31/07/25

2. Implement a classifier using Open Source Code

Aim: Implementing a classifier using a Open source dataset

Objective:

1. Load and Explore iris dataset
2. Preprocess the data
3. Apply logistic regression for classification
4. Evaluate the model using accuracy & other classification metrics

Pytorch:-

Organization: Facebook AI Research [Fair] (2016)

Main features: Dynamic Computational graph
Native pythonic syntax
strong GPU accelerating Support

Popular use cases: Research and Academic
Projects NLP models.
fast model prototyping

Graph type : Dynamic

Result : successfully Explored DLT Platform.

Pseudo code

1. Import required libraries
   - Sklearn, Pandas, numpy, matplotlib.

2. Load iris dataset using Sklearn datasets

3. Explore iris dataset:
   - Features: sepal length, Sepal width, petal length, petal width
   - Target: 3 classes

4. Split data:
   Train & test split

5. Train logistic regression model on training data

6. Predict labels on test data

7. Evaluate performances:
   - Accuracy

Observation:

1. Dataset

- Iris dataset contains 150 samples, equally divided into 3 classes
- Each sample has 4 features

2. model performance.

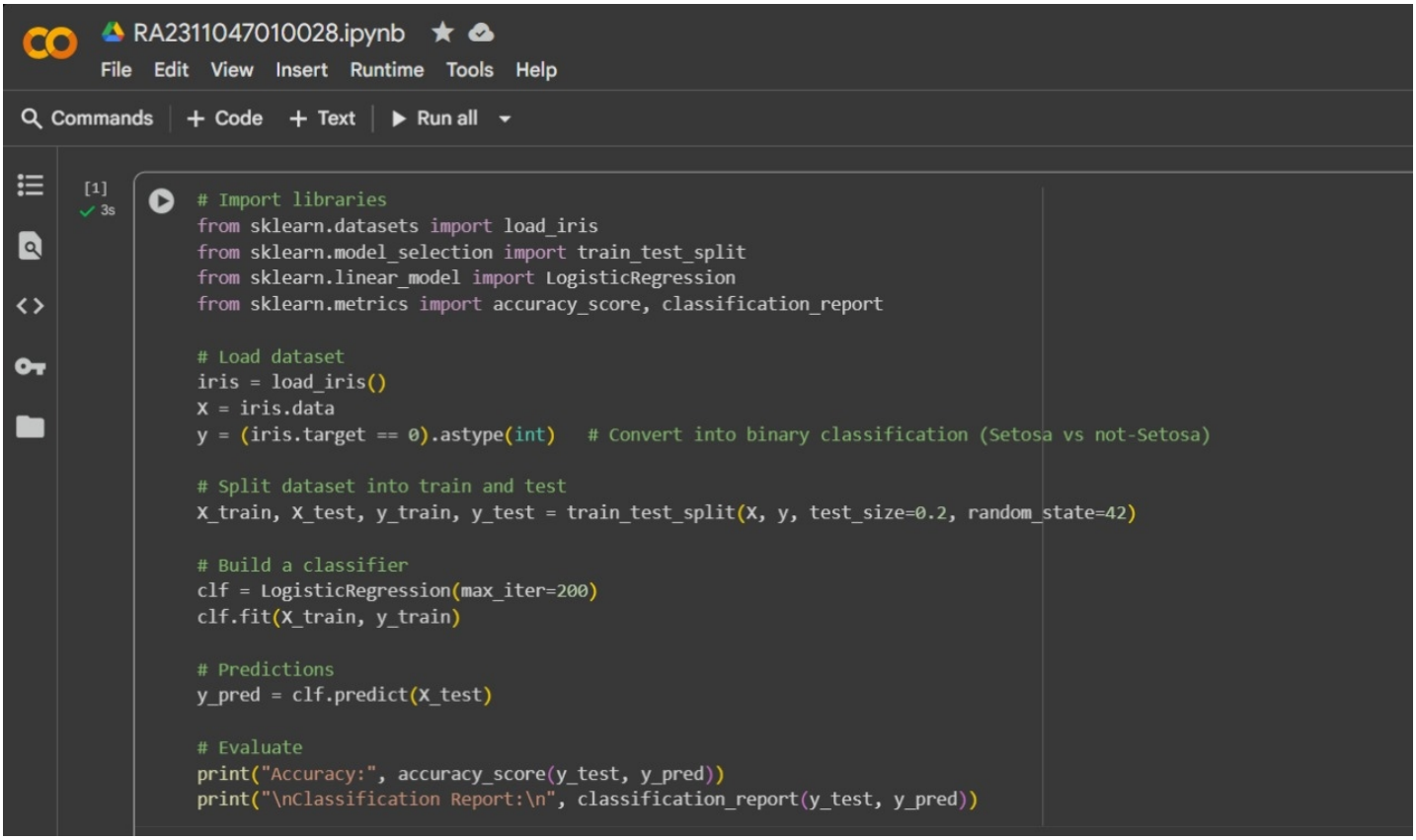
- Logistic regression achieved accuracy approximately.

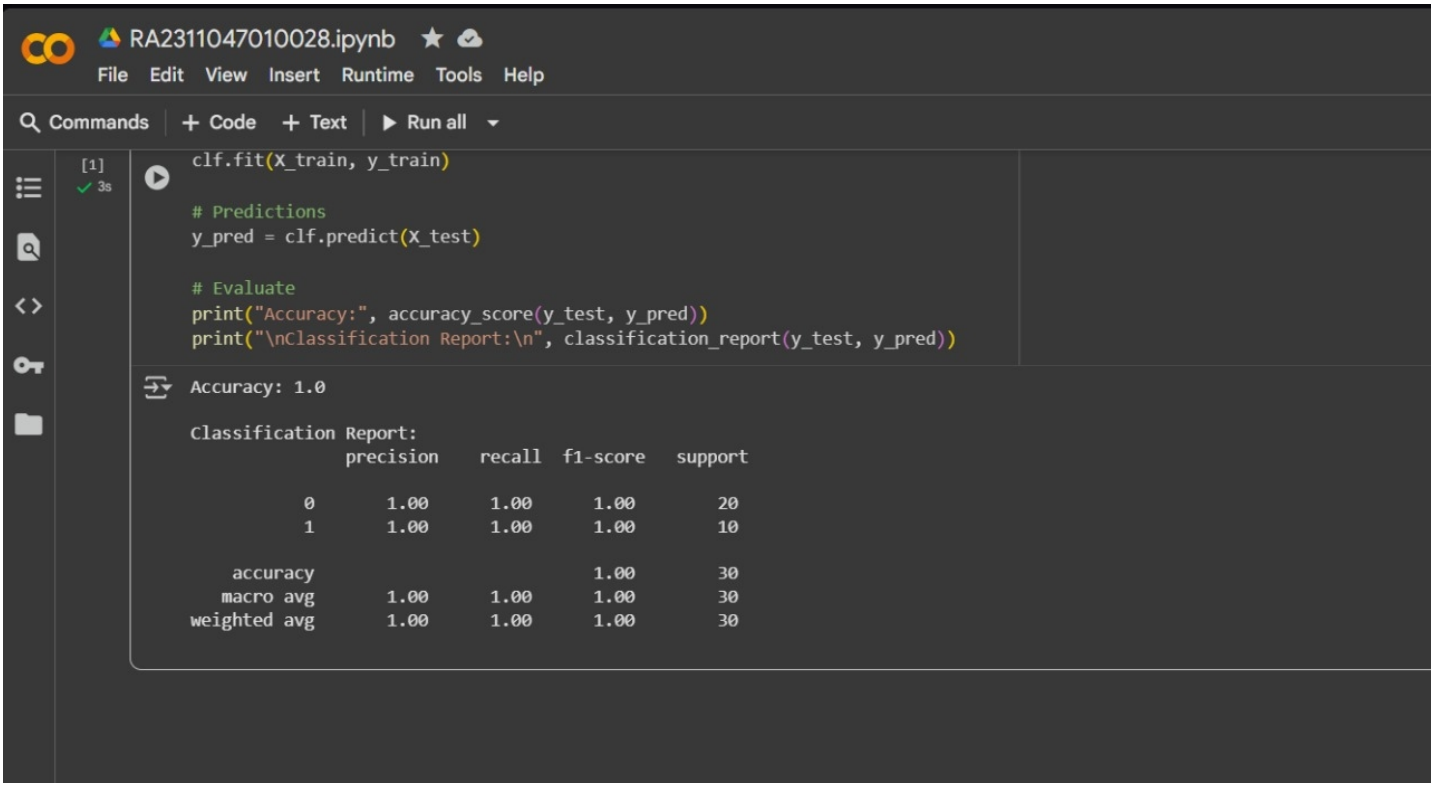
Result: Successfully implemented a classifier using open source dataset.

Output :-

Accuracy : 1.0

	Precision	recall	F1-Score	Support
setosa	1.00	1.00	1.00	10
Versicolor	1.00	1.00	1.00	9
Verginica	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.


RA2311047010028.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[1]
3s
# Import libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
# Load dataset
iris = load_iris()
X = iris.data
y = (iris.target == 0).astype(int) # Convert into binary classification (Setosa vs not-Setosa)
# Split dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Build a classifier
clf = LogisticRegression(max_iter=200)
clf.fit(X_train, y_train)
# Predictions
y_pred = clf.predict(X_test)
# Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))


RA2311047010028.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[1]
3s
clf.fit(X_train, y_train)
# Predictions
y_pred = clf.predict(X_test)
# Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
Accuracy: 1.0
Classification Report:
precision recall f1-score support
0 1.00 1.00 1.00 20
1 1.00 1.00 1.00 10
accuracy 1.00 30
macro avg 1.00 1.00 1.00 30
weighted avg 1.00 1.00 1.00 30