FITTING THE MODEL.

S.ABHISHEK
GARINE SRIJA

FITTING MAP

WHY DO WE FIT ?
HOW DO WE FIT?

Step 1

IMPORT LOGISTIC REGRESSION

Step 2

INSTANTIATE THE MODEL

Step 3

FIT THE MODEL WITH THE TRAINING DATA

Step 4

PREDICT THE OTPUT OF THE TEST SET

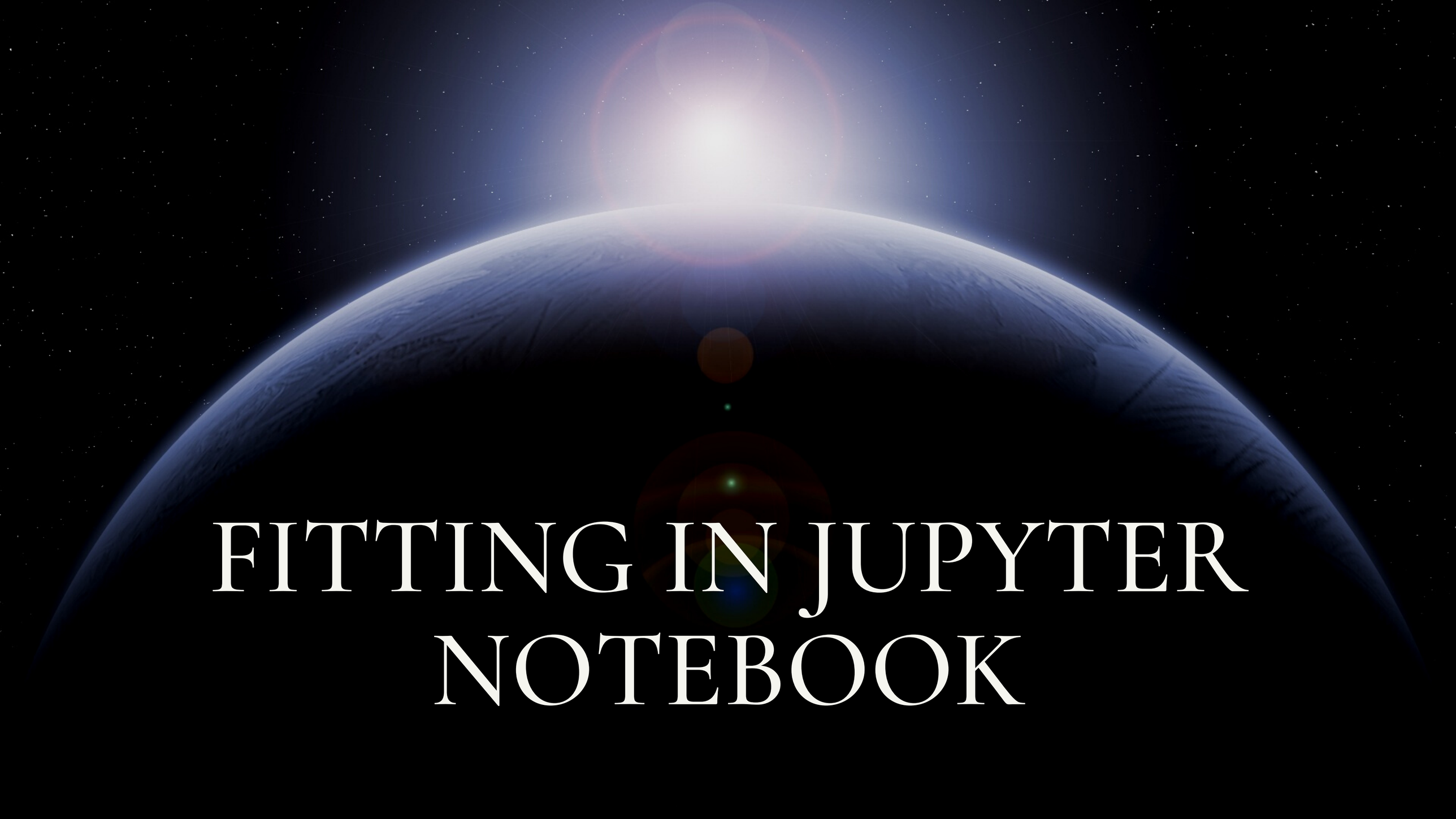
Why Do We Need To Fit A Model?

- Model fitting is a measure of how well a machine learning model generalizes to similar data to that on which it was trained.
- A model that is well-fitted produces more accurate outcomes.
- A model that is overfitted matches the data too closely.
- A model that is underfitted doesn’t match closely enough.
- Each machine learning algorithm has a basic set of parameters that can be changed to improve its accuracy.

- Then, you compare the outcomes to real, observed values of the target variable to determine their accuracy.
- Next, you use that information to adjust the algorithm's standard parameters to reduce the level of error, making it more accurate in uncovering patterns and relationships between the rest of its features and the target.
- You repeat this process until the algorithm finds the optimal parameters that produce valid, practical, applicable insights for your practical business problem.

Why is Model Fitting Important?

- Model fitting is the essence of machine learning.
- If your model doesn’t fit your data correctly, the outcomes it produces will not be accurate enough to be useful for practical decision-making.
- A properly fitted model has hyperparameters that capture the complex relationships between known variables and the target variable, allowing it to find relevant insights or make accurate predictions.
- Fitting is an automatic process that makes sure your machine learning models have the individual parameters best suited to solve your specific real-world business problem with a high level of accuracy.


FITTING IN JUPYTER
NOTEBOOK

[20]:
```
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression(max_iter = 1000)

# fit the model with data
logreg.fit(X_train,y_train)
```

[20]:
```
LogisticRegression(max_iter=1000)
```

[21]:
```
#predicting the output for out test set
y_pred=logreg.predict(X_test)
y_pred
```

[21]:
```
array(['1', '0', '0', '1', '0', '0', '1', '1', '0', '0', '1', '1', '0',
       '0', '0', '0', '1', '0', '0', '0', '1', '0', '0', '0', '0', '0',
       '0', '1', '0', '0', '0', '0', '0', '0', '0', '1', '0', '0', '0',
       '1', '0', '0', '0', '1', '1', '0', '0', '0', '0', '0', '0', '0',
       '1', '0', '0', '0', '0', '1', '0', '0', '1', '0', '0', '1', '1',
       '1', '1', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0', '1',
       '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '0', '0',
       '0', '0', '0', '1', '0', '0', '1', '1', '0', '0', '0', '0', '0',
       '1', '0', '0', '0', '0', '1', '0', '0', '1', '0', '1', '1', '0',
       '1', '0', '1', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0',
       '0', '0', '0', '1', '0', '0', '0', '0', '1', '0', '0', '1', '0',
       '0', '0', '0', '0', '0', '0', '0', '0', '1', '0', '0', '1', '0',
       '1', '0', '0', '1', '1', '1', '0', '0', '1', '0', '0', '0', '0',
       '0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '0', '1',
       '0', '1', '0', '0', '1', '0', '0', '0', '0', '0'], dtype=object)
```

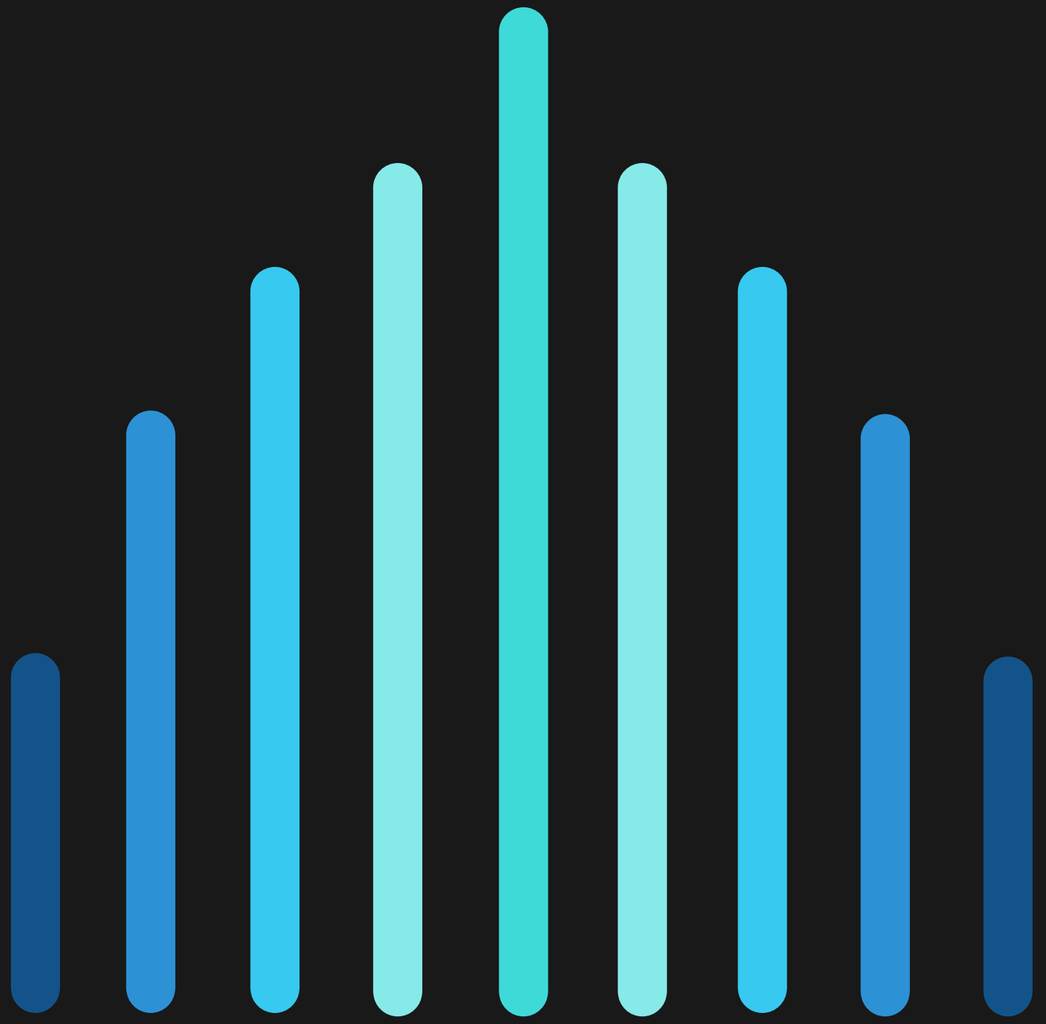
[22]:
```
X_test
```

[22]:

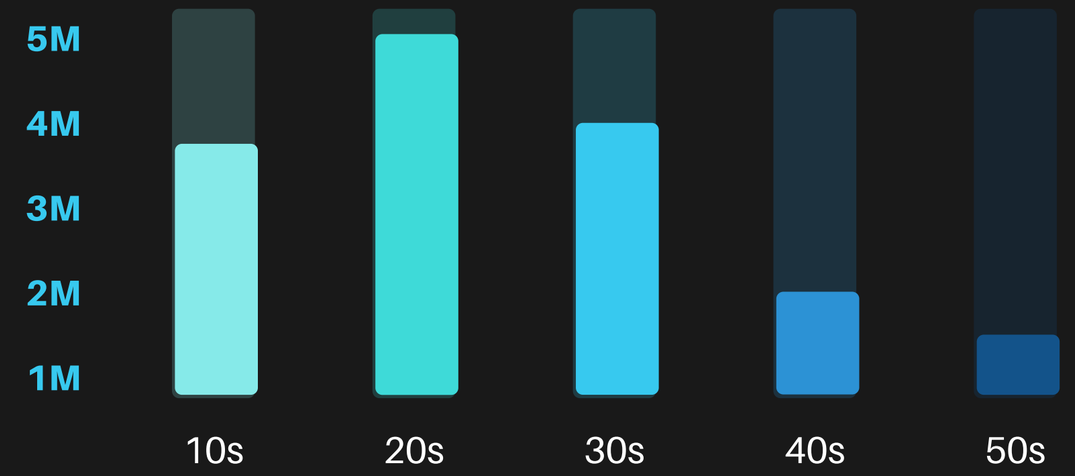
	pregnant	insulin	bmi	age	glucose	bp	pedigree
662	1	0	42.9	22	199	76	1.394
123	2	100	33.6	23	107	74	0.404
114	4	0	34	25	76	62	0.391
15	5	175	25.8	51	166	72	0.587
530	0	0	24.6	31	111	65	0.66
...	...	...	...	...	...	...	...
367	6	0	27.6	29	124	72	0.368
302	2	135	31.6	25	144	58	0.422
383	1	182	25.4	21	109	60	0.947
141	3	0	21.1	55	128	78	0.268
464	5	0	27.6	37	88	78	0.258

192 rows × 7 columns

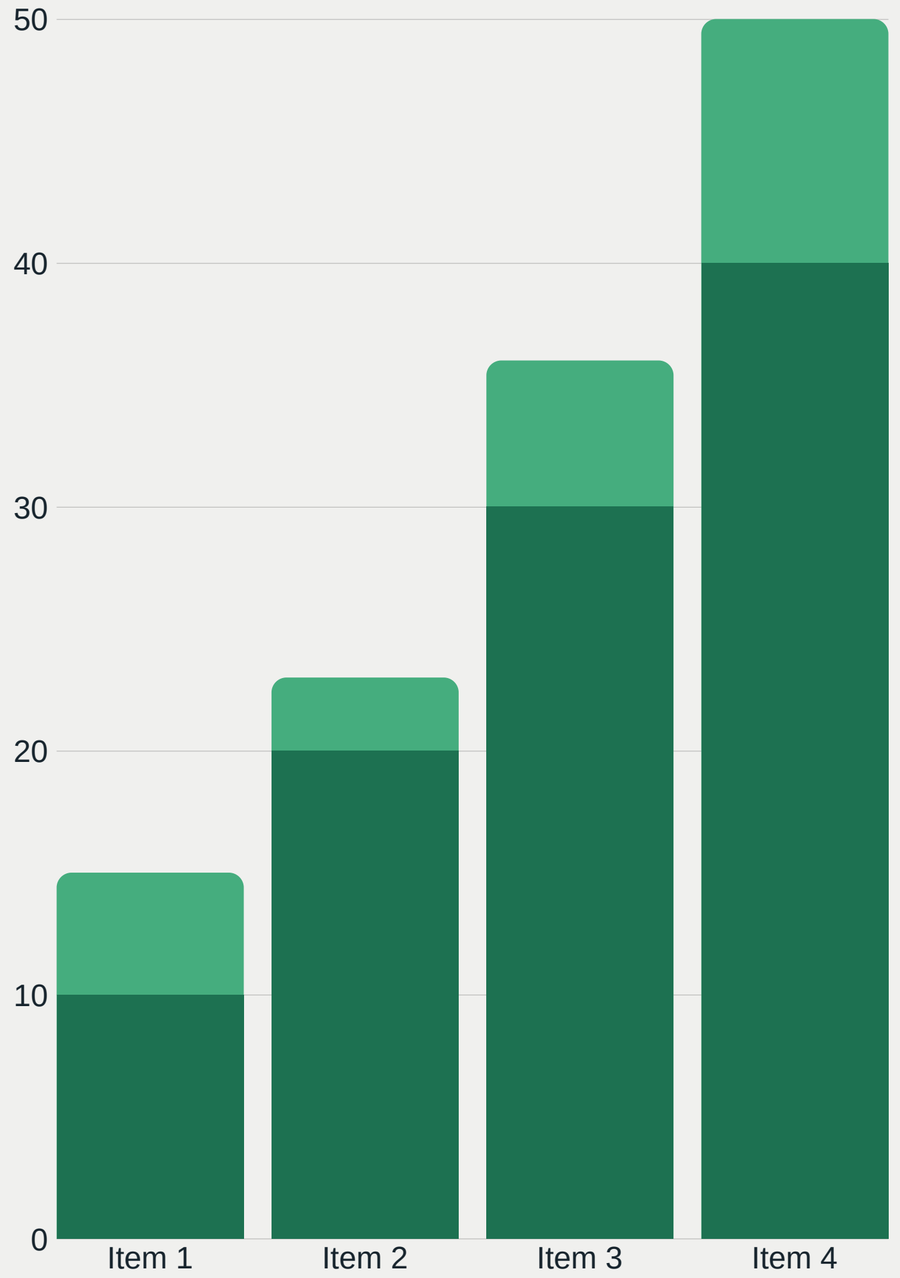
CONFUSION MATRIX



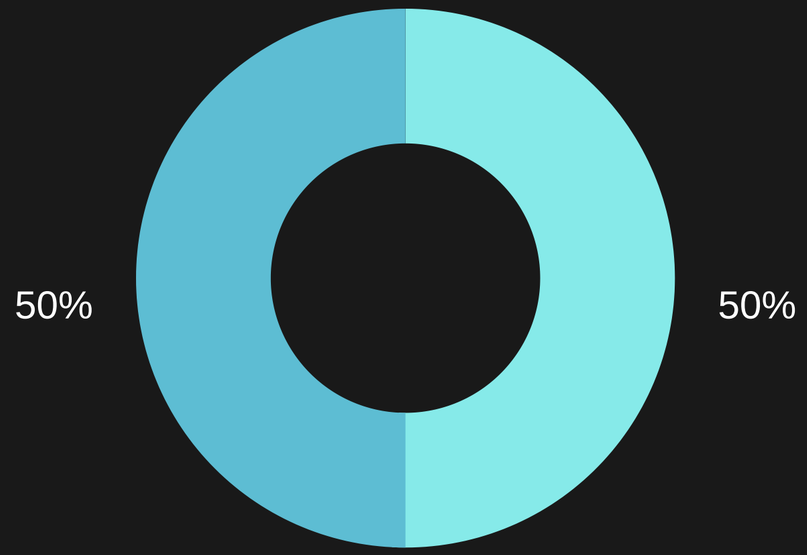
WHAT IS CONFUSION MATRIX?



- Confusion Matrix is generally used to measure the accuracy of a classification model.
- A classification model means the outcome or prediction is either yes or no (A binary decision of something happened or not)
- It is a table which is used to estimate the performance of a model.
- It tabulates the actual values and the predicted values in a 2×2 matrix.



WHAT IS THE NEED FOR CONFUSION MATRIX



- To describe the performance of a classification model on a set of data for which the true values are known.

- The Confusion Matrix is also known as Error Matrix or Table of Confusion

- A Confusion Matrix helps you visualize the performance of your classifier by displaying expected values vs predicted values in a matrix.

- True Positive (d): This denotes all of those records where the actual values are true and the predicted values are also true. So, these denote all of the true positives.

- False Negative (c): This denotes all of those records where the actual values are true, but the predicted values are false.

- False Positive (b): In this, the actual values are false, but the predicted values are true.

- True Negative (a): Here, the actual values are false and the predicted values are also false.

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Plotting Confusion Matrix :

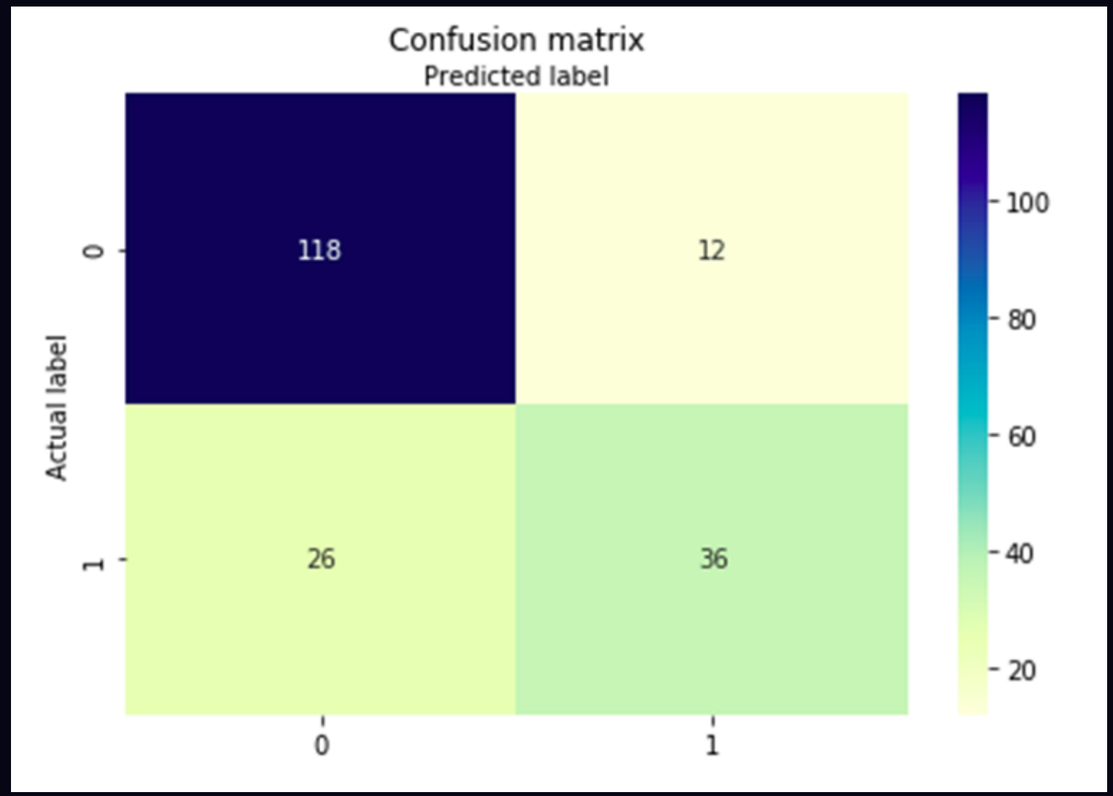
Visualizing Confusion matrix using Heat Map

- By visualizing the matrix we can get higher insights of Data and can understand the data even better

- We need to use matplotlib, numpy and seaborn modules to plot confusion matrix

Confusion Matrix Using Heat Map

- We need to input the data which is imported using pandas to draw heat map of confusion matrix



Annexure

- SUBPLOT FUNCTION IN MATPLOT LIB
- ARRANGE FUNCTION IN NUMPY
- TICKS IN MATPLOTLIB
- HEAT MAP FUNCTION IN SEABORN MODULE

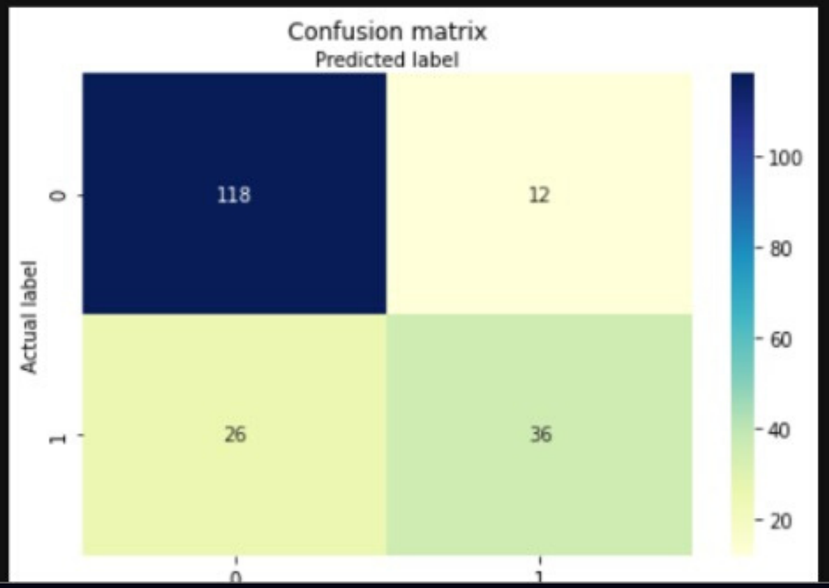
CONFUSION MATRIX USING JUPYTER NOTEBOOK

```
# import the metrics class to create confusion matrics
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

```
array([[118,  12],
       [ 26,  36]], dtype=int64)
```

```
#Ploting the Confusion Matrics
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')
```

```
Text(0.5, 257.44, 'Predicted label')
```



THANKYOU