33. Confusion Matrix (Practical) (Precision, Recall, F1-score)

In [3]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [4]:
```
dataset = pd.read_csv(r'Data/placement_3.csv')
dataset.head(3)
```

Out[4]:

	cgpa	score	placed
0	7.19	26	1
1	7.46	38	1
2	7.54	40	1

Split data into input and output data

In [5]:
```
x = dataset.iloc[:, :-1]
x
```

Out[5]:

	cgpa	score
0	7.19	26
1	7.46	38
2	7.54	40
3	6.42	8
4	7.23	17
...	...	...
995	8.87	44
996	9.12	65
997	4.89	34
998	8.62	46
999	4.90	10

1000 rows × 2 columns

In [6]:
```
y=dataset['placed']
y
```

Out[6]:
```
0      1
1      1
2      1
3      1
4      0
      ..
995    1
996    1
997    0
998    1
999    1
Name: placed, Length: 1000, dtype: int64
```

Split data into test and training data

In [7]:
```
from sklearn.model_selection import train_test_split
```

In [8]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_st
```

Build Model

In [9]:
```
from sklearn.linear_model import LogisticRegression
```

In [10]:
```
lg = LogisticRegression()
lg.fit(x_train, y_train)
```

Out[10]:
```
▾ LogisticRegression

LogisticRegression()
```

Checking model accuracy

In [11]:
```
lg.score(x_test, y_test)*100
```

Out[11]:
```
51.5
```

Confusion Matrix

In [12]:
```
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_sco
```

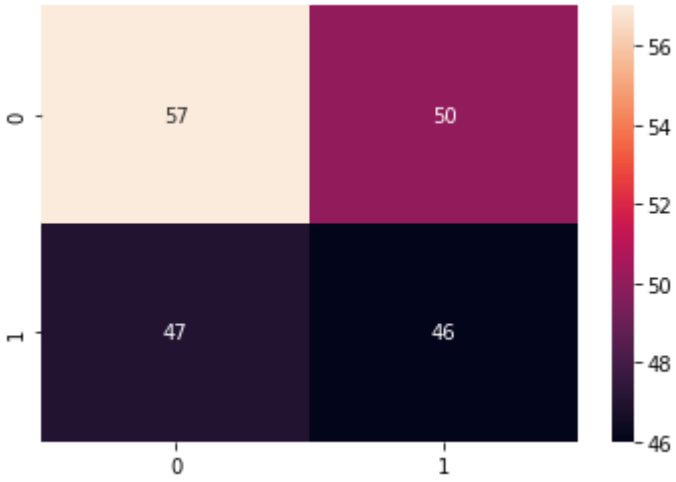
In [14]:
```
# Confusion matrix(y_true (actual), y_prediction)
cf = confusion_matrix(y_test,lg.predict(x_test))
cf
```

Out[14]:
```
array([[57, 50],
       [47, 46]], dtype=int64)
```

Graphically representing above results

In [15]:
```
sns.heatmap(cf, annot=True)
```

```
plt.show()
```



Interpretation of above graph:

- True Negative (TN)= 57
- True Positive (TP) = 46
- False Negative (FN) = 47
- False Positive (FP) = 50

Find Precision Score

In [17]:
```
# precision_score(y_true, y_pred)
precision_score(y_test, lg.predict(x_test))*100
```

Out[17]: 47.91666666666667

Find Recall Score

In [18]:
```
# recall_score(y_true, y_pred)
recall_score(y_test, lg.predict(x_test))*100
```

Out[18]: 49.46236559139785

Find F1-Score

In [20]:
```
# f1_score(y_true, y_pred)
f1_score(y_test, lg.predict(x_test))*100
```

Out[20]: 48.67724867724868

In [ ]:
```

```