ASSIGNMENT - 2

2228-CSE-5334-004-

DATA MINING

REPORT ON Classification – Decision Tree

SUBMITTED BY

SANJANA KONDABATHINI (1001984312)

MEGHANA KADALI (1002027499)

MUCHARLA RAJASHEKAR (1002027966)

Under the guidance of PROF. Dr. Elizabeth D Diaz
Teaching Assistant Kunal Mehta

UNIVERSITY OF TEXAS ARLINGTON

DEPARTMENT OF COMPUTER SCIENCE

Decision tree:

Introduction:

Decision Tree mining is a technique that is used to build classification models. The proposed model is described by a set a decision rules, and decision tree are simple to comprehend and adopt. It creates categorization models with structure like a tree. The suage of decision trees is applicable to both category and numerical data. It is used to crate data models that will predict class labels or values for the decision-making process.

The splitting technique used to crate the decision tree are GINI and ENTROPY.

GINI:

We may find out the likelihood of misclassifying an observation from Gini. A dataset's Gini impurity is a value between 0 and 0.5. When the node is pure, the Gini has a minimum value of 0 (all the items belong to the same class). When the odds of the two classes are equal, it is at its maximum that is 0.5.

Let the dataset D contains K classes. The probability of samples belonging to class I at a given node be Pi. Then GINI impurity is represented as

$$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$

ENTROPY:

Entropy, a measure of a random variable's uncertainty, identifies the impurity in any random collection of examples. More information is contained when entropy is higher. The highest value is reached when the probability of the two classes is equal, and the lowest value, which is zero, is reached when a node is pure.

$$\boxed{Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)}$$

Data Set Description:

The name of the data set is dataset_DT_NB (dataset_DT_NB.csv). The data set consists of Id, age, gender, height, weight, ap_hi (systolic blood pressure), ap_lo (Diastolic blood pressure), cholesterol, gluc (glucose), smoke (smoking), alco (Alcohol intake), active (Physical activity), and cardio are the attributes in the data set.

Below is the glimpse of the data set. Here we used head function to get the information of 1st 5 rows.

In [2]:
```
df_data=pd.read_csv('dataset_DT_NB.csv', sep=';')
df_data.head( )
```

Out[2]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0

To get the detailed description of each attribute in the data set we used the describe function

In [3]:
```
#detailed discription of dataset using describe()

df_data.describe().transpose()
```

Out[3]:

	count	mean	std	min	25%	50%	75%	max
id	70000.0	49972.419900	28851.302323	0.0	25006.75	50001.5	74889.25	99999.0
age	70000.0	19468.865814	2467.251667	10798.0	17664.00	19703.0	21327.00	23713.0
gender	70000.0	1.349571	0.476838	1.0	1.00	1.0	2.00	2.0
height	70000.0	164.359229	8.210126	55.0	159.00	165.0	170.00	250.0
weight	70000.0	74.205690	14.395757	10.0	65.00	72.0	82.00	200.0
ap_hi	70000.0	128.817286	154.011419	-150.0	120.00	120.0	140.00	16020.0
ap_lo	70000.0	96.630414	188.472530	-70.0	80.00	80.0	90.00	11000.0
cholesterol	70000.0	1.366871	0.680250	1.0	1.00	1.0	2.00	3.0
gluc	70000.0	1.226457	0.572270	1.0	1.00	1.0	1.00	3.0
smoke	70000.0	0.088129	0.283484	0.0	0.00	0.0	0.00	1.0
alco	70000.0	0.053771	0.225568	0.0	0.00	0.0	0.00	1.0
active	70000.0	0.803729	0.397179	0.0	1.00	1.0	1.00	1.0
cardio	70000.0	0.499700	0.500003	0.0	0.00	0.0	1.00	1.0

Data Cleaning and Pre-Processing:

Checking for any missing values in the data set by using isnull function

```
#checking if there are any missing values present in the dataset using isnull()

print(f"Missing values : {df_data.isnull().sum().any()}")
print(df_data.shape)
```

```
Missing values : False
(70000, 13)
```

Dropping ID and converting age to years and combining the weight and height to form BMI and then dropping the weight and height attributes.

```
#Categorized Data is preferable for decision trees.
#droping ID and converting age to years
#converting blood pressure
df = df_data
df = df.drop(['id'], axis=1)
df['age'] = df['age']/365

#It is possible to merge weight and height to get BMI
df['BMI'] = np.round(df['weight']/np.square(df['height']/100),1)
df = df.drop(['weight', 'height'], axis=1)
df.head()
df = df[(df["BMI"]>10) & (df["BMI"]<100)]
df = df[(df["ap_hi"]>20) & (df["ap_hi"]<250)]
df = df[(df["ap_lo"]>20) & (df["ap_lo"]<200)]
print(df.shape)
```

```
(68749, 11)
```

Converting the numeric values in cholesterol, and glucose to text then creating the dummy variables for cholesterol and glucose. After creating the dummy variables cholesterol and glucose are removed from the data set.

```
# numeric values in cholestrol, glucose are converted  to text
# Next Create Dummy Variable for Cholestrol and glucose
df['cholesterol'] = df['cholesterol'].map({ 1: 'Normal', 2: 'AboveNormal', 3: 'WellAboveNormal'})
df['gluc']=df['gluc'].map({ 1: 'Normal', 2: 'AboveNormal', 3: 'WellAboveNormal'})
dup = pd.get_dummies(df[['cholesterol','gluc']])
df = pd.concat([df,dup],axis=1)
```

```
df.drop(['cholesterol','gluc'],axis=1,inplace=True)
df.head()
```

	age	gender	ap_hi	ap_lo	smoke	alco	active	cardio	BMI	cholesterol_AboveNormal	cholesterol_Normal	cholesterol_WellAboveNormal	gluc_AboveNormal	gluc_Normal	gluc_WellAboveNormal
0	50.391781	2	110	80	0	0	1	0	22.0	0	1	0	0	1	0
1	55.419178	1	140	90	0	0	1	1	34.9	0	0	1	0	1	0
2	51.663014	1	130	70	0	0	0	1	23.5	0	0	1	0	1	0
3	48.282192	2	150	100	0	0	1	1	28.7	0	1	0	0	1	0
4	47.873973	1	100	60	0	0	0	0	23.0	0	1	0	0	1	0

Converting gender values from 1 and 2 into values of 1 and 0.

```
#round up to 0 and 1
df["gender"] = df["gender"] % 2
df.head()
```

	age	gender	ap_hi	ap_lo	smoke	alco	active	cardio	BMI	cholesterol_AboveNormal	cholesterol_Normal	cholesterol_WellAboveNormal	gluc_Abovel
0	50.391781	0	110	80	0	0	1	0	22.0	0	1	0	
1	55.419178	1	140	90	0	0	1	1	34.9	0	0	1	
2	51.663014	1	130	70	0	0	0	1	23.5	0	0	1	
3	48.282192	0	150	100	0	0	1	1	28.7	0	1	0	
4	47.873973	1	100	60	0	0	0	0	23.0	0	1	0	

Splitting the data set into 75% training and 25% testing.

```
B = df['cardio'] # assigning particular columns to varaible A and B for the creation of model
A = df.drop(['cardio'], axis=1)
A_train, A_test, B_train, B_test = train_test_split(A,B, test_size=0.25, random_state=0) # split
```

```
A_train
```

	age	gender	ap_hi	ap_lo	smoke	alco	active	BMI	cholesterol_AboveNormal	cholesterol_Normal	cholesterol_WellAboveNormal	gluc_AboveNorr
62698	54.249315	1	150	100	0	0	1	32.0	0	0	1	
55598	49.857534	1	110	70	0	0	1	29.7	0	1	0	
56216	57.698630	1	120	80	0	0	0	26.4	0	1	0	
23972	59.846575	0	150	90	1	0	0	25.2	1	0	0	
17966	45.572603	1	120	80	0	0	1	29.4	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	
21634	62.153425	0	150	90	0	1	1	27.7	0	1	0	
46728	49.876712	1	100	80	0	0	1	23.9	0	1	0	
43391	52.621918	0	110	70	0	1	0	21.9	0	1	0	
44358	50.454795	1	140	90	0	0	1	23.9	0	1	0	
69510	46.487671	0	120	80	0	0	0	25.2	0	1	0	

51561 rows × 14 columns

```
B_train
```

```
62698    1
55598    1
56216    0
23972    1
17966    0
        ..
21634    1
46728    0
43391    0
44358    1
69510    1
Name: cardio, Length: 51561, dtype: int64
```

```
A_test
```

	age	gender	ap_hi	ap_lo	smoke	alco	active	BMI	cholesterol_AboveNormal	cholesterol_Normal	cholesterol_WellAboveNormal	gluc_AboveN
32210	62.463014	0	120	80	0	0	0	27.3	0	1	0	
11709	50.493151	1	120	80	0	0	1	27.3	0	1	0	
55752	53.750685	1	130	80	0	0	1	29.4	0	1	0	
13734	52.605479	1	120	80	0	0	1	29.2	0	1	0	
40233	45.687671	1	110	70	0	0	1	23.2	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	
62577	58.197260	0	140	90	0	0	1	23.4	0	1	0	
65881	52.189041	1	120	80	0	0	1	28.1	0	1	0	
27875	57.624658	1	120	80	0	0	1	26.9	1	0	0	
35003	53.742466	1	110	70	0	0	1	24.2	0	1	0	
10870	39.775342	0	145	80	1	0	1	24.7	1	0	0	

```
B_test
```

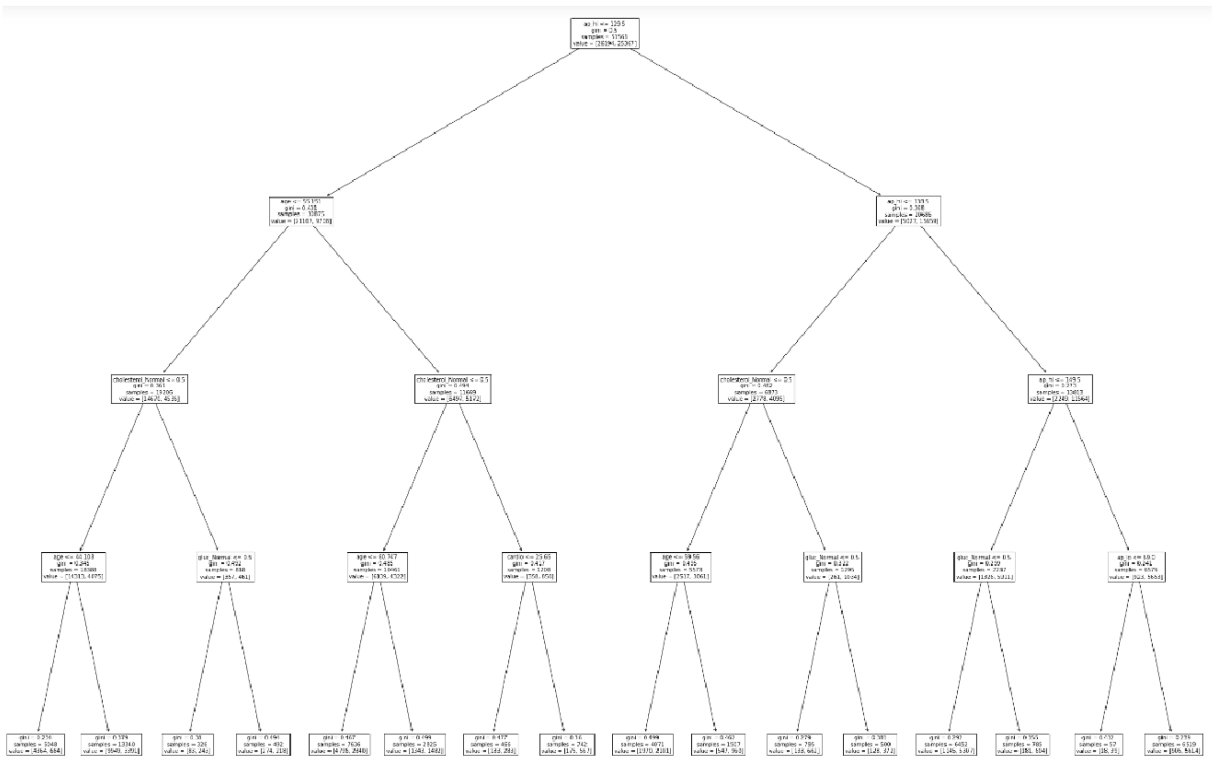
```
32210    1
11709    0
55752    0
13734    0
40233    0
        ..
62577    1
65881    0
27875    1
35003    0
10870    1
Name: cardio, Length: 17188, dtype: int64
```

Visualization of Decision Tree for GINI of depth 4

The below decision tree was constructed using the information gain and GINI index for every attribute.

```
#From the above train and test data we can determine the Decision tree using GINI with depth level 4
Decisiontree = tree.DecisionTreeClassifier(max_depth=4,criterion='gini')
Decisiontree = Decisiontree.fit(A_train,B_train)
B_pred_Gini=Decisiontree.predict(A_test)
fn = list(df.columns)
fig = p.subplots(figsize = (40,30))
print("Accuracy with Gini as hyper Parameter: : ",Decisiontree.score(A_test,B_test, sample_weight=None)*100)
tree.plot_tree(Decisiontree, feature_names = fn);
```

```
Accuracy with Gini as hyper Parameter: :  72.43425645799395
```



Confusion matrix (depth = 4)

```
confusion_matrix_gini=confusion_matrix(B_test, B_pred_Gini)
class_names = ['class 0', 'class 1']
print(confusion_matrix_gini)
print(classification_report(B_test, B_pred_Gini, target_names=class_names))
```

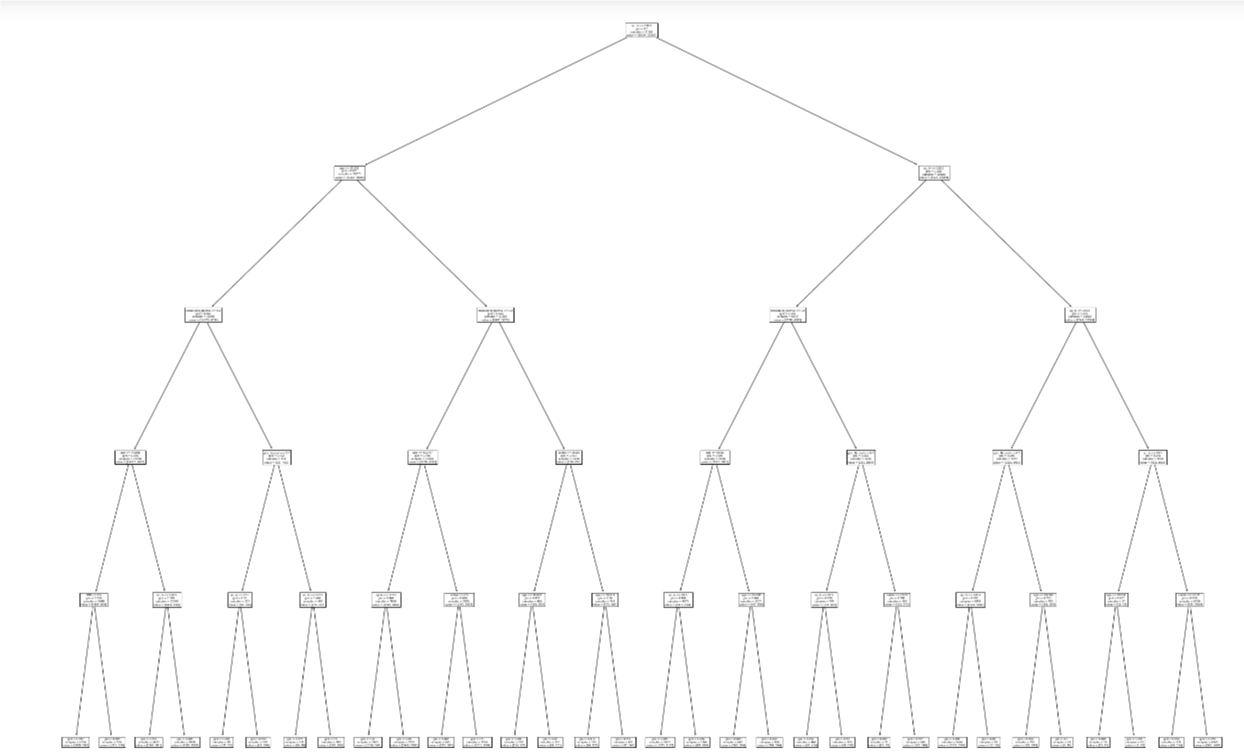
```
[[6259 2271]
 [2467 6191]]
              precision    recall  f1-score   support

     class 0       0.72      0.73      0.73      8530
     class 1       0.73      0.72      0.72      8658

    accuracy                           0.72     17188
   macro avg       0.72      0.72      0.72     17188
weighted avg       0.72      0.72      0.72     17188
```

Visualization of Decision Tree for GINI of depth 5

The below decision tree was constructed using the information gain and GINI index for every attribute.

```
#From the above train and test data we can determine the Decision tree using GINI with depth level 5
Decisiontree = tree.DecisionTreeClassifier(max_depth=5,criterion='gini')
Decisiontree = Decisiontree.fit(A_train,B_train)
B_pred_Gini=Decisiontree.predict(A_test)
fn = list(df.columns)
fig = p.subplots(figsize = (40,30))
print("Accuracy with Gini as hyper Parameter: ",Decisiontree.score(A_test,B_test, sample_weight=None)*100)
tree.plot_tree(Decisiontree, feature_names = fn);
```

```
Accuracy with Gini as hyper Parameter:  72.41680242029322
```



Confusion matrix (depth 5)

```
#confusion matrix
confusion_matrix_gini=confusion_matrix(B_test, B_pred_Gini)
class_names = ['class 0', 'class 1']
print(confusion_matrix_gini)
print(classification_report(B_test, B_pred_Gini, target_names=class_names))
```

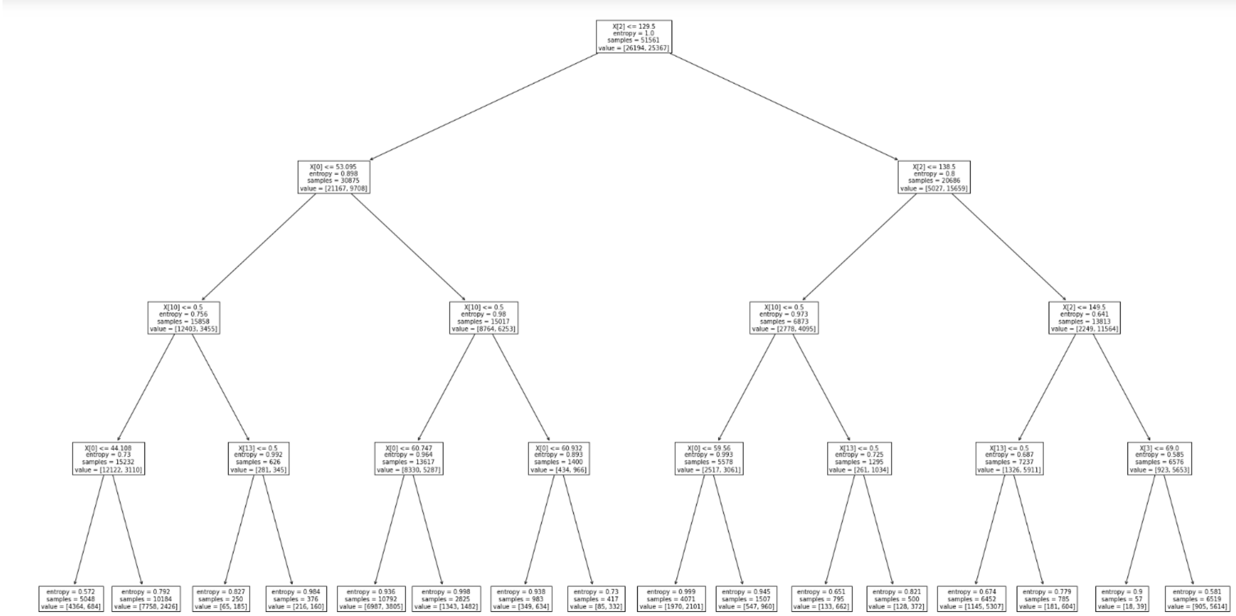
```
[[7033 1497]
 [3244 5414]]
              precision    recall  f1-score   support

     class 0       0.68      0.82      0.75      8530
     class 1       0.78      0.63      0.70      8658

    accuracy                           0.72     17188
   macro avg       0.73      0.72      0.72     17188
weighted avg       0.73      0.72      0.72     17188
```

Visualization of Decision Tree for ENTROPY of depth 4

The below decision tree was constructed using the ENTROPY for every attribute.

```
#From the above train and test data we can determine the Decision tree using Entropy with depth Level 4
decision_tree = tree.DecisionTreeClassifier(max_depth=4,criterion='entropy')
decision_tree = decision_tree.fit(A_train,B_train)
B_pred=decision_tree.predict(A_test)
p.figure(figsize=(35,21))
print("Accuracy with entropy as hyper parameter : ",decision_tree.score(A_test,B_test, sample_weight=None)*100)
tree.plot_tree(decision_tree);
```

```
Accuracy with entropy as hyper parameter :  72.40516639515941
```



Confusion matrix (depth 4)

```
confusion_matrix_entropy=confusion_matrix(B_test, B_pred)
class_names = ['class 0', 'class 1']
print(confusion_matrix_entropy)
print(classification_report(B_test, B_pred, target_names=class_names))
```

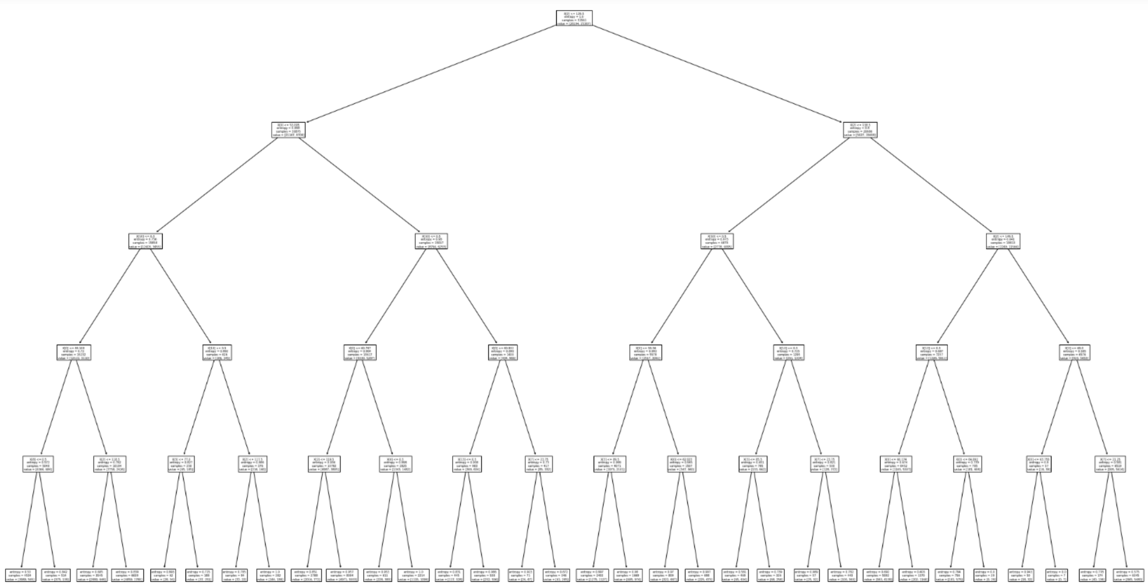
```
[[6237 2293]
 [2450 6208]]
              precision    recall  f1-score   support

     class 0       0.72      0.73      0.72      8530
     class 1       0.73      0.72      0.72      8658

    accuracy                           0.72     17188
   macro avg       0.72      0.72      0.72     17188
weighted avg       0.72      0.72      0.72     17188
```

Visualization of Decision Tree for ENTROPY of depth 5

The below decision tree was constructed using the ENTROPY for every attribute.

```
#From the above train and test data we can determine the Decision tree using Entropy with depth level 5
decision_tree = tree.DecisionTreeClassifier(max_depth=5,criterion='entropy')
decision_tree = decision_tree.fit(A_train,B_train)
B_pred=decision_tree.predict(A_test)
p.figure(figsize=(35,21))
print("Accuracy with entropy as hyper parameter : ",decision_tree.score(A_test,B_test, sample_weight=None)*100)
tree.plot_tree(decision_tree);
```

```
Accuracy with entropy as hyper parameter :  72.3877123574587
```



Confusion matrix (depth 5)

```
confusion_matrix_entropy=confusion_matrix(B_test, B_pred) # printing the confusion matri
class_names = ['class 0', 'class 1']
print(confusion_matrix_entropy)
print(classification_report(B_test, B_pred, target_names=class_names))
```

```
[[7011 1519]
 [3227 5431]]
              precision    recall  f1-score   support

     class 0       0.68      0.82      0.75      8530
     class 1       0.78      0.63      0.70      8658

    accuracy                           0.72     17188
   macro avg       0.73      0.72      0.72     17188
weighted avg       0.73      0.72      0.72     17188
```

Four most influential attributes on target attribute.

```
#Determine four most influential attributes on target attribute (with explanation) [5 points]
#using feature importance we have gathered the important variables that are influential on decision tree.

feature_dictionary= {}
for col, valu in sorted(zip(A_train.columns, Decisiontree.feature_importances_),key=lambda x:x[1],reverse=True):
  feature_dictionary[col]=valu
```

```
feature_dictionary = pd.DataFrame({'Feature':feature_dictionary.keys(),'Importance':feature_dictionary.values()})
```

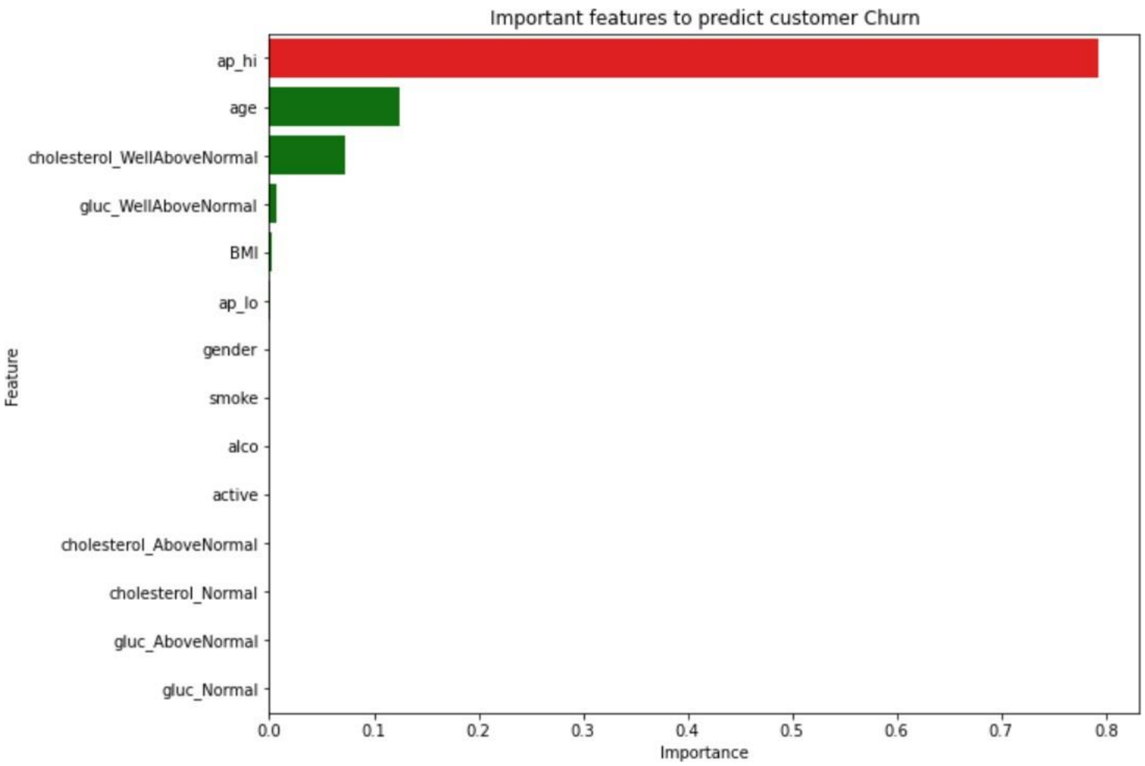
```
feature_dictionary
```

	Feature	Importance
0	ap_hi	0.792578
1	age	0.125337
2	cholesterol_WellAboveNormal	0.072949
3	gluc_WellAboveNormal	0.006549
4	BMI	0.002067
5	ap_lo	0.000519
6	gender	0.000000
7	smoke	0.000000
8	alco	0.000000
9	active	0.000000
10	cholesterol_AboveNormal	0.000000
11	cholesterol_Normal	0.000000
12	gluc_AboveNormal	0.000000
13	gluc_Normal	0.000000

The four most influence attributes on the target attribute are ap_hi (Systolic blood pressure), age, cholesterol_wellAboveNormal, and gluc_wellAboveNormal.

Systolic blood pressure (ap_hi) as the high impact on the customers compared to the other attributes. Nearly 80% of the customers are effect by the ap_hi attribute. Age is the second most influenced attribute for the customers. Cholesterol_wellAbovenormal and gluc_wellAboveNormal are third and forth attribute that effecting the customers both combined nearly to 0.8%

```
#4 infulential attributes are ap_hi, age, cholesterol_WellAboveNormal, gluc_WellAboveNormal
#visualization
values = feature_dictionary.Importance
idx = feature_dictionary.Feature
p.figure(figsize=(10,8))
clare = ['green' if (x < max(values)) else 'red' for x in values ]
sns.barplot(y=idx,x=values,palette=clare).set(title='Important features to predict customer Churn')
p.show()
```



Naïve Bayes:

Naive Bayes classification is a powerful algorithm for the classification. Simple Bayes and independent Bayes are other names for naive Bayes models. When predicting membership probabilities for each class, such as the likelihood that a given record or data point belongs to a specific class, the Nave Bayes Classifier applies the Bayes theorem. The most likely class is the one with the highest likelihood. A mathematical formula called the Bayes theorem is used to compute conditional probabilities.

The probability of an event happening given that another event has already happened is known as conditional probability.

Probability of B occurring given evidence A has already occurred

Probability of A occurring

$$P(\mathrm{A}|\mathrm{B}) = \frac{P(\mathrm{B}|A) \cdot P(A)}{P(B)}$$

Probability of A occurring given evidence B has already occurred

Probability of B occurring

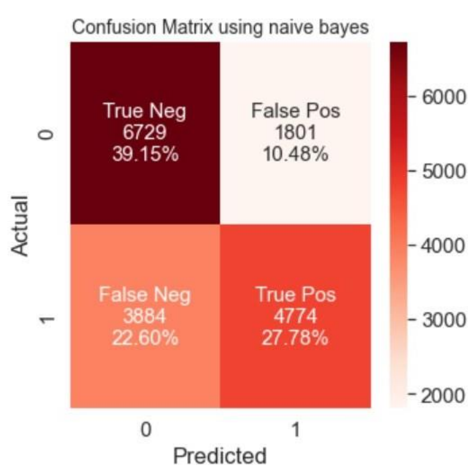
Gaussian Naïve Bayes:

The below is the confusion matrix using gaussian naïve bayes classifier.

```
#By using Naive bays classifier 'GaussianNB()' we can determine the confusion matrix and accuracy for the split data
bayes_model = GaussianNB()
bayes_model.fit(A_train, B_train)
B_pred = bayes_model.predict(A_test)
print(classification_report(B_test,B_pred))
con_mat=confusion_matrix(B_test,B_pred)
print("Gaussian Naive Bayes model accuracy(in %):", accuracy_score(B_test, B_pred)*100)
```

```
              precision    recall  f1-score   support

           0       0.63      0.79      0.70      8530
           1       0.73      0.55      0.63      8658

    accuracy                           0.67     17188
   macro avg       0.68      0.67      0.66     17188
weighted avg       0.68      0.67      0.66     17188

Gaussian Naive Bayes model accuracy(in %): 66.92459855713288
```

```
#heat map
groupnames = ['True Neg','False Pos','False Neg','True Pos']
groupcounts = ["{0:0.0f}".format(value) for value in
                con_mat.flatten()]
grouppercentages = ["{0:.2%}".format(value) for value in
                     con_mat.flatten()/np.sum(con_mat)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(groupnames,groupcounts,grouppercentages)]
labels = np.asarray(labels).reshape(2,2)
p.figure(figsize=(5,5))
sns.set(font_scale=1.4)
sns.heatmap(con_mat, annot=labels,annot_kws={"size": 15}, fmt='', cmap='Reds')
p.title("Confusion Matrix using naive bayes", fontsize=14);
p.ylabel("Actual")
p.xlabel("Predicted")
p.show()
```



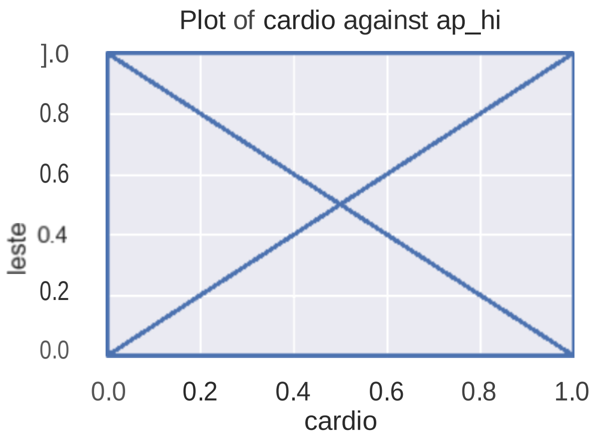
Comparison and interpretation of DT(Gini), DT (entropy) and Naïve bayes.

For the given dataset, GINI and ENTROPY has the same accuracy around 72%. There is a small difference in the true positive value, but both had the same true negative values. Real negative values are predicted as negative value for the entropy. There is a four percent different in the accuracy for the naïve bayes and Gini, entropy.

Visualizing the dataset for the target variable.

```
# storing the data frame 'df' in to temparrory varaible df_data1 ang plotting against target variable
df_data1 = df
p.plot(df_data1.age,df_data1.cardio)
p.xlabel('age')
p.ylabel('cardio')
p.title("Plot of cardio against ap_hi")
p.show()
```

```
# s towing the data frame dfl !n to tenpo rory voroTbt e dfi-_data1 ang p L offing aga!ns h target var! abL e
d -_data1 = d -
p.plot(df_data1.cardio,df_datal.cholesterol_Normal)
p.xlabel('cardio')
p.ylabel('cholesterol_Normal')
p.title("Plot of cardio against ap_hi")
p.show()
```



Contribution:

Sanjana Kondabathini	Worked on Classification – Decision Tree and Naive Bayes.
Meghana Kadali	Worked on Classification – Nearest Neighbors and report of knn.
Mucharla Rajashekar	Report on Classification – Decision Tree.

REFERENCES:

https://www.kaggle.com/code/alaaosamaawad/cardiovascular-disease-classification

https://www.geeksforgeeks.org/naive-bayes-classifiers/

https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html