Decision Tree Classification: Diabetes

Problem Statement:

> Create the model that can classify diabetes of the patients using Decision Tree Classifier, Predict diabetes for the follwing patient parameters, generate the classification report of the model, draw confusion matrix and create a tree of this dataset. Apply decision tree optimizers and try to come up with better accuracy.
>
> Apply Bagging (Ensemble Learning) Technique on this model and find accuracy.

Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age
1	85	66	29	0	26.6	0.351	31

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

In [2]:

```
df = pd.read_csv('diabetes.csv')
df.head()
```

Out[2]:

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

In [3]:

```
df.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 [4]:

```
df.shape
```

Out[4]:

```
(768, 9)
```

In [5]:

```
X = df.drop('Outcome',axis=1)
y = df['Outcome']
```

In [23]:

```
X_train, X_test, y_train, y_test = train_test_split(X.values, y.values, test_size=0.20,
```

In [24]:

```
print(len(X_train))
print(len(X_test))
print(len(y_train))
print(len(y_test))
```

```
614
154
614
154
```

In [25]:

```
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

Out[25]:

```
0.7402597402597403
```

In [26]:

```
y_pred = model.predict(X_test)
y_pred
```

Out[26]:

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

In [27]:

```
y_test
```

Out[27]:

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

In [28]:

```
accuracy_score(y_test, y_pred)
```

Out[28]:

```
0.7402597402597403
```

In [29]:

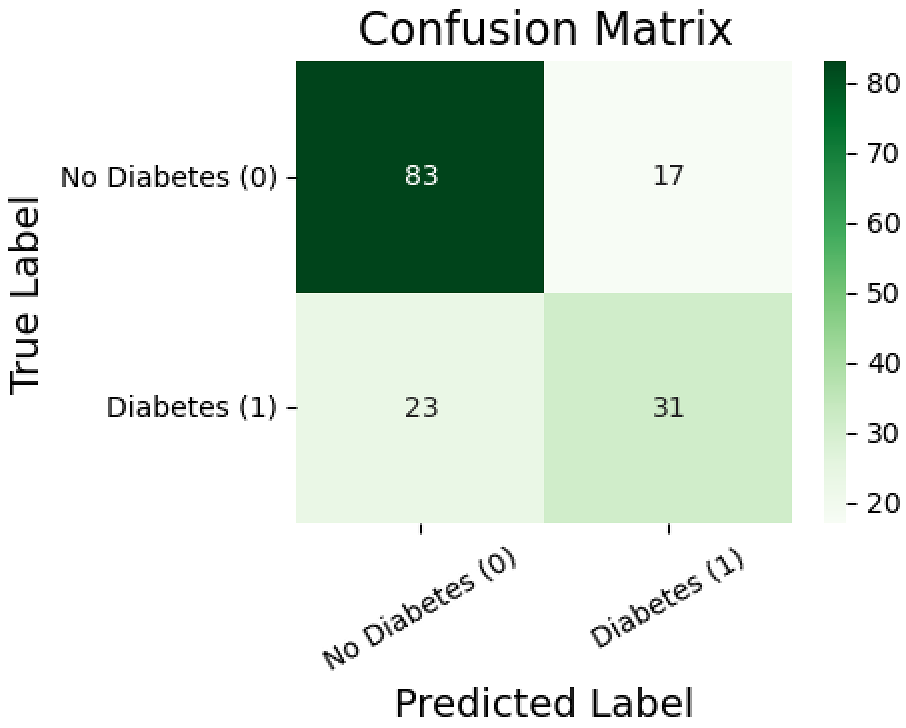
```
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[29]:

```
array([[83, 17],
       [23, 31]], dtype=int64)
```

In [30]:

```
plt.figure(figsize=(4,3))
g = sns.heatmap(cm, cmap='Greens', annot=True)
g.set_xticklabels(labels=['No Diabetes (0)' , 'Diabetes (1)'], rotation=30)
g.set_yticklabels(labels=['No Diabetes (0)' , 'Diabetes (1)'], rotation=0)
plt.ylabel('True Label', fontsize=14)
plt.xlabel('Predicted Label', fontsize=14)
plt.title('Confusion Matrix', fontsize=16)
plt.show()
```



In [31]:

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

```
              precision    recall  f1-score   support

           0       0.78      0.83      0.81       100
           1       0.65      0.57      0.61        54

    accuracy                           0.74       154
   macro avg       0.71      0.70      0.71       154
weighted avg       0.73      0.74      0.74       154
```

In [32]:

```
model.predict([[1, 85, 66, 29, 0,26.6 ,0.351, 31]])
```

Out[32]:

```
array([0], dtype=int64)
```

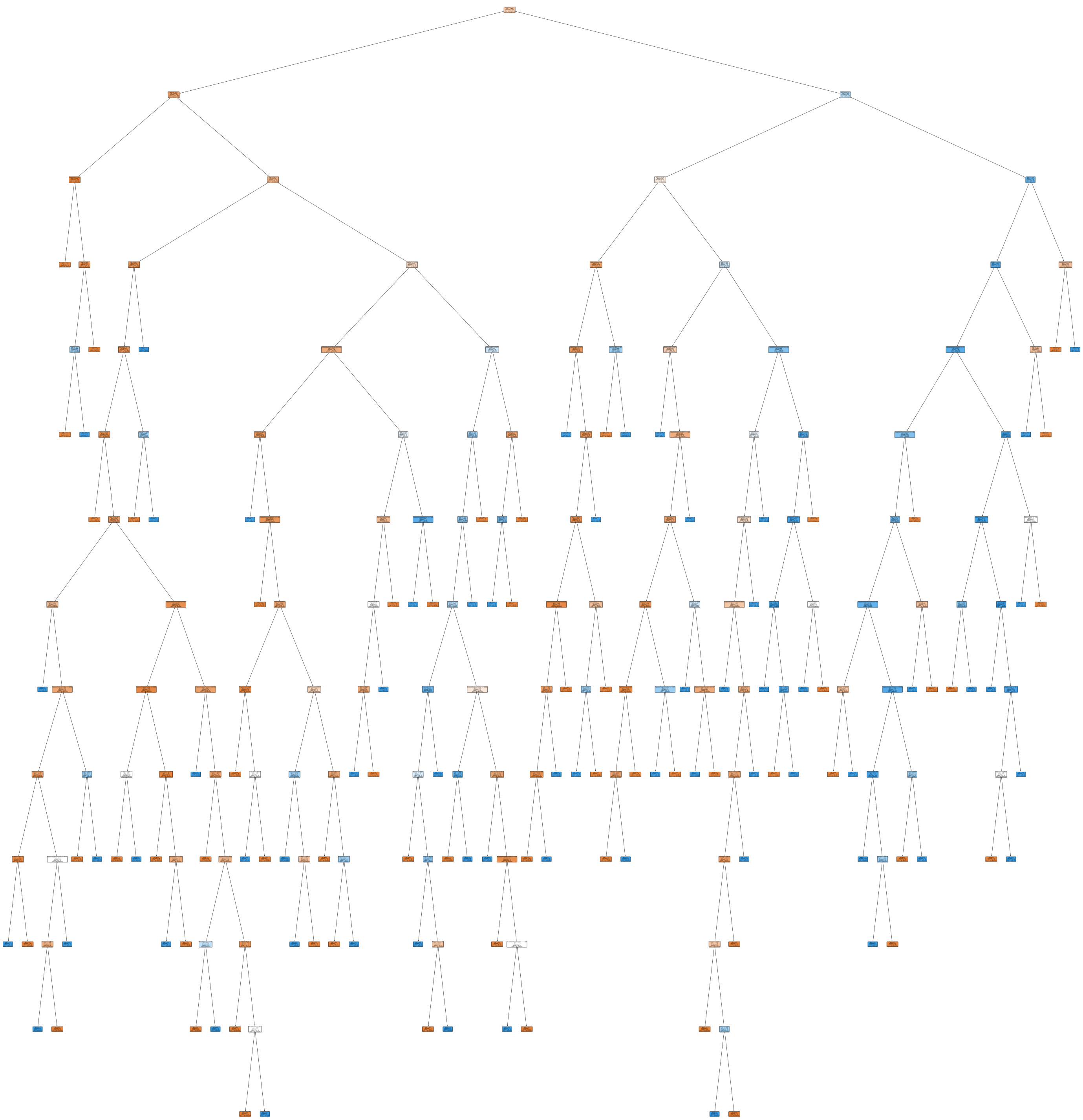
In [33]:

```
if model.predict([[1, 85, 66, 29, 0,26.6 ,0.351, 31]])[0] == 1:
  print("Having diabetes")
else:
  print("Not having diabetes")
```

```
Not having diabetes
```

In [34]:

```
from sklearn import tree
from matplotlib import pyplot as plt
plt.figure(figsize=(90,99))
tree.plot_tree(model,
               feature_names=X.columns,
               class_names={0:'No diabetes', 1:'Diabetes'},
               filled=True,
               rounded=True,
               fontsize=5)
#plt.savefig("fig.png", dpi=300)
plt.show()
```



Optimizing Decision Tree Performance

1. Gini index = Gini
2. Information gain = Entropy

In [39]:

```
modelO = DecisionTreeClassifier(criterion="entropy", max_depth=3) # default="gini"
modelO.fit(X_train,y_train)
modelO.score(X_test, y_test)
```

Out[39]:

```
0.7987012987012987
```

In [43]:

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

bag_model = BaggingClassifier(
    estimator=DecisionTreeClassifier(),
    n_estimators=70,
    max_samples=0.8,
    oob_score=True,
    random_state=10
)
bag_model.fit(X_train, y_train)
bag_model.score(X_test, y_test)
```

Out[43]:

```
0.7662337662337663
```