In [1]:

```
import pandas as pd
import numpy as np
from sklearn import tree
import graphviz
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score

from IPython.display import IFrame

# Decision tree demo for Titanic data

# read data (replace with your own path)
df = pd.read_csv('titanic_v2.csv', sep=';')

# drop rows with missing values
df.dropna(axis=0, how='any', inplace=True)
df.head(10)
```

Out[1]:

	pclass	sex	age	survived
0	1	female	29.0000	1
1	1	male	0.9167	1
2	1	female	2.0000	0
3	1	male	30.0000	0
4	1	female	25.0000	0
5	1	male	48.0000	1
6	1	female	63.0000	1
7	1	male	39.0000	0
8	1	female	53.0000	1
9	1	male	71.0000	0

In [2]:

```
# type conversions
# Note that DecisionTreeClassifier can't use strings in explanatory variables
df['sex'] = df['sex'].replace(['male', 'female'],[1,2])
df.dtypes
```

Out[2]:

```
pclass        int64
sex           int64
age         float64
survived      int64
dtype: object
```

In [3]:

```
# save column headings into a list
colnames = df.columns
colnames
```

Out[3]:

```
Index(['pclass', 'sex', 'age', 'survived'], dtype='object')
```

In [4]:

```
# extract explanatory variables into a data frame
X = df.loc[:, 'pclass':'age']

# extract response variable (class variable) into a series
Y = df.loc[:, 'survived']
```

In [5]:

```
# decision tree classification
classifier = tree.DecisionTreeClassifier(max_depth=2)
classifier.fit(X,Y)
```

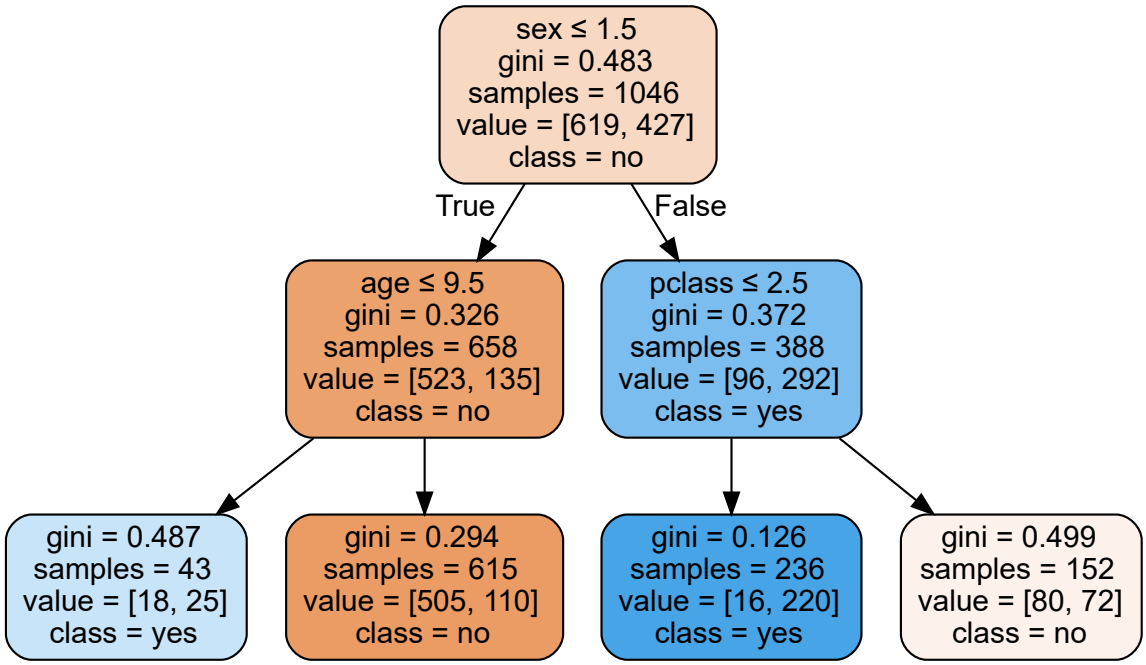
Out[5]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=2,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [6]:

```
# visualize
dot_data = tree.export_graphviz(classifier, out_file=None,
                                feature_names=colnames[:3],
                                class_names=['no','yes'],
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

Out[6]:



In [7]:

```
# predict
Y_pred = classifier.predict(X)

# output confusion matrix
cm = confusion_matrix(Y, Y_pred)
print("Confusion matrix:\n",cm)

accuracy = accuracy_score(Y, Y_pred)
print("Accuracy calculated from the training set = %.3f" % (accuracy))

print(classification_report(Y, Y_pred, target_names=['no', 'yes']))
```

```
Confusion matrix:
 [[585  34]
 [182 245]]
Accuracy calculated from the training set = 0.793
              precision    recall  f1-score   support

          no       0.76      0.95      0.84       619
         yes       0.88      0.57      0.69       427

    accuracy                           0.79      1046
   macro avg       0.82      0.76      0.77      1046
weighted avg       0.81      0.79      0.78      1046
```

In [8]:

```
# cross-validate
# number of folds
k = 10
scores = cross_val_score(estimator=classifier,
                         X=X,
                         y=Y,
                         scoring="accuracy",
                         cv=k)
print("Accuracies from %d individual folds:" % k)
print(scores)
print("Accuracy calculated using %d-fold cross validation = %.3f" % (k, scores.mean()))
```

```
Accuracies from 10 individual folds:
[0.83809524 0.86666667 0.84761905 0.82857143 0.76190476 0.83809524
 0.8        0.59615385 0.52884615 0.61165049]
Accuracy calculated using 10-fold cross validation = 0.752
```

In [ ]: