Model Evaluation

I used a car evaluation dataset. It has 6 features and is to evaluates cars according to the features. The original dataset has four target variables: unacc, acc, good and vgood. I simply converted the unacc to 0 and the rest of them to 1.

In [33]:

```
import pandas as pd
import random
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz
from sklearn.metrics import precision_score,recall_score,accuracy_score,confusion_matrix,roc_curve,classification_report,pr
from graphviz import Source
import graphviz
from IPython.display import display,SVG
import matplotlib.pyplot as plt
random.seed(42)

# Import dataset
df = pd.read_csv('car_evaluation.csv')

# Rename column names
col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
df.columns = col_names

# Split the features and target
y = df['class']
X_raw = df.drop('class', axis = 1)
X = pd.get_dummies(X_raw)
```

1. Split the dataset into training set and test set (80, 20)

In [34]:
```
# Split the dataset into training set and test set (80, 20)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

2. Using scikit-learn’s DecisionTreeClassifier, train a supervised learning model that can be used to gnerate predictions for your data.

In [35]:
```
# Import the classifier from sklearn
model = DecisionTreeClassifier(max_depth = 6)
model.fit(X_train,y_train)

# Making predictions
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)
```

2.1. The accuracy of your model on the test data

In [36]:
```
accuracy = accuracy_score(y_test, y_test_pred)
print('The accuracy is', accuracy)
```

```
The accuracy is 0.9393063583815029
```

2.2. The precision and recall values

In [37]:
```
precision = precision_score(y_test, y_test_pred)
print('The precision is', precision)
recall = recall_score(y_test, y_test_pred)
print('The recall is',recall)
```

```
The precision is 0.8532110091743119
The recall is 0.9489795918367347
```

2.3. A classification report (scikit-learn has a function that can create this for you)

In [38]:
```
classification = classification_report(y_test, y_test_pred)
print('The classification report is\n',classification)
```

```
The classification report is
               precision    recall  f1-score   support

           0       0.98      0.94      0.96       248
           1       0.85      0.95      0.90        98

    accuracy                           0.94       346
   macro avg       0.92      0.94      0.93       346
weighted avg       0.94      0.94      0.94       346
```

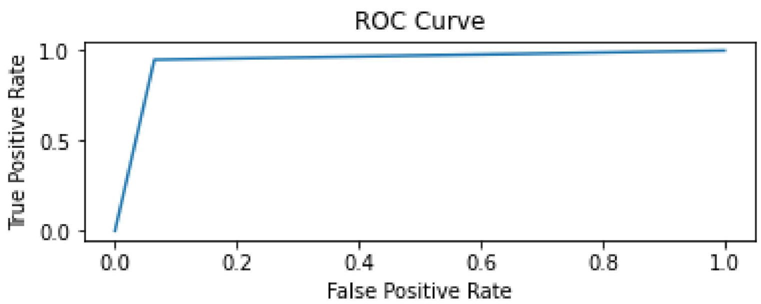
2.4. The confusion matrix for this experiment

In [39]:
```
confusion = confusion_matrix(y_test, y_test_pred)
print('The confusion matrix is\n',confusion)
```

```
The confusion matrix is
 [[232  16]
 [  5  93]]
```

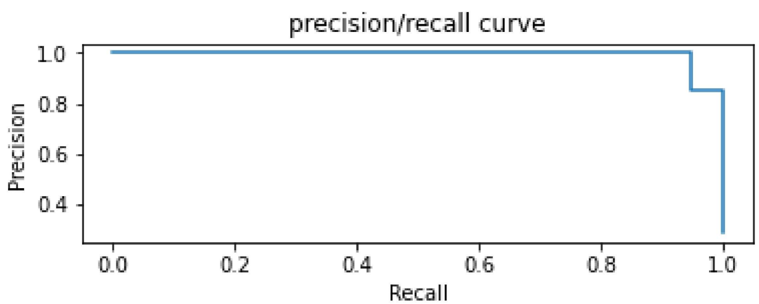
2.5. An ROC curve

In [40]:
```
fpr, tpr, thersholds = roc_curve(y_test, y_test_pred)
pre, rec, thresholds2 = precision_recall_curve(y_test, y_test_pred)

plt.figure()
plt.subplot(2,1,1)
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
```

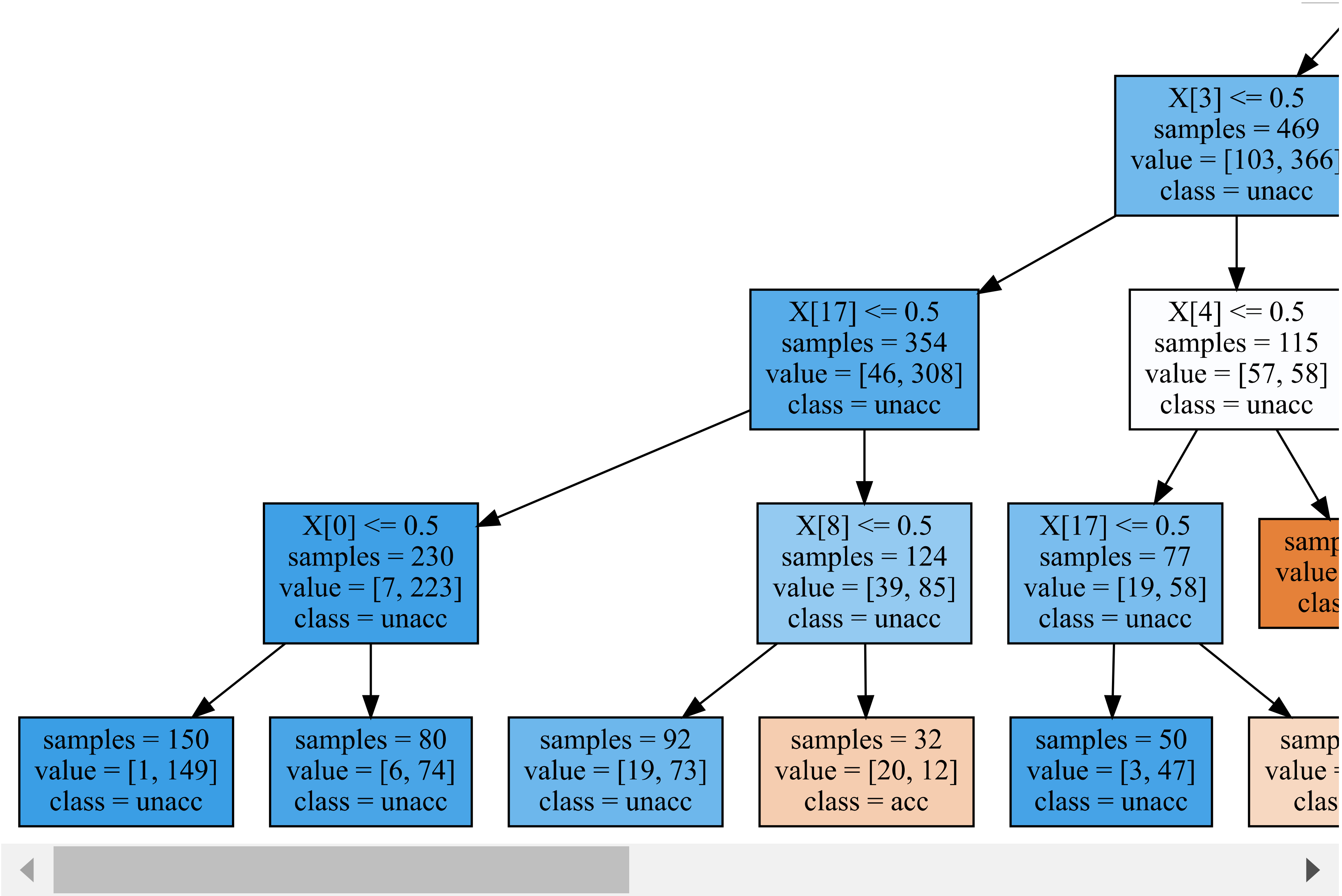


2.6. A Precision/Recall curve

In [41]:
```
plt.subplot(2,1,2)
plt.step(rec, pre)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('precision/recall curve')
plt.show()

graph = Source(tree.export_graphviz(model, out_file=None, class_names=['acc','unacc'], impurity=False,filled=True))
graph
```



Out[41]:



3. Similarly as in previous step, train another Decision Tree Classifier - but in this case set the maximum depth of the tree to 1 (max_depth = 1). Use the same training and test set as you used for the Decision Tree in the previous step.

3.1. Using scikit-learn’s DecisionTreeClassifier, train a supervised learning model that can be used to

gnerate predictions for your data.

In [42]:
```
# Import the classifier from sklearn
model_1 = DecisionTreeClassifier(max_depth = 1)
model_1.fit(X_train,y_train)

# Making predictions
y_test_pred_1 = model_1.predict(X_test)
```

3.1.1. The accuracy of your model on the test data

In [43]:
```
accuracy_1 = accuracy_score(y_test, y_test_pred_1)
print('The accuracy is', accuracy_1)
```

```
The accuracy is 0.7167630057803468
```

3.1.2. The precision and recall values

In [44]:
```
precision_1 = precision_score(y_test, y_test_pred_1)
print('The precision is', precision_1)
recall_1 = recall_score(y_test, y_test_pred_1)
print('The recall is',recall_1)
```

```
The precision is 0.0
The recall is 0.0
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision is i
ll-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

3.1.3. A classification report (scikit-learn has a function that can create this for you)

In [45]:
```
classification_1 = classification_report(y_test, y_test_pred_1)
print('The classification report is\n',classification_1)
```

```
The classification report is
               precision    recall  f1-score   support

           0       0.72      1.00      0.84       248
           1       0.00      0.00      0.00        98

    accuracy                           0.72       346
   macro avg       0.36      0.50      0.42       346
weighted avg       0.51      0.72      0.60       346
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

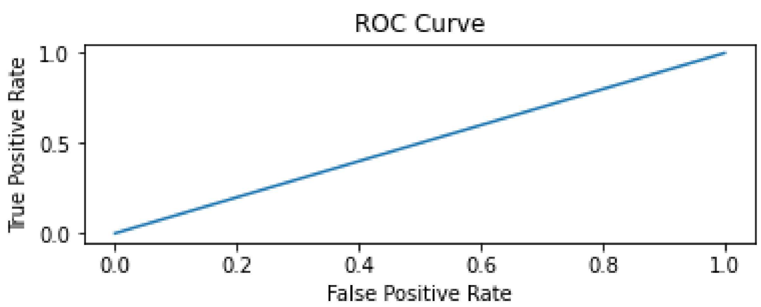
3.1.4. The confusion matrix for this experiment

In [46]:
```
confusion_1 = confusion_matrix(y_test, y_test_pred_1)
print('The confusion matrix is\n',confusion_1)
```

```
The confusion matrix is
 [[248   0]
 [ 98   0]]
```

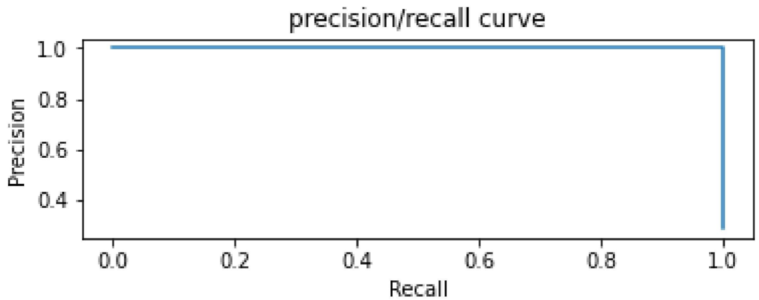
3.1.5. An ROC curve

In [47]:
```
fpr_1, tpr_1, thersholds_1 = roc_curve(y_test, y_test_pred_1)
pre_1, rec_1, thresholds2_1 = precision_recall_curve(y_test, y_test_pred_1)

plt.figure()
plt.subplot(2,1,1)
plt.plot(fpr_1, tpr_1)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
```

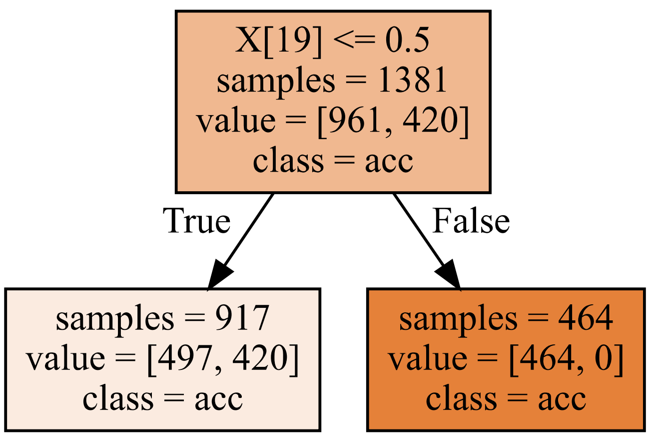


3.1.6. A Precision/Recall curve

In [48]:
```
plt.subplot(2,1,2)
plt.step(rec_1, pre_1)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('precision/recall curve')
plt.show()

graph = Source(tree.export_graphviz(model_1, out_file=None, class_names=['acc','unacc'], impurity=False,filled=True))
graph
```



Out[48]:



4. Report on the six evaluation metrics listed in objective for both the models, and compare their results.

The six evaluation metrics listed in objective for both the models are as shown above.

Compare the results, as you can see that when we use a decision tree with maximum depth = 1, the exact value of the result is much lower, with an accuracy of 0. Because it is trained by only one single feature and recision and the actual tag is missing from the predicted tag. In terms of the area of the ROC curve and the conflict matrix, the result of one depth is not as good as the full-deployment decision tree, so the full-depth decision tree will have better prediction results.