192425203
SET-9

: Aim: To Compare Decision Tree, Logistic Regression, KNN classifier using accuracy and execution time on Iris dataset

Algorithm:

1. load Iris dataset
2. Split into training and testing data
3. Train Decision Tree, logistic Regression, KNN
4. Measure accuracy and execution time
- Compare results.

ogram:

Python:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import logistic Regression
x, y = load_iris (return_X_y = True)
X_train, X_test, y_train, y_train = y_test_sp
    (X, y, test_size = 0.3)
```

tput:

Decision Tree Accuracy : 0.97 Time : 0.0012

istic Regression Accuracy : 0.98 Time : 0.0021

KNN Accuracy: 0.96 Time : 0.0015

2. Aim: To find the Most specific and Most general hypothesis consistent with training data

Algorithm:

1. Initialize S as Most Specific hypothesis
2. Initialize G as Most general hypothesis
3. Update S for Positive examples
4. Update G for Negative examples
5. Output final S and G

Program:

Python:

```
import Pandas as pd
data = [
['some', 'small', 'No', 'Affordable', 'Few', 'No']
['some', 'small', 'No', 'Affordable', 'Few', 'No']
['Many', 'Big', 'No', 'Expensive', 'Many', 'Yes']
    df = pd.DataFrame(data)
    x = df.iloc[:, :-1]
    y = df.iloc[:, :-1]
    S = ['0'] * len(x.columns)
    G = [['?'] * len(x.columns)]
for i in range(len(x)):
        if y[i] == 'Yes':
            for j in range(len(S)):
                if S[j] == '0':
                S[j] = x.iloc[i, j]
                elif S[j] != x.iloc[i, j]:
                    S[j] = '?'
```

```
else: G = [g for g in G if g != X.iloc[i]]
print ( ; s, "specific hypothesis ; G; "General )
```

Output:

specific hypothesis : ['Many', '?', 'No', '?', '?']

general hypothesis : [['?', '?', '?', '?', '?']]

3, Aim: To implement Polynomial regression and evaluate Model Performance.

Algorithm:

1. generate Sample data
2. Transform features to Polynomial.
3. Fit linear Regression
4. Predict Values.

Program:

Python:

```
import numpy as np
from sklearn.preprocessing import polynomialFeatures
from sklearn.linear_model import linear Regression
x = np.array([1,2,3,4,5]).reshape(-1,1)
y = np.array([2,5,10,17,26])
poly = PolynomialFeature(degree = 2)
x_poly = poly.fit_tranform(x)
Model = linearRegression()
Model.fit(x_poly, y)
```

Print ("Prediction for 6: ", model.predict(poly.transform
((([[6]]))

output:

Predictim for 6: [37.]

4. KNN Algorithm:

Aim: To implement K-Nearest Neighbours classification with a
example.

Algorithm:

1. load dataset
2. choose value of K
3. Calculate distance
4. Predict class using Majority vote.

ogram:

Python:

from sklearn.datasets import load_iris

from sklearn.neighbours import KNeighbours Classifier

x, y = load_iris (return_X_y=True)

knn = KNeighbour Classifier (n_neighbours = 3)

Knn.fit(x, y)

Print ("Predicted class", Knn.predict([[5.1, 3.5, 1.4
0.2]]))

tput:

Predicted class: [0]