

## Program – 14

**AIM:** Write a program in Python language to implement Model Evaluation(Cross Validation and Generate Confusion Matrix).

### Introduction and Theory

---

#### CROSS VALIDATION

Cross-validation, sometimes called rotation estimation, or out-of-sample testing is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of known data on which training is run (training dataset), and a dataset of unknown data (or first seen data) against which the model is tested (called the validation dataset or testing set). The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).

The general procedure is as follows (For K Fold Cross Validation):

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
4. Take the group as a hold out or test data set
5. Take the remaining groups as a training data set
6. Fit a model on the training set and evaluate it on the test set
7. Retain the evaluation score and discard the model
8. Summarize the skill of the model using the sample of model evaluation scores.

#### CONFUSION MATRIX

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another).

It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

	Predicted:		
	NO	YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
		55	110

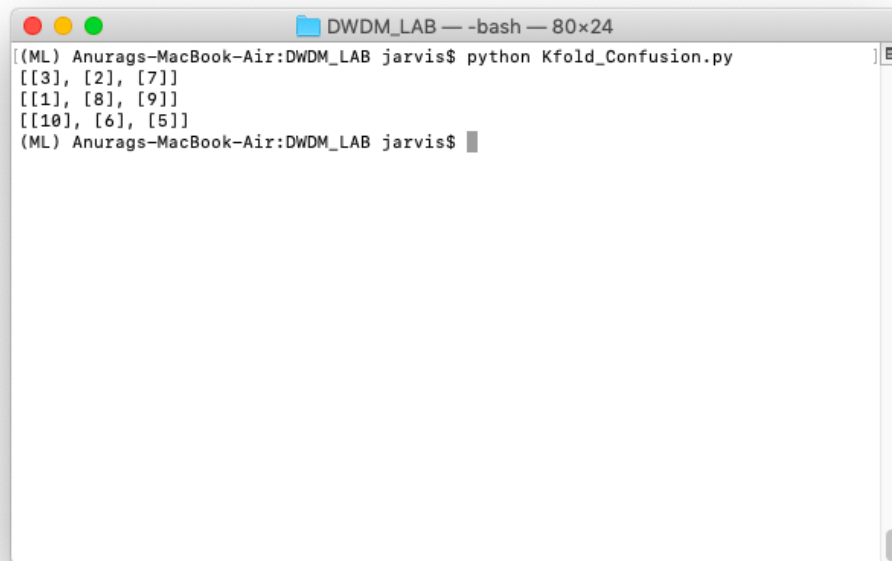
## Program – 14

### Code

```
1  from random import seed
2  from random import randrange
3
4
5  def cross_validation_split(dataset, folds=3):
6      dataset_split = list()
7      dataset_copy = list(dataset)
8      fold_size = int(len(dataset) / folds)
9      for i in range(folds):
10         fold = list()
11         while len(fold) < fold_size:
12             index = randrange(len(dataset_copy))
13             fold.append(dataset_copy.pop(index))
14         dataset_split.append(fold)
15     return dataset_split
16
17 seed(1)
18 dataset = [[1], [2], [3], [4], [5], [6], [7], [8], [9], [10]]
19 folds = cross_validation_split(dataset, 3)
20 for x in folds:
21     print(x)
22
23
24 def confusionmatrix(actual, predicted, normalize=False):
25     unique = sorted(set(actual))
26     matrix = [[0 for _ in unique] for _ in unique]
27     imap = {key: i for i, key in enumerate(unique)}
28
29     for p, a in zip(predicted, actual):
30         matrix[imap[p]][imap[a]] += 1
31
32     if normalize:
33         sigma = sum([sum(matrix[imap[i]]) for i in unique])
34         matrix = [row for row in map(lambda i: list(map(lambda j: j /
35 sigma, i)), matrix)]
36     return matrix
37
38
39 cm = confusionmatrix(
40     [1, 1, 2, 0, 1, 1, 2, 0, 0, 1], # actual
41     [0, 1, 1, 0, 2, 1, 2, 2, 0, 2] # predicted
42 )
43 print('actual : ', [1, 1, 2, 0, 1, 1, 2, 0, 0, 1])
44 print('predicted: ', [0, 1, 1, 0, 2, 1, 2, 2, 0, 2])
45 for x in cm:
46     print(x)
```

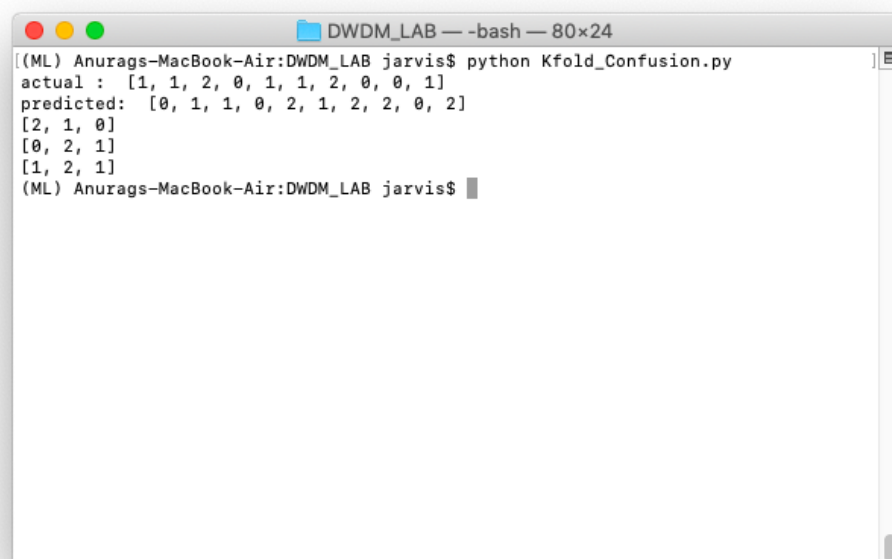
## Program – 14

### Results & Outputs



```
DWDM_LAB — -bash — 80x24
(ML) Anurags-MacBook-Air:DWDM_LAB jarvis$ python Kfold_Confusion.py
[[3], [2], [7]]
[[1], [8], [9]]
[[10], [6], [5]]
(ML) Anurags-MacBook-Air:DWDM_LAB jarvis$
```

#### K-fold cross validation



```
DWDM_LAB — -bash — 80x24
(ML) Anurags-MacBook-Air:DWDM_LAB jarvis$ python Kfold_Confusion.py
actual : [1, 1, 2, 0, 1, 1, 2, 0, 0, 1]
predicted: [0, 1, 1, 0, 2, 1, 2, 2, 0, 2]
[2, 1, 0]
[0, 2, 1]
[1, 2, 1]
(ML) Anurags-MacBook-Air:DWDM_LAB jarvis$
```

#### Confusion matrix

### Findings and Learnings:

1. What is cross validation and confusion matrix.
2. Cross validation helps correct over fitting.
3. Confusion Matrix helps identify skew and misclassification
4. How to code cross validation and confusion matrix in python.