Problem Statement

A dataset to train a model to predict whether a student thinks of dropping out.

Questionnaires

- Name
- Gender
- Date of Birth
- Degree
- Year of Study
- Academic Performance
- GPA
- Do you have any backlogs?
- Attendance
- Support System
- Involment in Extra Curricular Activities
- Financial Status
- Are you thinking about dropping out

Screenshots of the Dataset

Timestamp	Date of Birth	Name	Gender	Age	Degree	Year of Study	Acadamic Performance	GPA	Do you have any backlogs?	Attendance	Support System
11/06/2023 13:04:55	08/05/2001	manish	Male		B.E	3	Poor	7.2	Yes	Average	Family
25/06/2023 16:02:16	14/06/2001	Karthik J	Male		B.E	3	Good	8	No	Excellent	Family
03/06/2023 23:08:46	14/03/2002	Deeksha pk	Female		B.E	3	Good	8.3	no	Good	Family
03/06/2023 23:27:24	12/01/2002	Ranika	Female		BCA	3	Good	8		Excellent	Family
03/06/2023 23:29:03	09/11/2002	FOUZIYA UNAISA	Female		B.E	3	Good	8.5		Average	Family
03/06/2023 23:31:02	09/11/2002	Vignesh V	Male		B.E	3	Good	7.4		Good	Family
03/06/2023 23:34:47	15/11/2002	Ayishath Azhana	Female		B.E	3	Excellent	8.1		Excellent	Family
03/06/2023 23:37:48	27/03/2003	Nidhi Rai	Female		B.E	3	Good	8.4		Good	Family
03/06/2023 23:40:22	09/03/2002	Saahil S Pawar	Male		B.E	3	Good	9		Good	Family
03/06/2023 23:45:16	12/10/2002	Laniel	Male		BCA	3	Good	7.5		Good	Family
03/06/2023 23:47:18	07/06/2004	Vaishakh A	Male		B.E	4	Excellent	9.35		Excellent	Family
04/06/2023 12:18:35	21/09/2002	Shaun Crasta	Male		B.E	3	Good	8.25	No	Good	Family
04/06/2023 12:43:47	05/08/2002	Pranam rai	Male		B.E	3	Good	8.57	No	Average	Family
04/06/2023 12:20:52	27/04/2002	Ramith N	Male		B.E	3	Excellent	8.25	No	Good	Teachers
04/06/2023 12:33:08	09/03/2002	Ram Sai Rao U	Male		B.E	3	Good	8.86	No	Good	Family
04/06/2023 12:30:53	31/01/2002	Mohith	Male		BCA	3	Average	7	No	Good	Family
04/06/2023 13:20:52	27/04/2002	Ranith N	Male		B.E	3	Good	8.3	No	Excellent	Friends
04/06/2023 12:36:09	11/04/2002	Ankith K Ullal	Male		B.E	3	Good	8.2	No	Good	
04/06/2023 12:38:41	19/12/2001	Mohammed Basith	Male		B.E	3	Good	8.33	No	Good	Family
04/06/2023 12:28:46	24/06/2002	Amulya D	Female		B.E	3	Good	7.92	No	Good	Family
04/06/2023 15:11:22	24/03/2002	Syed hashim	Male		B.E	3	Good	7.25	No	Excellent	Friends
04/06/2023 12:25:14	05/05/2002	Anagha	Female		B.E	3	Good	8.5	No	Good	Family

Timestamp	Date of Birth	Name	Gender	Age	Degree	Year of Study	Acadamic Performance	GPA	Do you have any backlogs?	Attendance	Support System
28/06/2023 18:38:30	28/06/2000	Manoj	Male		B.Sc	4	Average	8	No	Good	Teachers
28/06/2023 18:39:54	18/11/2002	Anish	Male		B.Sc	3	Average	6.5	No	Average	Family
28/06/2023 18:44:44	28/06/2010	Oswald	Male		BCA	1	Excellent	8.6	No	Good	Family
28/06/2023 18:45:25	28/06/1999	Charlie	Male		B.E	4	Good	8.5	No	Good	Friends
28/06/2023 18:45:55	28/06/2001	Shwetha	Female		B.Sc	2	Average	7	No	Good	Friends
28/06/2023 18:46:22	28/06/2003	Sneha	Female		B.E	1	Average	7.8	No	Good	Family
28/06/2023 18:46:50	28/06/2001	Swathi	Female		B.Sc	2	Good	8.8	No	Good	Family
28/06/2023 18:47:15	28/06/2001	Yash	Male		BCA	2	Average	7.9	No	Average	Family
28/06/2023 18:47:42	28/06/2002	Krithik	Male		BCA	2	Good	8	No	Good	Friends
28/06/2023 18:48:32	28/06/2001	Jathin	Male		BCA	2	Good	8.5	No	Average	Friends
28/06/2023 18:49:04	28/06/2001	Lohith	Male		BCA	3	Good	8.8	No	Good	Friends
28/06/2023 18:51:11	01/07/2002	Sukesh	Male		B.Sc	3	Good	8.41	No	Excellent	Family
28/06/2023 19:01:37	08/06/2002	Jerome Joseph	Male		B.E	3	Poor	8.84	No	Poor	Haemoglobin
28/06/2023 19:06:16	01/06/2002	Shelton Dcunha	Male		B.E	3	Good	6.5	Yes	Excellent	Friends
28/06/2023 19:13:15	09/11/2002	Shreecharan Hebbar M	Male		B.E	3	Good	7.5	No	Good	Family
28/06/2023 19:24:09	07/01/2002	Suhas S Bhandary	Male		B.E	4	Good	8.2	No	Good	Family
28/06/2023 19:26:16	04/07/2002	Laxmee	Female		B.Sc	3	Good	6	No	Good	Family
28/06/2023 19:55:05	04/04/2002	R K Prem Iniyan	Male		B.E	3	Good	8	No	Poor	Family
28/06/2023 20:49:02	19/09/2002	Gagan	Male		B.Sc	3	Excellent	9	No	Excellent	Family
28/06/2023 22:16:46	20/10/2002	Pinto priya	Female		B.E	3	Good	8.3	No	Good	Family
29/06/2023 08:34:10	01/04/2000	Ripche Soshe	Male		B.Sc	3	Excellent	7.9	No	Excellent	Family

Screenshots of Programming Code & Execution

- **Reading CSV file**

```
# Importing libraries
import pandas as pd
import scipy
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import seaborn as sns
import matplotlib.pyplot as plt


# Load the dataset
df = pd.read_csv('/content/data1.csv')
df.head()
```

	Timestamp	Date of Birth	Name	Gender	Age	Degree	Year of Study	Acadamic Performance	GPA	Do you have any backlogs?	Attendance	Support System	Involment in Extra Curricular Activities	Financial Status	Are you thinking about droping out	Mention reasons below
0	11-06-2023 13:04	08-05-2001	manish	Male	NaN	B.E	3	Poor	7.20	Yes	Average	Family	Active	Excellent	Yes	Prefer Not to Say
1	03-06-2023 23:08	14-03-2002	Deeksha pk	Female	NaN	B.E	3	Good	8.30	NaN	Good	Family	Somewhat Active	Good	No	NaN
2	04-06-2023 12:13	13-06-2023	fluwafyugaff	Male	NaN	B.Sc	1	Excellent	43.00	NaN	Excellent	Family	Active	Excellent	No	NaN
3	04-06-2023 12:18	21-09-2002	Shaun Crasta	Male	NaN	B.E	3	Good	8.25	No	Good	Family	Not Active	Average	No	NaN
4	03-06-2023 23:19	19-04-2002	Srinivas A Rao	Male	NaN	B.E	3	Good	8.00	NaN	Good	Family	Somewhat Active	Average	No	NaN

- **Handling missing data**

Two of the columns have the missing data, which needs to be handled.

```
# Gives information about dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 302 entries, 0 to 301
Data columns (total 17 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   Timestamp                               302 non-null    object 
 1   Date of Birth                           302 non-null    object 
 2   Name                                    302 non-null    object 
 3   Gender                                  302 non-null    object 
 4   Age                                     0 non-null      float64
 5   Degree                                  302 non-null    object 
 6   Year of Study                           302 non-null    int64  
 7   Acadamic Performance                    302 non-null    object 
 8   GPA                                     302 non-null    float64
 9   Do you have any backlogs?               264 non-null    object 
 10  Attendance                              302 non-null    object 
 11  Support System                          302 non-null    object 
 12  Involment in Extra Curricular Activities  302 non-null    object 
 13  Financial Status                        302 non-null    object 
 14  Are you thinking about droping out      302 non-null    object 
 15  Mention reasons below                   55 non-null     object 
 16  Email address                           302 non-null    object 
dtypes: float64(2), int64(1), object(14)
memory usage: 40.2+ KB
```

```
# Check for missing values
df.isnull().sum()
```

```
Timestamp                                      0
Date of Birth                                  0
Name                                           0
Gender                                         0
Age                                          302
Degree                                         0
Year of Study                                  0
Acadamic Performance                           0
GPA                                            0
Do you have any backlogs?                     38
Attendance                                     0
Support System                                 0
Involment in Extra Curricular Activities       0
Financial Status                               0
Are you thinking about droping out             0
Mention reasons below                        247
Email address                                  0
dtype: int64
```

Since the age column has all the values missing, it needs to be dropped.

```
# Handling the missing values
data=data.drop('Age',axis=1)
data=data.drop('Mention reasons below',axis=1)
```

Next, the values which are missing are filled using the most frequent method.

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='most_frequent')
data_imputed = pd.DataFrame(imputer.fit_transform(data), columns=data.columns)
print("Data after imputation:\n", data_imputed)
```

```
Data after imputation:
            Timestamp Date of Birth            Name  Gender Degree  \
0    11-06-2023 13:04    08-05-2001          manish    Male    B.E
1    03-06-2023 23:08    14-03-2002       Deeksha pk  Female    B.E
2    04-06-2023 12:13    13-06-2023     fluwafyugaff    Male   B.Sc
3    04-06-2023 12:18    21-09-2002     Shaun Crasta    Male    B.E
4    03-06-2023 23:19    19-04-2002  Srinivas A Rao     Male    B.E
..                ...           ...              ...     ...    ...
297  28-06-2023 18:46    28-06-2001           Swathi  Female   B.Sc
298  28-06-2023 18:47    28-06-2001             Yash    Male    BCA
299  28-06-2023 18:47    28-06-2002          Krithik    Male    BCA
300  28-06-2023 18:48    28-06-2001           Jathin    Male    BCA
301  28-06-2023 18:49    28-06-2001           Lohith    Male    BCA

    Year of Study Acadamic Performance   GPA Do you have any backlogs?  \
0               3                 Poor   7.2                      Yes
1               3                 Good   8.3                       No
2               1            Excellent  43.0                       No
3               3                 Good  8.25                       No
4               3                 Good   8.0                       No
..            ...                  ...   ...                      ...
297             2                 Good   8.8                       No
298             2              Average   7.9                       No
299             2                 Good   8.0                       No
300             2                 Good   8.5                       No
301             3                 Good   8.8                       No
```

All the missing data has been handled.

```
data_imputed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 302 entries, 0 to 301
Data columns (total 15 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Timestamp                               302 non-null    object
 1   Date of Birth                           302 non-null    object
 2   Name                                    302 non-null    object
 3   Gender                                  302 non-null    object
 4   Degree                                  302 non-null    object
 5   Year of Study                           302 non-null    object
 6   Acadamic Performance                    302 non-null    object
 7   GPA                                     302 non-null    object
 8   Do you have any backlogs?               302 non-null    object
 9   Attendance                              302 non-null    object
 10  Support System                          302 non-null    object
 11  Involment in Extra Curricular Activities  302 non-null  object
 12  Financial Status                        302 non-null    object
 13  Are you thinking about droping out      302 non-null    object
 14  Email address                           302 non-null    object
dtypes: object(15)
memory usage: 35.5+ KB
```

- **Feature selection**

Only those attributes which contribute to the prediction are selected, rest are dropped.

The required attributes are mentioned below.

```
# Feature selection
data_2=data_1.drop(['Timestamp','Date of Birth','Name','Gender','Degree','Year of Study','Email address'],axis=1)
```

```
data_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 302 entries, 0 to 301
Data columns (total 8 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Acadamic Performance                    302 non-null    object
 1   GPA                                     302 non-null    object
 2   Do you have any backlogs?               302 non-null    object
 3   Attendance                              302 non-null    object
 4   Support System                          302 non-null    object
 5   Involment in Extra Curricular Activities  302 non-null  object
 6   Financial Status                        302 non-null    object
 7   Are you thinking about droping out      302 non-null    object
dtypes: object(8)
memory usage: 19.0+ KB
```

- **Discretization and Binarization**

All the categorical values need to be converted into numerical values.

Gpa column is not converted, since it is already in the numerical form.

```
# Convert the yes no and maybe to numerical values
# Define a mapping dictionary
mapping = {"Yes": 1, "No": 0,"Maybe":2}
# Apply the mapping to the column
data_3["Are you thinking about droping out"] = data_3["Are you thinking about droping out"].map(mapping)
```

```
data_3['GPA'] = data_3['GPA'].astype('float')
```

```
data_3.dtypes
```

```
Acadamic Performance                      object
GPA                                      float64
Do you have any backlogs?                 object
Attendance                                object
Support System                            object
Involment in Extra Curricular Activities  object
Financial Status                          object
Are you thinking about droping out         int64
dtype: object
```

```
# Converting Categorical variables into numeric values
from sklearn.preprocessing import LabelEncoder
categorical_attributes = ['Acadamic Performance',
    # 'GPA',
     'Do you have any backlogs?',
                          'Attendance',
                          'Support System',
                          'Involment in Extra Curricular Activities',
                          'Financial Status ',
                          'Are you thinking about droping out',
                          ]
encoder = LabelEncoder()
for attr in categorical_attributes:
    data_4[attr] = encoder.fit_transform(data_4[attr])
```

```
data_4.dtypes
```

```
Acadamic Performance                        int64
GPA                                       float64
Do you have any backlogs?                   int64
Attendance                                  int64
Support System                              int64
Involment in Extra Curricular Activities    int64
Financial Status                            int64
Are you thinking about droping out          int64
dtype: object
```

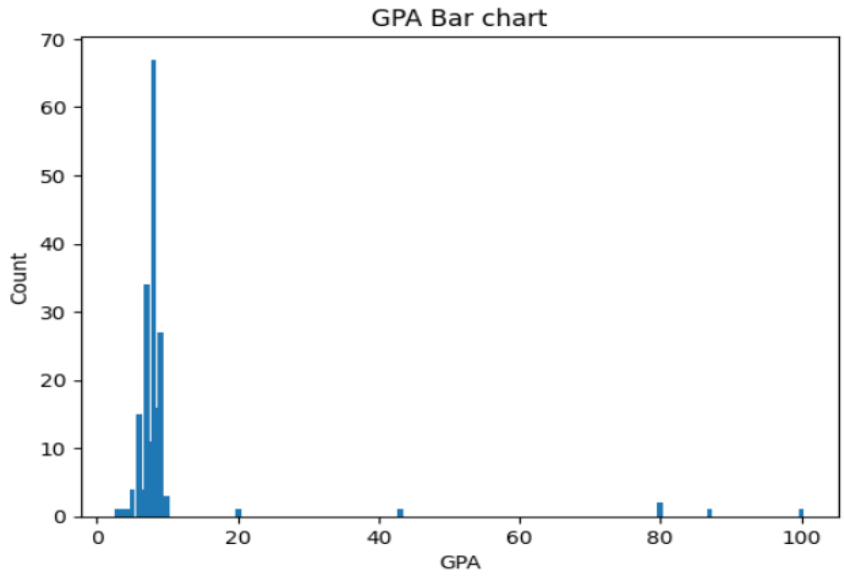
- **Identifing unique values of each Attribute**

```
# Identify unique values of each attribute
attribute_values = {'Acadamic Performance': data_4['Acadamic Performance'].unique(),
    'GPA': data_4['GPA'].unique(),
    'Do you have any backlogs?': data_4['Do you have any backlogs?'].unique(),
    'Attendance': data_4['Attendance'].unique(),
    'Support System': data_4['Support System'].unique(),
    'Involment in Extra Curricular Activities': data_4['Involment in Extra Curricular Activities'].unique(),
    'Financial Status ': data_4['Financial Status '].unique(),
    'Are you thinking about droping out': data_4['Are you thinking about droping out'].unique()
}

# Print unique values
for attribute, values in attribute_values.items():
    print(attribute)
    print(values)
    print()
```

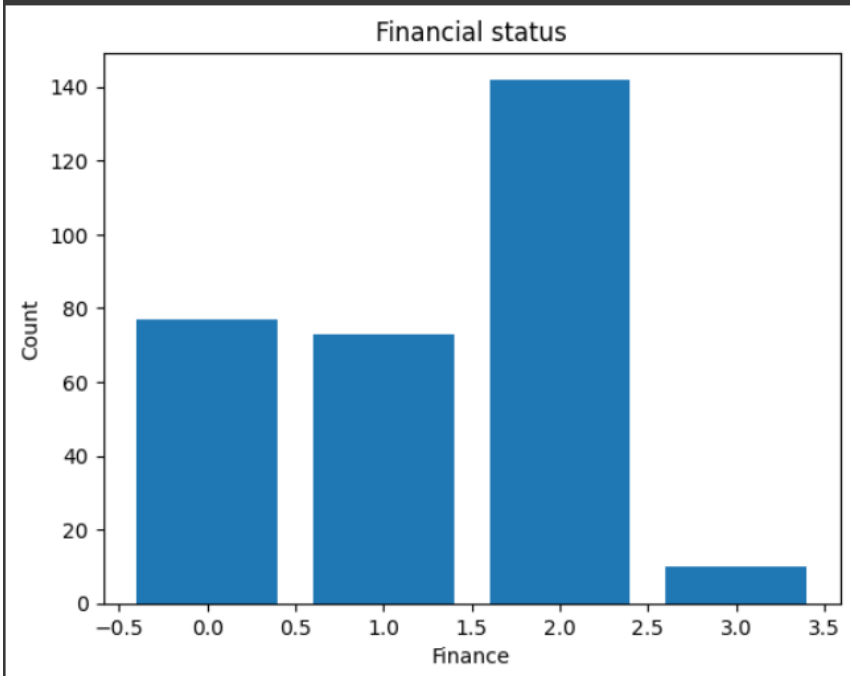
```
Acadamic Performance
[3 2 1 0]

GPA
[  7.2    8.3   43.     8.25   8.     8.57   8.86   7.    80.     8.33
   7.92   7.25   8.5    8.9    9.34   8.7    7.4    8.76   8.1    8.4
   9.     7.5    9.35   6.5    7.72   8.2    6.8    9.3    7.29   8.56
   6.2    6.     7.96   5.    87.     9.2    9.45   7.39   8.99   9.25
   8.8    6.9    9.19   6.75   8.47   6.4    8.75   9.8    9.9    9.65
   8.77   9.6    8.61   3.5    7.1  100.     3.     8.6    8.44   8.81
   9.56   7.66   4.35   6.63   6.7    7.9    7.89   7.7    9.5   20.
   7.8    6.3    7.68   4.     8.69   9.15   8.55   7.01   7.86   6.68
   8.29]

Do you have any backlogs?
[1 0]

Attendance
[0 2 1 3]

Support System
[ 2 10  5  4  6  3  9  1  8  7  0]

Involment in Extra Curricular Activities
[0 2 1]

Financial Status
[1 2 0 3]

Are you thinking about droping out
[1 0 2]
```

- **Data Visualization**

```
# Display bar chart
fcounts = data_4['GPA'].value_counts()
plt.bar(fcounts.index, fcounts.values)
plt.title('GPA Bar chart')
plt.xlabel('GPA')
plt.ylabel('Count')
plt.show()
```




```
fcounts = data_4['Financial Status '].value_counts()
plt.bar(fcounts.index, fcounts.values)
plt.title('Financial status')
plt.xlabel('Finance')
plt.ylabel('Count')
plt.show()
```



- **Training the Model**

The data has been split to 70% to train the model and 30% to test the model.

The Target class is ‘Are you thinking of dropping out’, which contains three values: Yes , No or Maybe which is 1,0,2 in numerical form respectively.

```
# Training model
features = [             'Acadamic Performance',
                         'GPA',
                         'Do you have any backlogs?',
                         'Attendance',
                         'Support System',
                         'Involment in Extra Curricular Activities',
                         'Financial Status ',
                         ]

target = 'Are you thinking about droping out'
```

```
# Training model
X_train, X_test, y_train, y_test = train_test_split(data_4[features], data_4[target], test_size=0.3, random_state=42)
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

Using these training data, the decision tree classifier algorithm is implemented.

- **Implementing confusion matrix to evaluate performance metrices**

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.97      0.94      0.95       247
           1       0.88      0.85      0.86        33
           2       0.62      0.82      0.71        22

    accuracy                           0.92       302
   macro avg       0.82      0.87      0.84       302
weighted avg       0.93      0.92      0.93       302
```

```
print(confusion_matrix(y_test,predictions))
```

```
[[233   3  11]
 [  5  28   0]
 [  3   1  18]]
```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.83      0.87        81
           1       0.43      0.50      0.46         6
           2       0.00      0.00      0.00         4

    accuracy                           0.77        91
   macro avg       0.45      0.44      0.44        91
weighted avg       0.85      0.77      0.80        91
```

```
print(confusion_matrix(y_test,y_pred))
```

```
[[67  3 11]
 [ 3  3  0]
 [ 3  1  0]]
```

```
# Find Accuracy, Error rate, Precision, Recall, F-Measure of trained data
accuracy = accuracy_score(y_test, y_pred)
error_rate = 1 - accuracy
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f_measure = f1_score(y_test, y_pred, average='weighted')

print("Accuracy:", accuracy)
print("Error Rate:", error_rate)
print("Precision:", precision)
print("Recall:", recall)
print("F-measure:", f_measure)
```

```
Accuracy: 0.7692307692307693
Error Rate: 0.23076923076923073
Precision: 0.8452076299434421
Recall: 0.7692307692307693
F-measure: 0.804942310436816
```

```
import joblib
joblib.dump(classifier, 'decision_tree_classifier.pkl')
```

```
['decision_tree_classifier.pkl']
```

```
# Test all the data based on the trained model
classifier = joblib.load('decision_tree_classifier.pkl')
X_test = data_4[features]



predictions = classifier.predict(X_test)

y_test = data_4['Are you thinking about droping out']
accuracy = accuracy_score(y_test, predictions)
error_rate = 1 - accuracy
precision = precision_score(y_test, predictions, average='weighted')
recall = recall_score(y_test, predictions, average='weighted')
f_measure = f1_score(y_test, predictions, average='weighted')

print("Accuracy:", accuracy)
print("Error Rate:", error_rate)
print("Precision:", precision)
print("Recall:", recall)
print("F-measure:", f_measure)
```

```
Accuracy: 0.9238410596026491
Error Rate: 0.07615894039735094
Precision: 0.9315596102295092
Recall: 0.9238410596026491
F-measure: 0.9265726327610455
```

The resulting performance metrices are as follows:

a) Accuracy: 92.3%

b) Error Rate:76.1%

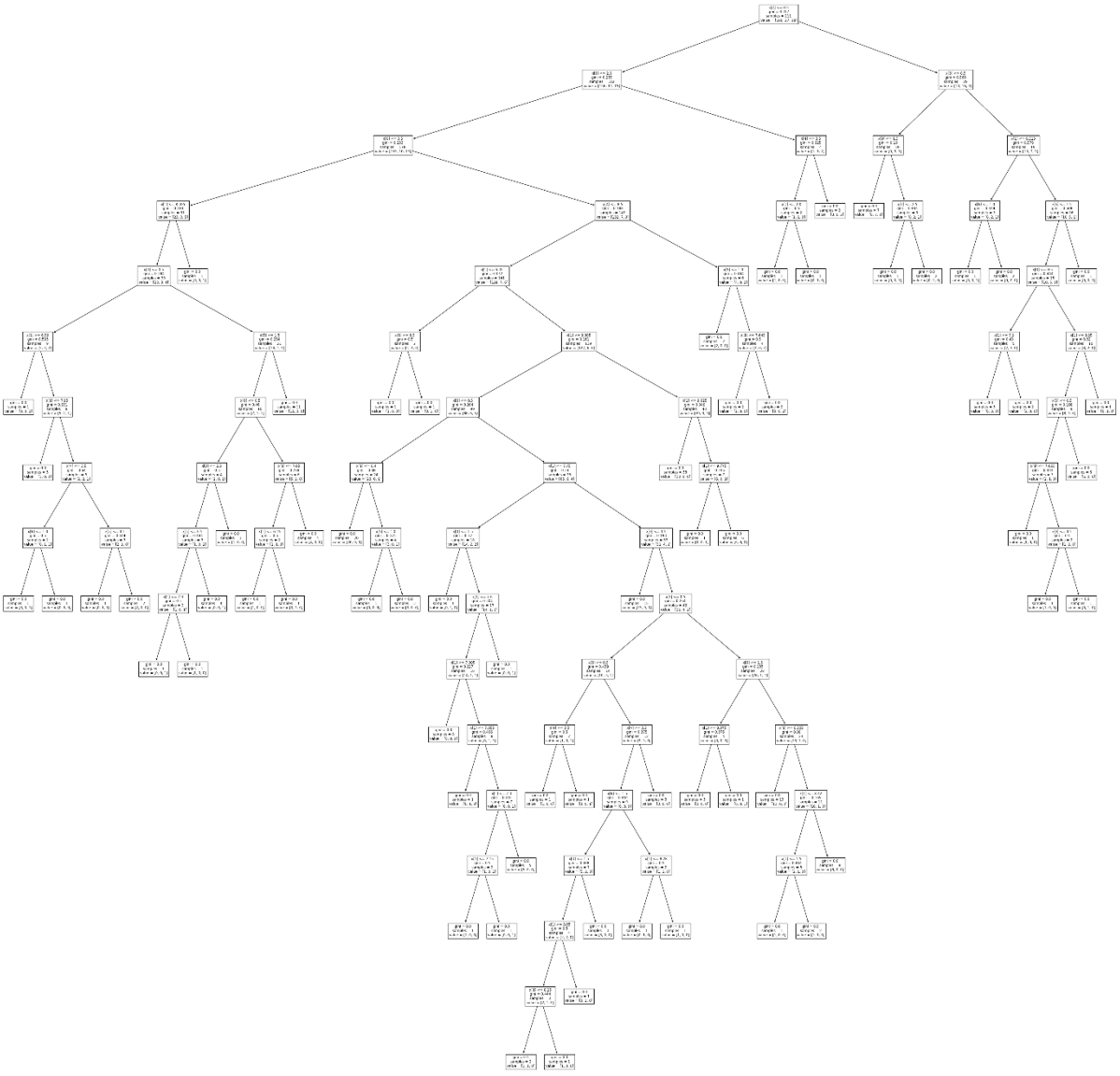
c) Precision:93.1%

d) Recall:92.3%

e) F-Measure:92.6%

- **Plotting Decision tree**

```
from sklearn import tree
fig, ax = plt.subplots(figsize=(50, 50))
tree.plot_tree(classifier, ax=ax)
plt.show()
```



```
# Outcomes of the data tested
print(predictions)
```

```
[1 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 2 0 0 0 0 0 0 0 0
 0 0 0 0 2 0 2 0 2 0 0 0 0 0 0 0 0 2 0 0 2 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 2 0 0 0 0 0 2 0 0 2 2 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 1 0 0
 0 2 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 2 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 2 0 0 1 0 0 2 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 2 0 0 0 0
 0 0 0 0 0 2 0 0 2 0 0 1 0 0 0 2 0 1 0 0 0 2 1 0 0 0 0 0 1 1 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0 2 1 1 0 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
 0 2 0 2 0 0 1 0 0 0 1 0 2 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 2 1 0 2 0 0 0
 0 0 0 0 0 0]
```

- **Using the model to predict whether a student thinks of dropping out.**

```
from pandas.core.window.expanding import ExpandingGroupby
from matplotlib.projections.polar import ThetaAxis
n = int(input("Enter no of Students who's status is to be determined :- "))

def Prediction() :
  Acadamic_Performance =int(input("Performance of student(Excellent:1, Good:2 , Average:0 ,poor:1) :- "))
  GPA = float(input("CGPA"))
  Bg = int(input("do u have any backlogs: yes(1),no(0)"))
  Attendance= int(input("attendance: Average(0),Good(2),Excellent(1),Poor(3) "))
  ss = int(input("Support system: family(2), friends(5), teacher(10)  "))
  Activities= int(input("involvement in extra curricular activities: active(0), not active(1), somewhat active(2) "))
  Fs = int(input("Financial status:Excellent(1), good(2), average(0), poor(3) "))

  new_student = np.array([Acadamic_Performance, GPA,Bg,Attendance,ss, Activities,Fs])

  new_student = new_student.reshape(1,-1) #converting to 2D array

  if classifier.predict(new_student) == 1:
     return "The student is thinking of dropping out!!!"
  elif classifier.predict(new_student)==2 :
    return "The student may have a thought of dropping out"
  elif classifier.predict(new_student)==0:
      return "The student doesnt have any thoughts of dropping out"

for i in range(n) :
   print(Prediction())
```

The figure below shows the input given to the model

```
Enter no of Students who's status is to be determined :- 1
Performance of student(Excellent:1, Good:2 , Average:0 ,poor:1) :- 1
CGPA4
do u have any backlogs: yes(1),no(0)1
attendance: Average(0),Good(2),Excellent(1),Poor(3) 0
Support system: family(2), friends(5), teacher(10)  2
involvement in extra curricular activities: active(0), not active(1), somewhat active(2) 2
Financial status:Excellent(1), good(2), average(0), poor(3) 0
The student is thinking of dropping out!!!
```

According the input of the user, the model predicts that the student is having a thought of dropping out.