Praticle 5 : Implementation of Decision Tree Classification Algorithms

In [30]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder

# Load dataset
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedi
pima = pd.read_csv("diabetes.csv", header=0, names=col_names)

# Preprocessing: Encode label column
label_encoder = LabelEncoder()
pima['label'] = label_encoder.fit_transform(pima['label'])

# Split dataset into features (X) and target (y)
X = pima.drop('label', axis=1)
y = pima['label']
```

In [31]:
```
# 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,

# Create Decision Tree classifier object
clf = DecisionTreeClassifier()

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

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

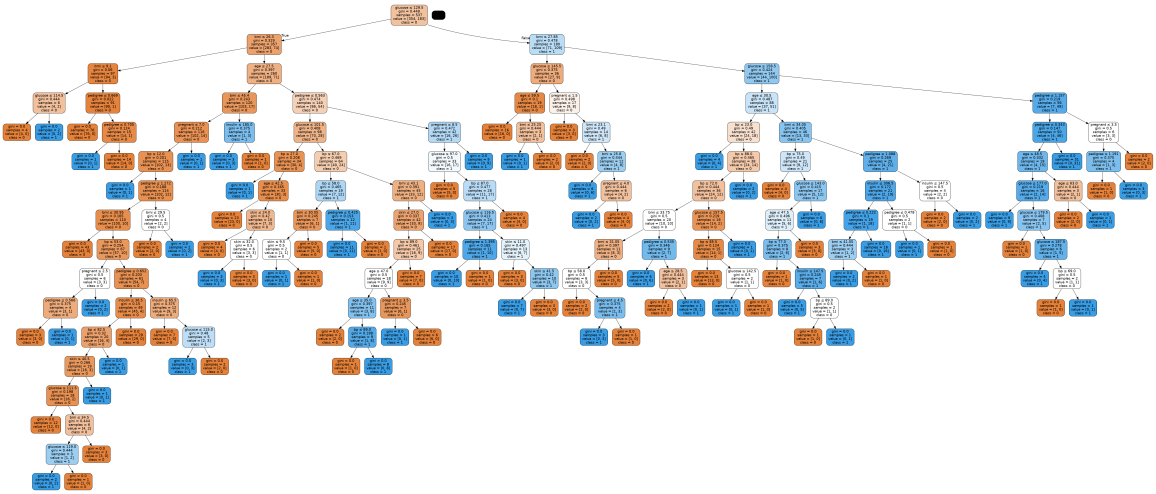
In [32]:
```
# Evaluate the model
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7056277056277056
```

In [33]:
```
from sklearn.tree import export_graphviz
from 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=col_names[:-1], cl
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

Out[33]:



Optimizing Decision Tree Performance

In [34]:
```
# 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)
```

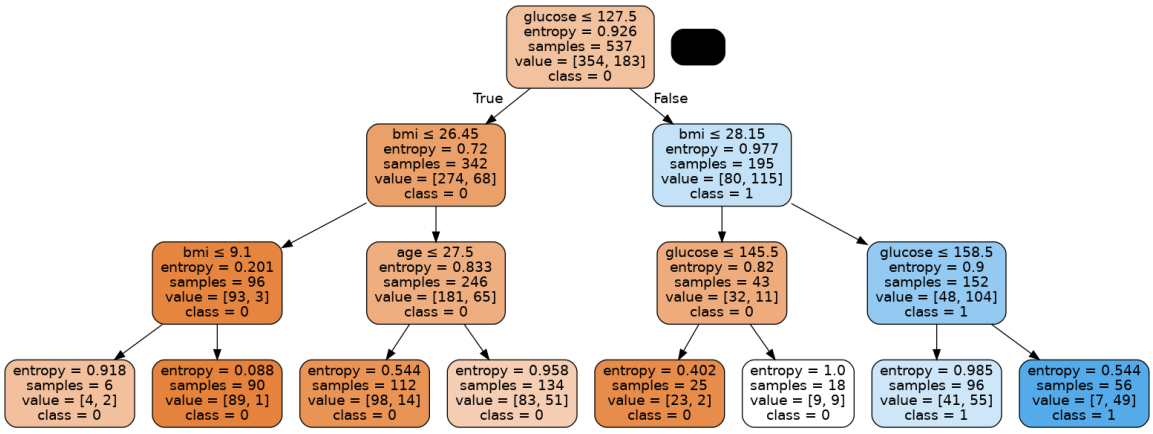
In [35]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7705627705627706
```

In [36]:
```
from sklearn.tree import export_graphviz
from 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=col_names[:-1], cl
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

Out[36]:



Some Steps Were Followed to create and optimize the tree:

- The necessary libraries and modules are imported, including export_graphviz for visualizing the decision tree, StringIO for creating a file-like object in memory, Image for displaying the image in Jupyter Notebook, and pydotplus for handling the graph data.
- The dataset is loaded from a CSV file into a pandas DataFrame, with column names specified.
- The dataset is split into training and test sets using the train_test_split function from scikit-learn. 70% of the data is used for training, and 30% is reserved for testing.
- A decision tree classifier object is created using the default settings.
- The decision tree classifier is trained on the training data using the fit method.
- The classifier is used to predict the labels for the test dataset using the predict method.
- The accuracy of the model is calculated by comparing the predicted labels with the actual labels from the test dataset.
- The decision tree is visualized using the export_graphviz function to generate the dot data, which is then converted into a PNG image using pydotplus. The image is saved as "diabetes.png" and displayed using Image.
- The decision tree classifier is redefined with some additional settings, such as using entropy as the criterion and setting a maximum depth of 3.
- Steps 5-8 are repeated with the updated decision tree classifier to visualize the new tree and calculate its accuracy.