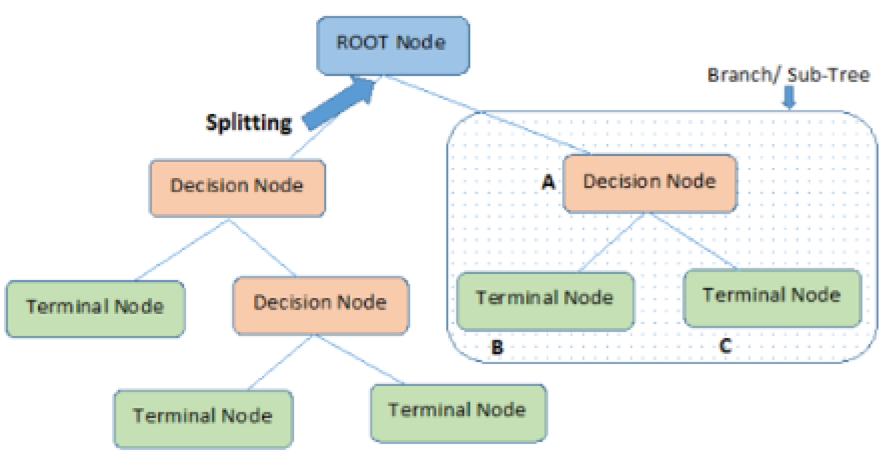
Day 18-100 Data Science

Decision Tree

- A decision tree is a non-parametric supervised learning algorithm for classification and regression tasks.
- It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaf nodes.
- Decision trees are used for classification and regression tasks, providing easy-to-understand models.
- It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.



Decision Tree Terminologies

- Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

- Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- Branch/Sub Tree: A tree formed by splitting the tree.
- Pruning: Pruning is the process of removing the unwanted branches from the tree.
- Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes

Attribute Selection Measures

Selecting the right Attribute Selection Measure (ASM) is crucial for building an effective decision tree in machine learning.

There are two popular techniques for ASM, which are:

- Entropy
- Information Gain
- Gini Index

Entropy

Entropy is the measure of the degree of randomness or uncertainty in the dataset.

- Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

```
- S= Total number of samples
- P(yes)= probability of yes
- P(no)= probability of no
```

Information Gain:

Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.

$$\text{Information Gain(H, A)} = H - \sum \frac{|H_V|}{|H|} H_v$$

where

- A is the specific attribute or class label
- |H| is the entropy of dataset sample S

- |HV| is the number of instances in the subset S that have the value v for attribute A

Gini Index

It is a measure of impurity or purity used while creating a decision tree.

$$\text{Gini Impurity} = 1 - \sum p_i^2$$

implementation of Decision Tree

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [ ]:

In [23]:
```
#load dataset
data = pd.read_csv("diabetes.csv")
data.head()
```

Out[23]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

In [24]:
```
data.isna().sum()
```

Out[24]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [25]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [27]:

```
data.describe()
```

Out[27]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [35]:

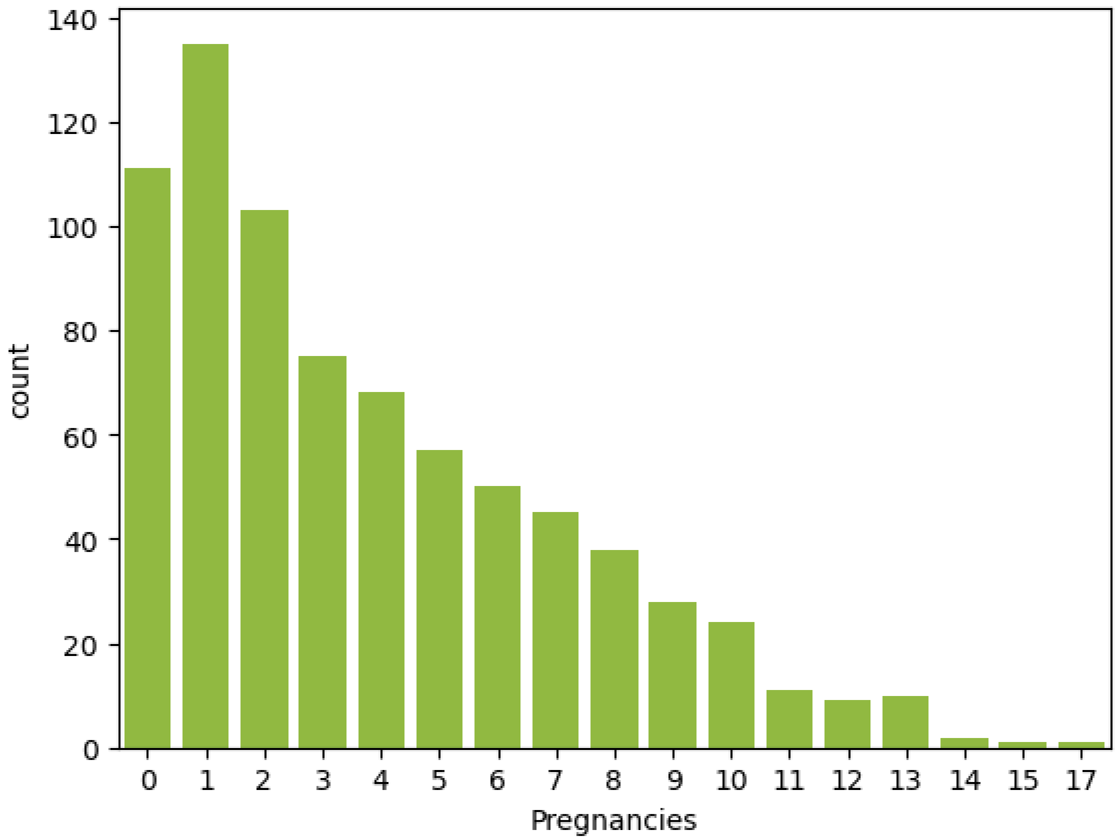
```
pre_value =data['Pregnancies'].value_counts()
pre_value
```

Out[35]:

```
Pregnancies
1     135
0     111
2     103
3      75
4      68
5      57
6      50
7      45
8      38
9      28
10     24
11     11
13     10
12      9
14      2
15      1
17      1
Name: count, dtype: int64
```

In [44]:
```
sns.countplot(data,x='Pregnancies',color='yellowgreen')
```

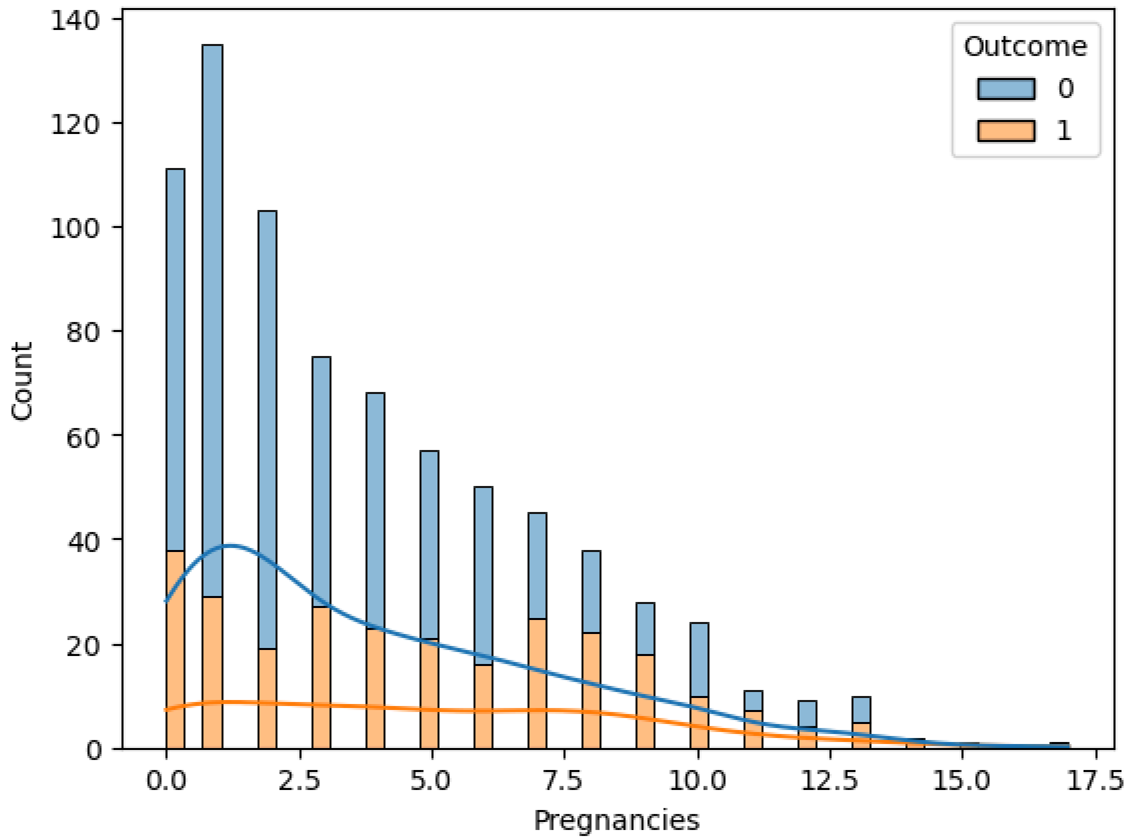
Out[44]:
```
<Axes: xlabel='Pregnancies', ylabel='count'>
```



In [ ]:

In [49]:
```
sns.histplot(data, x="Pregnancies", hue="Outcome", multiple="stack",bins = 50, kde=Tru
```

Out[49]:
```
<Axes: xlabel='Pregnancies', ylabel='Count'>
```



In [50]:
```
#model building
```

In [57]:
```
#split dataset in features and target variable

X = data.drop(['Outcome'], axis=1)

y = data['Outcome']
```

In [58]:
```
X.head()
```

Out[58]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

In [59]:
```
y.head()
```

Out[59]:
```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

In [60]:
```
# split X and y into training and testing sets

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_sta
```

In [61]:
```
# check the shape of X_train and X_test

X_train.shape, X_test.shape
```

Out[61]:
```
((514, 8), (254, 8))
```

In [62]:
```
# check data types in X_train

X_train.dtypes
```

Out[62]:
```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
dtype: object
```

In [63]:
```
# import DecisionTreeClassifier

from sklearn.tree import DecisionTreeClassifier
# instantiate the DecisionTreeClassifier model with criterion gini index

DTree = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=0)


# fit the model
DTree.fit(X_train, y_train)
```

Out[63]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, random_state=0)
```

In [66]:
```
y_pred = DTree.predict(X_test)
y_pred
```

Out[66]:
```
array([1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0,
       1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0], dtype=int64)
```

In [67]:
```
from sklearn.metrics import accuracy_score

print('Model accuracy score with criterion gini index: {0:0.4f}'. format(accuracy_scor
```

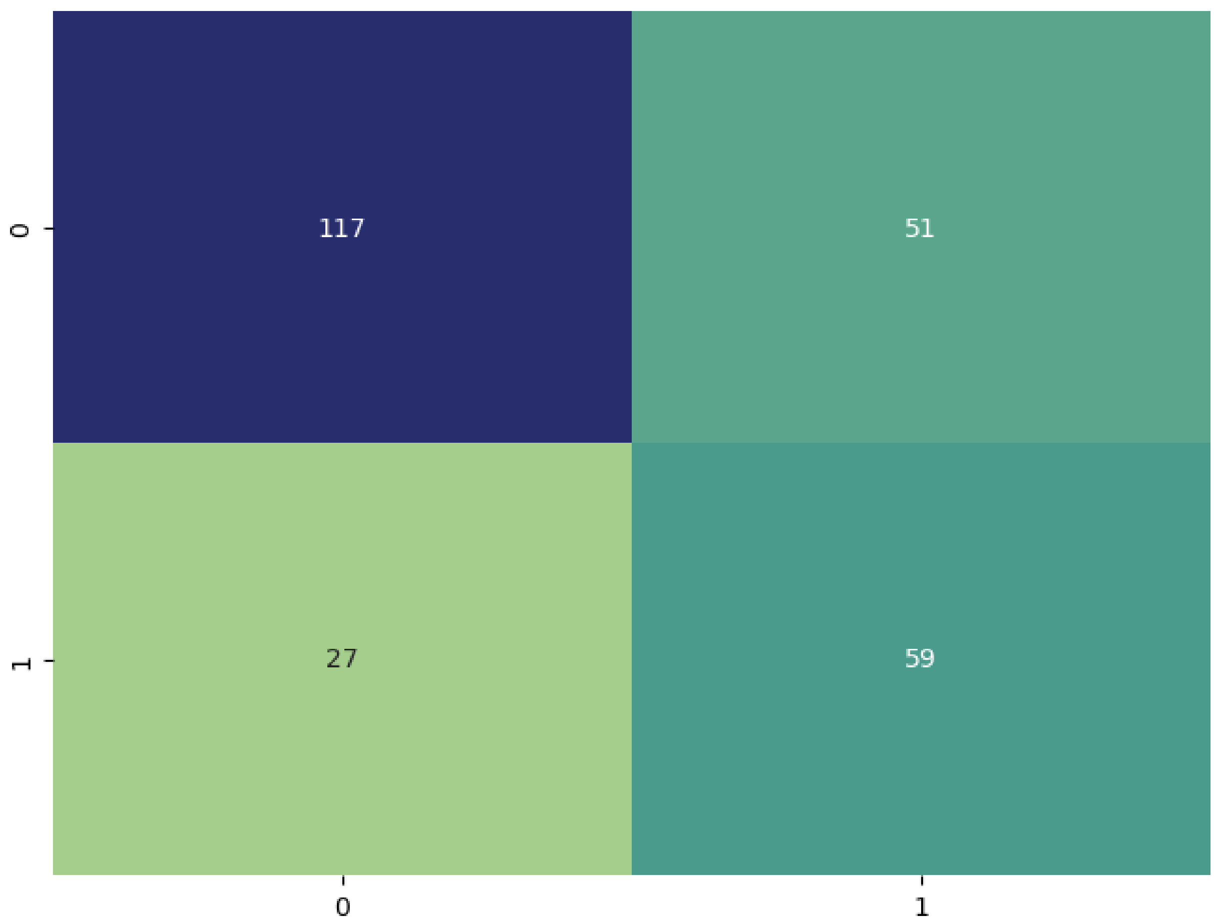
```
Model accuracy score with criterion gini index: 0.6929
```

In [71]:
```
# Create a confusion matrix
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test, y_pred)
conf_matrix
```

Out[71]:
```
array([[117,  51],
       [ 27,  59]], dtype=int64)
```

In [81]:
```
# Display the confusion matrix using Seaborn heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='crest', cbar=False)
```

Out[81]:
```
<Axes: >
```



In [84]:
```
# Create a classification report
from sklearn.metrics import classification_report
class_report = classification_report(y_test, y_pred)
print(class_report)
```

```
              precision    recall  f1-score   support

           0       0.81      0.70      0.75       168
           1       0.54      0.69      0.60        86

    accuracy                           0.69       254
   macro avg       0.67      0.69      0.68       254
weighted avg       0.72      0.69      0.70       254
```

In [72]:

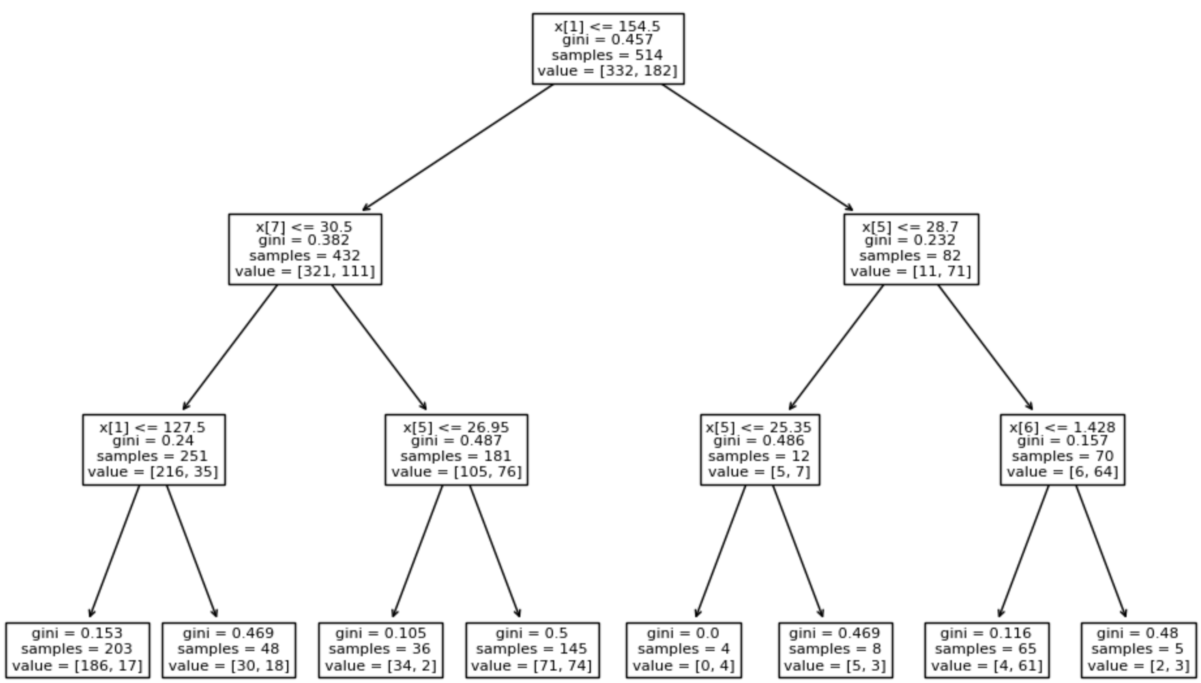
```python
plt.figure(figsize=(12,8))

from sklearn import tree

tree.plot_tree(DTree.fit(X_train, y_train))
```

Out[72]:

```
[Text(0.5, 0.875, 'x[1] <= 154.5\ngini = 0.457\nsamples = 514\nvalue = [332, 182]'),
 Text(0.25, 0.625, 'x[7] <= 30.5\ngini = 0.382\nsamples = 432\nvalue = [321, 111]'),
 Text(0.125, 0.375, 'x[1] <= 127.5\ngini = 0.24\nsamples = 251\nvalue = [216, 35]'),
 Text(0.0625, 0.125, 'gini = 0.153\nsamples = 203\nvalue = [186, 17]'),
 Text(0.1875, 0.125, 'gini = 0.469\nsamples = 48\nvalue = [30, 18]'),
 Text(0.375, 0.375, 'x[5] <= 26.95\ngini = 0.487\nsamples = 181\nvalue = [105, 76]'),
 Text(0.3125, 0.125, 'gini = 0.105\nsamples = 36\nvalue = [34, 2]'),
 Text(0.4375, 0.125, 'gini = 0.5\nsamples = 145\nvalue = [71, 74]'),
 Text(0.75, 0.625, 'x[5] <= 28.7\ngini = 0.232\nsamples = 82\nvalue = [11, 71]'),
 Text(0.625, 0.375, 'x[5] <= 25.35\ngini = 0.486\nsamples = 12\nvalue = [5, 7]'),
 Text(0.5625, 0.125, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(0.6875, 0.125, 'gini = 0.469\nsamples = 8\nvalue = [5, 3]'),
 Text(0.875, 0.375, 'x[6] <= 1.428\ngini = 0.157\nsamples = 70\nvalue = [6, 64]'),
 Text(0.8125, 0.125, 'gini = 0.116\nsamples = 65\nvalue = [4, 61]'),
 Text(0.9375, 0.125, 'gini = 0.48\nsamples = 5\nvalue = [2, 3]')]
```



In [ ]:

In [ ]:

In [ ]:

Follow us on Social Media

Linkedin: https://www.linkedin.com/company/eternaltek/about/?viewAsMember=true

Medium: https://medium.com/@eternaltek.info

WhatsApp Channel: https://whatsapp.com/channel/0029Va5onCbDjiOTi6D1vU36

Github: https://github.com/Vamsi-2203

In [ ]: