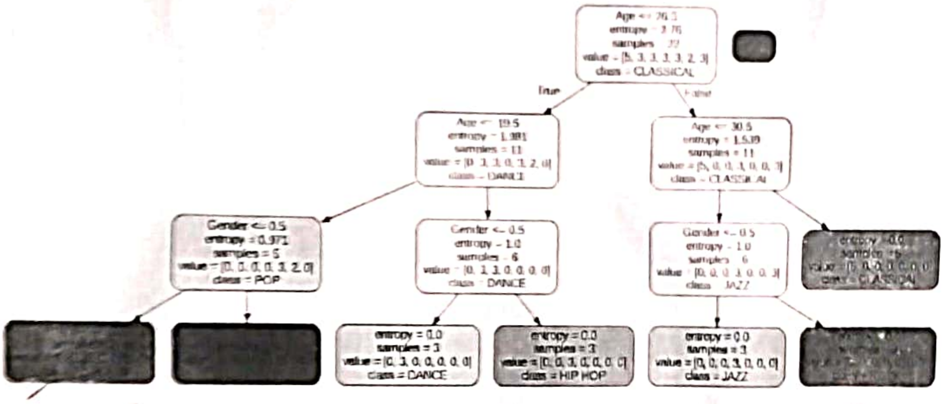
Experiment - 4

Aim -> write a program to demonstrate the working of the decission tree based ID3 algo. Use an approreiate did set for working and building of decision tree and apply this knowladge to classify a new sample

Code

```
import numpy as np
import pandas as pd

df_music = Pd.read_excel('/content/music.xsx')
df - music.head(5)

x = df-music.drop('Genre', axis = 1)
y = df-music ['Genre']

X.Shape, y.shape

X_train_split = int (0.8 * len(x))
x - train, Y - train = X[:x_train-split]
Y [:x-train - split]

x - train.shape, Y_ train . Shape
X - test.shape, Y_ test.shape
```

Output


Age <= 26.5
entropy = 2.76
samples = 22
value = [5, 3, 3, 3, 3, 2, 3]
class = CLASSICAL
True
False
Age <= 19.5
entropy = 1.981
samples = 11
value = [0, 3, 3, 0, 3, 2, 0]
class = DANCE
Age <= 30.5
entropy = 1.539
samples = 11
value = [5, 0, 0, 3, 0, 0, 3]
class = CLASSICAL
Gender <= 0.5
entropy = 0.971
samples = 5
value = [0, 0, 0, 0, 3, 2, 0]
class = POP
Gender <= 0.5
entropy = 1.0
samples = 6
value = [0, 3, 3, 0, 0, 0, 0]
class = DANCE
Gender <= 0.5
entropy = 1.0
samples = 6
value = [0, 0, 0, 3, 0, 0, 3]
class = JAZZ
entropy = 0.0
samples = 5
value = [5, 0, 0, 0, 0, 0, 0]
class = CLASSICAL
entropy = 0.0
samples = 3
value = [0, 3, 0, 0, 0, 0, 0]
class = DANCE
entropy = 0.0
samples = 3
value = [0, 0, 3, 0, 0, 0, 0]
class = HIP HOP
entropy = 0.0
samples = 3
value = [0, 0, 0, 3, 0, 0, 0]
class = JAZZ

```
from sklearn.tree import decision tree classifier
model = DecisionTreeClassifier (criterion='entropy')

model.fit (x-train, y-train)

prediction = model.predict ([[23,1]])
prediction

prediction = model.predict (x_test)
predictions

y-test

from sklearn.metrics import accuracy_score
accuracy_score (prediction, y_test)

print (f' correct prediction are as {accuracy-sco
            y_test, normalize = false)}')

from sklearn.tree import export graphviz

export_graphviz (model, out-file = 'music-recommender
-b2.dot"

feature_names = ['Age', 'Gender'], class_names =
Sorted (y.unique()), label='all', rounded
= true; filled = true )
```

```
import pydotplus
decision_tree = pydotplus.graph_from_dot_file
('music-recommender-b2.dot')
from Ipython.display import Image
Image (decision_treate.create_png())
```

14/2

Experiment-5

AIM: Write a program to implement the naive byes classifier for play-tennis dataset Store as a dot csv file. Compute the accuracy of classify fille dataset.

Code:

```
import numpy as np
import pandas as pd

play_tennis_df = pd.read_csv('PlayTennis.csv')
play_tennis_df.head(5)

import plotly.express as px

fig = px.parallel_categories(play_tennis_df[[
'Temperature', 'PlayTennis']], width=600, hei

fig.show()

from sklearn.preprocessing import LabelEncoder

number_encoder = LabelEncoder;

play_tennis_df['Outlook'] = number_encoder.fit_
  transform(play_tennis_df['Outlook']);
```

```
play_tennis_df['Temperature'] = number_encoder.fit_transform(
    play_tennis_df['Temperature'])

play_tennis_df['Humidity']=number_encoder.fit_transform('play_
tennis_df['Humidity'])

play_tennis_df['Wind'] = number_encoder.fit_transform(play_tennis
_df['wind'])

play_tennis_df['play Tennis'] = number_encoder.fit_transform(
    play_tennis_df['play Tennis'])

play_tennis_df.head()

X= play_tennis_df.drop(columns='play tennis',axis=1
X.head()

Y= play_tennis_df['play Tennis']
y.head(3)

from sklearn.model_selection import train_test_spli

from sklearn.navie_bayes import GaussianNB
model = GaussianNB()

model.fit(X_train,Y_train)
```

```
predict = model. predict ([[2, 0, 0, 0]])
predict

Y - pred = model. predict(x_test)
Y_pred

from sklearn. metrics import accuracy_store

accuracy_score (x_test, y_pred)
```

21/03/24

Experiment - 6

Aim write a program to implement K-Nearest Neighbor algo. to classify the iris, iris data set in scikit-learn. Print both correct & wrong prediction.

Code

```
from sklearn.dataset import load iris
iris = load- iris ()
print (iris.data (0:5))

iris.target [0:5]

iris.data [0:5]

iris.target names

iris.feature_names

import pandas as pd
df = pd.Dataframe (iris.data, columns = iris.feature
names)

df.head( )

df[ 'species' ] = pd.Categorical.form-codes (
    iris.target, iris.target-names)
```

Output

Accuracy of the model is 1.0

Incorrect predictions : 0

Correct predictions 38

```
df_haad(5)

x = df.drop('species', axis=1)
y = df['species']

from sklearn.neighbours import KNeighbours
knn = KNeighbours Classifiers (n-neighbours=5 )

knn.fit(x_train , y_train )

prediction = knn. predict (x_test )

from sklearn import metrics
accurracy = metrics.accuracy_score(y_test, predicti

printf (f' Accurracy of the model is {accuracy}')

print (f' Incorrect predictions : {len(y test) - metrics
 accuracy_score( y-test, predictions, normalize
== false )})

print( f' Correct prediction : {metrics.accuracy
 scorre (y - test ) predictions , normalize = false}')
```

Experiment -7

Aim: Build an artifical neural network for predicition of logical gates

Code

```
# importing necessary libraries
import numpy as np
from tensorflow.keras.model import Sequential
from tensorflow.keras.layers impor Dense

#

x = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([[0],[1],[1],[0]])

model = Sequential ()

model.add (Dense (4, input_dim=2, activation
   ='relu'))

activation function (binary output)

model.add (Dense (1, activation ='sigmoid')

model. compile (optimizer='adam', loss='binary
        crossentropy');

model.fit (X, Y, epochs=100, verbose=1)
```

Output

predictions

```
[ [0]
  [1]
  [1]
  [1] ]
```

Actual

```
[ [0]
  [1]
  [1]
  [0] ]
```

```
predictions = model. predict (X)

actual

print (" predictions:")
print ( np. round ( predictions ))

print (" Actual:")
print (y)
```

Experiment - 8

Aim Write a program for character recognition using CNN.

Code

```
pip install tensorflow
import tensorflow as tf

import matplotlib. pylot as plt

#Load the MNIST dataset
 (X_train ,Y_train ), (X_test, Y_test ) = dataset . mnist
load data()

x_train , x_test = x_train /255.0

x_train = x_train . reshape ((x_train . shape [0], 28,
                                          28 , 1))
x_test = x_test . reshape ((x_test . shape [0] 28, 28)

model  =   model . Sequential ([
             layer . Cony2D (32, (3,3), activation ='relu
imput shape= (28,28,1)),
                              layer . Maxpooling ((2,2)),

   layer . Cony2D (64, 63,3), activation = 'relu'
    layer . Maxpooling 2D ((2,2))
```

output

Test accuracy : 0.9912

```
layers. Flatten (),
layers. Dense (64, activation ='relu');
layers. Dense (10, activation ='softmax')
])

model. Compile (optimize ='adam', loss ='sparse
 categorical _      metrics = ['accuracy']).

model. fit (x_ train , y_ train , epoch=5 );

print( f' Test accuracy: { test_acc : 4f }')

plt. figure (figsize =(10,10))

for i in rang (25 ):

    plt. subplot (5, 5 , i+1)
   plt. x tricks ([])
  plt. ytricks [[])
  plt. grid (false)

  plt. imshow (x_ test [i]. reshap(28,28).
              camp= plt. Cm.binary )

  prt. show ()
```