,11,12,9,8,12,11,10,9,8,13,12, 20,22,19,21,23,20,22,21,19,23,22,21,

              30,29,28,31,32,30,29,31,28,32,30,29, 15,16,14,17,18,16,15,17,14,18,16,17,

              25,26,24,27,28,26,25,27,24,28,26,27],


    "feat3": [2,3,1,2,4,5,2,3,1,2,4,5, 6,7,5,8,9,7,6,8,5,9,7,8,

              1,2,3,1,4,5,3,2,1,4,3,5, 7,8,6,9,10,8,7,9,6,10,8,9,

              3,4,2,5,6,4,3,5,2,6,4,5],


    "feat4": [1,1,2,1,3,2,1,2,1,3,2,1, 4,4,5,4,6,5,4,5,4,6,5,4,

              7,8,6,7,9,8,7,8,6,9,8,7, 2,3,1,2,4,3,2,3,1,4,3,2,

              5,6,4,5,7,6,5,6,4,7,6,5],

}


labels = [0]*12 + [1]*12 + [2]*12 + [0]*12 + [1]*12  # 60 labels total


df = pd.DataFrame(data)

df["label"] = labels


X = df[["feat1", "feat2", "feat3", "feat4"]]
```

```
y = df["label"]


df.head()
```

K - Means Clustering:

```
from sklearn.cluster import KMeans


kmeans = KMeans(n_clusters=3, random_state=0)

clusters_km = kmeans.fit_predict(X)


plt.figure(figsize=(7,5))

plt.scatter(X["feat1"], X["feat2"], c=clusters_km)

plt.title("K-Means Clustering")

plt.xlabel("feat1")

plt.ylabel("feat2")

plt.show()
```


K-Means Clustering
feat2
feat1
30
25
20
15
10
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5

Experiment 12:

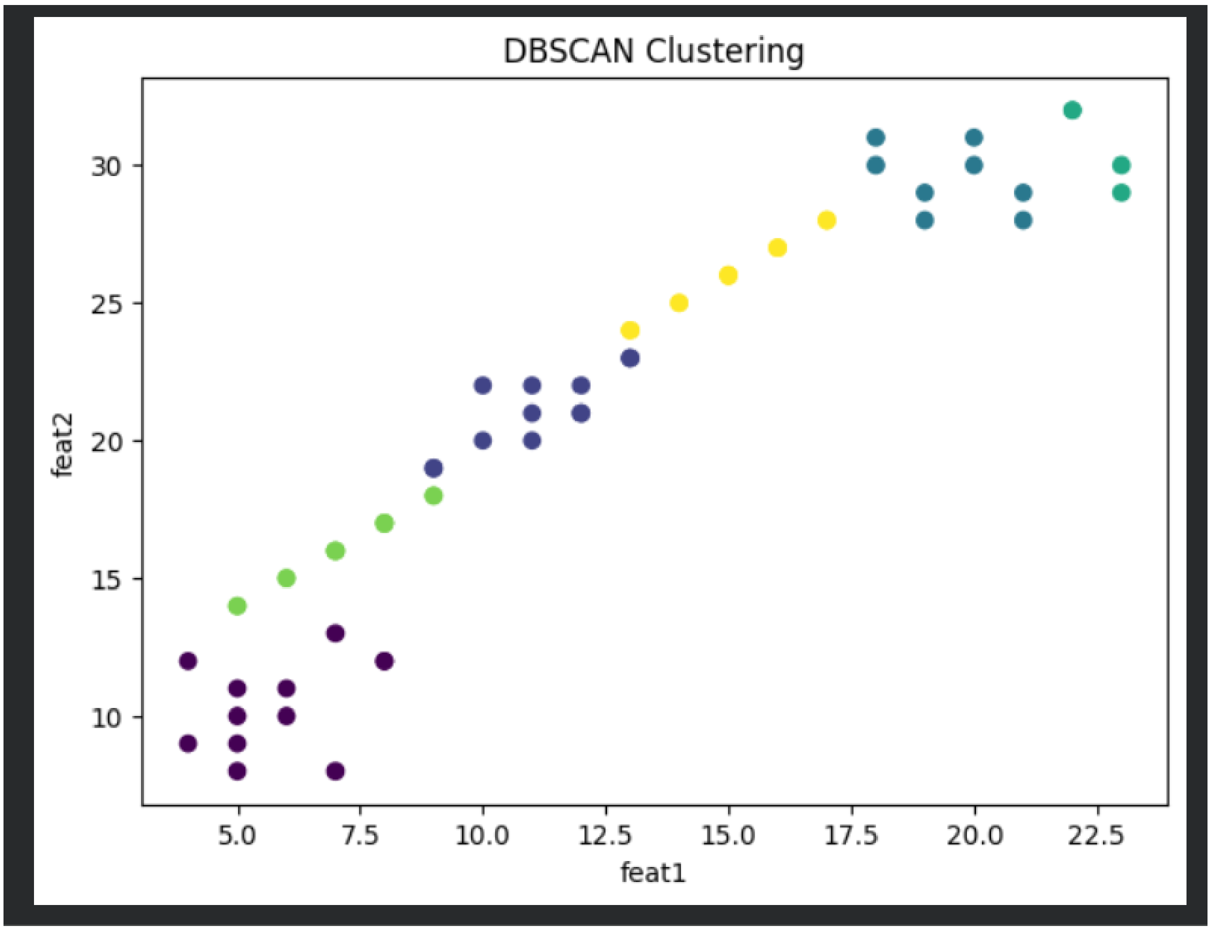
Creating dataset:

```
import pandas as pd

import numpy as np


data = {
    "feat1": [5,6,4,5,7,8,5,6,4,5,7,8, 10,11,9,12,13,11,10,12,9,13,12,11,
              20,21,19,18,22,23,19,20,21,22,18,23, 6,7,5,8,9,7,6,8,5,9,7,8,
              14,15,13,16,17,15,14,16,13,17,15,16],

    "feat2": [10,11,12,9,8,12,11,10,9,8,13,12, 20,22,19,21,23,20,22,21,19,23,22,21,
              30,29,28,31,32,30,29,31,28,32,30,29, 15,16,14,17,18,16,15,17,14,18,16,17,
              25,26,24,27,28,26,25,27,24,28,26,27],

    "feat3": [2,3,1,2,4,5,2,3,1,2,4,5, 6,7,5,8,9,7,6,8,5,9,7,8,
              1,2,3,1,4,5,3,2,1,4,3,5, 7,8,6,9,10,8,7,9,6,10,8,9,
              3,4,2,5,6,4,3,5,2,6,4,5],

    "feat4": [1,1,2,1,3,2,1,2,1,3,2,1, 4,4,5,4,6,5,4,5,4,6,5,4,
              7,8,6,7,9,8,7,8,6,9,8,7, 2,3,1,2,4,3,2,3,1,4,3,2,
              5,6,4,5,7,6,5,6,4,7,6,5],
}

labels = [0]*12 + [1]*12 + [2]*12 + [0]*12 + [1]*12  # 60 labels total

df = pd.DataFrame(data)

df["label"] = labels

X = df[["feat1", "feat2", "feat3", "feat4"]]
```

```
y = df["label"]

df.head()
```

DBSCAN Clustering:

```
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=3, min_samples=3)
clusters_db = db.fit_predict(X)

plt.figure(figsize=(7,5))
plt.scatter(X["feat1"], X["feat2"], c=clusters_db)
plt.title("DBSCAN Clustering")
plt.xlabel("feat1")
plt.ylabel("feat2")
plt.show()
```


DBSCAN Clustering
feat1
feat2
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
10
15
20
25
30