VELLORE INSTITUTE OF TECHNOLOGY
CSE4020 Machine Learning
Lab Assessment - 4

17BCE0581

SATYAM SINGH CHAUHAN

Classification - Decision Tree

Importing the Required Libraries

In [1]:

```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifie
r
from sklearn.model_selection import train_test_split # Import train_test_split f
unction
from sklearn import metrics #Import scikit-learn metrics module for accuracy cal
culation
```

Loading Data

In [6]:

```
col_names = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin',
'BMI','DiabetesPedigreeFunction','Age','label']
# load dataset
pima = pd.read_csv("diabetes.csv", header=None, names=col_names)

#Inspecting dataset
pima.head()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

Feature Selection

In [22]:

```
#split dataset in features and target variable
feature_cols = ['Pregnancies', 'Insulin', 'BMI', 'Age','Glucose','BloodPressure'
,'DiabetesPedigreeFunction']

# Features
X = pima[feature_cols]

# Target variable
y = pima.label
```

Splitting Data

In [23]:

```
# Split dataset into training set and test set

# 70% training and 30% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_
state=1)
```

Building Decision Tree Model

In [24]:

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

Evaluating Model

In [25]:

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.6796536796536796
```

In [26]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Out[26]:

```
array([[113,  33],
       [ 41,  44]])
```

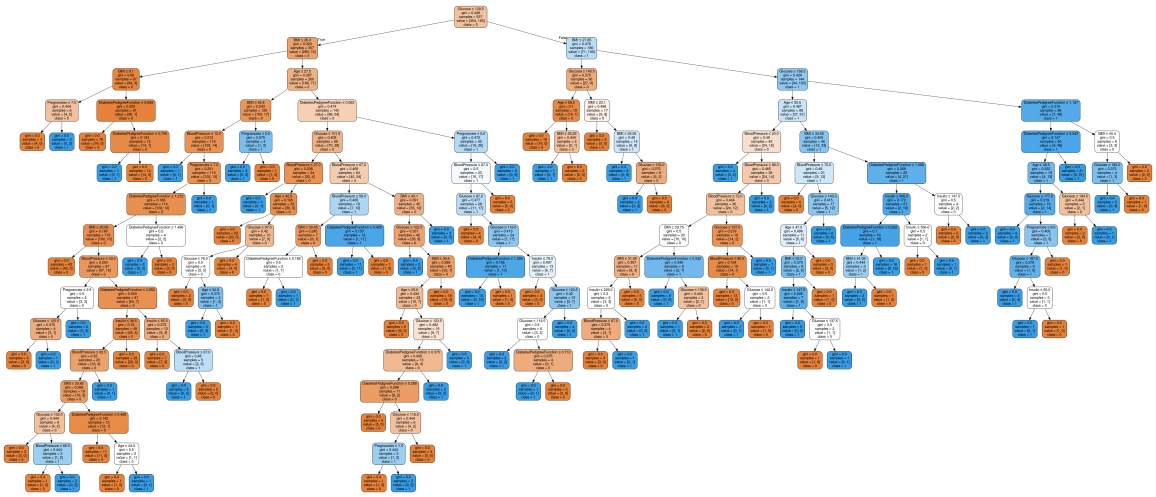
Visualizing Decision Trees

In [27]:

```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names
=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

Out[27]:



Optimizing Decision Tree Performance

In [28]:

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

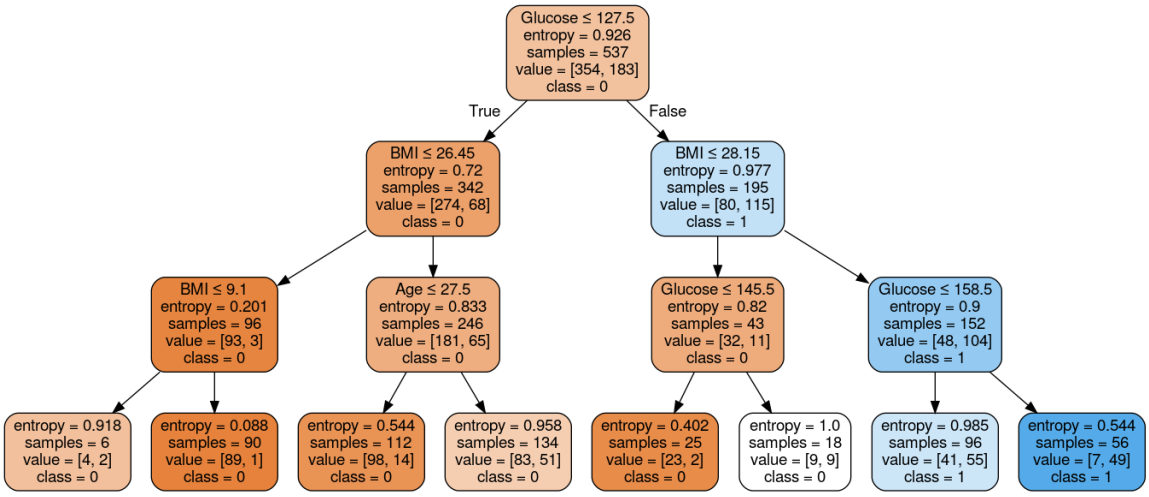
```
Accuracy: 0.7705627705627706
```

Visualizing Decision Trees

In [29]:

```
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True, feature_names = feature_cols,class_name
s=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

Out[29]:



Displaying Confusion Matrix

In [30]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Out[30]:

```
array([[124,  22],
       [ 31,  54]])
```

precision_recall_fscore_support

In [49]:

```
from sklearn.metrics import precision_recall_fscore_support
print('Binary: ',precision_recall_fscore_support(y_test, y_pred, average='binar
y'))
print('Micro: ',precision_recall_fscore_support(y_test, y_pred, average='micro'
))
print('Macro: ',precision_recall_fscore_support(y_test, y_pred, average='macro'
))
print('Weighted: ',precision_recall_fscore_support(y_test, y_pred, average='weig
hted'))
```

```
Binary:  (0.7105263157894737, 0.6352941176470588, 0.670807453416149
1, None)
Micro:  (0.7705627705627706, 0.7705627705627706, 0.7705627705627706,
None)
Macro:  (0.7552631578947369, 0.7423045930701047, 0.74736385959844, N
one)
Weighted:  (0.7670767828662566, 0.7705627705627706, 0.76758005343878
51, None)
```

F-Score

In [53]:

```
from sklearn.metrics import f1_score
print('Binary: ',f1_score(y_test, y_pred, average='binary'))
print('Micro: ',f1_score(y_test, y_pred, average='micro'))
print('Macro: ',f1_score(y_test, y_pred, average='macro'))
print('Weighted: ',f1_score(y_test, y_pred, average='weighted'))
```

```
Binary:  0.6708074534161491
Micro:  0.7705627705627706
Macro:  0.74736385959844
Weighted:  0.7675800534387851
```

Conclusion

When we observe the results we get the Accuracy of 0.77 that is 77%. This is the Accuracy after we Optimized the Decision Tree Performance. We optimized the performance of 67.9% to 77%.

In [ ]: