```
[1] from google.colab import drive
    drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
[2] # Import required libraries and modules
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import confusion_matrix, classification_report
    from sklearn.neural_network import MLPClassifier
    from sklearn.metrics import mean_squared_error
    from math import sqrt
    from sklearn.metrics import r2_score
    import seaborn as sns
    import matplotlib.pyplot as plt
    from sklearn.metrics import confusion_matrix
```

```
[3] path_to_csv = '/content/gdrive/My Drive/glass.csv'
```

```
[4] # Load data
    glass_df = pd.read_csv(path_to_csv)
```

```
[5] print(glass_df.shape)
    glass_df.describe()
```

[5] (214, 10)

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	0.175047	0.057009	2.780374
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	0.497219	0.097439	2.103739
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.000000	0.000000	1.000000
25%	1.516522	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	0.000000	0.000000	1.000000
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	0.000000	0.000000	2.000000
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	0.000000	0.100000	3.000000
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	3.150000	0.510000	7.000000

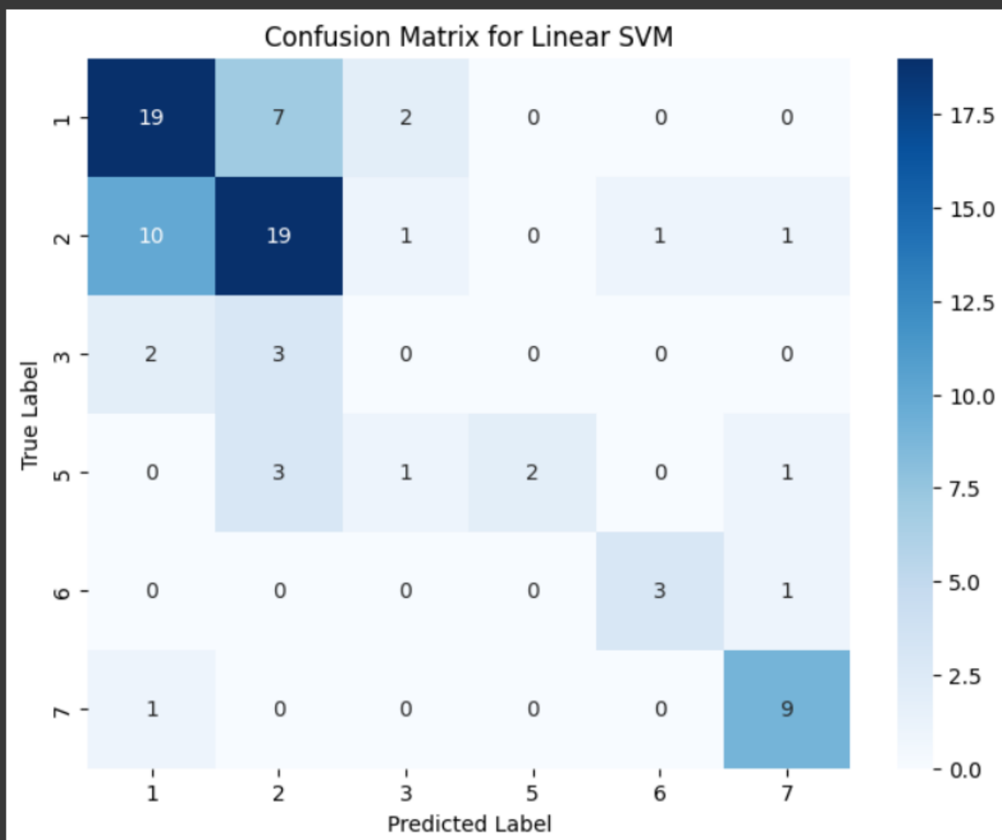
```
# Create arrays for the features and the response variable
X = glass_df.drop('Type', axis=1)
y = glass_df['Type']
```

```
[7] # Create training and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4, random_state=42)

    # Scaling the feature variables
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
```

Linear SVM

```
from sklearn.svm import LinearSVC

# Initializing the model
linear_svc = LinearSVC(max_iter=10000, random_state=42)

# Training the model
linear_svc.fit(X_train_scaled, y_train)

# Making predictions
y_pred_svc = linear_svc.predict(X_test_scaled)

# Evaluating the model
print(confusion_matrix(y_test, y_pred_svc))
print(classification_report(y_test, y_pred_svc))


# Predictions from Linear SVM
y_pred_svc = linear_svc.predict(X_test_scaled)

# Generate a confusion matrix
conf_matrix_svc = confusion_matrix(y_test, y_pred_svc)

# Plot the confusion matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_svc, annot=True, fmt='d', cmap='Blues', xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix for Linear SVM')
plt.show()
```

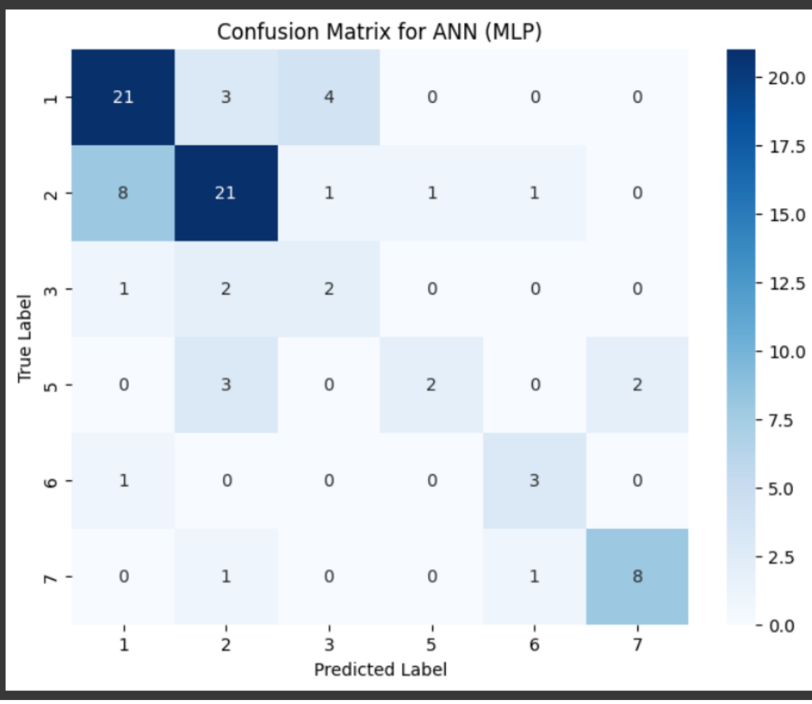
```
[[19  7  2  0  0  0]
 [10 19  1  0  1  1]
 [ 2  3  0  0  0  0]
 [ 0  3  1  2  0  1]
 [ 0  0  0  0  3  1]
 [ 1  0  0  0  0  9]]
              precision    recall  f1-score   support

           1       0.59      0.68      0.63        28
           2       0.59      0.59      0.59        32
           3       0.00      0.00      0.00         5
           5       1.00      0.29      0.44         7
           6       0.75      0.75      0.75         4
           7       0.75      0.90      0.82        10

    accuracy                           0.60        86
   macro avg       0.61      0.53      0.54        86
weighted avg       0.62      0.60      0.59        86
```



Multi-Layer Perceptron (ANN)

```python
from sklearn.neural_network import MLPClassifier

# Initializing the model
mlp_clf = MLPClassifier(hidden_layer_sizes=(100,), max_iter=10000, random_state=42)

# Training the model
mlp_clf.fit(X_train_scaled, y_train)

# Making predictions
y_pred_mlp = mlp_clf.predict(X_test_scaled)

# Evaluating the model
print(confusion_matrix(y_test, y_pred_mlp))
print(classification_report(y_test, y_pred_mlp))


# Predictions from ANN (MLP)
y_pred_mlp = mlp_clf.predict(X_test_scaled)

# Generate a confusion matrix
conf_matrix_mlp = confusion_matrix(y_test, y_pred_mlp)

# Plot the confusion matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_mlp, annot=True, fmt='d', cmap='Blues', xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix for ANN (MLP)')
plt.show()
```

```
[[21  3  4  0  0  0]
 [ 8 21  1  1  1  0]
 [ 1  2  2  0  0  0]
 [ 0  3  0  2  0  2]
 [ 1  0  0  0  3  0]
 [ 0  1  0  0  1  8]]
              precision    recall  f1-score   support

           1       0.68      0.75      0.71        28
           2       0.70      0.66      0.68        32
           3       0.29      0.40      0.33         5
           5       0.67      0.29      0.40         7
           6       0.60      0.75      0.67         4
           7       0.80      0.80      0.80        10

    accuracy                           0.66        86
   macro avg       0.62      0.61      0.60        86
weighted avg       0.67      0.66      0.66        86
```



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

The ANN algorithm outperformed the Linear SVM with a higher accuracy and higher F1-Scores due to its ability to handle non-linear data and model intricate relationships, crucial for the given dataset's multiple classes and observed imbalance. In essence, the ANN's complexity and advanced capabilities make it more suitable for this dataset, providing superior accuracy and balance between precision and recall compared to the Linear SVM.

GitHub Repo: https://github.com/dheerukarra/BigDataAnalytics/tree/main/ICP_5

YouTube Video Link: https://youtu.be/rGKVGPBixA8