In [1]:
```python
#importing the libraries
import pandas as pd
from sklearn import preprocessing
import matplotlib.pyplot as plt
plt.rc("font", size=14)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import seaborn as sns
import numpy as np
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)

#getting the dataset
data_set = pd.read_csv("D:\Detecting parkinsons disease\parkinsons.data")

#getting first 5 records
data_set.head()
```

Out[1]:

| | name | MDVP:Fo(Hz) | MDVP:Fhi(Hz) | MDVP:Flo(Hz) | MDVP:Jitter(%) | MDVP:Jitter(Abs) | MDVP:RAP | MDVP:PPQ | Jitter:DDP | MDVP:S |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | phon_R01_S01_1 | 119.992 | 157.302 | 74.997 | 0.00784 | 0.00007 | 0.00370 | 0.00554 | 0.01109 | |
| 1 | phon_R01_S01_2 | 122.400 | 148.650 | 113.819 | 0.00968 | 0.00008 | 0.00465 | 0.00696 | 0.01394 | |
| 2 | phon_R01_S01_3 | 116.682 | 131.111 | 111.555 | 0.01050 | 0.00009 | 0.00544 | 0.00781 | 0.01633 | |
| 3 | phon_R01_S01_4 | 116.676 | 137.871 | 111.366 | 0.00997 | 0.00009 | 0.00502 | 0.00698 | 0.01505 | |
| 4 | phon_R01_S01_5 | 116.014 | 141.781 | 110.655 | 0.01284 | 0.00011 | 0.00655 | 0.00908 | 0.01966 | |

5 rows × 24 columns

In [2]:
```python
#statistical summary of all the quantitative variables
data_set.describe()
```

Out[2]:

| | MDVP:Fo(Hz) | MDVP:Fhi(Hz) | MDVP:Flo(Hz) | MDVP:Jitter(%) | MDVP:Jitter(Abs) | MDVP:RAP | MDVP:PPQ | Jitter:DDP | MDVP:Shimmer | MI |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | |
| mean | 154.228641 | 197.104918 | 116.324631 | 0.006220 | 0.000044 | 0.003306 | 0.003446 | 0.009920 | 0.029709 | |
| std | 41.390065 | 91.491548 | 43.521413 | 0.004848 | 0.000035 | 0.002968 | 0.002759 | 0.008903 | 0.018857 | |
| min | 88.333000 | 102.145000 | 65.476000 | 0.001680 | 0.000007 | 0.000680 | 0.000920 | 0.002040 | 0.009540 | |
| 25% | 117.572000 | 134.862500 | 84.291000 | 0.003460 | 0.000020 | 0.001660 | 0.001860 | 0.004985 | 0.016505 | |
| 50% | 148.790000 | 175.829000 | 104.315000 | 0.004940 | 0.000030 | 0.002500 | 0.002690 | 0.007490 | 0.022970 | |
| 75% | 182.769000 | 224.205500 | 140.018500 | 0.007365 | 0.000060 | 0.003835 | 0.003955 | 0.011505 | 0.037885 | |
| max | 260.105000 | 592.030000 | 239.170000 | 0.033160 | 0.000260 | 0.021440 | 0.019580 | 0.064330 | 0.119080 | |

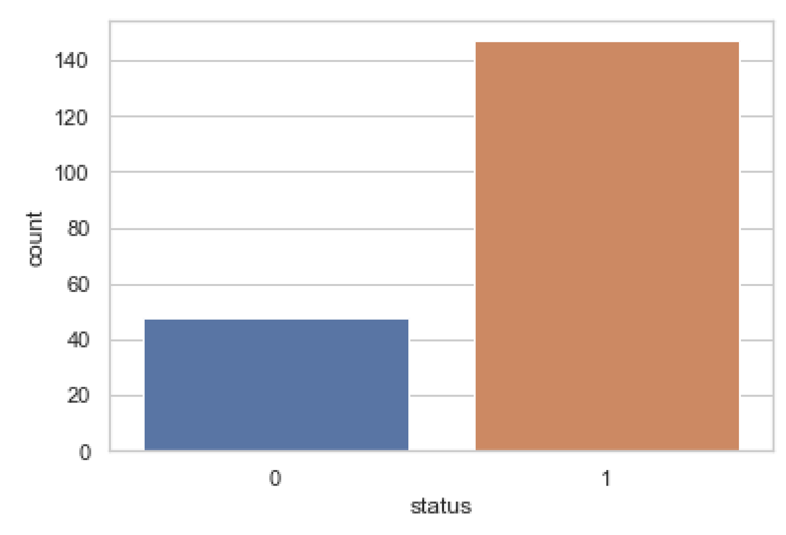
8 rows × 23 columns

In [3]:
```python
#counting number of people with status 1 and 0
data_set['status'].value_counts()
```

Out[3]:
```
1    147
0     48
Name: status, dtype: int64
```

In [4]:
```python
#plotting  status graphs
sns.countplot(x='status',data=data_set)
plt.show()

plt.savefig('status_count')
```




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

In [5]:
```python
parkinsons = len(data_set[data_set['status']==1])
no_parkinsons = len(data_set[data_set['status']==0])


per_parkinsons = parkinsons/(parkinsons+no_parkinsons)
per_no_parkinsons = no_parkinsons/(parkinsons+no_parkinsons)

print("Percentage of people having parkinsons:",per_parkinsons*100)
print("Percentage of people not having parkinsons:",per_no_parkinsons*100)
```

```
Percentage of people having parkinsons: 75.38461538461539
Percentage of people not having parkinsons: 24.615384615384617
```

In [6]:
```python
#getting column names
data_set.columns
```

Out[6]:
```
Index(['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 [7]:
```python
my_cols=set(data_set.columns)
my_cols.remove('status')
my_cols.remove('name')
#independent variables
X=data_set[my_cols]

#depemdent variable
y=data_set.status
```

In [8]:
```python
#importing class
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

#splitting training and testing sets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)

#importing class
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
log_reg = LogisticRegression(max_iter=2000)

#fitting the model
log_reg.fit(X_train,y_train)

#predicting the values
y_pred=log_reg.predict(X_test)

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

Out[8]:

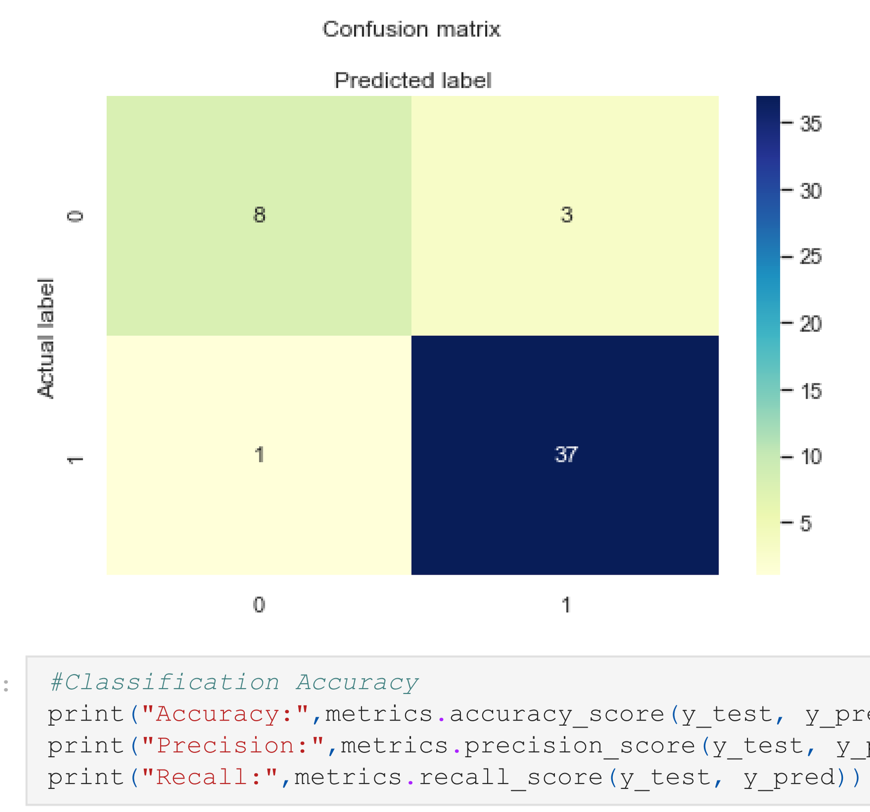
| | actual status | predicted status: |
|---|---|---|
| 83 | 1 | 1 |
| 12 | 1 | 1 |
| 33 | 0 | 0 |
| 113 | 1 | 1 |
| 171 | 0 | 1 |
| 134 | 1 | 1 |
| 163 | 1 | 1 |
| 124 | 1 | 1 |
| 74 | 1 | 1 |
| 18 | 1 | 1 |
| 7 | 1 | 1 |
| 5 | 1 | 1 |
| 125 | 1 | 1 |
| 161 | 1 | 1 |
| 170 | 0 | 0 |
| 181 | 1 | 1 |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

In [9]:
```python
#Model Evaluation
```

In [10]:
```python
#Confusion Matrix
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[10]:
```
array([[ 8,  3],
       [ 1, 37]], dtype=int64)
```

In [11]:
```python
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(cnf_matrix), 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 label')
plt.xlabel('Predicted label')
plt.savefig("HeatMap")
```



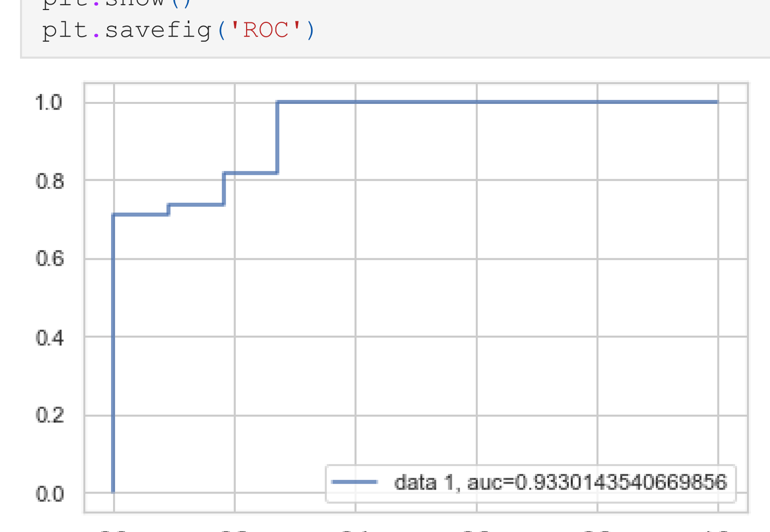

In [12]:
```python
#Classification Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.9183673469387755
Precision: 0.925
Recall: 0.9736842105263158
```

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

```
Logloss: 2.82
```

In [14]:
```python
#ROC curve
y_pred_proba = log_reg.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)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
plt.savefig('ROC')
```




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

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

```
F1 score: 0.948718
```

In [ ]: