**Program to implement Decision tree algorithm**

import pandas as pd

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

data=load\_iris()

x=data.data

y=data.target

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,random\_state=50,test\_size=0.25)

#default criterion gini

classifier=DecisionTreeClassifier()

classifier.fit(x\_train,y\_train)

y\_pred=classifier.predict(x\_test)

print('accuracy on train data using gini:',accuracy\_score(y\_train,classifier.predict(x\_train)))

print('accuracy on test data using gini:',accuracy\_score(y\_test,y\_pred))

#entropy

classifier\_entropy=DecisionTreeClassifier(criterion='entropy')

classifier\_entropy.fit(x\_train,y\_train)

y\_pred\_entropy=classifier\_entropy.predict(x\_test)

print('accuracy on train data using entropy:',accuracy\_score(y\_train,classifier\_entropy.predict(x\_train)))

print('accuracy on test data using entopy:',accuracy\_score(y\_test,y\_pred\_entropy))

#entropy with min\_samples\_split

classifier\_entropy1=DecisionTreeClassifier(criterion='entropy',min\_samples\_split=50)

classifier\_entropy1.fit(x\_train,y\_train)

y\_pred\_entropy1=classifier\_entropy1.predict(x\_test)

print('accuracy on train data using entropy:',accuracy\_score(y\_true=y\_train,y\_pred=classifier\_entropy1.predict(x\_train)))

print('accuracy on test data using entopy:',accuracy\_score(y\_true=y\_test,y\_pred=y\_pred\_entropy1))

#visualize

from sklearn.tree import plot\_tree

import matplotlib.pyplot as plt

# Visualize the decision tree

plt.figure(figsize=(15, 10))

plot\_tree(classifier,

          feature\_names=data.feature\_names,

          class\_names=data.target\_names,

          filled=True,

          rounded=True)

plt.title("Decision Tree Visualization (Gini Criterion)")

plt.show()

**Program to implement Naivebayes for categorical data(using LabelEncoder)**

# Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

from sklearn.naive\_bayes import CategoricalNB

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, classification\_report

# -----------------------------

# 1. Load Data

# -----------------------------

data = pd.read\_csv("tennis.csv")

print("Dataset:\n", data)

# Separate features (X) and target (y)

X = data.drop("play", axis=1).copy()

y = data["play"]

# -----------------------------

# 2. Encode categorical features

# -----------------------------

le\_outlook = LabelEncoder()

X["outlook"] = le\_outlook.fit\_transform(X["outlook"])

le\_temp = LabelEncoder()

X["temp"] = le\_temp.fit\_transform(X["temp"])

le\_humidity = LabelEncoder()

X["humidity"] = le\_humidity.fit\_transform(X["humidity"])

le\_windy = LabelEncoder()

X["windy"] = le\_windy.fit\_transform(X["windy"])

print("\nEncoded Features (X):\n", X)

# Encode target

le\_play = LabelEncoder()

y = le\_play.fit\_transform(y)

print("\nEncoded Target (y):\n", y)

# -----------------------------

# 3. Train-test split

# -----------------------------

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=0.20, stratify=y, random\_state=42

)

# -----------------------------

# 4. Train model

# -----------------------------

classifier = CategoricalNB()

classifier.fit(X\_train, y\_train)

# -----------------------------

# 5. Evaluate model

# -----------------------------

y\_pred = classifier.predict(X\_test)

print("\nAccuracy:", accuracy\_score(y\_test, y\_pred))

print("\nClassification Report:\n",

      classification\_report(y\_test, y\_pred, target\_names=le\_play.classes\_))

# -----------------------------

# 6. Predict new sample using array

# -----------------------------

# New input: ('sunny','hot','high','False')

new\_sample = np.array([[

    le\_outlook.transform(["sunny"])[0],

    le\_temp.transform(["hot"])[0],

    le\_humidity.transform(["high"])[0],

    le\_windy.transform(["False"])[0]

]])

prediction = classifier.predict(new\_sample)

predicted\_label = le\_play.inverse\_transform(prediction)[0]

print("\nPrediction for ('sunny','hot','high','False'):", predicted\_label)

**Program to implement Naivebayes for Text data**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score, classification\_report

data = {'label': ['ham', 'spam', 'ham', 'ham', 'spam', 'ham', 'spam', 'ham'],

    'message': [

        "Hello, how are you?",

        "Congratulations! You've won a lottery of $1000!",

        "Don't forget our meeting tomorrow.",

        "Your subscription is renewed.",

        "Click here to claim your prize!",

        "See you at the gym later.",

        "Limited time offer! Get your discount now!",

        "Have a great day!"

       ]}

df = pd.DataFrame(data)

X = df['message']

y = df['label']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5,

                                                    random\_state=42, stratify=y)

vectorizer = CountVectorizer()

x\_train\_vectorized = vectorizer.fit\_transform(x\_train)

x\_test\_vectorized = vectorizer.transform(x\_test)

# Build model

model = MultinomialNB()

model.fit(x\_train\_vectorized, y\_train)

y\_pred = model.predict(x\_test\_vectorized)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy \* 100:.2f}%")

# Predict a new message

new\_message = ['how are you']  # New message to predict

new\_message\_vectorized = vectorizer.transform(new\_message)  # Use the same vectorizer

new\_pred = model.predict(new\_message\_vectorized)

print(f"Predicted label for new message: {new\_pred[0]}")

**Program to implement Naivebayes for categorical data(using OrdinalEncoder)**

# Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.preprocessing import OrdinalEncoder, LabelEncoder

from sklearn.naive\_bayes import CategoricalNB

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, classification\_report

# -----------------------------

# 1. Load Data

# -----------------------------

data = pd.read\_csv("tennis.csv")

print("Dataset:\n", data)

# Separate features (X) and target (y)

X = data.drop("play", axis=1).copy()

y = data["play"]

# -----------------------------

# 2. Encode categorical features

# -----------------------------

encoder = OrdinalEncoder()

X\_encoded = encoder.fit\_transform(X)

print("\nEncoded Features (X):\n", X\_encoded)

# Encode target

le\_play = LabelEncoder()

y\_encoded = le\_play.fit\_transform(y)

print("\nEncoded Target (y):\n", y\_encoded)

# -----------------------------

# 3. Train-test split

# -----------------------------

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X\_encoded, y\_encoded, test\_size=0.20, stratify=y\_encoded, random\_state=42

)

# -----------------------------

# 4. Train model

# -----------------------------

classifier = CategoricalNB()

classifier.fit(X\_train, y\_train)

# -----------------------------

# 5. Evaluate model

# -----------------------------

y\_pred = classifier.predict(X\_test)

print("\nAccuracy:", accuracy\_score(y\_test, y\_pred))

print("\nClassification Report:\n",

      classification\_report(y\_test, y\_pred, target\_names=le\_play.classes\_))

# -----------------------------

# 6. Predict new sample using array

# -----------------------------

# New input: ('sunny','hot','high','False')

new\_sample = np.array([["sunny", "hot", "high", "False"]])

# Transform new sample using the same encoder

new\_sample\_encoded = encoder.fit\_transform(new\_sample)

prediction = classifier.predict(new\_sample\_encoded)

predicted\_label = le\_play.inverse\_transform(prediction)[0]

print("\nPrediction for ('sunny','hot','high','False'):", predicted\_label)