#!/usr/bin/env python

# coding: utf-8

# In[1]:

# Load libraries

import pandas as pd

from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier

from sklearn.model\_selection import train\_test\_split # Import train\_test\_split function

from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

fr

# In[2]:

# load dataset

from pandas import DataFrame

pima = DataFrame.from\_csv('D://ML//diabetes.csv')

pima.head()

# In[3]:

# Get Predictor Names (all but 'class')

X = list(pima.columns)

print("List of Attributes:", X)

# In[6]:

X.remove('Outcome') #Remove the class attribute

print("Predicting Attributes:", X)

# In[5]:

X=pima[X]

y=pima.Outcome

# In[7]:

# Split dataset into training set and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=1) # 70% training and 30% test

# Create Decision Tree classifer object

clf = DecisionTreeClassifier()

# Train Decision Tree Classifer

clf = clf.fit(X,y)

# In[8]:

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)

graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

Image(graph.create\_png())

# In[9]:

# Create Decision Tree classifer object

clf = DecisionTreeClassifier(criterion="entropy", max\_depth=3)

# Train Decision Tree Classifer

#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))

# In[10]:

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)

graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

graph.write\_png('diabetes.png')

Image(graph.create\_png())

# In[ ]: