In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
from matplotlib import pyplot
```

In [2]:
```
parameters = pd.read_csv(r"D:\Detecting parkinsons disease\cleaned-data.csv")
parameters.head()
```

Out[2]:

	Unnamed: 0	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:D
0	0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.000070	0.00370	0.00554	0.011
1	1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.000080	0.00465	0.00576	0.013
2	2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.000090	0.00544	0.00576	0.016
3	3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.000090	0.00502	0.00576	0.015
4	4	phon_R01_S01_5	116.014	141.781	110.655	0.01101	0.000037	0.00593	0.00576	0.017

5 rows × 25 columns

In [3]:
```
columns = list(parameters.columns)
columns.remove('name')
columns
```

Out[3]:
```
['Unnamed: 0',
 'MDVP:Fo(Hz)',
 'MDVP:Fhi(Hz)',
 'MDVP:Flo(Hz)',
 'MDVP:Jitter(%)',
 'MDVP:Jitter(Abs)',
 'MDVP:RAP',
 'MDVP:PPQ',
 'Jitter:DDP',
 'MDVP:Shimmer',
 'MDVP:Shimmer(dB)',
 'Shimmer:APQ3',
 'Shimmer:APQ5',
 'MDVP:APQ',
 'Shimmer:DDA',
 'NHR',
 'HNR',
 'status',
 'RPDE',
 'DFA',
 'spread1',
 'spread2',
 'D2',
 'PPE']
```

In [4]:
```
feature_columns = columns
feature_columns.remove('status')
feature_columns
```

Out[4]:
```
['Unnamed: 0',
 'MDVP:Fo(Hz)',
 'MDVP:Fhi(Hz)',
 'MDVP:Flo(Hz)',
 'MDVP:Jitter(%)',
 'MDVP:Jitter(Abs)',
 'MDVP:RAP',
 'MDVP:PPQ',
 'Jitter:DDP',
 'MDVP:Shimmer',
 'MDVP:Shimmer(dB)',
 'Shimmer:APQ3',
 'Shimmer:APQ5',
 'MDVP:APQ',
 'Shimmer:DDA',
 'NHR',
 'HNR',
 'RPDE',
 'DFA',
 'spread1',
 'spread2',
 'D2',
 'PPE']
```

In [5]:
```
x = parameters[feature_columns]
y = parameters.status
x_train,x_test,y_train,y_test = train_test_split(x, y, test_size=0.3, random_state=1)
```

In [6]:
```
clf=RandomForestClassifier(n_estimators=100)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
```

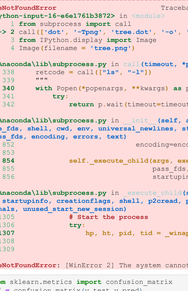
In [7]:
```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8644067796610017
```

In [8]:
```
clf=RandomForestClassifier(n_estimators=100)
clf.fit(x_train,y_train)
```

Out[8]:
```
RandomForestClassifier()
```

In [9]:
```
feature_imp = pd.Series(clf.feature_importances_,index=feature_columns).sort_values(ascending=False)
feature_imp
from matplotlib import pyplot
pyplot.bar(feature_columns,clf.feature_importances_)
pyplot.xticks(rotation='vertical')
pyplot.show()
```



In [10]:
```
sel = SelectFromModel(RandomForestClassifier(n_estimators = 100))
sel.fit(x_train, y_train)
sel.get_support()
```

Out[10]:
```
array([ True,  True,  True,  True, False, False,  True, False,  True,
       False, False, False, False, False, False, False, False, False,
       False,  True,  True,  True,  True, False])
```

In [11]:
```
selected_feat= x_train.columns[(sel.get_support())]
len(selected_feat)
```

Out[11]:
```
8
```

In [12]:
```
selected_feat
```

Out[12]:
```
Index(['Unnamed: 0', 'MDVP:Fo(Hz)', 'MDVP:Fhi(Hz)', 'MDVP:Flo(Hz)', 'MDVP:RAP',
       'Jitter:DDP', 'spread1', 'PPE'],
      dtype='object')
```

In [13]:
```
x = parameters[selected_feat]
y = parameters.status
x_train,x_test,y_train,y_test = train_test_split(x, y, test_size=0.3, random_state=1)
```

In [14]:
```
clf=RandomForestClassifier(n_estimators=50)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9491525423728814
```

In [15]:
```
from sklearn.tree import export_graphviz
estimator = clf.estimators_[5]
export_graphviz(estimator,
                out_file='tree.dot',
                feature_names = selected_feat,
                class_names = ['0','1'],
                rounded = True, proportion = False,
                precision = 2, filled = True)
```

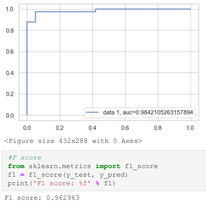
In [16]:
```
from subprocess import call
call(['dot', '-Tpng', 'tree.dot', '-o', 'tree.png', '-Gdpi=600'])
from IPython.display import Image
Image(filename = 'tree.png')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-16-e6e1761b3872> in <module>
      1 from subprocess import call
----> 2 call(['dot', '-Tpng', 'tree.dot', '-o', 'tree.png', '-Gdpi=600'])
      3 from IPython.display import Image
      4 Image(filename = 'tree.png')

D:\Anaconda\lib\subprocess.py in call(timeout, *popenargs, **kwargs)
    338     retcode = call(["ls", "-l"])
    339     """
--> 340     with Popen(*popenargs, **kwargs) as p:
    341         try:
    342             return p.wait(timeout=timeout)

D:\Anaconda\lib\subprocess.py in __init__(self, args, bufsize, executable, stdin, stdout, stderr, preexec_fn, c
lose_fds, shell, cwd, env, universal_newlines, startupinfo, creationflags, restore_signals, start_new_session,
pass_fds, encoding, errors, text)
    852                                 encoding=encoding, errors=errors)
    853
--> 854             self._execute_child(args, executable, preexec_fn, close_fds,
    855                                 pass_fds, cwd, env,
    856                                 startupinfo, creationflags, shell,

D:\Anaconda\lib\subprocess.py in _execute_child(self, args, executable, preexec_fn, close_fds, pass_fds, cwd, e
nv, startupinfo, creationflags, shell, p2cread, p2cwrite, c2pread, c2pwrite, errread, errwrite, unused_restore_
signals, unused_start_new_session)
   1305                 # Start the process
   1306                 try:
-> 1307                     hp, ht, pid, tid = _winapi.CreateProcess(executable, args,
   1308                                              # no special security
   1309                                              None, None,

FileNotFoundError: [WinError 2] The system cannot find the file specified
```

In [17]:
```
from sklearn.metrics import confusion_matrix
cnf = confusion_matrix(y_test,y_pred)
cnf
```

Out[17]:
```
array([[17,  2],
       [ 1, 39]], dtype=int64)
```

In [18]:
```
import seaborn as sns
import numpy as np
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
class_names=[0,1] # name of classes
fig, ax = pyplot.subplots()
tick_marks = np.arange(len(class_names))
pyplot.xticks(tick_marks, class_names)
pyplot.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
pyplot.tight_layout()
pyplot.title('Confusion matrix', y=1.1)
pyplot.ylabel('Actual label')
pyplot.xlabel('Predicted label')
pyplot.savefig("HeatMap")
```



In [19]:
```
from sklearn.metrics import precision_score, recall_score
print("Precision :",precision_score(y_test, y_pred))
print("Recall",recall_score(y_test, y_pred))
```

```
Precision :  0.9512195121951219
Recall 0.975
```

In [20]:
```
y_pred_proba = clf.predict_proba(x_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
pyplot.plot(fpr,tpr,label="data 1, auc="+str(auc))
pyplot.legend(loc=4)
pyplot.show()
pyplot.savefig('ROC')
```


```
<Figure size 432x288 with 0 Axes>
```

In [21]:
```
#F score
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print('F1 score: %f' % f1)
```

```
F1 score: 0.962963
```

In [22]:
```
#LogLoss
from sklearn.metrics import log_loss
logloss=log_loss(y_test,y_pred)
print("LogLoss: %.2f" % (logloss))
```

```
LogLoss: 1.76
```

In [ ]:
```

```