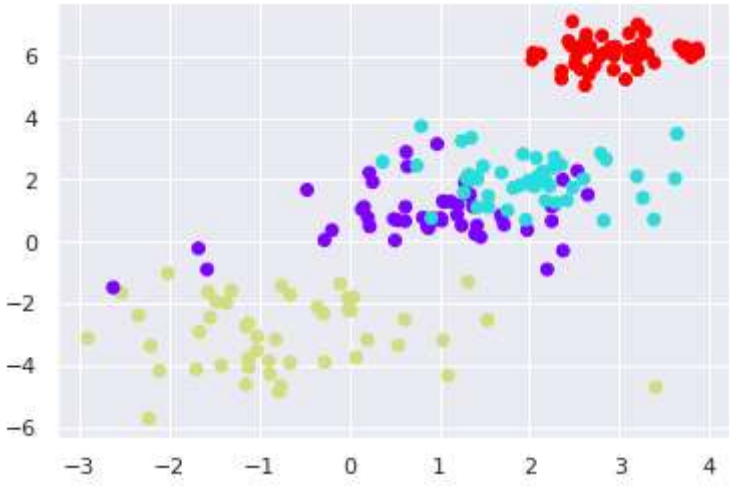
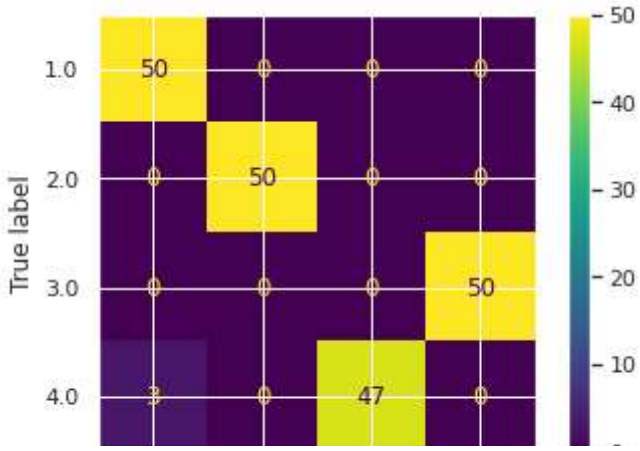
Part 1: K Means Clustering

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
# Reading the data
df = pd.read_csv('cluster_table.csv')
df
#Selecting the matrix
x = df.iloc[:,1:3]

#assign number of clusters and applying kmeans.fit
kmeans = KMeans(4)
model = kmeans.fit(x)
predicted_clusters = kmeans.fit_predict(x)

#Scatter plot with clustering
df_clustered = df.copy()
df_clustered['Clusters'] = predicted_clusters +1
plt.scatter(df_clustered['X'],df_clustered['Y'],c=df_clustered['Clusters'] , cmap='rainbow')
plt.show()
print(df_clustered.head(10))

# Preparing data for confusion matrix
k_labels = kmeans.labels_  # to identify cluster labels
k_labels_matched = np.empty_like(k_labels)

actual_label = df_clustered['Cluster label']
kMeans_label = df_clustered['Clusters']

# Compute confusion matrix
cm = confusion_matrix(actual_label, k_labels_matched)

# Plot confusion matrix
ConfusionMatrixDisplay.from_predictions(actual_label, kMeans_label)
```



	X	Y	Z	Cluster label	Clusters
0	1.014886	0.750362	1.156124	1.0	1
1	-0.197795	0.362302	0.884961	1.0	1
2	0.628812	2.889340	0.019449	1.0	1
3	0.642886	2.423315	0.329615	1.0	1
4	2.197576	-0.903305	1.581067	1.0	1
5	1.975973	0.372510	0.929662	1.0	1
6	1.717956	0.536231	1.209356	1.0	1
7	-0.280465	0.043773	-0.449671	1.0	1
8	1.282532	1.902875	1.147130	1.0	1
9	0.814000	0.765601	0.328497	1.0	1

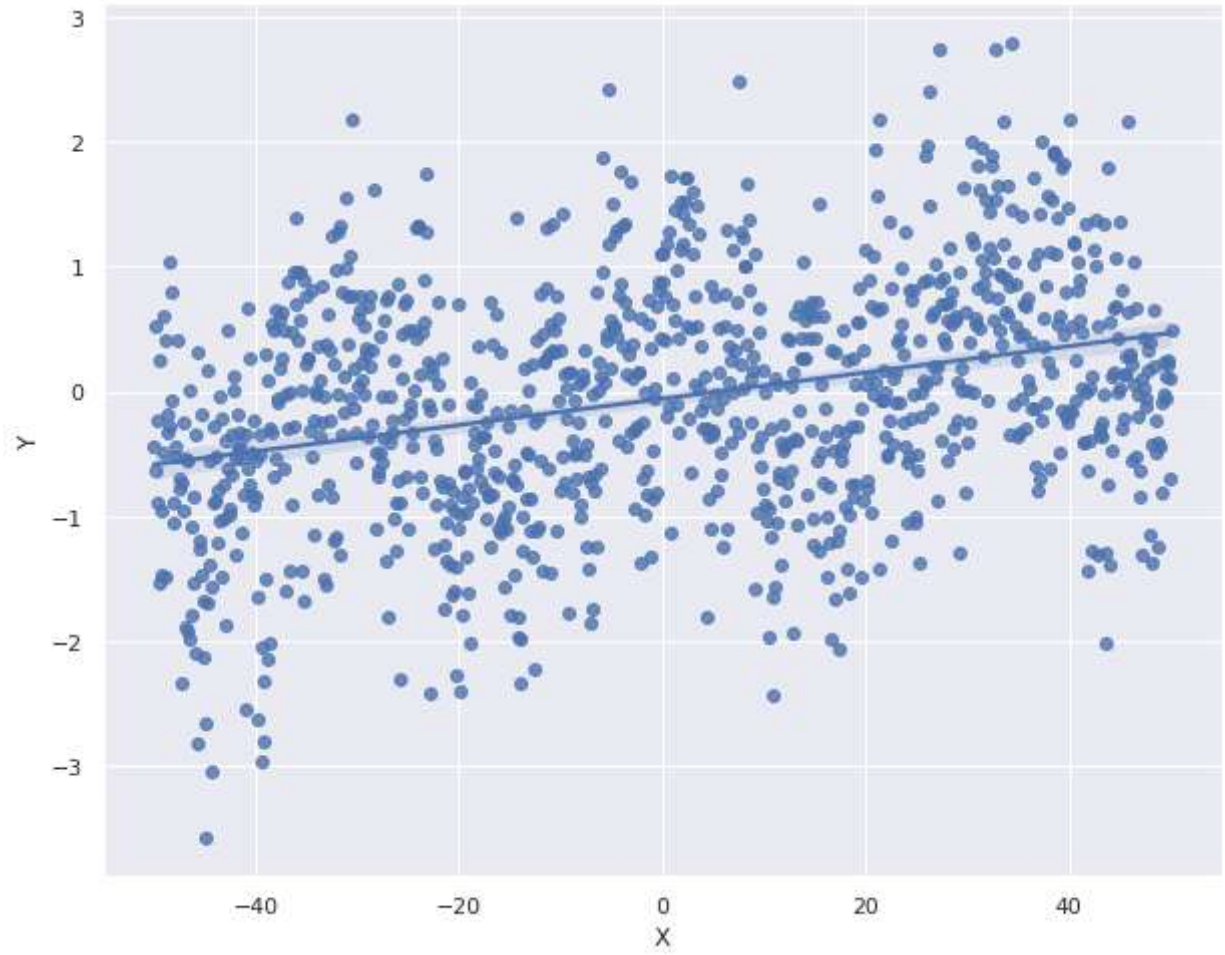
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fb70ff822d0>
```



Part 2: Regression

```
import numpy as np
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
sns.set_theme(color_codes=True)


# importing data
values = pd.read_csv("regression_table.csv")
```

```
#plot size
f, ax = plt.subplots(figsize=(10, 8))

#scatter plot with regression line
sns.regplot(x="X", y="Y", data=values, ax=ax);
```




```
# Regression Metrics
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

x = np.array(values['X']).reshape((-1, 1))
y= np.array(values['Y'])
model = LinearRegression().fit(x, y)

# Print the y axis Intercept:
print('intercept:', model.intercept_)

# Predict a Response and print it:
y_pred = model.predict(x)
```

```
intercept: -0.05685208077628467
```

```
#regression coefficient
regcoeff = model.coef_
print('Regression coefficient:', regcoeff)

#mean square error
mse = mean_squared_error(y, y_pred)
print('Mean Square Error:', mse)

#Coefficient of determination: r2_score (y, y_predicted_values)
coef_det = r2_score(y, y_pred)
print('Coefficient of determination:', coef_det)
```

```
Regression coefficient: [0.01052991]
Mean Square Error: 0.8115142707620799
Coefficient of determination: 0.10240499306072204
```

Comments:

1. Coeff. of Determination (R2) varies from 0-1. If R2 close to 1 it means Regression line is able to explain a major portion of variability. However in our case R2 = 0.1024(very close to 0) which means the model does not fully represent the relaationship b/w the variables.

✓ 0s completed at 7:10 PM