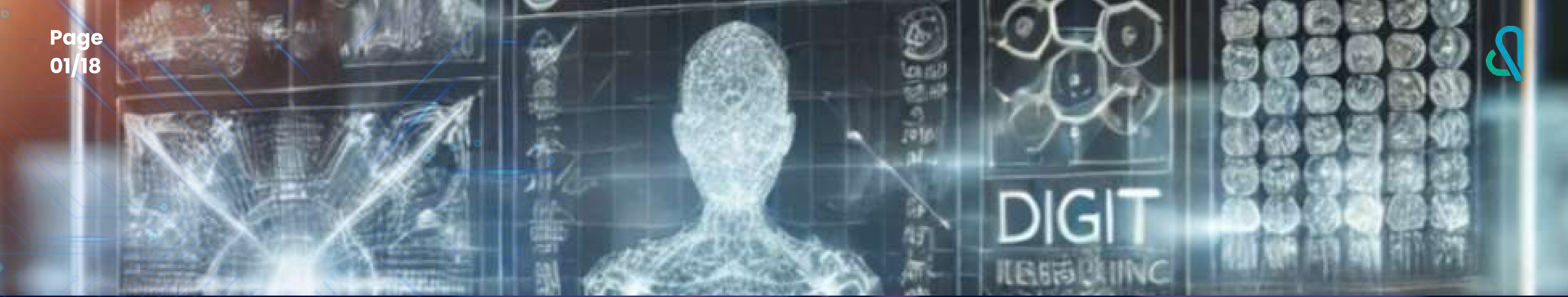


MACHINE LEARNING

HANDWRITTEN DIGIT RECOGNITION WITH MACHINE LEARNING CLASSIFICATION MODELS

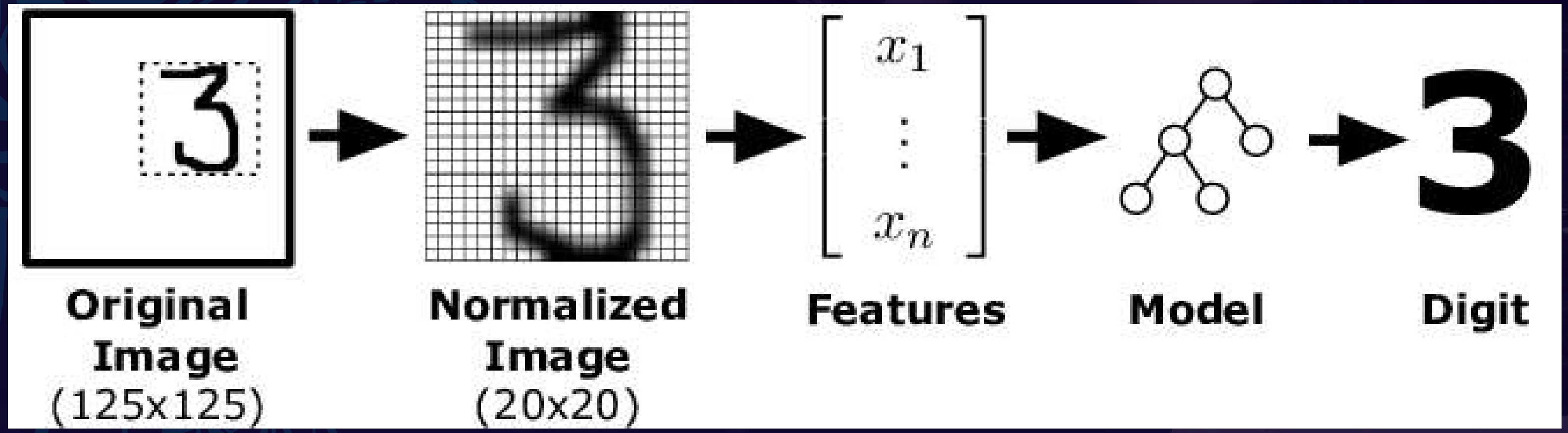
Github

By Ni Gusti Ayu Agung Indraswari

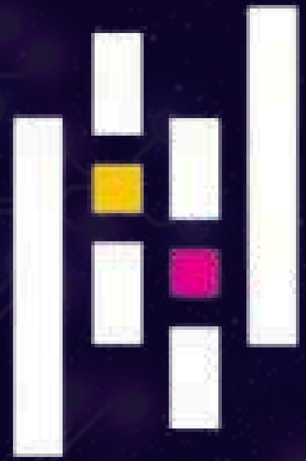
February 2025

Project Overview

This project aims to accurately classify handwritten digits using the Digits dataset from scikit-learn. With 1,797 samples in an 8×8 pixel matrix, machine learning models will be applied to improve classification accuracy. The results are expected to benefit automated character recognition, such as document processing and digital identity verification.



Tools and Library



By Ni Gusti Ayu Agung Indraswari

Machine Learning Process

1

Data

Uses the Digits dataset from scikit-learn

2

EDA

Class distribution, displays, sample digit images, correlations, ect.

3

Preprocessing

Splits datas and applies normalization.

4

Training

Trains classification models

5

Validation

Evaluates model performance using accuracy, classification reports, and confusion matrices.

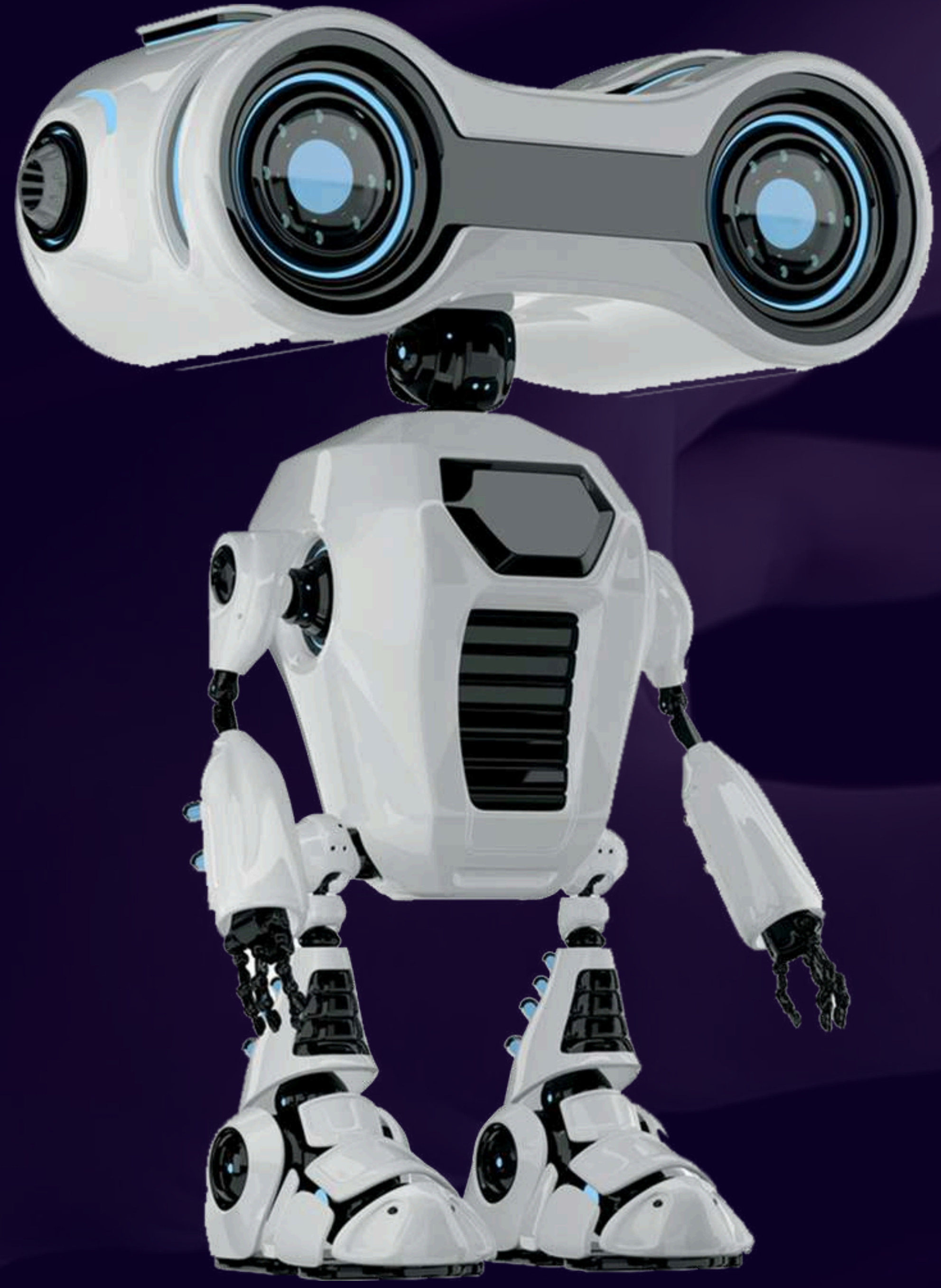
6

Compare

Compares different models to find the one with the best accuracy and performance.

By Ni Gusti Ayu Agung Indraswari

Dataset Overview

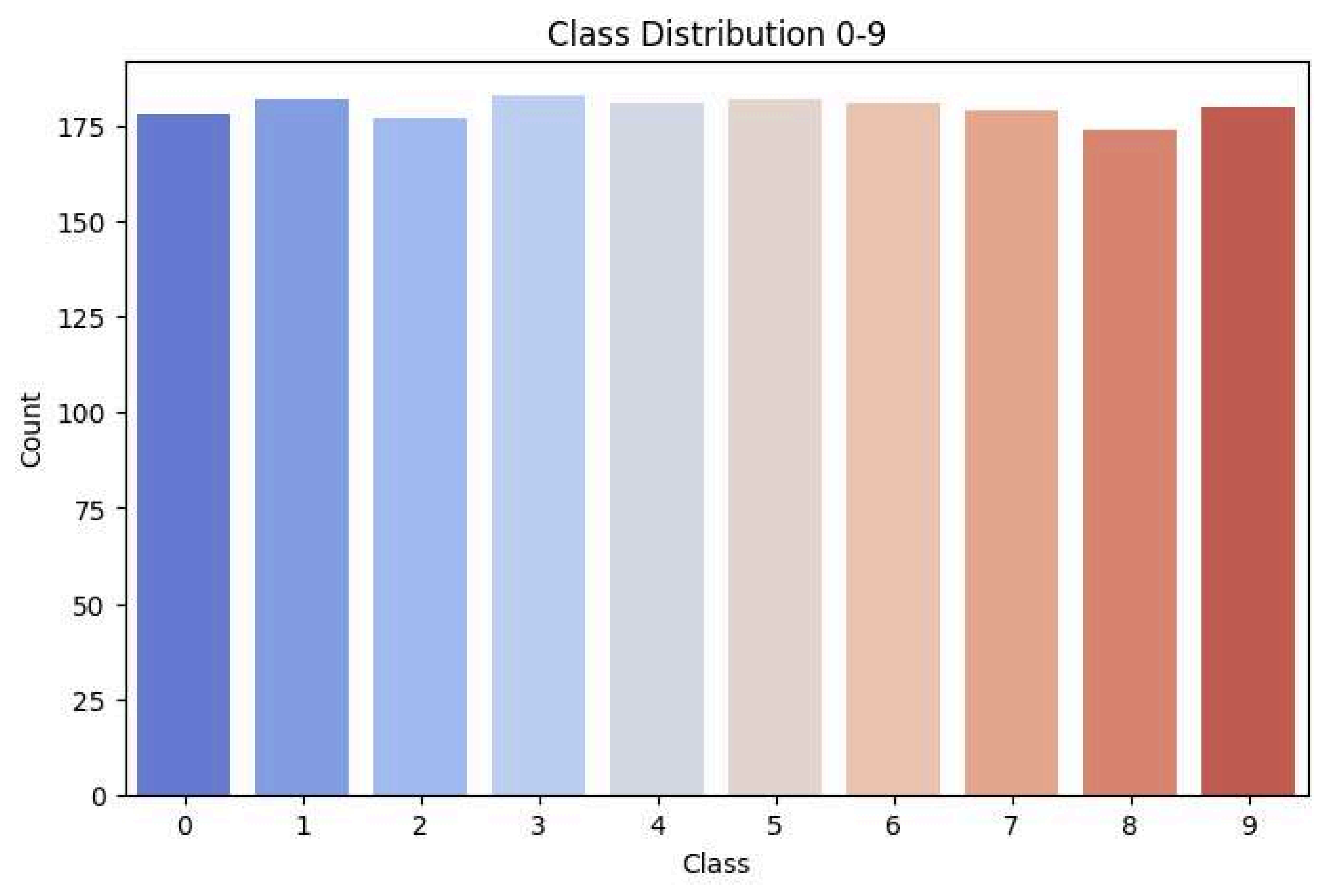


Source	Scikit-learn
Data Type	Handwritten digit images (0-9)
Number of Samples	1,797 images
Image Dimensions	8x8 pixels (64 features per image)
Target Classes	10 classes (digits 0 to 9)

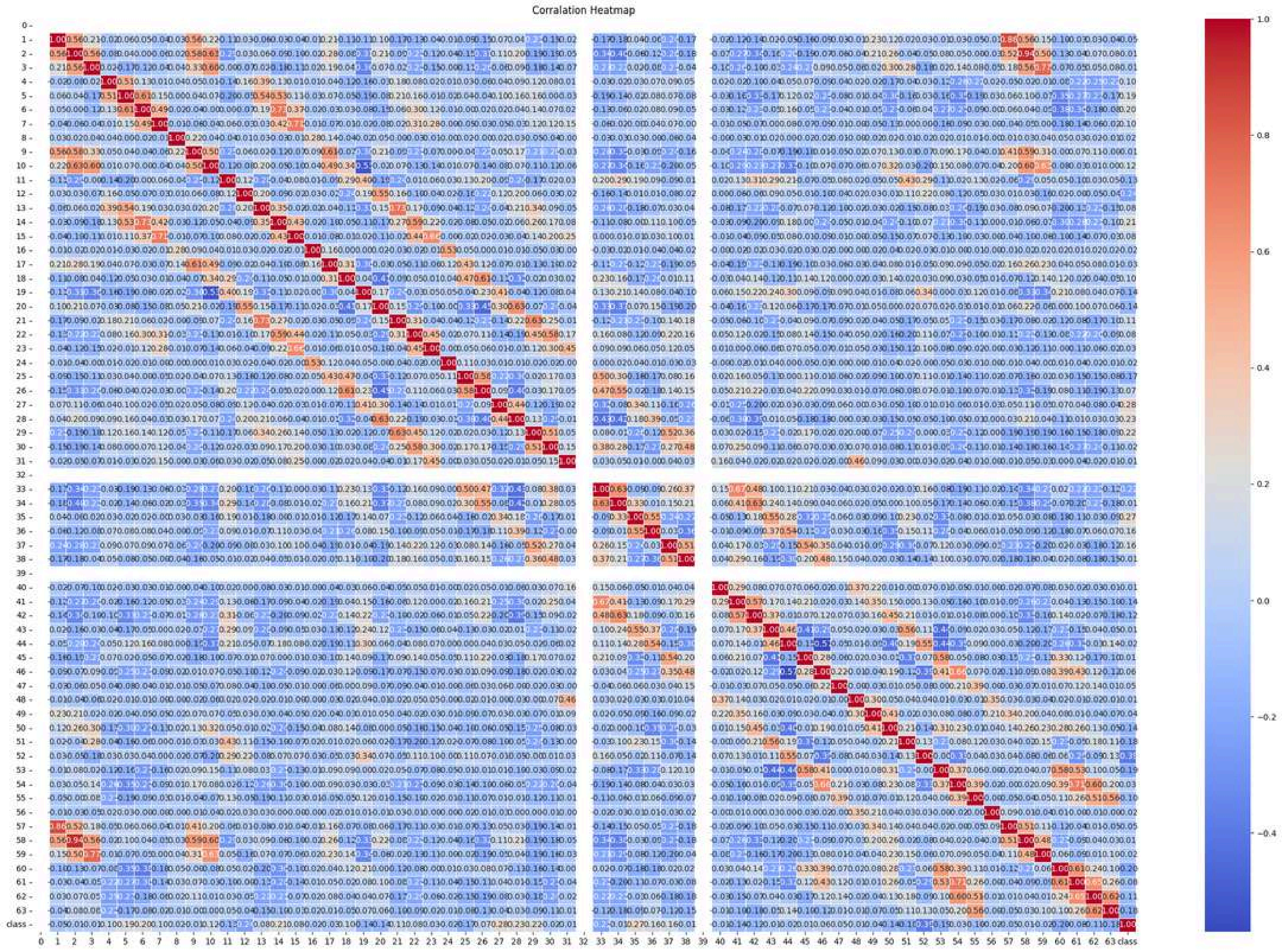
By Ni Gusti Ayu Agung Indraswari

Exploration Data Analysis (EDA)

Class Distribution



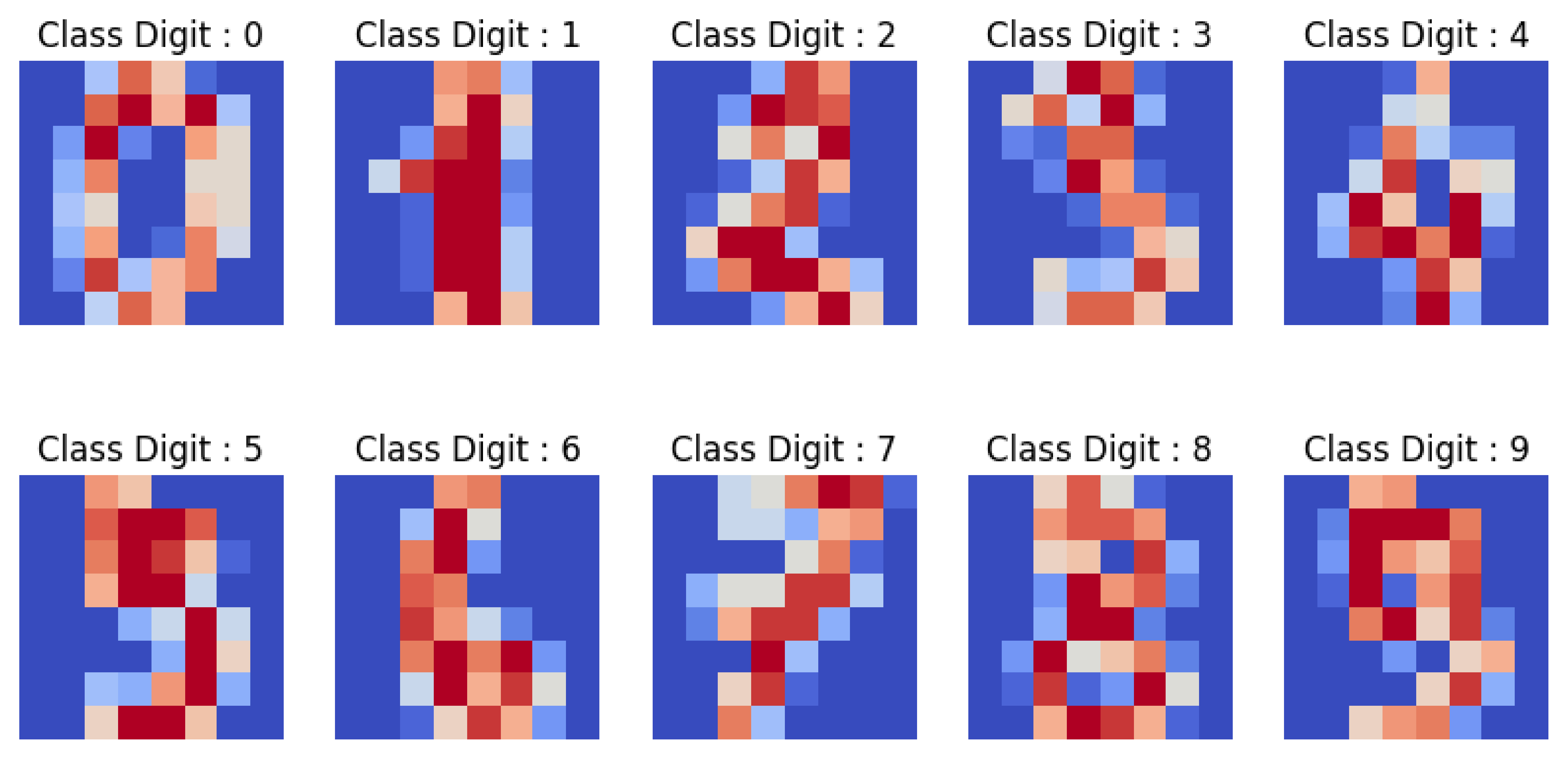
Correlation between feature



By Ni Gusti Ayu Agung Indraswari

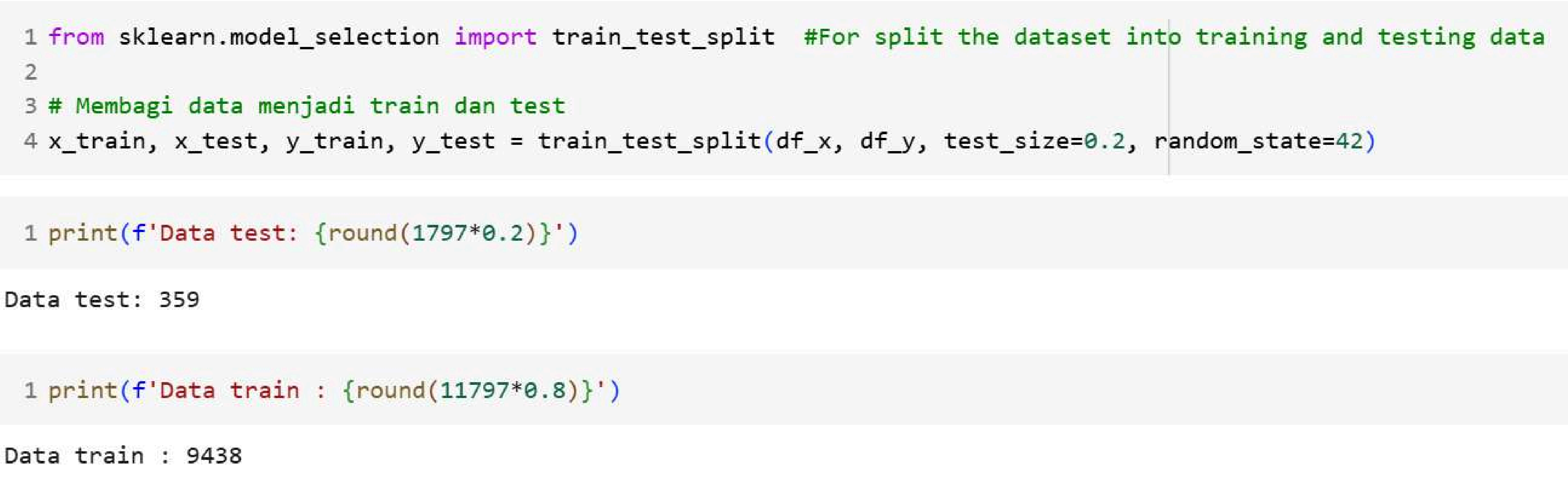
Exploration Data Analysis (EDA)

Visualize some example digit images



Data Preprocessing

Data Spliting



```
from sklearn.model_selection import train_test_split  #For split the dataset into training and testing data

# Membagi data menjadi train dan test
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.2, random_state=42)
```

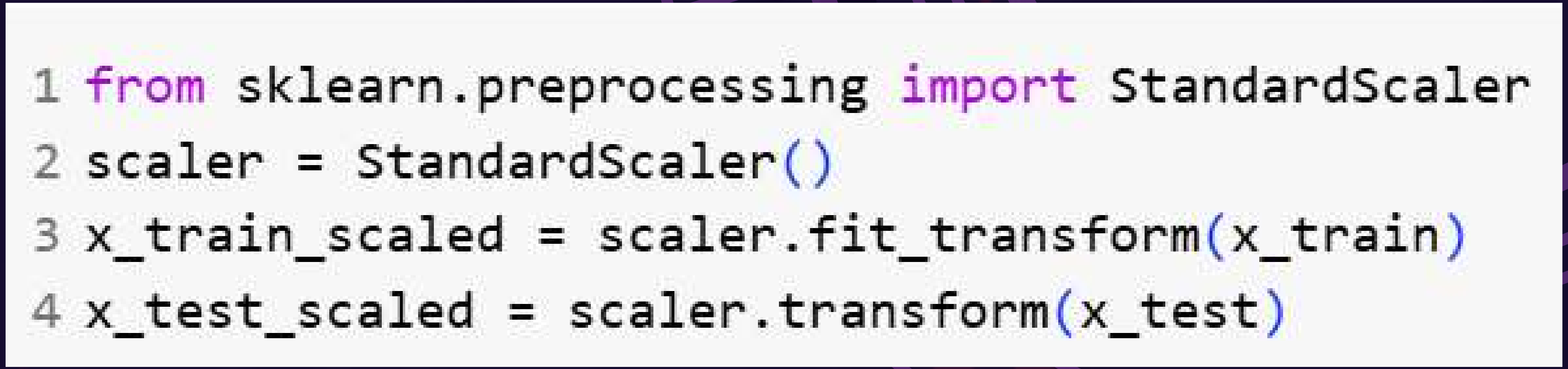
```
print(f'Data test: {round(1797*0.2)}')
```

```
Data test: 359
```

```
print(f'Data train : {round(11797*0.8)}')
```

```
Data train : 9438
```

Normalization / Scaling



```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

By Ni Gusti Ayu Agung Indraswari

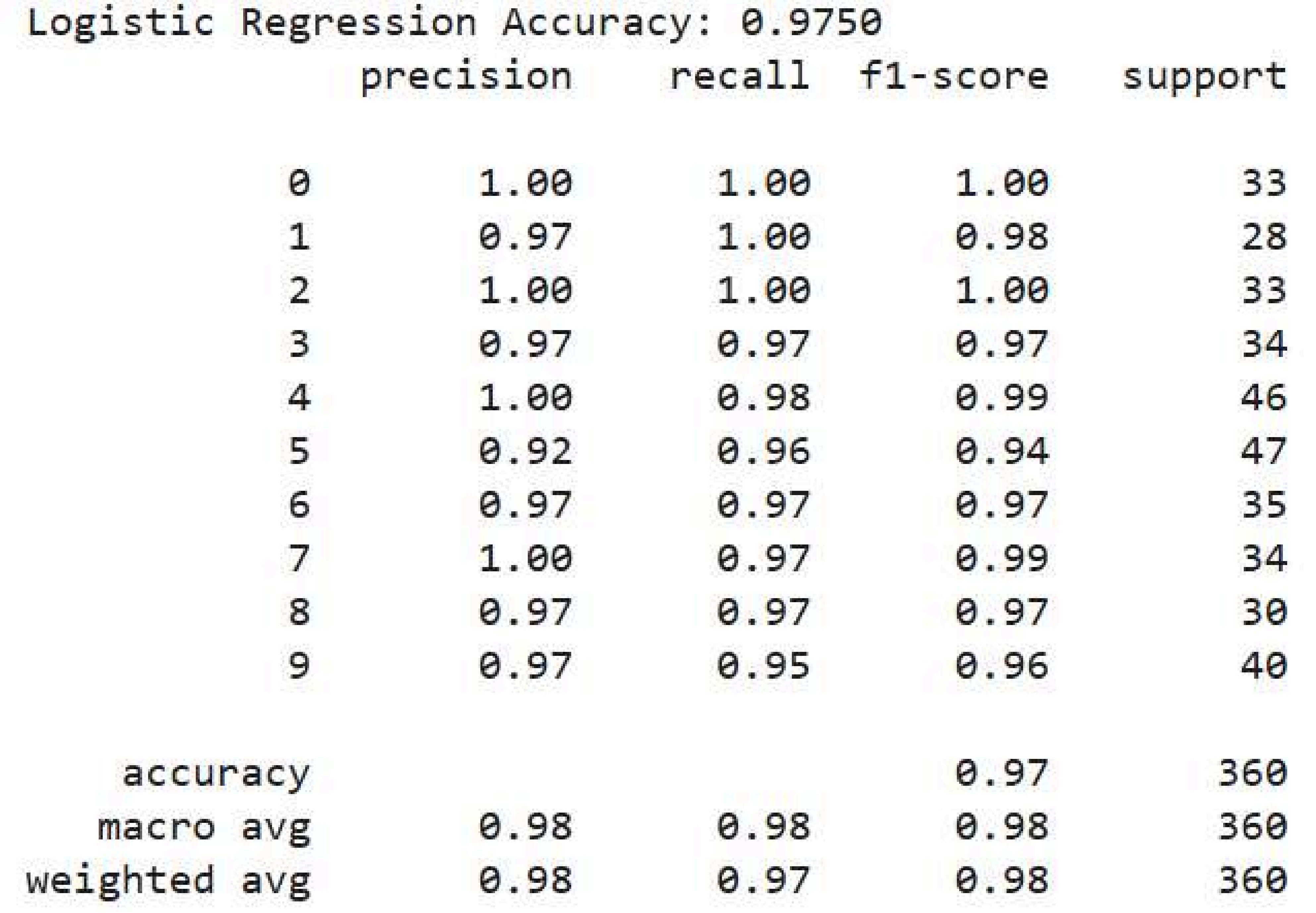
Machine Learning Models

1. Logistic Regression
2. K-Nearest Neigbors (KNN)
3. Support Vector Machine (SVM)
4. Random Forest
5. Neural Network

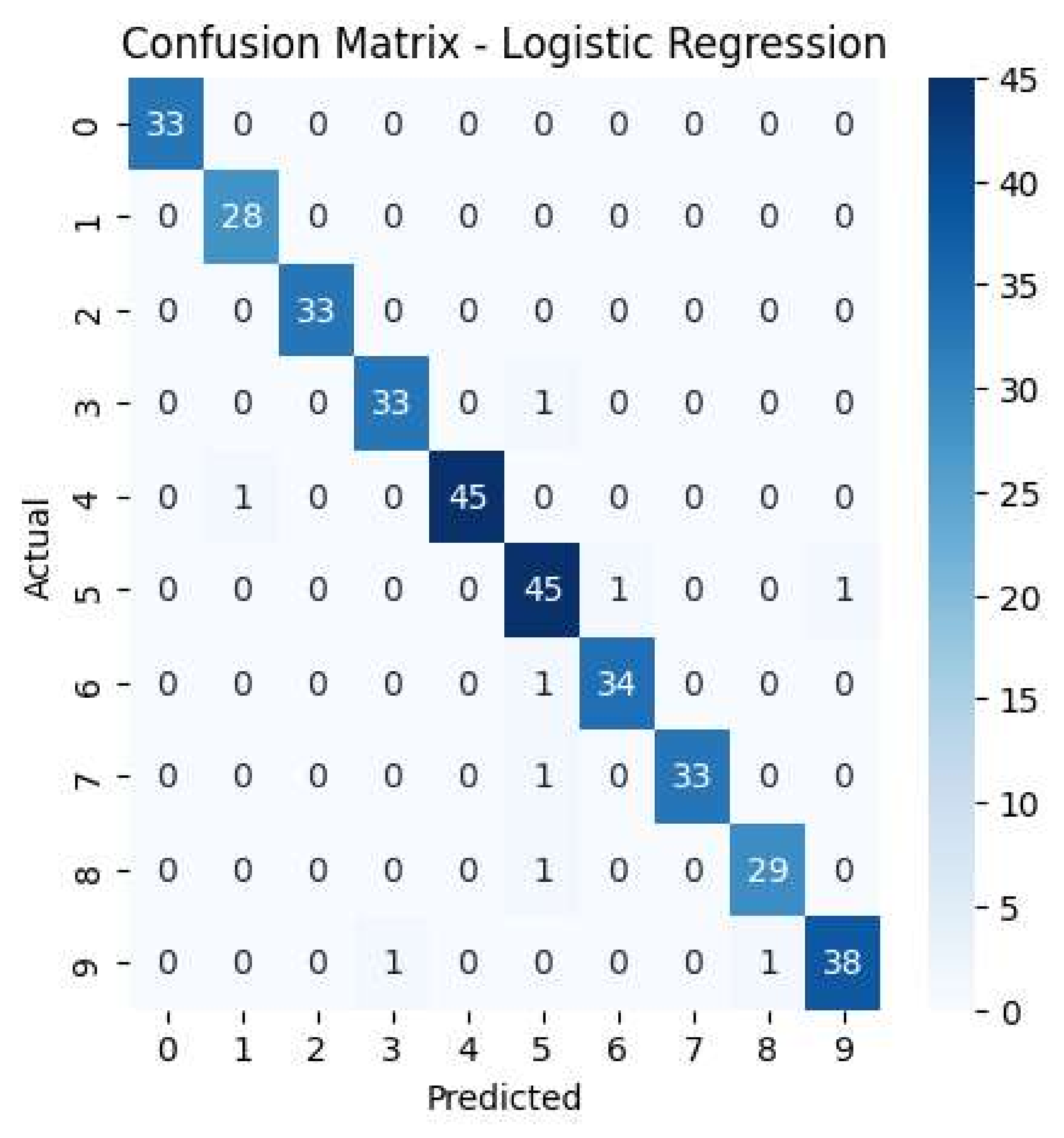
By Ni Gusti Ayu Agung Indraswari

Train and Validation Models

Logistic Regression



```
Logistic Regression Accuracy: 0.9750
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.97      1.00      0.98        28
           2       1.00      1.00      1.00        33
           3       0.97      0.97      0.97        34
           4       1.00      0.98      0.99        46
           5       0.92      0.96      0.94        47
           6       0.97      0.97      0.97        35
           7       1.00      0.97      0.99        34
           8       0.97      0.97      0.97        30
           9       0.97      0.95      0.96        40

    accuracy                           0.97       360
   macro avg       0.98      0.98      0.98       360
weighted avg       0.98      0.97      0.98       360
```

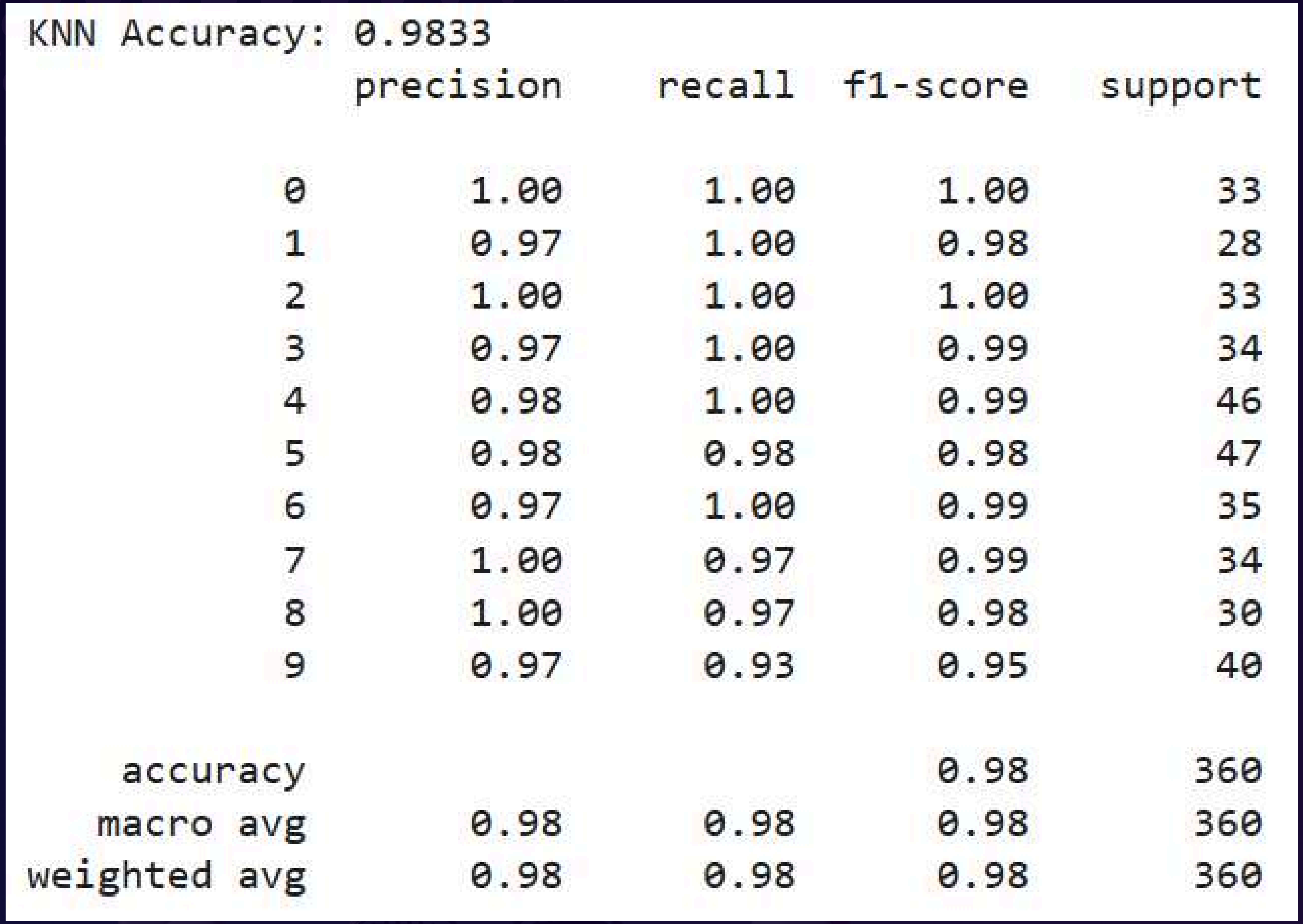


Confusion Matrix - Logistic Regression

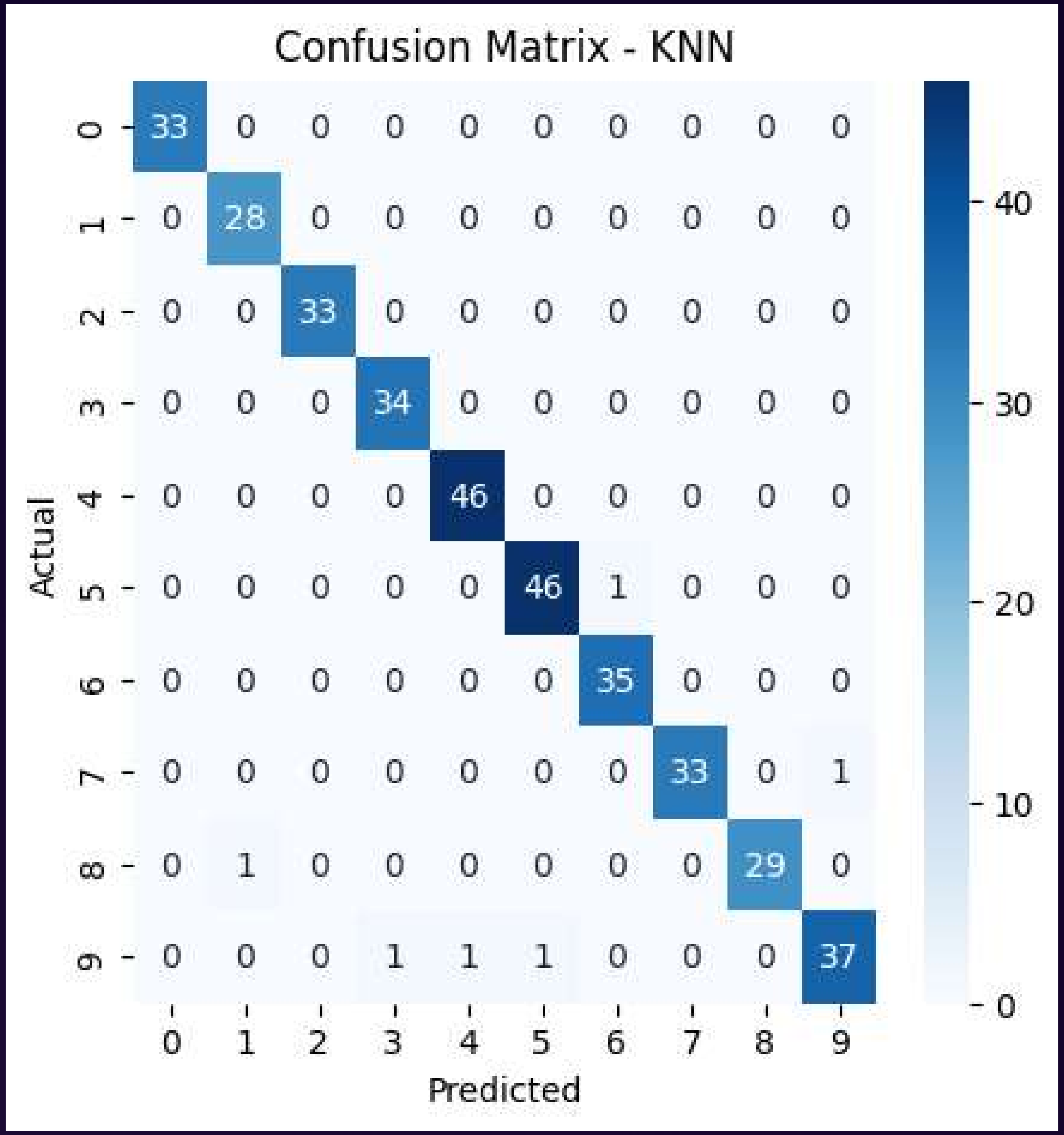
Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	33	0	0	0	0	0	0	0	0	0
1	0	28	0	0	0	0	0	0	0	0
2	0	0	33	0	0	0	0	0	0	0
3	0	0	0	33	0	1	0	0	0	0
4	0	1	0	0	45	0	0	0	0	0
5	0	0	0	0	0	45	1	0	0	1
6	0	0	0	0	0	1	34	0	0	0
7	0	0	0	0	0	1	0	33	0	0
8	0	0	0	0	0	1	0	0	29	0
9	0	0	0	1	0	0	0	0	1	38

Train and Validation Models

K-Nearest Neigbors (KNN)



```
KNN Accuracy: 0.9833
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.97      1.00      0.98        28
           2       1.00      1.00      1.00        33
           3       0.97      1.00      0.99        34
           4       0.98      1.00      0.99        46
           5       0.98      0.98      0.98        47
           6       0.97      1.00      0.99        35
           7       1.00      0.97      0.99        34
           8       1.00      0.97      0.98        30
           9       0.97      0.93      0.95        40

    accuracy                           0.98       360
   macro avg       0.98      0.98      0.98       360
weighted avg       0.98      0.98      0.98       360
```



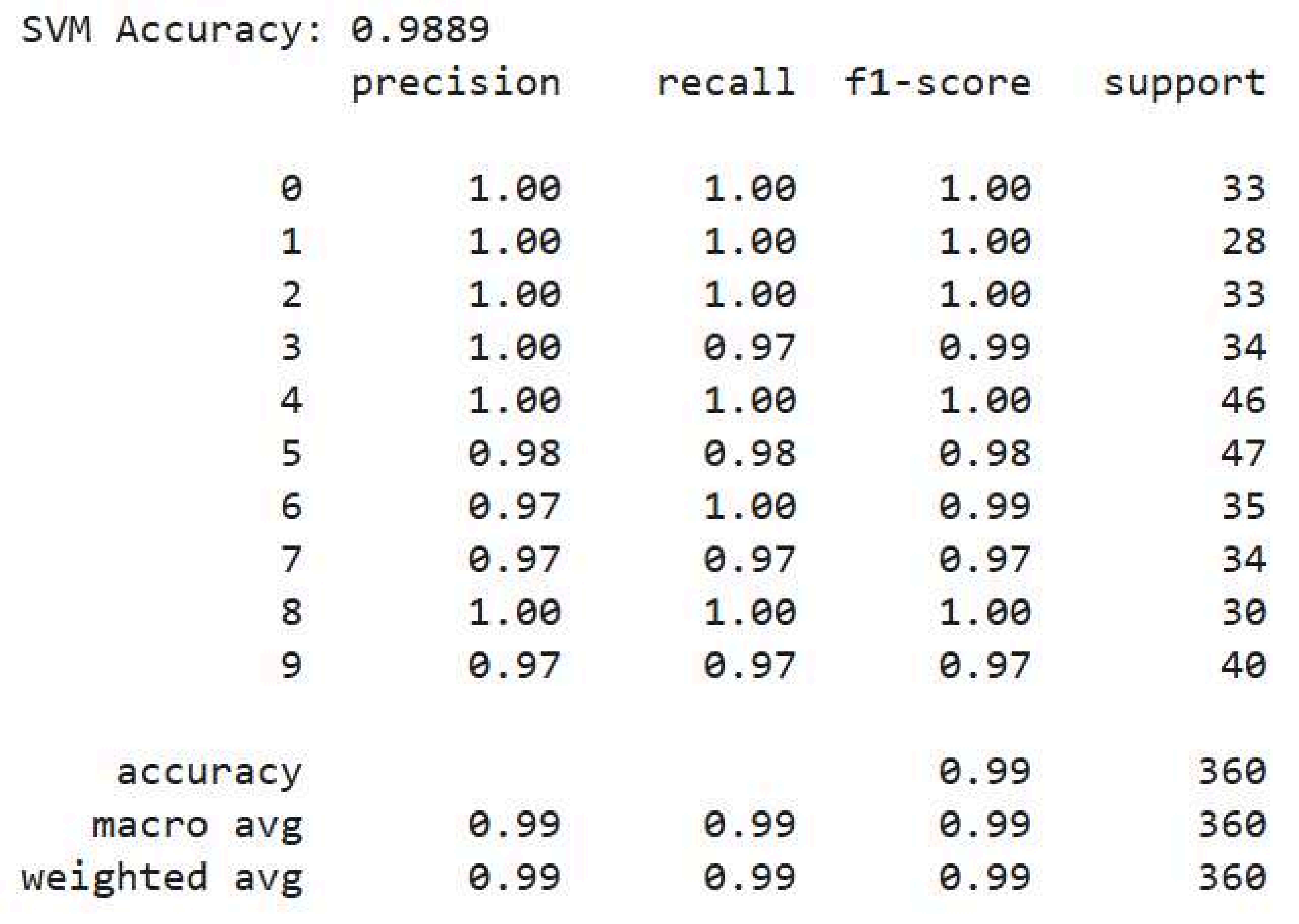
Confusion Matrix - KNN

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	33	0	0	0	0	0	0	0	0	0
1	0	28	0	0	0	0	0	0	0	0
2	0	0	33	0	0	0	0	0	0	0
3	0	0	0	34	0	0	0	0	0	0
4	0	0	0	0	46	0	0	0	0	0
5	0	0	0	0	0	46	1	0	0	0
6	0	0	0	0	0	0	35	0	0	0
7	0	0	0	0	0	0	0	33	0	1
8	0	1	0	0	0	0	0	0	29	0
9	0	0	0	1	1	1	0	0	0	37

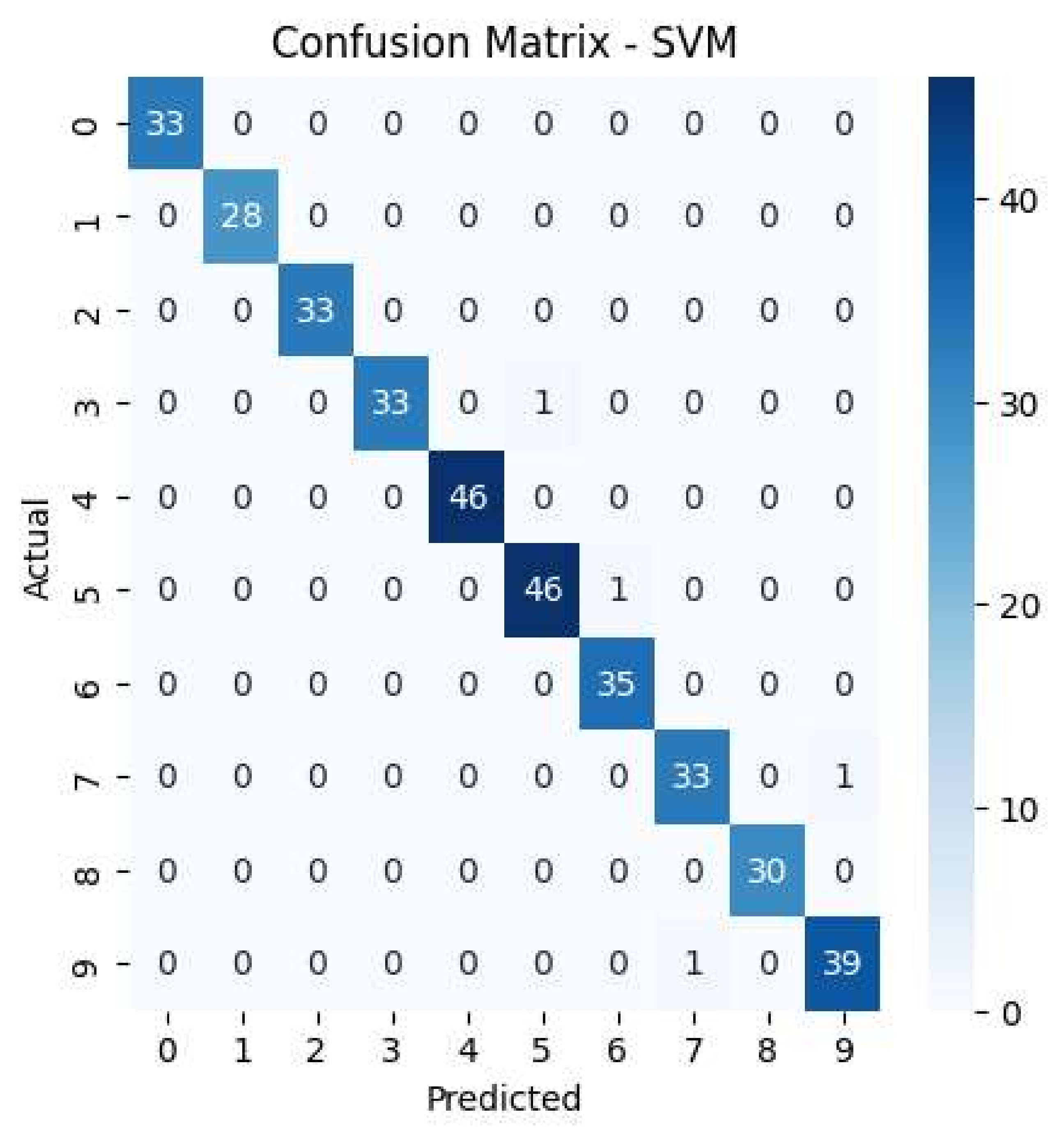
By Ni Gusti Ayu Agung Indraswari

Train and Validation Models

Support Vector Machine (SVM)



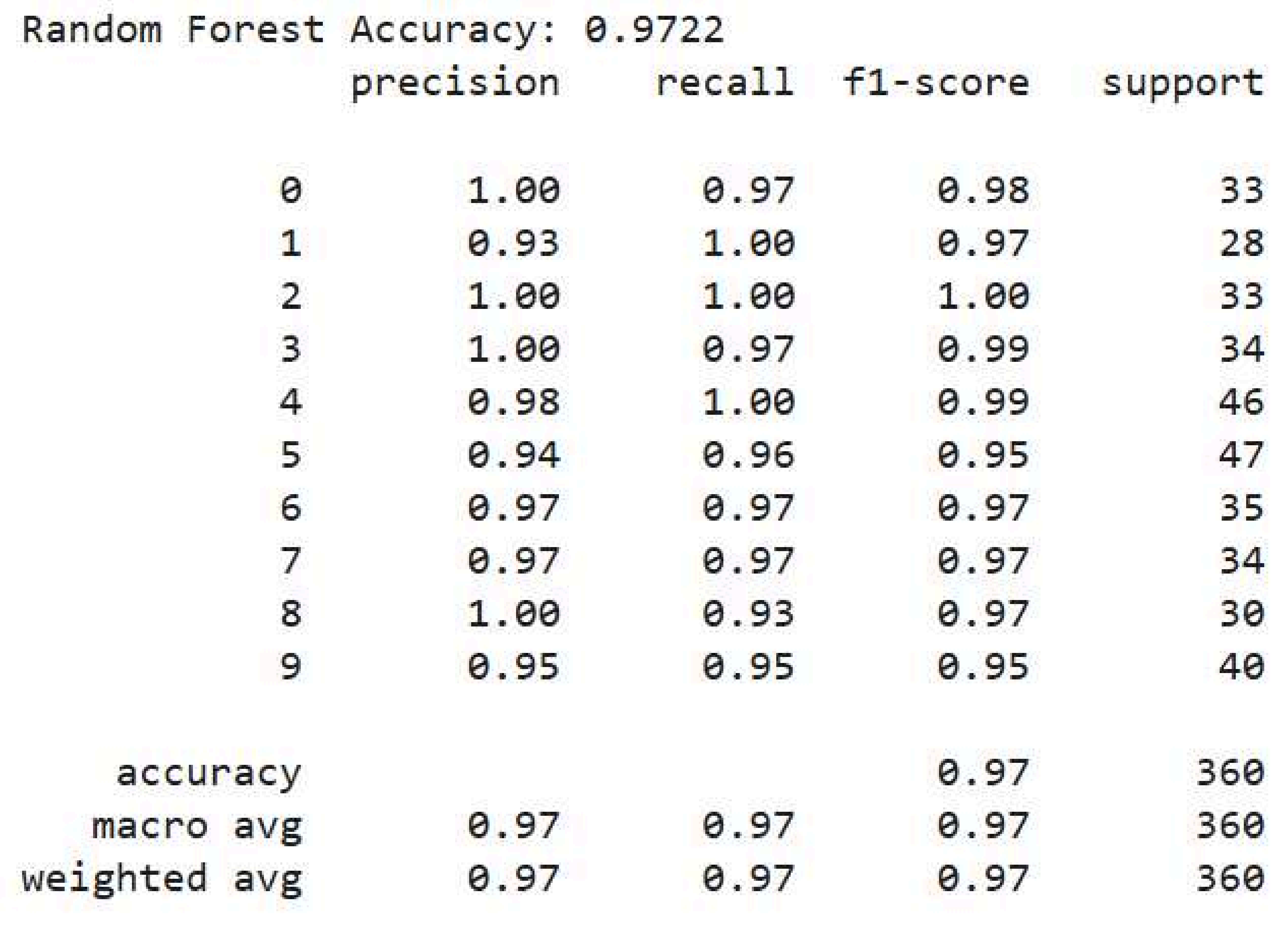
```
SVM Accuracy: 0.9889
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       1.00      1.00      1.00        28
           2       1.00      1.00      1.00        33
           3       1.00      0.97      0.99        34
           4       1.00      1.00      1.00        46
           5       0.98      0.98      0.98        47
           6       0.97      1.00      0.99        35
           7       0.97      0.97      0.97        34
           8       1.00      1.00      1.00        30
           9       0.97      0.97      0.97        40

    accuracy                           0.99       360
   macro avg       0.99      0.99      0.99       360
weighted avg       0.99      0.99      0.99       360
```



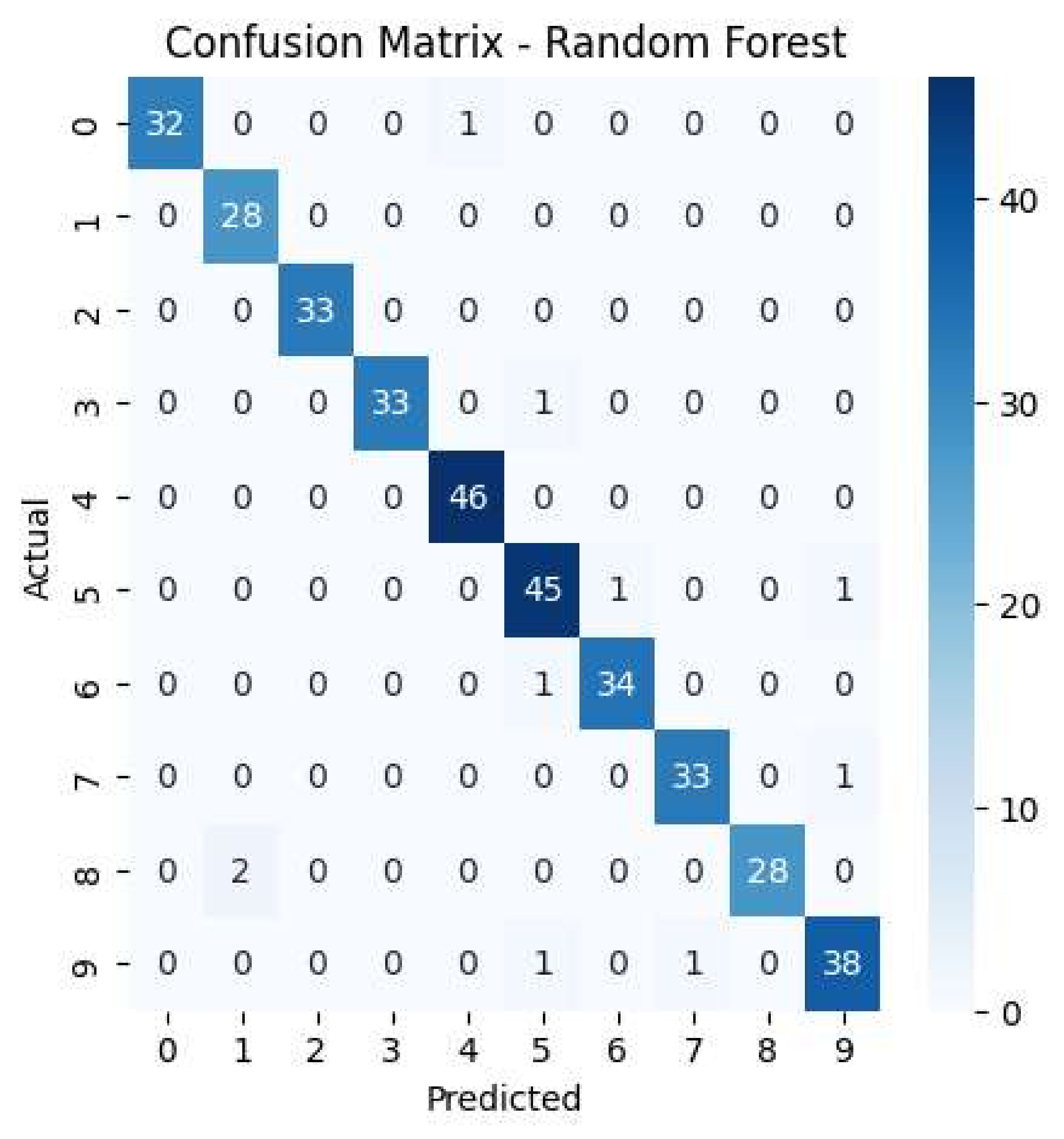
Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	33	0	0	0	0	0	0	0	0	0
1	0	28	0	0	0	0	0	0	0	0
2	0	0	33	0	0	0	0	0	0	0
3	0	0	0	33	0	1	0	0	0	0
4	0	0	0	0	46	0	0	0	0	0
5	0	0	0	0	0	46	1	0	0	0
6	0	0	0	0	0	0	35	0	0	0
7	0	0	0	0	0	0	0	33	0	1
8	0	0	0	0	0	0	0	0	30	0
9	0	0	0	0	0	0	0	1	0	39

Train and Validation Models

Random Forest



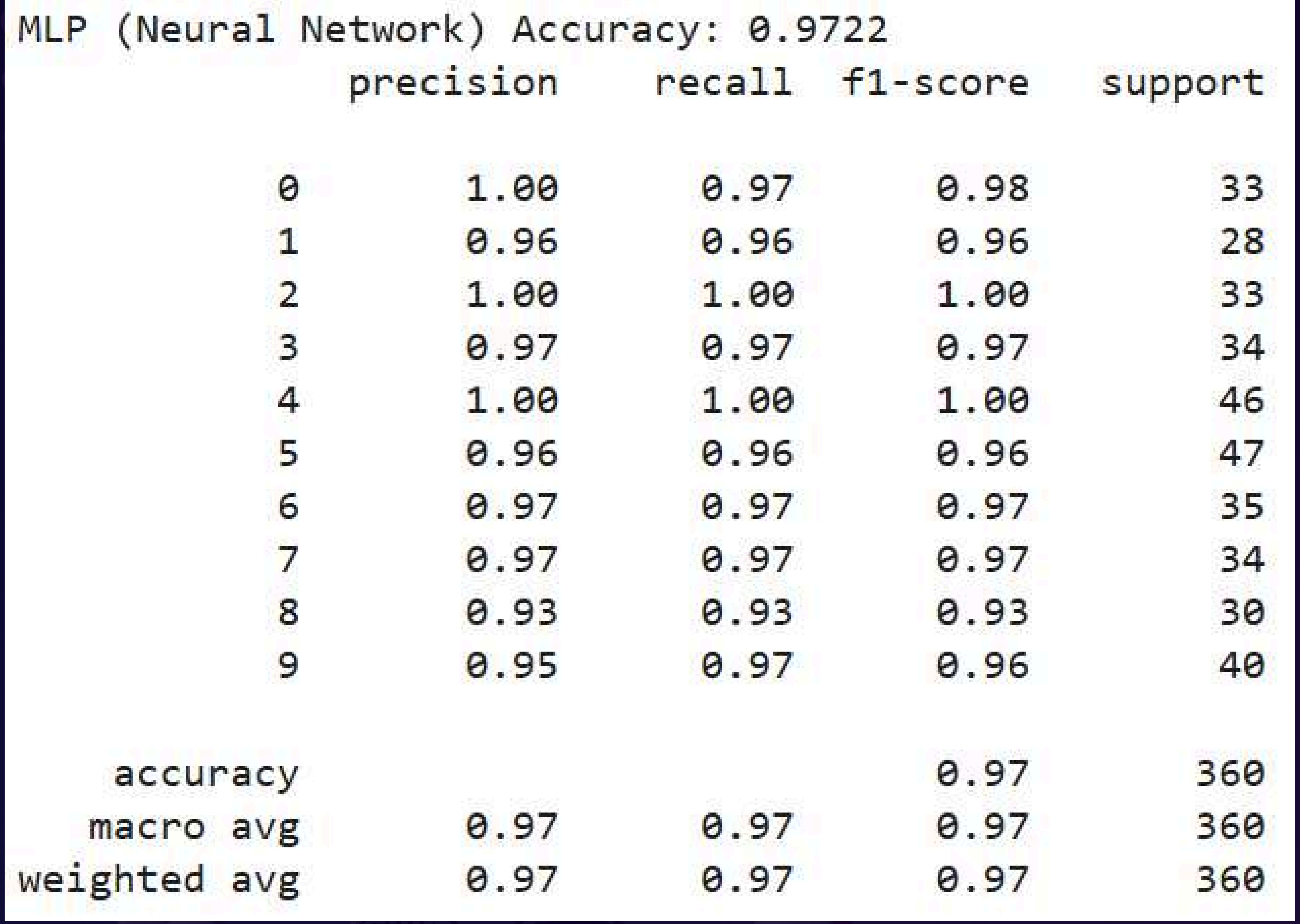
```
Random Forest Accuracy: 0.9722
              precision    recall  f1-score   support

           0       1.00      0.97      0.98        33
           1       0.93      1.00      0.97        28
           2       1.00      1.00      1.00        33
           3       1.00      0.97      0.99        34
           4       0.98      1.00      0.99        46
           5       0.94      0.96      0.95        47
           6       0.97      0.97      0.97        35
           7       0.97      0.97      0.97        34
           8       1.00      0.93      0.97        30
           9       0.95      0.95      0.95        40

    accuracy                           0.97       360
   macro avg       0.97      0.97      0.97       360
weighted avg       0.97      0.97      0.97       360
```



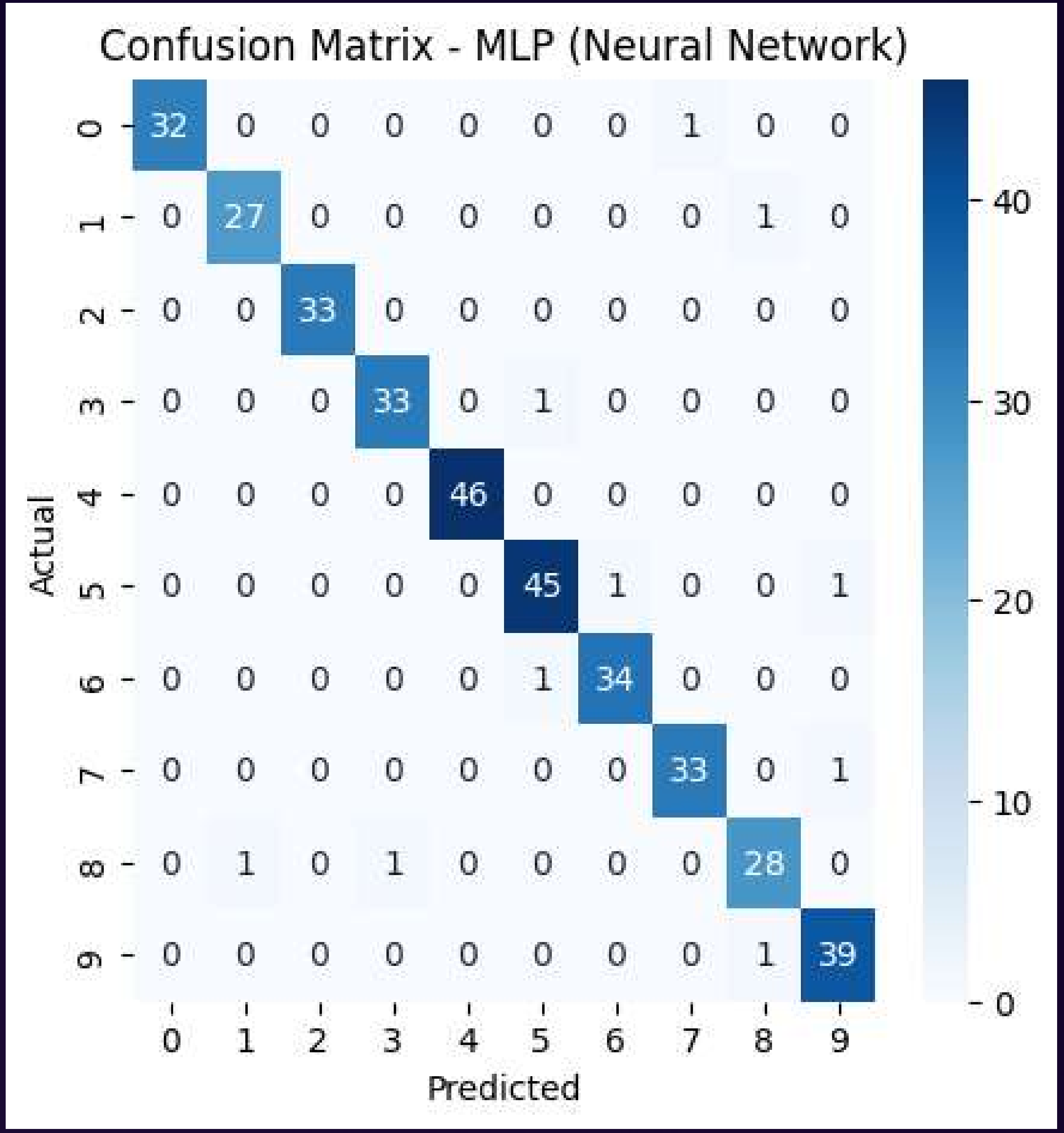
Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	32	0	0	0	1	0	0	0	0	0
1	0	28	0	0	0	0	0	0	0	0
2	0	0	33	0	0	0	0	0	0	0
3	0	0	0	33	0	1	0	0	0	0
4	0	0	0	0	46	0	0	0	0	0
5	0	0	0	0	0	45	1	0	0	1
6	0	0	0	0	0	1	34	0	0	0
7	0	0	0	0	0	0	0	33	0	1
8	0	2	0	0	0	0	0	0	28	0
9	0	0	0	0	0	1	0	1	0	38

Train and Validation Models

Neural Network



```
MLP (Neural Network) Accuracy: 0.9722
              precision    recall  f1-score   support

           0       1.00      0.97      0.98        33
           1       0.96      0.96      0.96        28
           2       1.00      1.00      1.00        33
           3       0.97      0.97      0.97        34
           4       1.00      1.00      1.00        46
           5       0.96      0.96      0.96        47
           6       0.97      0.97      0.97        35
           7       0.97      0.97      0.97        34
           8       0.93      0.93      0.93        30
           9       0.95      0.97      0.96        40

    accuracy                           0.97       360
   macro avg       0.97      0.97      0.97       360
weighted avg       0.97      0.97      0.97       360
```



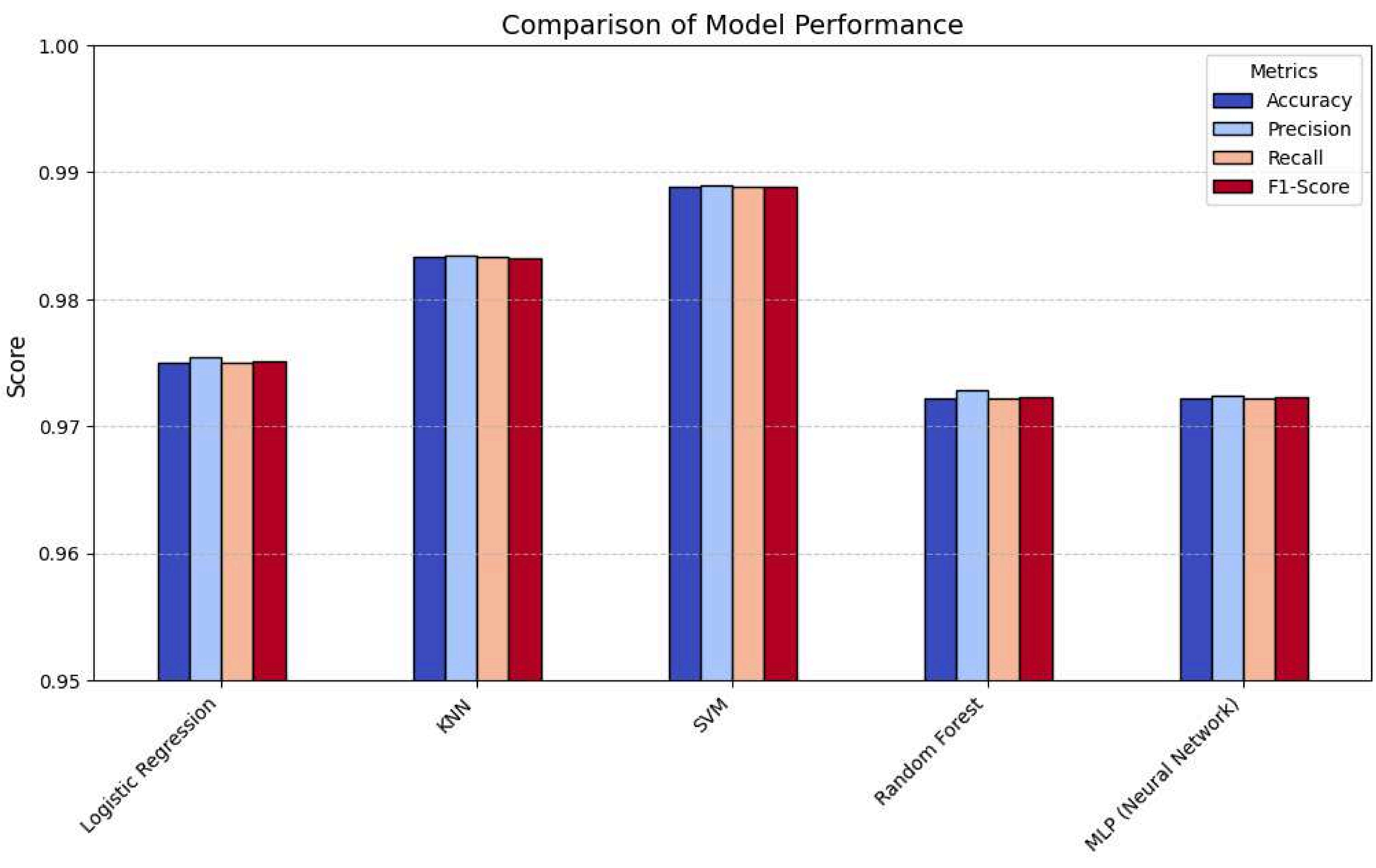
Confusion Matrix - MLP (Neural Network)

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	32	0	0	0	0	0	0	1	0	0
1	0	27	0	0	0	0	0	0	1	0
2	0	0	33	0	0	0	0	0	0	0
3	0	0	0	33	0	1	0	0	0	0
4	0	0	0	0	46	0	0	0	0	0
5	0	0	0	0	0	45	1	0	0	1
6	0	0	0	0	0	1	34	0	0	0
7	0	0	0	0	0	0	0	33	0	1
8	0	1	0	1	0	0	0	0	28	0
9	0	0	0	0	0	0	0	0	1	39

By Ni Gusti Ayu Agung Indraswari

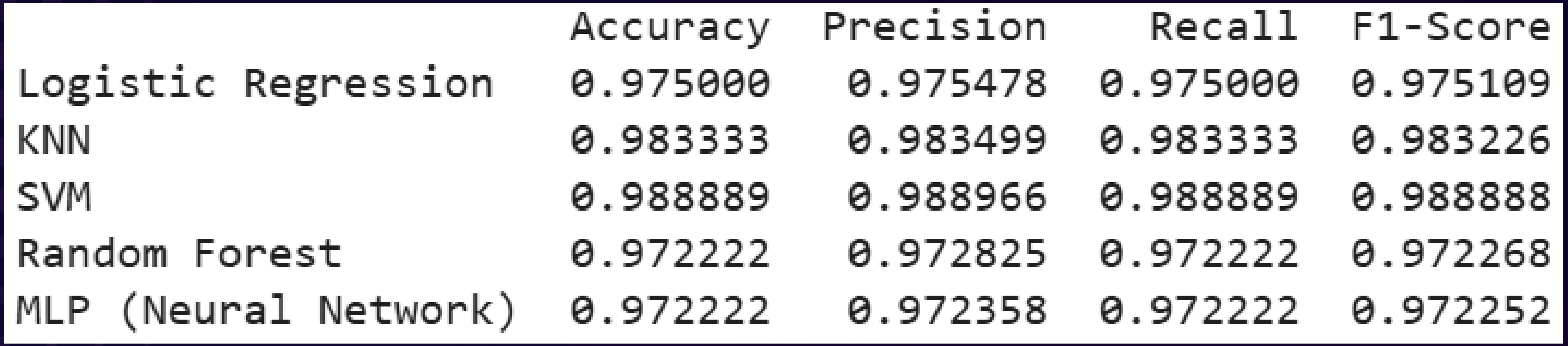
Page 15/18

Comparison Models



By Ni Gusti Ayu Agung Indraswari

Comparison Models



```
                       Accuracy  Precision    Recall  F1-Score
Logistic Regression    0.975000   0.975478  0.975000  0.975109
KNN                    0.983333   0.983499  0.983333  0.983226
SVM                    0.988889   0.988966  0.988889  0.988888
Random Forest          0.972222   0.972825  0.972222  0.972268
MLP (Neural Network)   0.972222   0.972358  0.972222  0.972252
```

Based on the bar chart comparing accuracy, precision, recall, and F1-score and data, SVM outperforms all other models in every metric.

- **Indications that SVM is the best model:**
  - **Highest accuracy among all models.**
  - **Superior precision, recall, and F1-score, ensuring balanced predictions.**
  - **Consistently strong performance, effectively identifying all classes.**

Result and Conclusion

The Support Vector Machine (SVM) model proved to be the best for classifying handwritten digits in the Digits dataset, achieving an accuracy of 98.89%. This result demonstrates SVM's high performance and strong generalization ability on test data. With this level of accuracy, the model is highly suitable for applications in automated character recognition, such as document processing and digital identity verification.

By Ni Gusti Ayu Agung Indraswari

Thank You!

Feel free to reach out if you'd like to collaborate, discuss ideas, or explore exciting opportunities.

@ag.swarii

Github

Linkedin

agungindraswari03@gmail.com