Ex: 2. Implement a Classifier using an Open Source dataset.

06.08.25.

6/8/25

Aim: To implement a supervised machine learning using an open source dataset.

Pseudo code:

```
1. Import necessary libraries.
   • pandas, scikit-learn, dataset, metrics, KNeighboursClassifier.

2. Load the dataset.
   • use datasets. load_iris (). (iris dataset)

3. Prepare the data
   • Assign features to x, target to y.

4. Split into training & testing sets :-
   • Use train_test_split (x, y, test_size 0.3,
     random. state = 42)

5. ~~Res~~ Instantiate the KNN classifier :
   • knn = KNeighboursClassifier(n-neighbours=3

6. Train the Model

7. Make Predictions
   • y-pred = knn. predict (x_test)

8. Evalute the classifier :-
   • calculate accuracy : metric - accuracy_
     score (y-test, y-pred
```

Observation: • The KNN classifier is trained on the iris dataset and tested w/ unseen data

• Output is displayed.

• Lowering 'k' can make the the model more sensitive, to ~~make~~ noise, while larger 'k' can smoother decisions boundaries.

Result: KNN ~~classified~~ r classifier was successfully implemented & tested using an open-source dataset.

OBSERVATION:

Accuracy: 1.0

Classification Report:

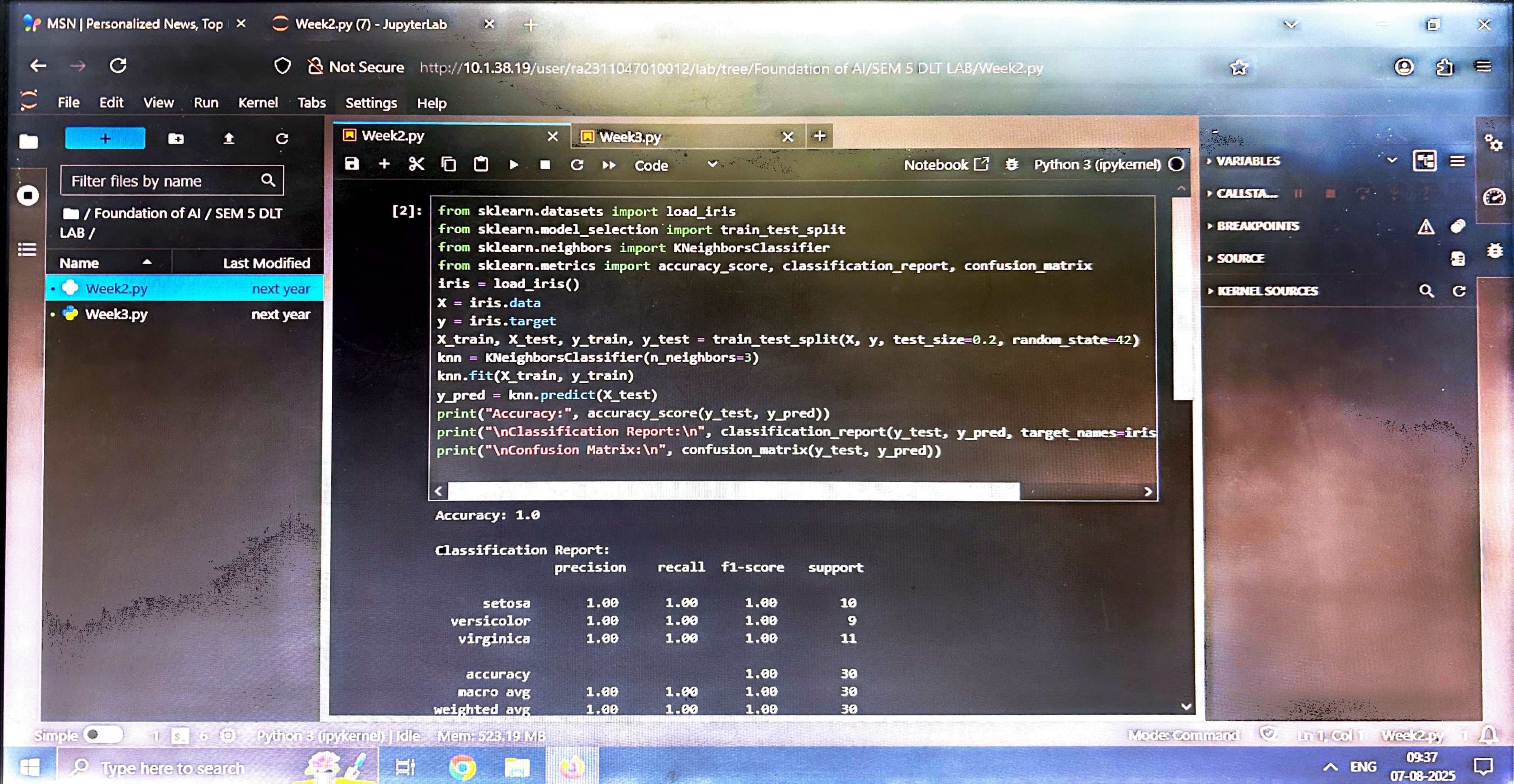
```
              ~~preci~~
              precision    recall    f1 score    support

setosa          1.00        1.00       1.00        10
versicolor      1.00        1.00       1.00         9
virginica       1.00        1.00       1.00        11

accuracy                               1.00        30
macro avg.      1.00        1.00       1.00        30
weighted avg.   1.00        1.00       1.00        30
```

Confusion Matrix:

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```



[2]:

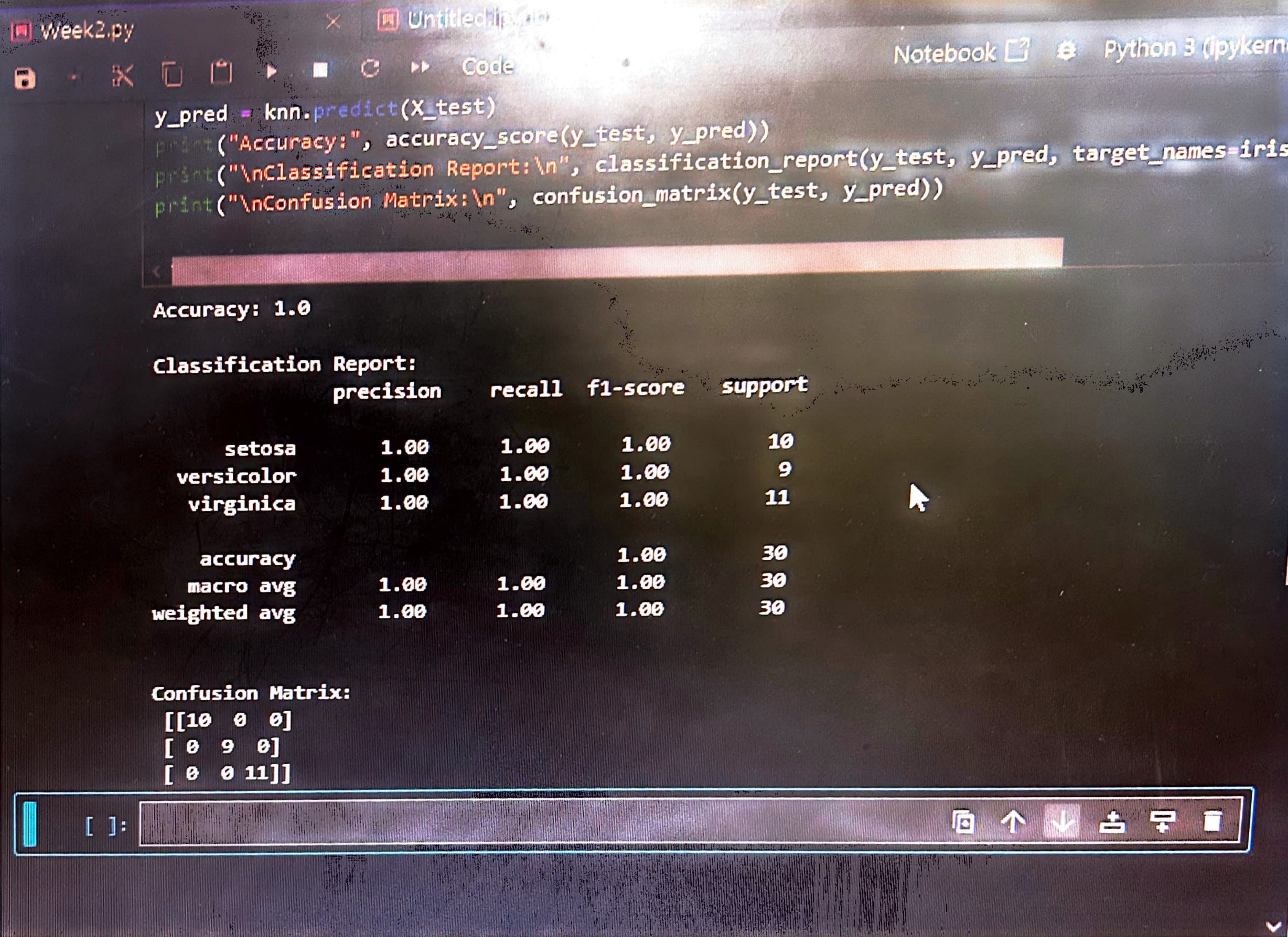
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred, target_names=iris
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 1.0

Classification Report:
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```



```
y_pred = knn.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred, target_names=iris
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 1.0

Classification Report:
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30


Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```