Sardar Patel Institute of Technology,Mumbai
Department of Electronics and Telecommunication Engineering
B.E. Sem-VII- PE-IV (2024-2025)
IT 24 - AI in Healthcare

Exp 2-Experiment: Decision Tree (ID3) algorithm

Name: Adwait Purao UID : 2021300101 Batch : D Date: 27/8/24

Objective: Write Python program to demonstrate the working of the decision tree based ID3 algorithm by using appropriate medical data set for building the decision tree and apply this knowledge to forecast.

Outcomes:

1. Find entropy of data and follow steps of the algorithm to construct a tree.
2. Representation of hypothesis using decision tree.
3. Apply Decision Tree algorithm to classify the given data.
4. Interpret the output of Decision Tree.

System Requirements:

Linux OS with Python and libraries or R or windows with MATLAB

Theory:

The decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

Entropy
A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.

E(S) is the Entropy of the entire set, while the second term E(S, A) relates to an Entropy of an attribute A.

$$E(S) = \sum_{x \in X} -P(x) \log_2 P(x)$$

$$E(S, A) = \sum_{x \in X} [P(x) * E(S)]$$

Information Gain
The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

$$IG(S,A) = E(S) - E(S,A)$$

Dataset Description:

Code:

```
#Importing Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


#For ignoring warning

import warnings

warnings.filterwarnings("ignore")

df=pd.read_csv('./survey lung cancer.csv')

df
```

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN	LUNG_CANCER
0	M	69	1	2	2	1	1	2	1	2	2	2	2	2	2	YES
1	M	74	2	1	1	1	2	2	2	1	1	1	2	2	2	YES
2	F	59	1	1	1	2	1	2	1	2	1	2	2	1	2	NO
3	M	63	2	2	2	1	1	1	1	1	2	1	1	2	2	NO
4	F	63	1	2	1	1	1	1	1	2	1	2	2	1	1	NO
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
304	F	56	1	1	1	2	2	2	1	1	2	2	2	2	1	YES
305	M	70	2	1	1	1	1	2	2	2	2	2	2	1	2	YES
306	M	58	2	1	1	1	1	1	2	2	2	2	1	1	2	YES
307	M	67	2	1	2	1	1	2	2	1	2	2	2	1	2	YES
308	M	62	1	1	1	2	1	2	2	2	2	1	1	2	1	YES

309 rows × 16 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 276 entries, 0 to 283
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   GENDER                 276 non-null    object
 1   AGE                    276 non-null    int64
 2   SMOKING                276 non-null    int64
 3   YELLOW_FINGERS         276 non-null    int64
 4   ANXIETY                276 non-null    int64
 5   PEER_PRESSURE          276 non-null    int64
 6   CHRONIC DISEASE        276 non-null    int64
 7   FATIGUE                276 non-null    int64
 8   ALLERGY                276 non-null    int64
 9   WHEEZING               276 non-null    int64
 10  ALCOHOL CONSUMING      276 non-null    int64
 11  COUGHING               276 non-null    int64
 12  SHORTNESS OF BREATH    276 non-null    int64
 13  SWALLOWING DIFFICULTY  276 non-null    int64
 14  CHEST PAIN             276 non-null    int64
 15  LUNG_CANCER            276 non-null    object
dtypes: int64(14), object(2)
memory usage: 36.7+ KB
```

```
from sklearn import preprocessing

le=preprocessing.LabelEncoder()

df['GENDER']=le.fit_transform(df['GENDER'])

df['LUNG_CANCER']=le.fit_transform(df['LUNG_CANCER'])

df['SMOKING']=le.fit_transform(df['SMOKING'])

df['YELLOW_FINGERS']=le.fit_transform(df['YELLOW_FINGERS'])

df['ANXIETY']=le.fit_transform(df['ANXIETY'])

df['PEER_PRESSURE']=le.fit_transform(df['PEER_PRESSURE'])

df['CHRONIC DISEASE']=le.fit_transform(df['CHRONIC DISEASE'])

df['FATIGUE ']=le.fit_transform(df['FATIGUE '])

df['ALLERGY ']=le.fit_transform(df['ALLERGY '])

df['WHEEZING']=le.fit_transform(df['WHEEZING'])
```

```python
df['ALCOHOL CONSUMING']=le.fit_transform(df['ALCOHOL CONSUMING'])

df['COUGHING']=le.fit_transform(df['COUGHING'])

df['SHORTNESS OF BREATH']=le.fit_transform(df['SHORTNESS OF BREATH'])

df['SWALLOWING DIFFICULTY']=le.fit_transform(df['SWALLOWING DIFFICULTY'])

df['CHEST PAIN']=le.fit_transform(df['CHEST PAIN'])

df['LUNG_CANCER']=le.fit_transform(df['LUNG_CANCER'])

df
```

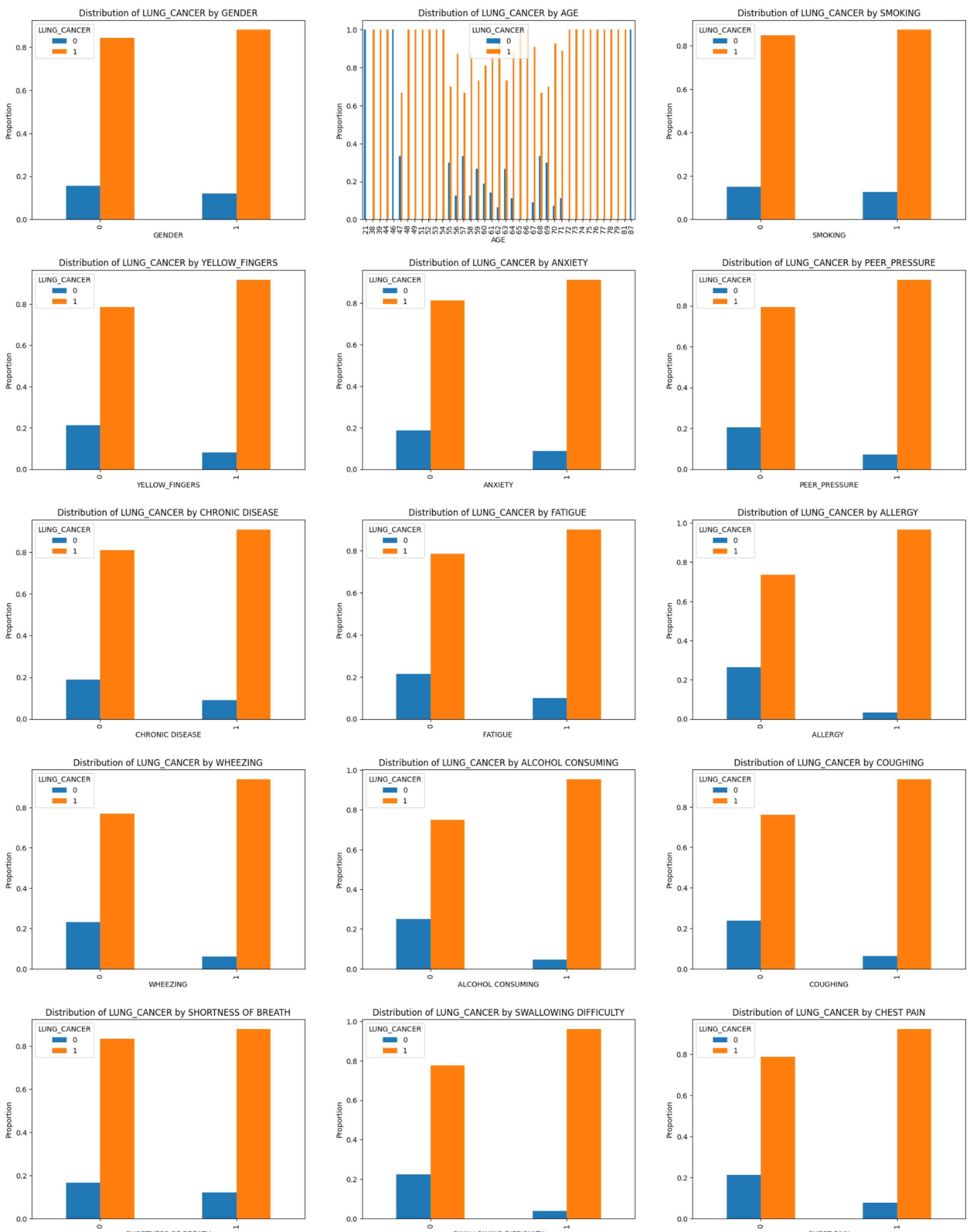
	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN	LUNG_CANCER
0	1	69	0	1	1	0	0	1	0	1	1	1	1	1	1	1
1	1	74	1	0	0	0	1	1	1	0	0	0	1	1	1	1
2	0	59	0	0	0	1	0	1	0	1	0	1	1	0	1	0
3	1	63	1	1	1	0	0	0	0	0	1	0	0	1	1	0
4	0	63	0	1	0	0	0	0	0	1	0	1	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
279	0	59	0	1	1	1	0	0	1	1	0	1	0	1	0	1
280	0	59	1	0	0	0	1	1	1	0	0	0	1	0	0	0
281	1	55	1	0	0	0	0	1	1	0	0	0	1	0	1	0
282	1	46	0	1	1	0	0	0	0	0	0	0	0	1	1	0
283	1	60	0	1	1	0	0	1	0	1	1	1	1	1	1	1

276 rows × 16 columns

Note: Male=1 & Female=0. Also for other variables, YES=1 & NO=0

```python
import matplotlib.pyplot as plt


# Define a list of columns you want to plot

columns = ['GENDER', 'AGE', 'SMOKING', 'YELLOW_FINGERS', 'ANXIETY',
'PEER_PRESSURE',

           'CHRONIC DISEASE', 'FATIGUE ', 'ALLERGY ', 'WHEEZING', 'ALCOHOL
CONSUMING',

           'COUGHING', 'SHORTNESS OF BREATH', 'SWALLOWING DIFFICULTY',
'CHEST PAIN']


# Create a figure with a grid of subplots

fig, axes = plt.subplots(nrows=5, ncols=3, figsize=(20, 25))  # Adjust the
grid size as needed

fig.tight_layout(pad=5.0)  # Adjust padding between plots
```

```python
# Flatten the axes array for easy iteration
axes = axes.flatten()


# Iterate over columns and plot each in its subplot
for i, col in enumerate(columns):

df.groupby(col)['LUNG_CANCER'].value_counts(normalize=True).unstack().plot
(kind='bar', ax=axes[i])
    axes[i].set_title(f'Distribution of LUNG_CANCER by {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Proportion')


# Remove any empty subplots (if the grid is larger than the number of
plots)
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])


plt.show()
```


Distribution of LUNG_CANCER by GENDER
GENDER
Distribution of LUNG_CANCER by AGE
AGE
Distribution of LUNG_CANCER by SMOKING
SMOKING
Distribution of LUNG_CANCER by YELLOW_FINGERS
YELLOW_FINGERS
Distribution of LUNG_CANCER by ANXIETY
ANXIETY
Distribution of LUNG_CANCER by PEER_PRESSURE
PEER_PRESSURE
Distribution of LUNG_CANCER by CHRONIC DISEASE
CHRONIC DISEASE
Distribution of LUNG_CANCER by FATIGUE
FATIGUE
Distribution of LUNG_CANCER by ALLERGY
ALLERGY
Distribution of LUNG_CANCER by WHEEZING
WHEEZING
Distribution of LUNG_CANCER by ALCOHOL CONSUMING
ALCOHOL CONSUMING
Distribution of LUNG_CANCER by COUGHING
COUGHING
Distribution of LUNG_CANCER by SHORTNESS OF BREATH
SHORTNESS OF BREATH
Distribution of LUNG_CANCER by SWALLOWING DIFFICULTY
SWALLOWING DIFFICULTY
Distribution of LUNG_CANCER by CHEST PAIN
CHEST PAIN
Proportion
LUNG_CANCER
0
1

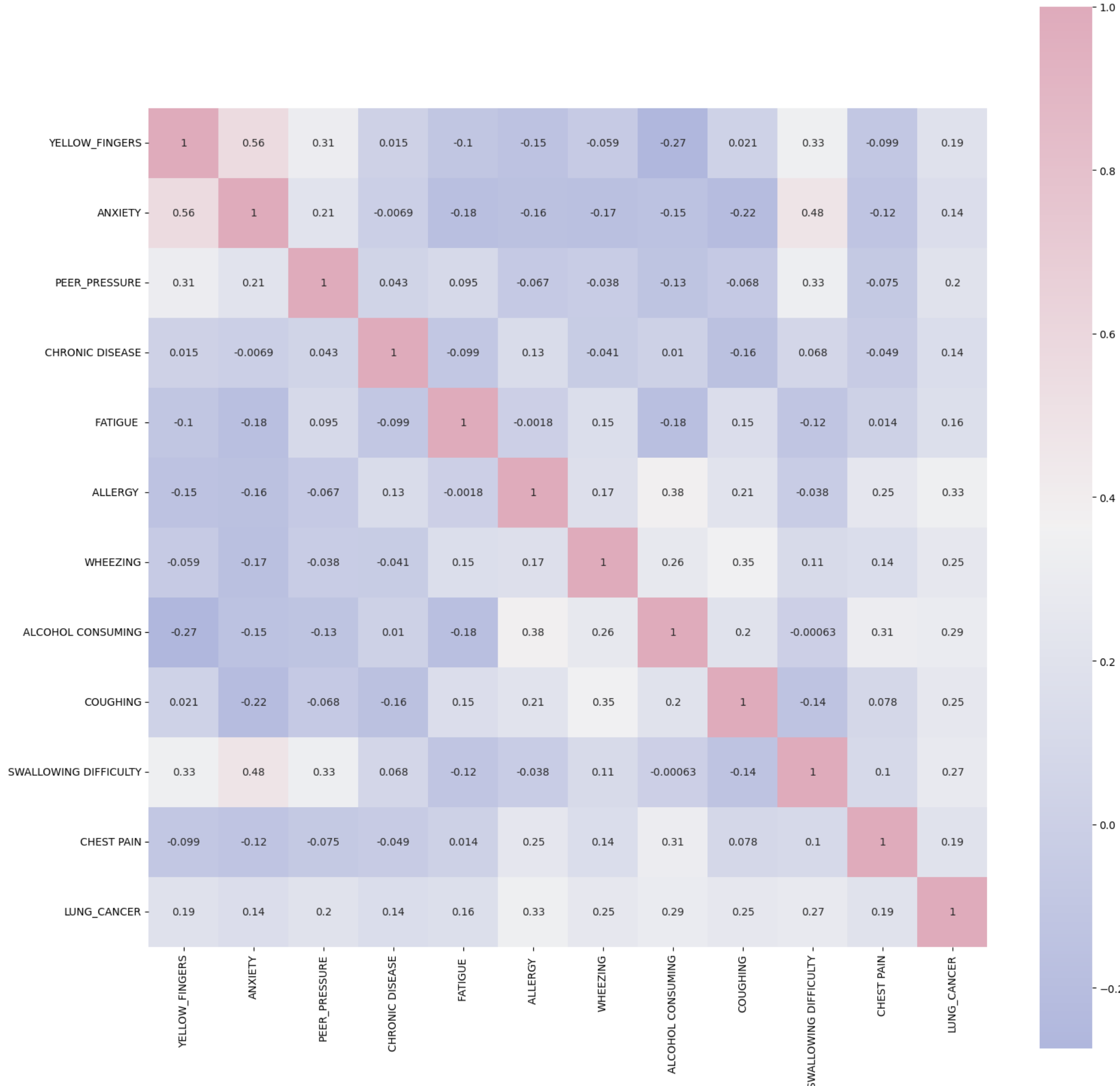
From the visualizations, it is clear that in the given dataset, the features GENDER, AGE, SMOKING and SHORTNESS OF BREATH don't have that much relationship with LUNG CANCER. So let's drop those features to make this dataset more clean.

```
df_new=df.drop(columns=['GENDER','AGE', 'SMOKING', 'SHORTNESS OF BREATH'])

df_new
```

	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SWALLOWING DIFFICULTY	CHEST PAIN	LUNG_CANCER
0	1	1	0	0	1	0	1	1	1	1	1	1
1	0	0	0	1	1	1	0	0	0	1	1	1
2	0	0	1	0	1	0	1	0	1	0	1	0
3	1	1	0	0	0	0	0	1	0	1	1	0
4	1	0	0	0	0	0	1	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
279	1	1	1	0	0	1	1	0	1	1	0	1
280	0	0	0	1	1	1	0	0	0	0	0	0
281	0	0	0	0	1	1	0	0	0	0	1	0
282	1	1	0	0	0	0	0	0	0	1	1	0
283	1	1	0	0	1	0	1	1	1	1	1	1

276 rows × 12 columns

```
#Correlation

cmap=sns.diverging_palette(260,-10,s=50, l=75, n=6,

as_cmap=True)

plt.subplots(figsize=(18,18))

sns.heatmap(cn,cmap=cmap,annot=True, square=True)

plt.show()



kot = cn[cn>=.40]

plt.figure(figsize=(12,8))

sns.heatmap(kot, cmap="Blues")
```



Feature Engineering

Feature Engineering is the process of creating new features using existing features.

The correlation matrix shows that ANXIETY and YELLOW_FINGERS are correlated more than 50%. So, lets create a new feature combining them.

```
df_new['ANXYELFIN']=df_new['ANXIETY']*df_new['YELLOW_FINGERS']

df_new
```

	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SWALLOWING DIFFICULTY	CHEST PAIN	LUNG_CANCER	ANXYELFIN
0	1	1	0	0	1	0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	0	0	0	1	1	1	0
2	0	0	1	0	1	0	1	0	1	0	1	0	0
3	1	1	0	0	0	0	0	1	0	1	1	0	1
4	1	0	0	0	0	0	1	0	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
279	1	1	1	0	0	1	1	0	1	1	0	1	1
280	0	0	0	1	1	1	0	0	0	0	0	0	0
281	0	0	0	0	1	1	0	0	0	0	1	0	0
282	1	1	0	0	0	0	0	0	0	1	1	0	1
283	1	1	0	0	1	0	1	1	1	1	1	1	1

276 rows × 13 columns

```
#Splitting independent and dependent variables
X = df_new.drop('LUNG_CANCER', axis = 1)
y = df_new['LUNG_CANCER']
from imblearn.over_sampling import ADASYN
adasyn = ADASYN(random_state=42)
X, y = adasyn.fit_resample(X, y)


len(X)
```

```
477
```

Decision Tree

```
#Splitting data for training and testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(X, y, test_size= 0.25, random_state=0)
#Fitting training data to the model
from sklearn.tree import DecisionTreeClassifier
dt_model= DecisionTreeClassifier(criterion='entropy', random_state=0)
dt_model.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
#Predicting result using testing data

y_dt_pred= dt_model.predict(X_test)

y_dt_pred
```

```
array([1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 1, 1, 0])
```

```
#Model accuracy

from sklearn.metrics import classification_report, accuracy_score,
f1_score

dt_cr=classification_report(y_test, y_dt_pred)

print(dt_cr)
```

```
              precision    recall  f1-score   support

           0       0.93      0.97      0.95        64
           1       0.96      0.91      0.94        56

    accuracy                           0.94       120
   macro avg       0.94      0.94      0.94       120
weighted avg       0.94      0.94      0.94       120
```

This model is 94% accurate.

```
from sklearn import metrics


print("Accuracy:",metrics.accuracy_score(y_test, y_dt_pred))
```

```
confusion_matrix = metrics.confusion_matrix(y_test, y_dt_pred)

print(confusion_matrix)
```

```
Accuracy: 0.9416666666666667
[[62  2]
 [ 5 51]]
```

```
# Forecasting


sample_index = 2  # change this index to predict different samples

# Access the row using .iloc for integer-location based indexing

sample = X_test.iloc[sample_index].values.reshape(1, -1)

prediction = dt_model.predict(sample)


print(f"Predicted class for sample {sample_index}: {prediction[0]}")
```

```
Predicted class for sample 2: 0
```

Conclusion:

- We learned to calculate the entropy of the dataset and information gain of each attribute to decide the root node and subsequently the leaf nodes.
- We also used scikit to run the Decision Tree algorithm on a larger dataset and estimate the accuracy f the model created.
- In Decision Tree as the depth of the tree increases the model overfits the data and accuracy reduces to avoid these, parameters for pruning the tree should be passed to the classifier.