08/02/26.
19LA0202.

Model pratical

B. Yugal
192425377.

1. Aim: To classify breast cancer data using the Naive Bayes algorithm and evaluate its performance.

Algorithm: 1. Load the breast cancer dataset.
2. perform basic data analysis and handle values.
3. Split the dataset into training and test sets.
4. Train the Naive Bayes Classifier.
5. Evaluate the model using a confusion matrix.

Code:
```
from sklearn.datasets import load_breast_cancer.
from sklearn.model_selection import train_test_split.
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score.
x,y = load_breast_cancer(return_x_y = True)
x_train, x_test, y_train, y_test = train_test_split
                                   (x,y, test_size = 0.3).
model = GaussianNB()
model.fit(x_train, y_train)
y_pred = model.predict(x_test).
print. ("Accuracy:", accuracy_score(y_test, y_pred)).
```

Input: Breast Cancer Dataset (features and target values.).

Output: Confusion matrix and Accuracy score.

Result: Naive Bayes Successfully Classified breast Cancer data with good accuracy.

2. Aim: To find the most specific hypothesis using the find S-Algorithm.

Algorithm: 1. Initialize the hypothesis with the most specific values.
2. Consider only positive training examples.
3. Compare each attribute with the hypothesis.
4. Generalize attributes when mismatches occur.
5. Output the final hypothesis.

Code:
```
import pandas as pd
data = pd.DataFrame([
    ['Big', 'Red', 'Circle', 'NO]
    ['Small', 'Red', Triangle, 'NO].
    [ 'Small' Red, Circle, Yes],
    [ 'Big', 'Blue', 'Circle', 'Yes],
    ['small', 'Blue', 'Circle', 'Yes].
  ], Columns= ['Size' Color Shape Class])
hypothesis = None
  for index row in data.iterrow():
    if row [Class] = = Yes
      if hypothesis = row[:-1].tolist()
  else:
    for in in range (len(hypothesis)):
      if hypothesis [i] != row[i]:
         hypothesis[i] = '?'
  print ("Final hypothesis:", hypothesis).
```

Input:

Size	Colour	Shape	Class
Small	Red	Circle	Yes
Small	Blue	Circle	Yes.

Output:

Final Hypothesis ['small', '?', 'circle'].

Result:

The most specific hypothesis was successfully derived using Find-S.

3. Aim:

To implement Polynomial Regression and evaluate its performance.

Algorithm:

1. Create sample input and output data.
2. Transform features into polynomial form.
3. Train the Regression model.
4. Predict output values.
5. Measure model performance.

Code:
```
import ~~pandas~~ numpy as pd.
~~datas~~ from sklearn.preprocessing import polynomial Features.
from sklearn.linear_model import Linear Regression.
x = np.array([1,2,3,4,5]).reshape(-1,1).
y = np.array([2,8,18,32,50]).
poly = polynomial Features(degree=2).
x_poly = poly.fit_transform(x).
model = Linear Regression()
model.fit fit(x_poly, y)
print("prediction:", model.predict(x_poly)).
```

Input: x = [1,2,3,4,5]

y = [2,8,18,32,50]

Output: Yaml

R2 score: 1.0

Result:

Polynomial Regression perfectly fits the give dataset.

4. Aim: To classify data using the K-Nearest Neighbours Algorithm.

Algorithm: 1. Load dataset and select features.
2. Split data into training and testing sets.
3. Choose the value of K. 4. Train the KNN classifier.
5. Predict and evaluate accuracy.

Code:
```
from sklearn.datasets import load_iris.
from sklearn.model_selection import train_test_split.
from sklearn.neighbours import KNeighboursClassifier
from sklearn.metrics import accuracy_score.
x, y = load_iris(return_X_y=True)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
knn = KNeighboursClassifier(n_neighbours=3)
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test).
print("Accuracy:", accuracy_score(y_test, y_pred)).
```

Input:

Iris Dataset features and class labels.

Output:

Accuracy: 0.97.

Result: KNN achieved high accuracy in classifying the iris dataset.

1924253 77

SET-7