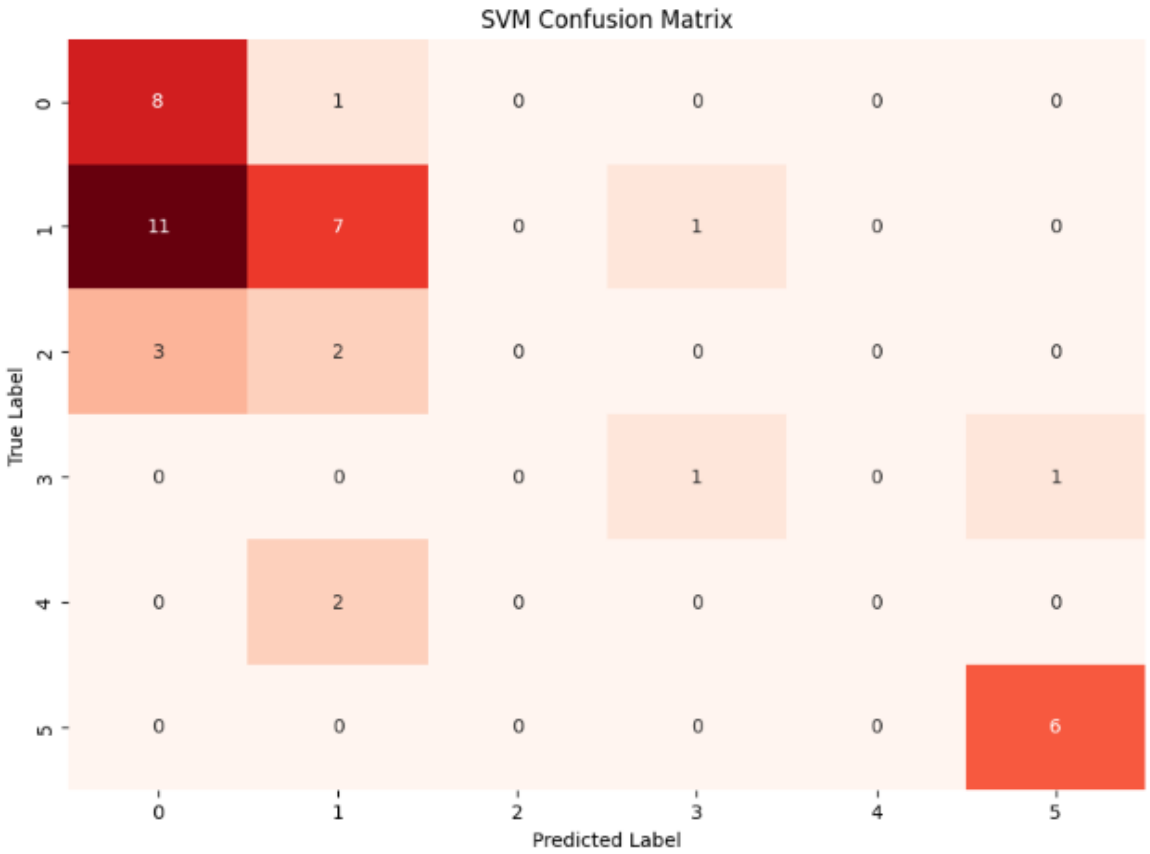
GIT LINK-
https://github.com/Harishwar-reddi/ICP-5

YouTube Link-
https://youtu.be/NXjjWboMQvo

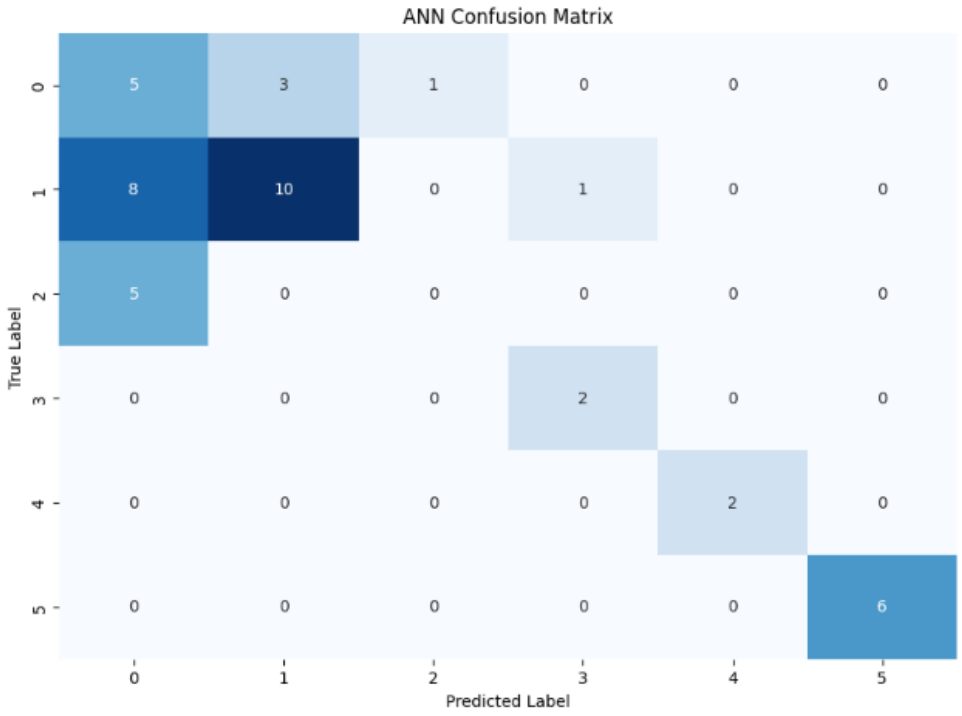
```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

# Load the dataset
url = 'https://drive.google.com/uc?id={}'.format('1ntvqcDKivieS9a1MWnmOzUVMadSL-zOS')
df = pd.read_csv(url)

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

# Separate the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# SVM model training
svm_model = SVC(kernel='linear', random_state=0)
svm_model.fit(X_train, y_train)

# Prediction using the test set
y_pred_svm = svm_model.predict(X_test)

# Score of accuracy
svm_accuracy = accuracy_score(y_test, y_pred_svm)
print(f"SVM Accuracy: {svm_accuracy}")

# Printing Classification report
print(classification_report(y_test, y_pred_svm))

# Visualization
conf_matrix = confusion_matrix(y_test, y_pred_svm)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, cmap="Reds", fmt='g', cbar=False)
plt.title('SVM Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
SVM Accuracy: 0.5116279069767442
              precision    recall  f1-score   support

           1       0.36      0.89      0.52         9
           2       0.58      0.37      0.45        19
           3       0.00      0.00      0.00         5
           5       0.50      0.50      0.50         2
           6       0.00      0.00      0.00         2
           7       0.86      1.00      0.92         6

    accuracy                           0.51        43
   macro avg       0.38      0.46      0.40        43
weighted avg       0.48      0.51      0.46        43
```


SVM Confusion Matrix
True Label
Predicted Label
0
1
2
3
4
5
8 1 0 0 0 0
11 7 0 1 0 0
3 2 0 0 0 0
0 0 0 1 0 1
0 2 0 0 0 0
0 0 0 0 0 6

```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

# Load the dataset
url = 'https://drive.google.com/uc?id={}'.format('1ntvqcDKivieS9a1MWnmOzUVMadSL-zOS')
df = pd.read_csv(url)

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

# Separate the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# ANN model Training
ann_model = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=2000, random_state=0)
ann_model.fit(X_train, y_train)

# Prediction using the test set
y_pred_ann = ann_model.predict(X_test)

# Score of Accuracy
ann_accuracy = accuracy_score(y_test, y_pred_ann)
print(f"ANN Accuracy: {ann_accuracy}")

# Printing Classification report
print(classification_report(y_test, y_pred_ann))

# Visualization
conf_matrix_ann = confusion_matrix(y_test, y_pred_ann)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix_ann, annot=True, cmap="Blues", fmt='g', cbar=False)
plt.title('ANN Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
ANN Accuracy: 0.5813953488372093
              precision    recall  f1-score   support

           1       0.28      0.56      0.37         9
           2       0.77      0.53      0.62        19
           3       0.00      0.00      0.00         5
           5       0.67      1.00      0.80         2
           6       1.00      1.00      1.00         2
           7       1.00      1.00      1.00         6

    accuracy                           0.58        43
   macro avg       0.62      0.68      0.63        43
weighted avg       0.62      0.58      0.58        43
```



CONCLUSION

Comparing the accuracy of the ANN model to the SVM model using the same dataset, the accuracy of the ANN model was found to be higher, at usually 58.14 percent (58.14%). The ANN's ability to identify intricate, non-linear patterns in the Data set may be one explanation for its improved performance. In any case, it is crucial to take into account the dataset, the concept behind the problem, and the available computer resources when choosing a technique. To optimize performance for a particular task, hyperparameters should be evaluated and modified.