Accuracy and Confusion Matrix

In [7]:
```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,roc_auc_score,confusion_matrix
```

In [8]:
```
from  sklearn.datasets import load_breast_cancer
```

In [9]:
```
Cancer = load_breast_cancer()
```

In [10]:
```
X = pd.DataFrame(Cancer.data, columns=Cancer.feature_names)
X.head()
```

Out[10]:

mean ıcavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension
0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890
0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902
0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758
0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300
0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678

In [11]:
```
X.shape
```

Out[11]:
```
(569, 30)
```

In [12]:
```
y = Cancer.target
```

In [14]:
```
y.mean()
```

Out[14]:
```
0.6274165202108963
```

In [15]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [18]:
```
clf = KNeighborsClassifier()
clf.fit(X,y)
```

Out[18]:
```
KNeighborsClassifier()
```

In [21]:
```
print(accuracy_score(clf.predict(X),y)*100)

print(confusion_matrix(clf.predict(X),y))
#94.72% -> Accurately classified
#5.28% -> Misclassified
```

```
94.72759226713534
[[191   9]
 [ 21 348]]
```

In [24]:
```
(191+348)/X.shape[0]
```

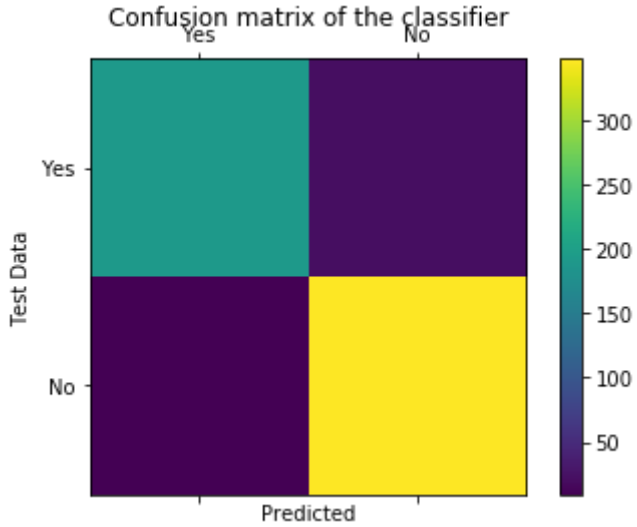
Out[24]:
```
0.9472759226713533
```

In [54]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(25, 25))
labels = ['Yes','No']
cm = confusion_matrix(y, clf.predict(X))
print(cm)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Test Data')
```

```
[[191  21]
 [  9 348]]
```

Out[54]: Text(0,0.5,'Test Data')

<Figure size 1800x1800 with 0 Axes>



RoC_AUC

Roc - > Reciever Operator Char.

Auc - > Area Under the Curve

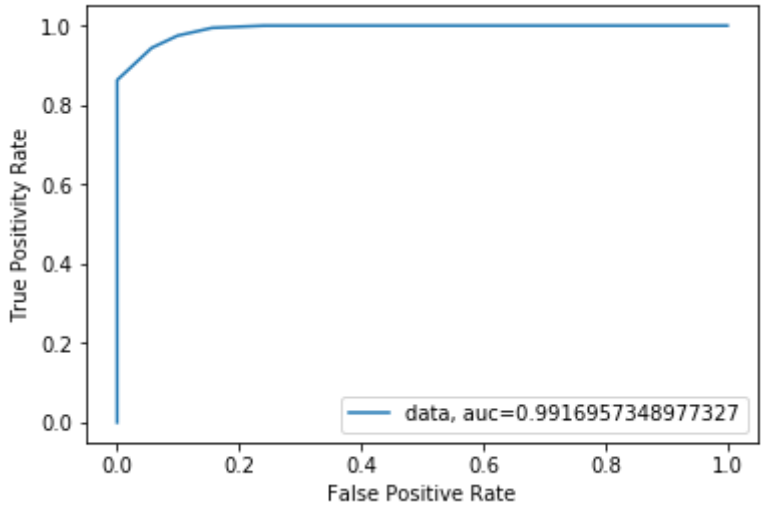
True Positivity rate(Sensitivity) = Tp/(Fn+Tp)

True Negative rate(Specifity) = Tn/(Fp+Tn)

False Positive rate = Fp/(Tn+Fp)

In [59]:
```
1 from sklearn import metrics
```

In [62]:
```
y_pred_proba = clf.predict_proba(X)[::,1]
fpr, tpr, _ = metrics.roc_curve(y,  y_pred_proba)
auc = metrics.roc_auc_score(y, y_pred_proba)
plt.plot(fpr,tpr,label="data, auc="+str(auc))
plt.legend(loc=4)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positivity Rate')
plt.show()
```



In [63]:
```
from sklearn.metrics import roc_auc_score
```

In [64]:
```
roc_auc_score(clf.predict(X),y)
```

Out[64]: 0.9490447154471544

Classification Report => Accuracy,Recall,Precision,F1 Score(Combination of precision and Recall)

In [65]:
```
from sklearn.metrics import classification_report
```

In [66]:
```
print(classification_report(clf.predict(X),y))
```

```
              precision    recall  f1-score   support

           0       0.90      0.95      0.93       200
           1       0.97      0.94      0.96       369

    accuracy                           0.95       569
   macro avg       0.94      0.95      0.94       569
weighted avg       0.95      0.95      0.95       569
```

In [ ]:
```

```

Precision

Precision = Tp/Tp+Fp

Ratio of correctly predicted positives to total predicted positives

What proportion of predicted positive values are actually positive

Model Effectiveness on the captured variation

Example :

Business use case is (We want to decrease credit limit - > Target[Whether to decrease limit or not])) and our priority is customer satisfaction so we want to avoid all cases where we decrease limit of a customer that can afford the higher limit.

Classification model: 1 -> Can decrease limit 0 -> Cannot decrease limit

False Positive Case -> We decrease cr limit of a person who can afford it (Predicted : 1 actual : 0)

Advantage of high precision:

To decrease FP cases we increase threshold : 0.9 :: Higher Threshold of positive classification == Higher precision value

Disadvantage of chasing a high precision with higher thresholds:

1.) Since we have a high threshold of prob for default to decrease the credit limit (i.e. 90%) we give higher credit limits to people with 80% chance of default that means we are giving riskier loans and hence there is a higher chance of losing money.

Recall

recall = Tp/(Tp+Fn)

Ratio of all correctly predicted positive values to all the positive values

What proportion of actual positives are predicted positives

Of how well your data is engineered and also a combination of how well you've selected your model on the basis of your data(Recall depends a lot on your data)

F-1 Score

Harmonic Mean of Precision and Recall

F1 = 2*(Precision X recall)/(Precision+recall)

Weighted Average of acc, Precision and Recall