

DATA WAREHOUSING AND DATA MINING

MODEL LAB

SIVARAM.BS

18BCS201

1. List the attributes and its type in a word Doc:

My dataset is fully Numeric.

2. Build decision tree classifier with Entropy criteria. Perform Prediction for test dataset using Entropy and print the results in the form of confusion matrix, accuracy and classification report. visualize the decision tree:

```
In [1]: import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")
```

```
In [2]: def importdata():
balance_data = pd.read_csv('balance-scale.csv')
print ("Dataset Length: ", len(balance_data))
print ("Dataset Shape: ", balance_data.shape)
print ("Dataset: ", balance_data.head())
return balance_data
```

```
In [3]: def splitdataset(balance_data):

# Separating the target variable
X = balance_data.values[:, 1:5]
Y = balance_data.values[:, 0]

# Splitting the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(
X, Y, test_size = 0.3, random_state = 100)

return X, Y, X_train, X_test, y_train, y_test
```

```
In [4]: def train_using_entropy(X_train, X_test, y_train):

clf_entropy = DecisionTreeClassifier(
    criterion = "entropy", random_state = 100,
    max_depth = 3, min_samples_leaf = 5)

clf_entropy.fit(X_train, y_train)
return clf_entropy
```

```

In [5]: def prediction(X_test, clf_object):

        y_pred = clf_object.predict(X_test)
        print("Predicted values:")
        print(y_pred)
        return y_pred

In [6]: def cal_accuracy(y_test, y_pred):

        print("Confusion Matrix: ",
              confusion_matrix(y_test, y_pred))

        print ("Accuracy : ",
              accuracy_score(y_test,y_pred)*100)



        print("Report : ",
              classification_report(y_test, y_pred))

In [7]: def main():







        data = importdata()
        X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
        clf_entropy = tarin_using_entropy(X_train, X_test, y_train)

        print("Results Using Entropy:")
        y_pred_entropy = prediction(X_test, clf_entropy)
        cal_accuracy(y_test, y_pred_entropy)

```

 **jupyter** Untitled Last Checkpoint: a few seconds ago (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

      Code

```

In [8]: main()

Dataset Length: 767
Dataset Shape: (767, 9)
Dataset:
 6 148 72 35 0 33.6 0.627 50 1
0 1 85 66 29 0 26.6 0.351 31 0
1 8 183 64 0 0 23.3 0.672 32 1
2 1 89 66 23 94 28.1 0.167 21 0
3 0 137 40 35 168 43.1 2.288 33 1
4 5 116 74 0 0 25.6 0.201 30 0
Results Using Entropy:
Predicted values:
[1. 1. 1. 1. 4. 0. 0. 1. 3. 0. 0. 1. 0. 0. 0. 1. 0. 0. 1. 0. 0. 5. 0. 1. 1. 0.
 0. 0. 0. 5. 1. 0. 0. 0. 1. 1. 4. 1. 0. 1. 1. 0. 0. 3. 1. 1. 0. 0. 1. 5.
 0. 0. 1. 1. 4. 3. 0. 0. 1. 0. 4. 0. 0. 0. 1. 0. 0. 0. 0. 4. 0. 1. 0. 1.
 1. 1. 5. 0. 1. 0. 4. 1. 1. 5. 1. 0. 3. 1. 1. 0. 0. 0. 0. 0. 0. 4. 0.
 1. 1. 1. 1. 3. 1. 1. 3. 1. 1. 0. 0. 1. 0. 1. 1. 1. 1. 0. 1. 0. 5. 1. 0.
 5. 3. 1. 3. 0. 1. 0. 4. 0. 4. 1. 5. 1. 0. 0. 1. 1. 0. 0. 1. 0. 0. 1. 1.
 5. 4. 0. 0. 0. 1. 1. 0. 0. 0. 1. 5. 3. 1. 0. 0. 1. 1. 0. 0. 0. 4. 1.
 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 3. 0. 3. 0. 1. 0. 1. 1. 5. 0. 1. 1. 0. 1.
 0. 1. 0. 1. 0. 0. 0. 0. 3. 0. 4. 0. 1. 0. 4. 0. 1. 4. 1. 0. 5. 1. 3.
 1. 3. 3. 1. 1. 0. 0. 1. 5. 1. 3. 4. 0. 0. 0.]
Confusion Matrix: [[22 15 0 3 3 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [17 21 0 4 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 6 13 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [10 8 0 2 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 9 6 0 3 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 7 2 0 1 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 4 6 0 1 3 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 8 2 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 7 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 8 1 0 0 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 4 2 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 1 2 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [ 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
Accuracy : 20.346320346320347
Report :

```

	precision	recall	f1-score	support
0.0	0.21	0.50	0.30	44
1.0	0.25	0.48	0.33	44
2.0	0.00	0.00	0.00	20
3.0	0.12	0.09	0.10	23
4.0	0.07	0.05	0.06	20
5.0	0.08	0.08	0.08	13
6.0	0.00	0.00	0.00	15
7.0	0.00	0.00	0.00	12
8.0	0.00	0.00	0.00	11
9.0	0.00	0.00	0.00	12
10.0	0.00	0.00	0.00	7
11.0	0.00	0.00	0.00	4
12.0	0.00	0.00	0.00	3
13.0	0.00	0.00	0.00	2
14.0	0.00	0.00	0.00	1
accuracy			0.20	231
macro avg	0.05	0.08	0.06	231
weighted avg	0.11	0.20	0.14	231