MLAU202 7924 25 130
G. Mohith SET-1

(1) Mobile Price Range Prediction

Aim:- To Predict the Price range of mobile phones using Naive Bayes classifier

Algorithm:-

1. Load dataset
2. Preprocess data
3. Split into train & test.
4. Train Naive Bayes model

~~Aim:~~ Program:-

Python:

```
Import pandas as pd
from sklearn.model_selection import train_test_split
from Sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
data=pd.read_csv("mobile_price_classification.csv")
data.fillna(data.mode().iloc[0], inplace=True)
  X=data.drop("Price_range", axis=1)
  y=data["Price_range"]
X_train, X_test, y_train, y_test = train_test_split(
test_size=0.2)

y_Pred =model.predict(X_test)
Print('Accuracy', accuracy_score.
```

Output:-

Accuracy: 0.82 (approx)

Result:-

Naive Bayes Succesfully classified mobile Price ranges with good accuracy.

2. Find-S-Algorithm

Aim:-
To find the most specific hypothesis using Find-S algorithm.

Algorithm:
1. Initialize hypothesis
2. Update using positive examples
3. Ignore negative examples

Program:

```
import numpy as np.
data = np.array ([
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'Yes'],
['Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same', 'Yes'],
['Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change', 'No'],
['Sunny', 'Warm', 'High', 'Strong', 'Cool', 'Change', 'Yes'],
]).
hypothesis = data[0, :-1]
for row in data:
    if row[-1] == 'Yes':
        for i in range (len (hypothesis)):
            if row[i] != hypothesis[i]:
                hypothesis[i] = '?'
print (hypothesis)
```

Output

['Sunny', 'Warm', '?', 'Strong', '?', '?']

Result:-
Find-S algorithm produced the most specific consistent hypothesis.

3. Linear ~~Algorithm~~ Regression

Aim:-

To implement Linear Regression & evaluate performance

Algorithm

1. Load dataset
2. split data
3. Train Model
4. Predict values

Program:-

```
from sklearn.linear_model import LinearRegression
import pandas as pd
data = pd.DataFrame({'X':[1,2,3,4,5],'Y':[2,4,6,8,10]})
X = data[['X']]
y = data['Y']
model = LinearRegression()
model.fit(X,y)
print("Coefficient:", model.coef_)
```

Output:-

Coefficient: [2.]

Result:-

Linear regression successfully modeled the linear relationship

4. K-Nearest Neighbors (KNN)

Aim:
To classify data using KNN algorithm.

Algorithm:
1. Load dataset
2. split data
3. choose K value
4. Train KNN
5. Predict output

Program:

```
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier

iris = load_iris()

X, y = iris.data, iris.target

knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X, y)
print("Prediction:", knn.predict([X[2]]))
```

outputs

Prediction: [2]

Result:
KNN Correctly classified the data based on nearest neighbours