

Experiment 2

Implement using an open-source dataset

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

Pseudocode:

- 1) import necessary libraries
 - pandas, scikit-learn, dataset, metrics, KNeighboursClassifier.
- 2) Load the dataset
 - use datasets.load_iris()
- 3) Prepare the data
 - Assign features to x target to y.
- 4) Split into training and testing sets
 - use train-test-split (x, y, test-size = 0.2, random-state = 42)
- 5) Instantiate the KNN classifier:
 - KNN = KNeighboursClassifier (n-neighbours = 3)
- 6) Train the model
- 7) Make predictions
- 8) Evaluation the classifier:
 - calculate accuracy: metrics.accuracy_score(y_test, y_pred)

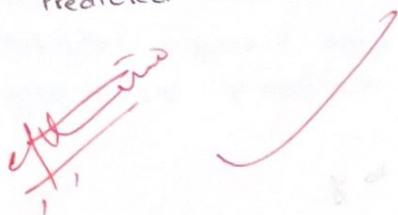
Observations:

- KNN classifier is trained on the Iris dataset
- and tested with unseen data
- Output is displayed
- Lowering 'K' can make the model even more sensitive to noise, while larger K can smoothen decision boundaries

Result:

Accuracy: 1.00

Predicted class for sample $[5.1, 3.5, 1.4, 0.2]$: Setosa



File Edit View Run Kernel Tabs Settings Help

Filter files by name

Name Last Modified

Lab_2.ipynb Lab_3.ipynb next year

[5]:

```
import numpy as np
import pandas as pd
import sklearn
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

[6]:

```
df = load_iris()
```

[7]:

```
X=df.data
y=df.target
```

[8]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)
```

[9]:

```
km = KNeighborsClassifier(n_neighbors=3)
km.fit(X_train, y_train)
y_pred = km.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
sample = [[5.1, 3.5, 1.4, 0.2]]
predicted_class = df.target_names[km.predict(sample)[0]]
print(f"Predicted class for sample {sample}: {predicted_class}")
```

Accuracy: 1.00

Predicted class for sample [[5.1, 3.5, 1.4, 0.2]]: setosa

Experiment -3

Study of classifiers with
respect to statistical parameters

Aim: To study the performance of different classifiers using statistical parameters like accuracy, precision, recall, & f1 score.

Description:

1. Accuracy -

The ratio of correctly predicted instances to the total instances in the dataset.

$$\text{accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{total predictions}}$$

2. Precision -

The ratio of correctly predicted positive instances to all instances predicted as positive.

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

3. Recall -

The ratio of correctly predicted positive instances to all actual positive instances. It measures how well the model identifies actual positives.

$$\text{recall} = \frac{\text{True Positives}}{\text{True positives} + \text{False negatives}}$$

4. F1 score -

The harmonic mean of precision and recall, providing a balance between the two metrics. It is a single metric used to evaluate the trade-off between precision & recall.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

5 Confusion Matrix:

This is a table showing the correct and incorrect predictions across classes. It helps visualize model performance with True Positive (TP), False Positive (FP), False negatives(FN), and true Negatives (TN).

Procedure :

- (i) Load the open source dataset
- (ii) Split dataset into training and testing sets
- (iii) Train the classifier
- (iv) Predict labels on test data
- (v) Evaluate each classifier, using accuracy, precision, etc.
- (vi) Visualize the confusion matrix

Observation:

Classifier	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.93	0.98	0.98	0.98
KNN	0.96	0.98	0.99	0.97
Decision Tree	0.94	0.97	0.94	0.95

Logistic Regression:

61(TP)	2(FN)
2(FP)	166(TN)

where malignancy
concerns

↓
+ve

Benign (non-concerns)

↑
-ve

KNN

57(TP)	6(FN)
1(FP)	107(TN)

Decision Tree

60(TP)	3(FN)
7(FP)	101(TN)

Result:

The classification of logistic regression, KNN and Decision tree were successfully implemented. All models achieved high scores with logistic regression having the highest overall performance

~~141828~~

File Edit View Run Kernel Help

Filter files by name

Untracked files

Name Last Modified

- 0 days 3 minutes ago
- 0 days 6 months ago
- 0 days 4 months ago
- 0 days 3 months ago

Run Cell

Untitled.ipynb

```
for name, model in models.items():
    model, x_train, y_train
    y_pred = model.predict(x_test)
    print("Model: ", name)
    print("Precision: ", accuracy_precision(x_test, y_pred))
    print("Accuracy: ", accuracy_accuracy(x_test, y_pred))
    print("Recall: ", recall_recall(x_test, y_pred))
    print("F1 Score: ", f1score_f1score(x_test, y_pred))
    print("Confusion Matrix: \n", confusion_matrix(x_test, y_pred))
    print("\n")
```

Model: LogisticRegression

Accuracy: 0.9766031772849629

Precision: 0.993830303030303

Recall: 0.993830303030303

F1 Score: 0.993830303030303

Confusion Matrix:

1 63	2
1 2	1061

Model: SVM

Accuracy: 0.9506043279483981

Precision: 0.9506043279483981

Recall: 0.9997487487487487

F1 Score: 0.98125791652936

Confusion Matrix:

1 57	6
1 1	1071

Model: DecisionTree

Accuracy: 0.9200205040235008

Precision: 0.9524081186799463

Recall: 0.955051051051051052

F1 Score: 0.9439252326465398

Confusion Matrix:

1 59	51
1 7	1041