In [1]:
```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
#Reading the data set
symptoms=pd.read_csv("D:\Detecting parkinsons disease\cleaned-data.csv")
symptoms.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
symptoms.columns
```

Out[3]:
```
Index(['Unnamed: 0', 'name', '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'],
      dtype='object')
```

In [4]:
```
#independent variables
x=symptoms.drop(['status','name'],axis=1)

#dependent variable
y=symptoms['status']
```

In [5]:
```
#splitting the data set into training and test data set
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
```

In [6]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
classifier.fit(x_train, y_train)
```

Out[6]:
```
KNeighborsClassifier()
```

In [7]:
```
#predicting the results
y_pred=classifier.predict(x_test)
pd.DataFrame({'actual status':y_test,"predicted status:":y_pred})
```

Out[7]:

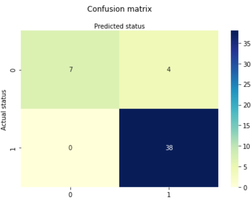
	actual status	predicted status:
83	1	1
12	1	1
33	0	0
113	1	1
171	0	0
134	1	1
163	1	1
124	1	1
74	1	1
18	1	1
7	1	1
5	1	1
125	1	1
161	1	1
170	0	1
181	1	1
123	1	1
60	0	1
44	0	0
141	1	1
56	1	1
173	0	0
136	1	1
89	1	1
63	0	0
55	1	1
110	1	1
166	0	1
175	0	0
45	0	0
22	1	1
155	1	1
66	1	1
37	1	1
4	1	1
80	1	1
178	1	1
106	1	1
160	1	1
26	1	1
139	1	1
143	1	1
71	1	1
8	1	1
61	0	1
130	1	1
122	1	1
101	1	1
118	1	1

In [8]:
```
#Model Evaluation
```

In [9]:
```
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
conf_mat= confusion_matrix(y_test, y_pred)
conf_mat
```

Out[9]:
```
array([[ 7,  4],
       [ 0, 38]], dtype=int64)
```

In [10]:
```
import seaborn as sns
#Heatmap for confusionmatrix
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(conf_mat), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual status')
plt.xlabel('Predicted status')
plt.savefig("HeatMap")
```



In [11]:
```
from sklearn.metrics import accuracy_score
from sklearn import metrics
accuracy=accuracy_score(y_test,y_pred)*100
print("Accuracy_score is :",accuracy)
print("Precision:",metrics.precision_score(y_test, y_pred)*100)
print("Recall:",metrics.recall_score(y_test, y_pred)*100)
```

```
Accuracy_score is : 91.83673469387756
Precision: 90.47619047619048
Recall: 100.0
```

In [12]:
```
#LogLoss
from sklearn.metrics import log_loss
logLoss=log_loss(y_test,y_pred)
print("Logloss: %.2f" % (logLoss))
```

```
Logloss: 2.82
```

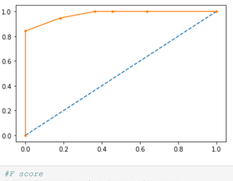
In [14]:
```
#ROC
from sklearn.metrics import roc_auc_score, roc_curve

# 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: 97.61%
```



In [15]:
```
#F score
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print('F1 score: %f' % f1)
```

```
F1 score: 0.950000
```

In [ ]:
```

```