Neural Network and Deep Learning- Assignment 1

1: Implement Naïve Bayes method using scikit-learn library
Use dataset available with name glass
Use train_test_split to create training and testing part Evaluate the model on test part using score and classification_report(y_true, y_pred)

In [2]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

# Load the dataset from the CSV file
glass_data = pd.read_csv("glass.csv")

# Split the dataset into features (X) and target (y)
X = glass_data.drop('Type', axis=1)
y = glass_data['Type']

# Split the dataset into training and testing parts
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the Naïve Bayes classifier
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = classifier.predict(X_test)

# Evaluate the model
score = classifier.score(X_test, y_test)
report = classification_report(y_test, y_pred)

print("Accuracy:", score)
print("Classification Report:")
print(report)
```

Output:

```
Accuracy: 0.5581395348837209
Classification Report:
              precision    recall  f1-score   support

           1       0.41      0.64      0.50        11
           2       0.43      0.21      0.29        14
           3       0.40      0.67      0.50         3
           5       0.50      0.25      0.33         4
           6       1.00      1.00      1.00         3
           7       0.89      1.00      0.94         8

    accuracy                           0.56        43
   macro avg       0.60      0.63      0.59        43
weighted avg       0.55      0.56      0.53        43
```

Neural Network and Deep Learning- Assignment 1

2: Implement linear SVM method using scikit-learn

Use the same dataset above

Use train_test_split to create training and testing part Evaluate the model on test part using score and classification_report(y_true, y_pred)

In [10]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report

# Load the dataset from the CSV file
glass_data = pd.read_csv("glass.csv")

# Split the dataset into features (X) and target (y)
X = glass_data.drop('Type', axis=1)
y = glass_data['Type']

# Split the dataset into training and testing parts
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the Linear SVM classifier
classifier = LinearSVC(max_iter=10000)
classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = classifier.predict(X_test)

# Evaluate the model
score = classifier.score(X_test, y_test)
report = classification_report(y_test, y_pred, zero_division=0)

print("Accuracy:", score)
print("Classification Report:")
print(report)
```

Output:

```
Accuracy: 0.5813953488372093
Classification Report:
              precision    recall  f1-score   support

           1       0.56      0.82      0.67        11
           2       0.50      0.50      0.50        14
           3       0.00      0.00      0.00         3
           5       1.00      0.25      0.40         4
           6       0.00      0.00      0.00         3
           7       0.67      1.00      0.80         8

    accuracy                           0.58        43
   macro avg       0.45      0.43      0.39        43
weighted avg       0.52      0.58      0.52        43
```

Neural Network and Deep Learning- Assignment 1

Which algorithm you got better accuracy? Can you justify why?

To determine which algorithm performs better on a specific dataset, it is essential to evaluate their performance using appropriate metrics such as accuracy, precision, recall, and F1-score. These metrics help to assess the model's ability to correctly classify instances from different classes and handle class imbalances. So based on the accuracy I got linear SVM method using scikit-learn algorithm got the more accuracy.

3. Implement Linear Regression using scikit-learn a) Import the given "Salary_Data.csv" b) Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset. c) Train and predict the model. d) Calculate the mean_squared error. e) Visualize both train and test data using scatter plot.

In [14]:
```
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

# Import the dataset from "Salary_Data.csv"
data = pd.read_csv("Salary_Data.csv")

# Split the data into features (X) and target (y)
X = data["YearsExperience"].values.reshape(-1, 1)
y = data["Salary"].values

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

# Create and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Calculate the mean squared error
mse = mean_squared_error(y_test, y_pred)

print("Mean Squared Error:", mse)

# Visualize the training and test data
plt.scatter(X_train, y_train, color='blue', label='Training data')
plt.scatter(X_test, y_test, color='red', label='Test data')
plt.plot(X_test, y_pred, color='green', linewidth=2, label='Linear Regression')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Linear Regression - Salary Prediction')
plt.legend()
plt.show()
```

Neural Network and Deep Learning- Assignment 1

Output:

```
Mean Squared Error: 35301898.88713492
```