03-02-2026

MLA0202

V. Rakshana

192425137

Set 7-7

MODEL PRACTICAL

1) Aim:- To study whether Breast Cancer Classification problem can solve using Naive Bayes Classifier, analyze dataset, process it & evaluates its performance using a Confusion matrix.

Algorithm:-

→ Load Breast Cancer dataset

→ The dataset is split into training & testing sets & a Naive Bayes Classifier using training data.

→ The model evaluate Confusion matrix and accuracy score.

Program:-

```
import pandas as pd
from sklearn.datasets import load_breast_cancer.
data = load_breast_cancer( )
print ("First five rows:\n", df.head())
print ("\n Null values:\n", df.isnull().sum())
df.fillna(df.mode().iloc[0], inplace=True)
x = df.drop ('target', axis=1)
y = df['target']
print ("\n Confusion Matrix:\n", Confusion_matrix(y_test,
                                                 y_pred))
print ("Accuracy", accuracy_score(ytest, y_pred)
```

Output:-

Accuracy ≈ 93-96 %

Result:-

The program executed successfully.

2)

Aim:- To find most specific hypothesis using Find s-algorithm using given dataset.

Algorithm:-

→ Initialize hypothesis with positive example.

⇒ Generalize only when attribute value differ.

⇒ Ignore negative examples

⇒ Output final hypothesis.

Program:-

```
data = [['Big', "Red', 'circle', 'No'], ['Small', 'Red', 'Triangle',
                                                         'Yes']]
hypothesis = None.
for row in data:
        if row[-1] == 'Yes':
              if hypothesis is None:
                      hythopesis = row[:-1]
              else:
                 for i in range(len(hypothesis)):
                     if hypothesis[i] != row[i]:
                          hypothesis[i] = '?'
print ("Most specific hypothesis:", hypothesis)
```

Output:-

```
Most specific hypothesis : ['Small', '?', 'Circle']
```

Result:-

The program executed Succesfully.

3) Aim :- To implement polynomial Regression & analyze its performance.

Program :-

```
import numpy as np
from sklearn.preprocessing import polynomial Features
x = np.array [[1,2,3,4,5]).reshape(-1,1)
y = np.array ([1,4,9,16,25])
Poly = Polynomial Features (degree=2)
x_poly = poly.fit_transform(x)
model = Linear Regression(·)
model.fit (x_poly, y)
y_pred = model.predict(x_poly)
print ("R2 Score:", r2_score (y, y_pred))
```

Output :-

```
R2 Score : 1.0
```

Result :-

The program has executed successfully.

Algorithm :-

⇒ Data preparation & Transformation.

⇒ A Linear Regression model is trained using transformed polynomial features & target values.

⇒ The trained model predicts Outputs for given data & its performance is evaluated using R^2 Score, indicating how well model fits data.

4) Aim:- To implement the KNN classification algorithm using a sample dataset.

Algorithm:-

⇒ The Iris dataset is loaded & divided into training & testing sets ~~to~~ to evaluate performance.

⇒ A KNN classifier is created with K=5 neighbors & trained using training data.

⇒ The trained model predicts class labels for test data, performance is measured using accuracy score.

Program:-

```
from sklearn.datasets. import .load_iris
from sklearn.metrics import accuracy_scores.
 iris = load_iris( )
 x = iris.data
 y = iris.target
 x_train, x_test, y_train, y_test = train_test_split (x, y,
 test_size = 0.3, random_state = 42)
 knn = KNeighbors Classifier (n_neighbors = 5)
 knn.fit ( x_train, y_train)
 y_pred = knn.predict (x_test)
 print ("KNN Accuracy:", accuracy_score(y_test,
                                      y_pred))
```

Output:-

```
KNN Accuracy: ~97%
```

Result:-

The program has been executed successfully.