Name : Sidhantha Poddar

Reg : 17BCE2044 ¶

Topic: Classification – Decision Tree Experiment

Outcomes:(for given binary dataset)

1. Consider the ID3 algorithm using Information Gain
2. All Metrics should be display as outcome - Accuracy, Confusion matrix, F-Score, Precision, Recall and also the Decision Tree graph
3. Workout on sample dataset with paper and pencil and prove the same on system. This is for understanding and how it works.

Problem Statement:(for big dataset)

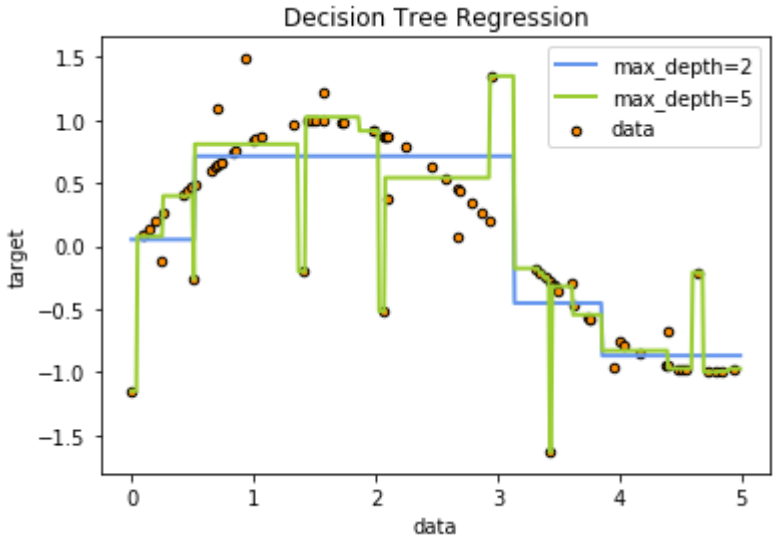
4. Now take real-time Indians Diabetes Database (given below) which is suitable for classification. Objective is to predict the onset of diabetes based on diagnostic measures.
5. Display all metrics.
6. Finally, do the cross validation and then display all the metrics.

In [57]:
```
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sn
import matplotlib.pyplot as plt
```

descision tree classification using random generated data set

In [58]:
```
print(__doc__)
# Import the necessary modules and libraries
import numpy as np
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt
# Create a random dataset
rng = np.random.RandomState(1)
X = np.sort(5 * rng.rand(80, 1), axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - rng.rand(16))
# Fit regression model
regr_1 = DecisionTreeRegressor(max_depth=2)
regr_2 = DecisionTreeRegressor(max_depth=5)
regr_1.fit(X, y)
regr_2.fit(X, y)
# Predict
X_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]
y_1 = regr_1.predict(X_test)
y_2 = regr_2.predict(X_test)
# Plot the results
plt.figure()
plt.scatter(X, y, s=20, edgecolor="black",
c="darkorange", label="data")
plt.plot(X_test, y_1, color="cornflowerblue",
label="max_depth=2", linewidth=2)
plt.plot(X_test, y_2, color="yellowgreen", label="max_depth=5", linewidth=2)
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```

```
Automatically created module for IPython interactive environment
```



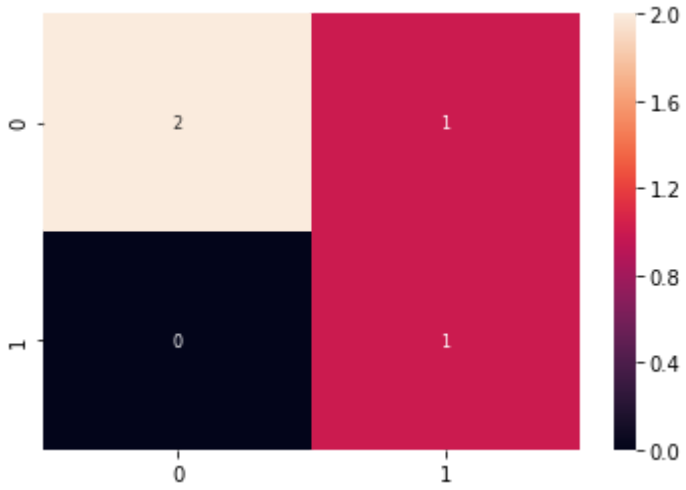
BINARY DATASET matrices, outcome and Tree representatiom

In [63]:
```
data=pd.read_csv('a.csv')
data
X=data.iloc[:,:4]
y=data.iloc[:,4]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

In [64]:
```
clf = tree.DecisionTreeClassifier()# defining classifier
clf = clf.fit(X_train, y_train) #fitting model
y_pred = clf.predict(X_test)
```

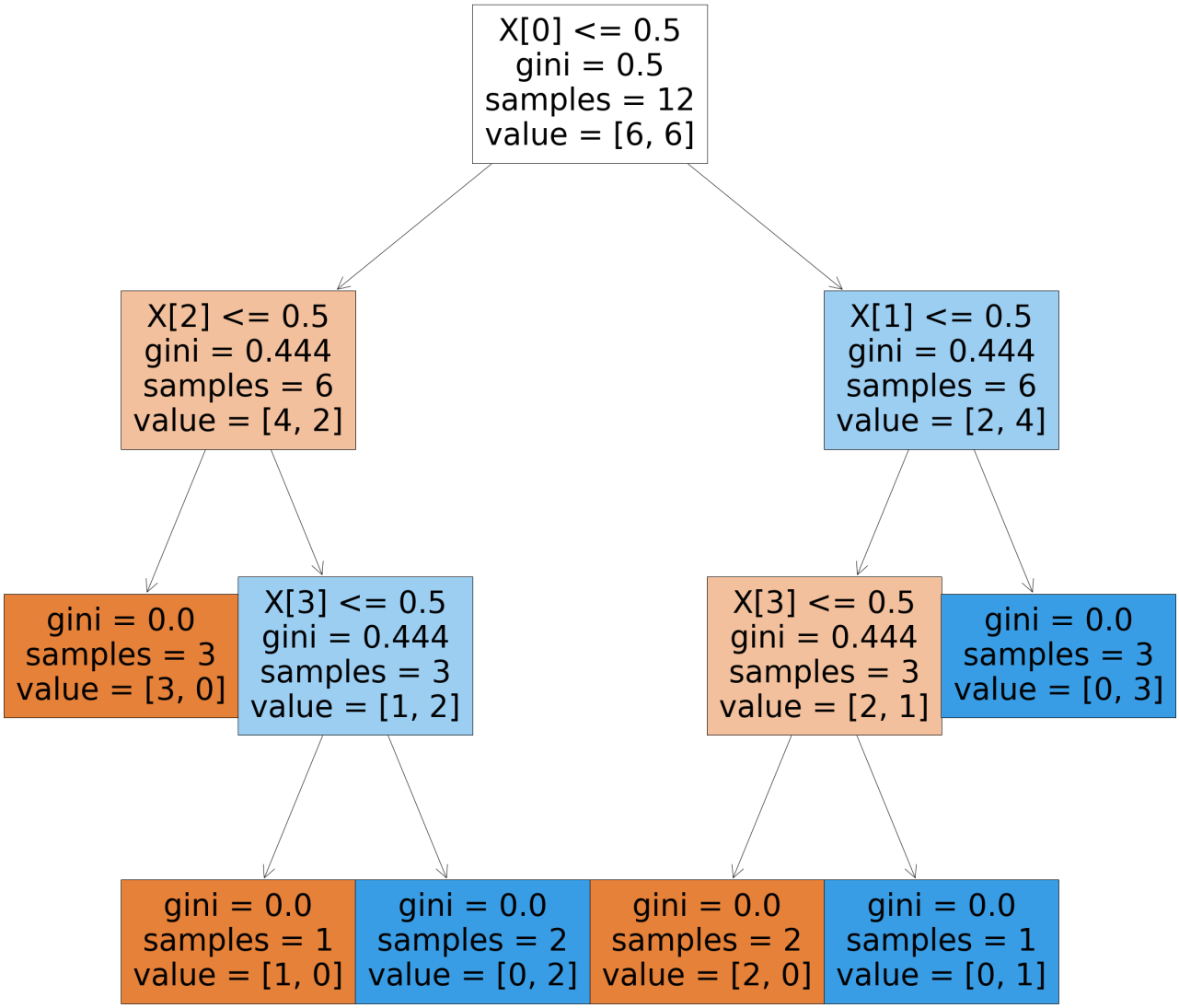
In [65]:
```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
sn.heatmap(confusion_matrix(y_test, y_pred), annot=True, annot_kws={"size": 8}
plt.show()
```

```
[[2 1]
 [0 1]]
              precision    recall  f1-score   support

           0       1.00      0.67      0.80         3
           1       0.50      1.00      0.67         1

    accuracy                           0.75         4
   macro avg       0.75      0.83      0.73         4
weighted avg       0.88      0.75      0.77         4
```



Decision Tree Graph

In [66]:
```
plt.figure(figsize=(40,40))
tree.plot_tree(clf, filled=True)
plt.show()
```



Descision tree Classification using Dibabetes Dataset

In [52]:
```
data=pd.read_csv('d.csv')
data
```

739	1	102	74	0	0	39.5	0.293	
740	11	120	80	37	150	42.3	0.785	
741	3	102	44	20	94	30.8	0.400	
742	1	109	58	18	116	28.5	0.219	
743	9	140	94	0	0	32.7	0.734	
744	13	153	88	37	140	40.6	1.174	
745	12	100	84	33	105	30.0	0.488	

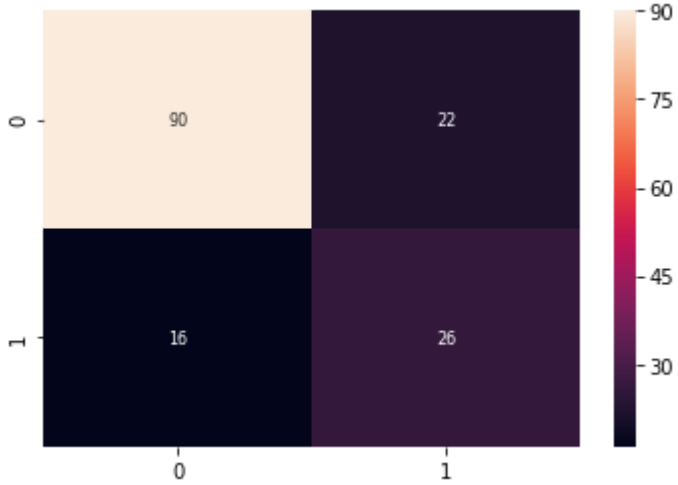
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
746	1	147	94	41	0	49.3	0.358	
747	1	81	74	41	57	46.3	1.096	
748	3	187	70	22	200	36.4	0.408	
749	6	162	62	0	0	24.3	0.178	

In [45]:
```
X=data.iloc[:,:8]
y=data.iloc[:,8]
X train, X test, y train, y test = train test split(X, y, test size=0.20)
```

In [46]:
```
clf = tree.DecisionTreeClassifier()# defining classifier
clf = clf.fit(X_train, y_train) #fitting model
y_pred = clf.predict(X_test)
```

In [55]:
```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
sn.heatmap(confusion_matrix(y_test, y_pred), annot=True, annot_kws={"size": 8}
plt.show()
```

```
[[90 22]
 [16 26]]
              precision    recall  f1-score   support

           0       0.85      0.80      0.83       112
           1       0.54      0.62      0.58        42

    accuracy                           0.75       154
   macro avg       0.70      0.71      0.70       154
weighted avg       0.77      0.75      0.76       154
```

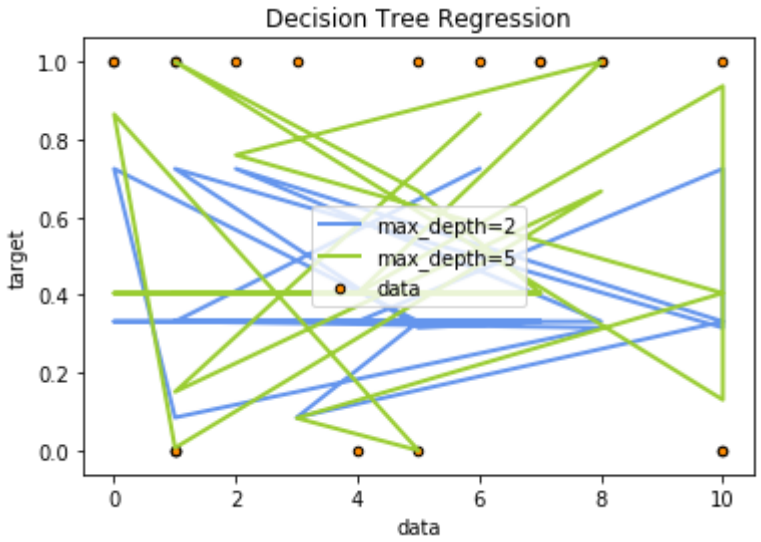


In [ ]:
```
plt.figure(figsize=(40,40))
tree.plot_tree(clf, filled=True)
plt.show()
```

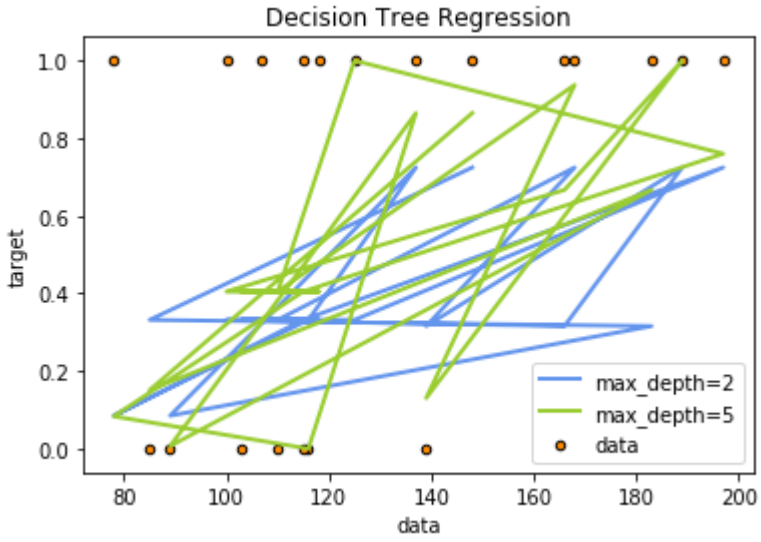
Descision Tree Regression

In [39]:
```
print(__doc__)
# Import the necessary modules and libraries
import numpy as np
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt


#X = X
y =  Y

# Fit regression model
regr_1 = DecisionTreeRegressor(max_depth=2)
regr_2 = DecisionTreeRegressor(max_depth=5)
regr_1.fit(X, Y)
regr_2.fit(X, Y)
# Predict
X_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]
y_1 = regr_1.predict(X)
y_2 = regr_2.predict(X)
# Plot the results
plt.figure()
plt.scatter(X.iloc[:20,0], y[:20], s=20, edgecolor="black",c="darkorange", lab
plt.plot(X.iloc[:20,0], y_1[:20], color="cornflowerblue",label="max_depth=2",
plt.plot(X.iloc[:20,0], y_2[:20], color="yellowgreen", label="max_depth=5", li
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```

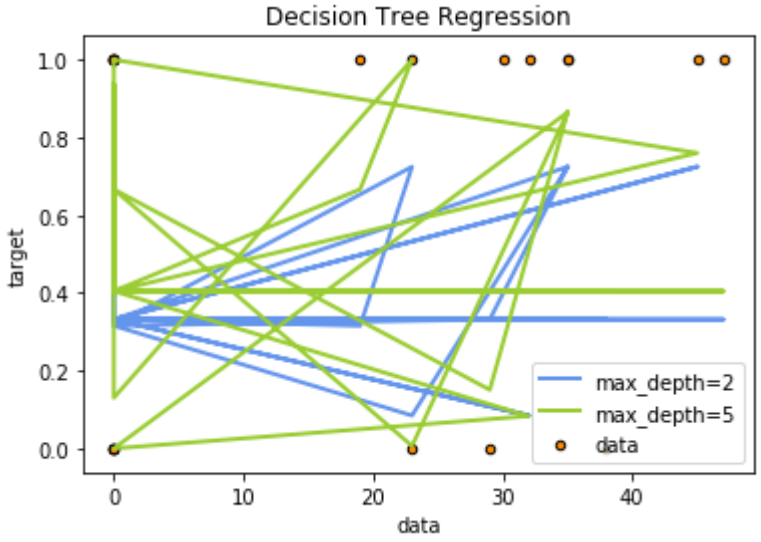
```
Automatically created module for IPython interactive environment
```



In [40]:
```
plt.figure()
plt.scatter(X.iloc[:20,1], y[:20], s=20, edgecolor="black",c="darkorange", lab
plt.plot(X.iloc[:20,1], y_1[:20], color="cornflowerblue",label="max_depth=2",
plt.plot(X.iloc[:20,1], y_2[:20], color="yellowgreen", label="max_depth=5", li
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```



In [41]:
```
plt.figure()
plt.scatter(X.iloc[:20,3], y[:20], s=20, edgecolor="black",c="darkorange", lab
plt.plot(X.iloc[:20,3], y_1[:20], color="cornflowerblue",label="max_depth=2",
plt.plot(X.iloc[:20,3], y_2[:20], color="yellowgreen", label="max_depth=5", li
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```



In [14]:
```
tree.plot tree(clf.fit(X, Y))
```

Out[14]:
```
[Text(241.92052165354332, 356.4, 'X[1] <= 127.5\ngini = 0.454\nsamples = 768\nvalue = [500, 268]'),
 Text(104.51820866141733, 330.0, 'X[7] <= 28.5\ngini = 0.313\nsamples = 485\nvalue = [391, 94]'),
 Text(54.92125984251969, 303.59999999999997, 'X[5] <= 45.4\ngini = 0.155\nsamples = 271\nvalue = [248, 23]'),
 Text(36.61417322834646, 277.2, 'X[5] <= 30.95\ngini = 0.139\nsamples = 267\nvalue = [247, 20]'),
 Text(11.716535433070867, 250.79999999999995, 'X[0] <= 7.5\ngini = 0.026\nsamples = 151\nvalue = [149, 2]'),
 Text(7.811023622047244, 224.39999999999998, 'X[6] <= 0.672\ngini = 0.013\nsamples = 150\nvalue = [149, 1]'),
 Text(3.905511811023622, 197.99999999999997, 'gini = 0.0\nsamples = 131\nvalue = [131, 0]'),
 Text(11.716535433070867, 197.99999999999997, 'X[6] <= 0.686\ngini = 0.1\nsamples = 19\nvalue = [18, 1]'),
 Text(7.811023622047244, 171.59999999999997, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(15.622047244094489, 171.59999999999997, 'gini = 0.0\nsamples = 18\nva
```

Tree Structure

In [21]:
```
plt.figure(figsize=(40,40))
tree.plot_tree(clf, filled=True)
plt.show()
```