In [1]:
```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

import pandas as pd
import numpy as np
```

In [2]:
```
#getting the dataset
data_set = pd.read_csv("D:\Detecting parkinsons disease\cleaned-data.csv")

#getting first 5 records
data_set.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]:
```
#independent and dependent varibale
X=data_set.drop(['name','status'],axis=1)
y=data_set['status']
#splitting the dataset into test and train data set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [4]:
```
# Create adaboost classifer object
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(n_estimators=100)
classifier = AdaBoostClassifier(n_estimators=50,learning_rate=1,base_estimator=clf)
```

In [5]:
```
# Train Adaboost Classifer
model = classifier.fit(X_train, y_train)
```

In [6]:
```
#Predict the response for test dataset
y_pred = model.predict(X_test)
```

In [7]:
```
pd.DataFrame({'actual status':y_test,"predicted status:":y_pred})
```

Out[7]:

	actual status	predicted status:
179	1	1
192	0	0
84	1	1
119	1	1
90	1	1
172	0	0
160	1	1
128	1	1
178	1	1
123	1	1
169	0	0
157	1	1
117	1	1
100	1	1
24	1	1
55	1	1
126	1	1
95	1	1
89	1	1
142	1	1
69	1	1
27	1	1
53	0	1
48	0	1
1	1	1
138	1	1
14	1	1
80	1	1
102	1	1
11	1	1
60	0	0
25	1	1
122	1	1
10	1	1
18	1	1
9	1	1
152	1	1
176	0	0
139	1	1
79	1	1
63	0	0
54	1	1
153	1	1
159	1	1
184	0	1
45	0	0
180	1	1
88	1	1
86	1	1
112	1	1
129	1	1
22	1	1
133	1	1
191	0	1
173	0	0
91	1	1
114	1	0
37	1	1
57	1	1

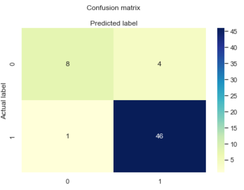
In [8]:
```
#Model Evaluation
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9152542372881356
```

In [9]:
```
#Confusion Matrix
from sklearn.metrics import confusion_matrix
cnf = confusion_matrix(y_test,y_pred)
cnf
```

Out[9]:
```
array([[ 8,  4],
       [ 1, 46]], dtype=int64)
```

In [10]:
```
from matplotlib import pyplot
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 [11]:
```
from sklearn.metrics import precision_score, recall_score
print("Precision : ",precision_score(y_test, y_pred))
print("Recall",recall_score(y_test, y_pred))
```

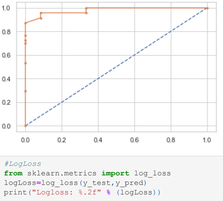
```
Precision :  0.92
Recall 0.9787234042553191
```

In [12]:
```
#ROC
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt
# predict probabilities
probs = classifier.predict_proba(X_test)
# keep probabilities for the positive outcome only
probs = probs[:, 1]

auc = roc_auc_score(y_test, probs)
print('AUC - Test Set: %.2f%%' % (auc*100))

# calculate roc curve
fpr, tpr, thresholds = roc_curve(y_test, probs)
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
# show the plot
plt.show()
```

```
AUC - Test Set: 98.05%
```



In [13]:
```
#LogLoss
from sklearn.metrics import log_loss
logLoss=log_loss(y_test,y_pred)
print("Logloss: %.2f" % (logLoss))
```

```
Logloss: 2.93
```

In [14]:
```
#F score
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print('F1 score: %f' % f1)
```

```
F1 score: 0.948454
```

In [ ]:
```

```