In [22]:
```
pip install xlrd==1.2.0
```

```
Requirement already satisfied: xlrd==1.2.0 in /Users/saharshamuddagou
ni/opt/anaconda3/lib/python3.9/site-packages (1.2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [29]:
```
#Importing the libraries
import pandas as pd
import numpy as np
import pandas as pd
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.svm import SVC

# reading the dataset file
df = pd.read_csv('glass.csv')

X = df.drop(['Type'], axis=1)
Y = df["Type"]

#splitting the dataset into training set and testing set
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X, Y, test_size=0.

#instantiating the Naive Bayes model and fitting it with traning set
gnb = GaussianNB()
gnb.fit(X_Train,Y_Train)

# Predicting the Test set result
Y_Pred = gnb.predict(X_Test)

acc_knn = round(gnb.score(X_Train, Y_Train) * 100, 2)
print('Train Accuracy: ', acc_knn)
acc_knn = round(gnb.score(X_Test, Y_Test) * 100, 2)
print('Test Accuracy: ', acc_knn)

print('\nClassification Report: \n', classification_report(Y_Test, Y_P
```

```
Train Accuracy:  56.14
Test Accuracy:  55.81

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
```

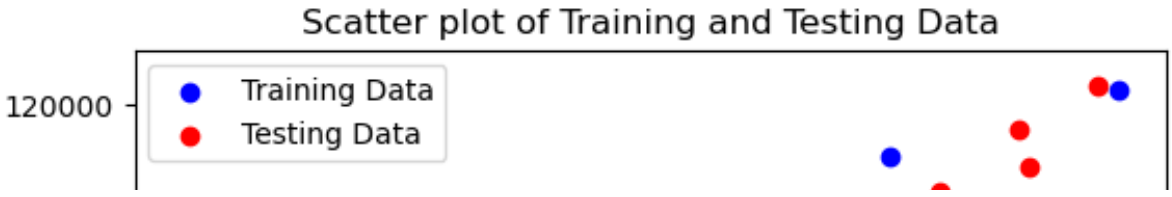
In [30]:
```
#instantiating the linear SVM model and fitting it with traning set
svm = SVC(kernel = 'linear')

svm.fit(X_Train, Y_Train)

Y_pred = svm.predict(X_Test)
acc_knn = round(svm.score(X_Train, Y_Train) * 100, 2)
print('Train Accuracy: ', acc_knn)
acc_knn = round(svm.score(X_Test, Y_Test) * 100, 2)
print('Test Accuracy: ', acc_knn)
print('Classification Report: \n', classification_report(Y_Test, Y_Pre
```

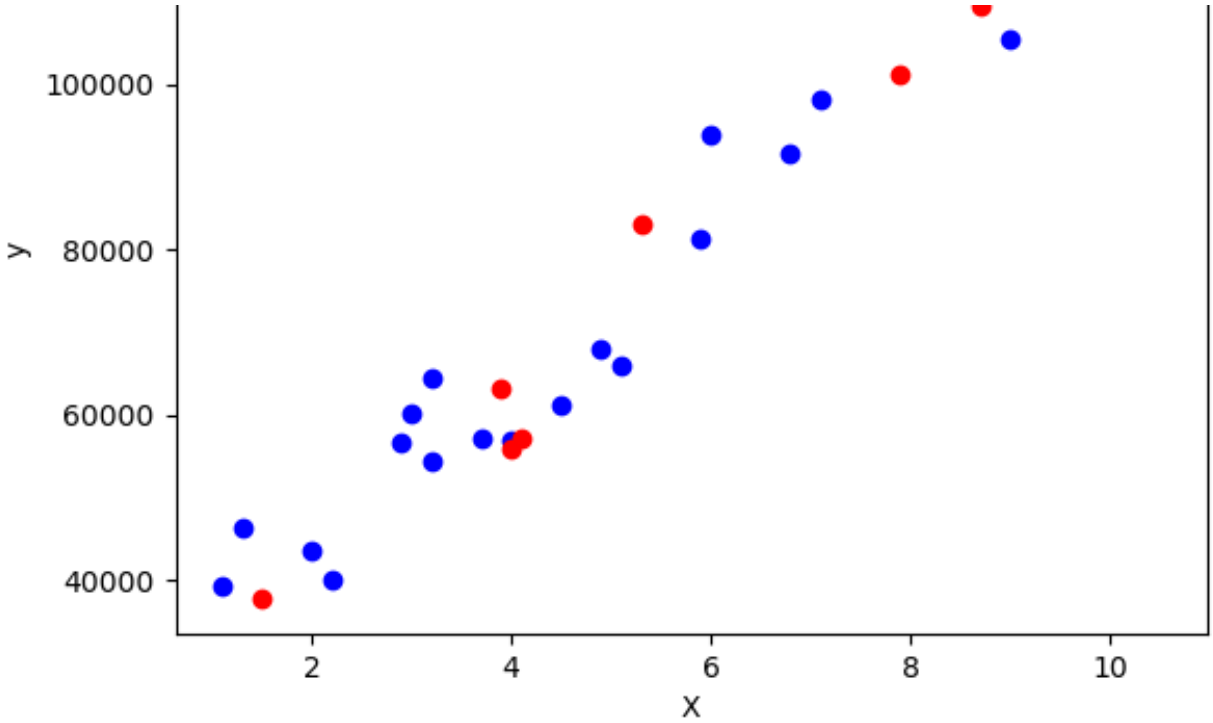
```
Train Accuracy:  66.67
Test Accuracy:  74.42
Classification Report:
               precision    recall  f1-score   support

           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
```

In [31]:
```

```

```
data = pd.read_csv('Salary_Data.csv')

X = data.iloc[:, :-1].values
y = data.iloc[:, 1].values

from sklearn.model_selection import train_test_split
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X,y,test_size=1/3,

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_Train, Y_Train)

Y_Pred = regressor.predict(X_Test)

from sklearn.metrics import mean_squared_error

# Assuming you have trained your linear regression model and obtained

# Calculate the mean squared error
mse = mean_squared_error(Y_Test, Y_Pred)
print("Mean Squared Error:", mse)

import matplotlib.pyplot as plt

# Assuming you have split the data into X_train, X_test, y_train, y_te

# Scatter plot for training data
plt.scatter(X_Train, Y_Train, color='blue', label='Training Data')

# Scatter plot for testing data
plt.scatter(X_Test, Y_Test, color='red', label='Testing Data')

# Add labels and title to the plot
plt.xlabel('X')
plt.ylabel('y')
plt.title('Scatter plot of Training and Testing Data')

# Add legend
plt.legend()

# Display the plot
plt.show()
```

```
Mean Squared Error: 21026037.329511303
```





In [ ]: