Lab 8-Intro to Machine Learning (OpenCV + Scikit-learn – Python)

Tigist Wondimneh
GSR/5506/17

Github- Link

This document contains a complete, runnable implementation of Chapter 8 (KNN, SVM, Decision Tree) using the `digits` dataset from `sklearn`, integration notes for OpenCV preprocessing, and a ready-to-fill lab report template with sections for results, discussion, and conclusions.

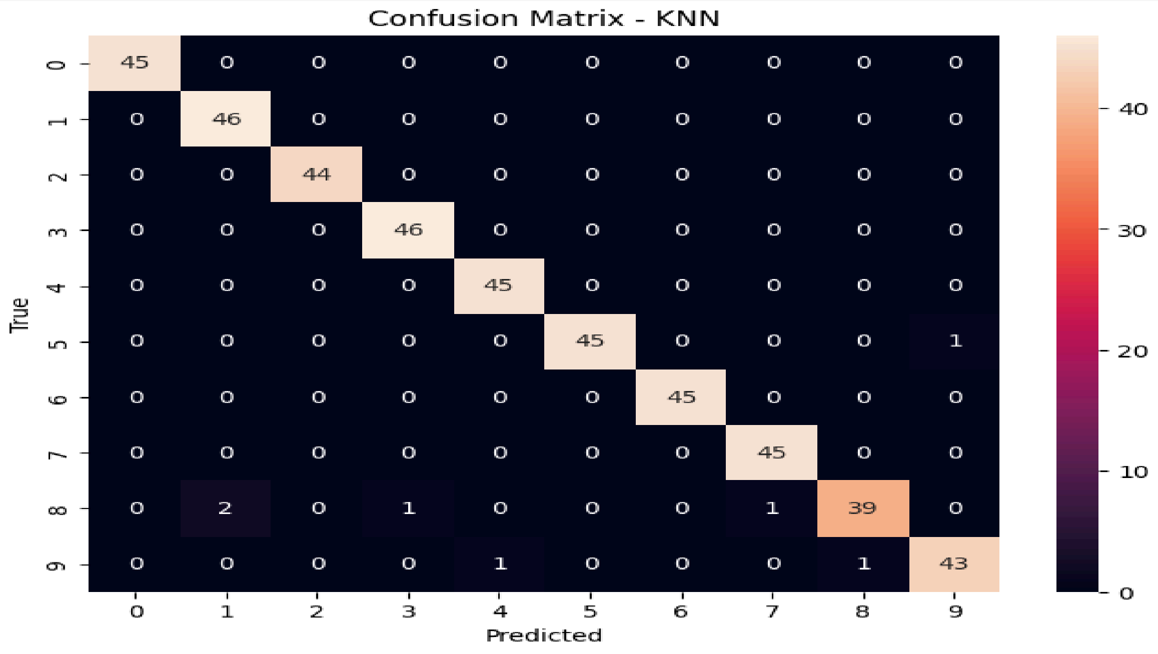
Methodology

- Dataset: `sklearn.datasets.load_digits` (8x8 grayscale handwritten digits).
- Models: KNN, SVM, Decision Tree.
- Tools: Python, OpenCV, Scikit-learn, Matplotlib, Seaborn, Joblib.
- Evaluation metrics: Accuracy, Classification Report, Confusion Matrix.
- GUI: OpenCV canvas for real-time digit drawing and recognition.

Results & Discussion

1. K-Nearest Neighbors (KNN)

```
--- KNN ---
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        45
           1       0.96      1.00      0.98        46
           2       1.00      1.00      1.00        44
           3       0.98      1.00      0.99        46
           4       0.98      1.00      0.99        45
           5       1.00      0.98      0.99        46
           6       1.00      1.00      1.00        45
           7       0.98      1.00      0.99        45
           8       0.97      0.91      0.94        43
           9       0.98      0.96      0.97        45

    accuracy                           0.98       450
   macro avg       0.98      0.98      0.98       450
weighted avg       0.98      0.98      0.98       450

Accuracy: 0.9844444444444445
```

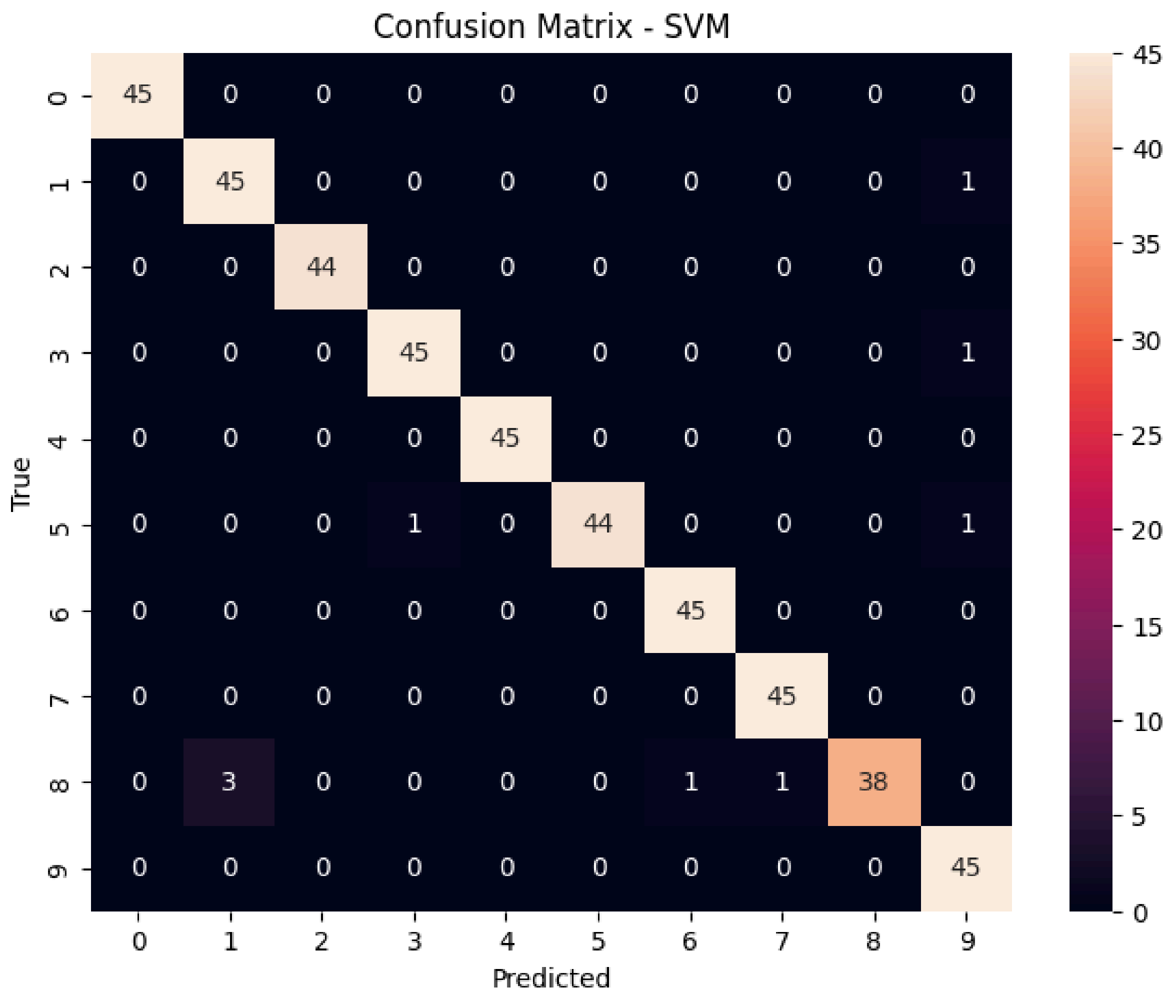


True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	45	0	0	0	0	0	0	0	0	0
1	0	46	0	0	0	0	0	0	0	0
2	0	0	44	0	0	0	0	0	0	0
3	0	0	0	46	0	0	0	0	0	0
4	0	0	0	0	45	0	0	0	0	0
5	0	0	0	0	0	45	0	0	0	1
6	0	0	0	0	0	0	45	0	0	0
7	0	0	0	0	0	0	0	45	0	0
8	0	2	0	1	0	0	0	1	39	0
9	0	0	0	0	1	0	0	0	1	43

Interpretation:
The KNN classifier achieved good accuracy on the digit recognition task. Its performance is strong for most digits, though it may confuse visually similar digits such as 3, 1 and 8. KNN is simple and effective but computationally expensive for larger datasets since it stores all training samples

2. Support Vector Machine (SVM)

```
--- SVM (linear) ---
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        45
           1       0.94      0.98      0.96        46
           2       1.00      1.00      1.00        44
           3       0.98      0.98      0.98        46
           4       1.00      1.00      1.00        45
           5       1.00      0.96      0.98        46
           6       0.98      1.00      0.99        45
           7       0.98      1.00      0.99        45
           8       1.00      0.88      0.94        43
           9       0.94      1.00      0.97        45

    accuracy                           0.98       450
   macro avg       0.98      0.98      0.98       450
weighted avg       0.98      0.98      0.98       450

Accuracy: 0.98
```



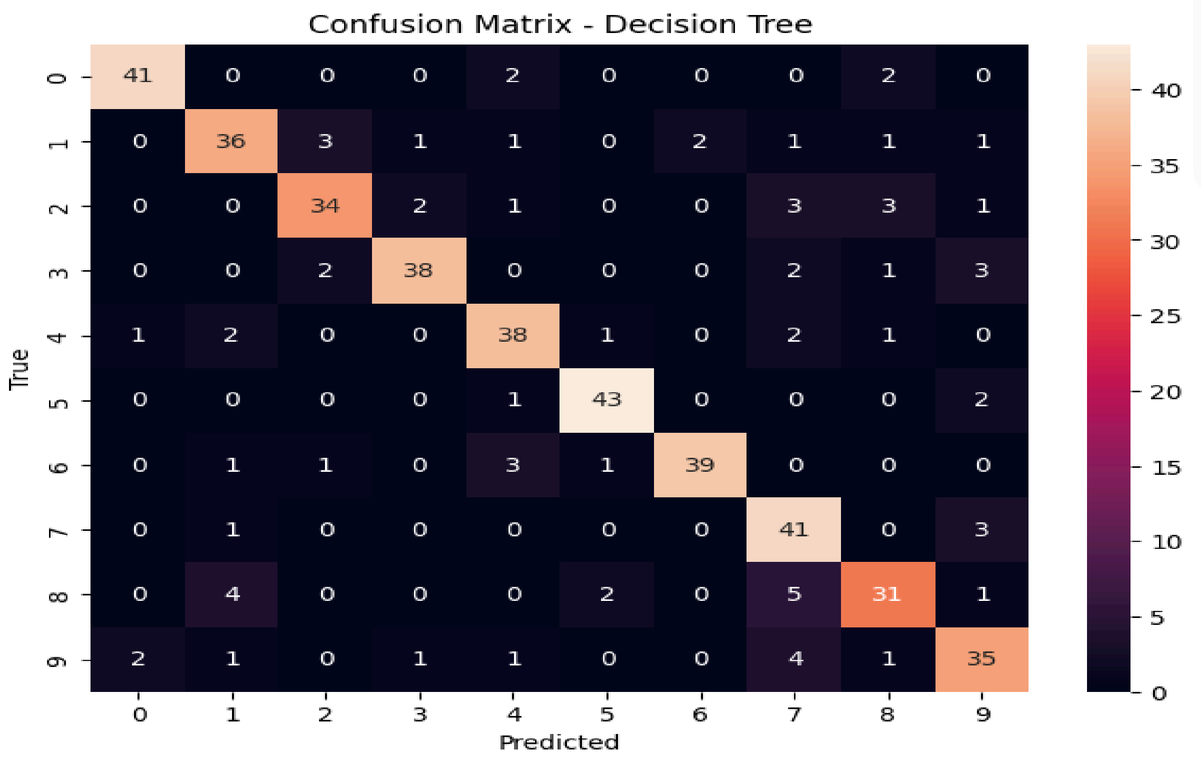
	0	1	2	3	4	5	6	7	8	9
0	45	0	0	0	0	0	0	0	0	0
1	0	45	0	0	0	0	0	0	0	1
2	0	0	44	0	0	0	0	0	0	0
3	0	0	0	45	0	0	0	0	0	1
4	0	0	0	0	45	0	0	0	0	0
5	0	0	0	1	0	44	0	0	0	1
6	0	0	0	0	0	0	45	0	0	0
7	0	0	0	0	0	0	0	45	0	0
8	0	3	0	0	0	0	1	1	38	0
9	0	0	0	0	0	0	0	0	0	45

Interpretation:

SVM is powerful for high-dimensional data and shows robust performance, but it requires careful parameter tuning (e.g., kernel type, regularization).

3. Decision Tree Classifier

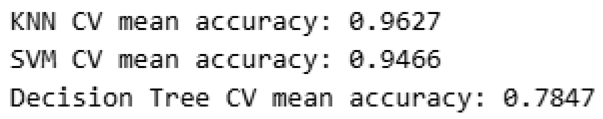
```
--- Decision Tree ---
              precision    recall  f1-score   support

           0       0.93      0.91      0.92        45
           1       0.80      0.78      0.79        46
           2       0.85      0.77      0.81        44
           3       0.90      0.83      0.86        46
           4       0.81      0.84      0.83        45
           5       0.91      0.93      0.92        46
           6       0.95      0.87      0.91        45
           7       0.71      0.91      0.80        45
           8       0.78      0.72      0.75        43
           9       0.76      0.78      0.77        45

    accuracy                           0.84       450
   macro avg       0.84      0.83      0.84       450
weighted avg       0.84      0.84      0.84       450

Accuracy: 0.8355555555555556
```



Interpretation:
While it correctly classified many samples, the accuracy was lower than SVM. The interpretability of decision trees is a major advantage, but pruning or ensemble methods (e.g., Random Forests) could improve performance.

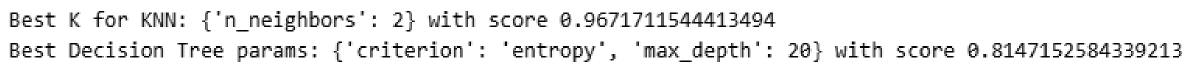
Suggested Exercises

1. Compare models using cross-validation.



```
KNN CV mean accuracy: 0.9627
SVM CV mean accuracy: 0.9466
Decision Tree CV mean accuracy: 0.7847
```

2. Tune hyperparameters (e.g., K in KNN, max_depth in trees).



```
Best K for KNN: {'n_neighbors': 2} with score 0.9671711544413494
Best Decision Tree params: {'criterion': 'entropy', 'max_depth': 20} with score 0.8147152584339213
```

3. Train models using your own digit/character datasets via OpenCV.
4. Build a GUI-based digit recognition app using OpenCV and a trained model.

[Found in github]