**Submitted by –**

**Chandrashekhar M**

**Cross Validation**

**Definition :**

Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model. In typical cross-validation, the training and validation sets must cross-over in successive rounds such that each data point has a chance of being validated against. The basic form of cross-validation is k-fold cross-validation. Other forms of cross-validation are special cases of k-fold cross-validation or involve repeated rounds of k-fold cross-validation.

**What is Cross-Validation?**

For any model in Machine learning, it is considered as a best practice if the model is tested with an independent data set. Normally, any prediction model works on a known data set which is also known as the training set.

But in a real-life scenario, the model will be tested for its efficiency and accuracy with an altogether different and unique data set. Under those circumstances, you’d want your model to be efficient enough or at least to be at par with the same efficiency that it shows for the training set. Basically this testing is known as cross-validation in Machine Learning so that it is fit to work with any model in the future.

The basic purpose of cross-validation is to assess how the model will perform with an unknown data set. For instance, you are trying to score a goal in an empty goal. It looks pretty easy, and you could even score from a considerable distance too. But the real test starts when there is a goalkeeper and a bunch of defenders. That’s why you need to get trained in a real match facing all the heat and still score the goal.

Similarly, a statistical model is trained in such a way that it excels in its efficiency with other unknown data sets using cross-validation.

**Types Of Cross-Validation :**

There are two types of cross-validation techniques in Machine Learning.

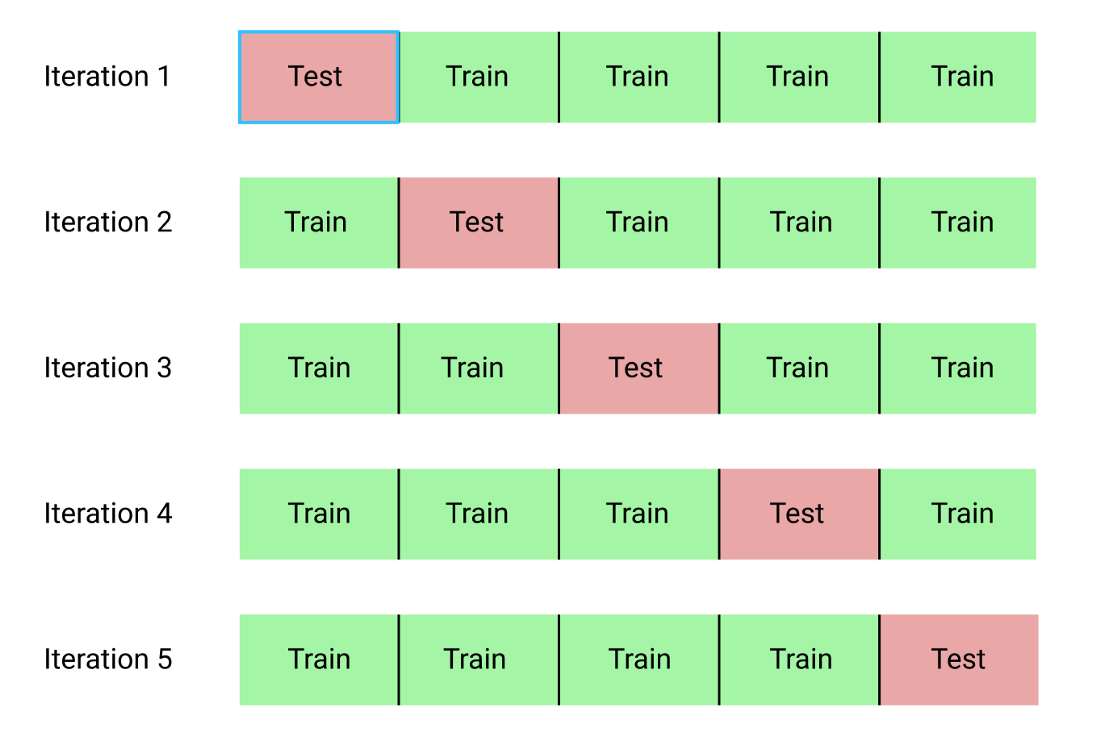
1. **Exhaustive Cross-Validation** – This method basically involves testing the model in all possible ways, it is done by dividing the original data set into training and validation sets. Example: Leave-p-out Cross-Validation, Leave-one-out Cross-validation.
2. **Non-Exhaustive Cross-Validation** – In this method, the original data set is not separated into all the possible permutations and combinations. Example: K-fold Cross-Validation, Holdout Method.

Let’s get into more details about various types of cross-validation in Machine Learning.

**1.K-Fold Cross-Validation :**

In Machine Learning, there is never enough data to train the model. Even then, if we remove some part of the data, it poses a threat of overfitting the Machine Learning model. It is also possible that it may not recognize a dominant pattern if enough data is not provided for the training phase.

By reducing the data, we also face the risk of reduced accuracy due to the error induced by bias. To overcome this problem, we need a method that would provide ample data for training and some data for testing. K-fold Cross-validation does exactly that.



The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

If k=5 the dataset will be divided into 5 equal parts and the below process will run 5 times, each time with a different holdout set.

1. Take the group as a holdout or test data set

2. Take the remaining groups as a training data set

3. Fit a model on the training set and evaluate it on the test set

4. Retain the evaluation score and discard the model

At the end of the above process Summarize the skill of the model using the sample of model evaluation scores.

**How does it work?**

In this cross-validation technique, the data is divided into k subsets. We take one subset from the bunch and treat it as the validation set for the model. And we keep the k-1 subset for training the model. The error estimation is averaged for all the ‘k trials’ to get the effective readiness of the model. Each k subset will be in the validation set at least once. It is also included in the k-1 training set at least once. This significantly reduces the error induced by bias. It also reduces the variance as each of the k subsets is used in the validation.

### ****Stratified K-fold Cross-Validation****

In this technique, a slight change is made in the k-fold Cross-Validation. It changes such that each fold will have an approximately equal percentage of samples of each target class as the whole set.  In the case of prediction problems, the mean responsive value is approximately equal in all the folds.

In some cases, there is a large imbalance in the responsive variables. Let us understand this with an example. In a house pricing problem, the prices of some houses can be much more than the other houses. Also, in classification problems, the samples may have more negative examples than the positive samples. To tackle this discrepancy we follow the stratified k-fold Cross-Validation technique in Machine Learning.



### ****2.Holdout Method****

This is the simplified cross-validation method among all. In this method, we randomly assign data points to two data sets. The size is not relevant in this case.

The basic idea behind this is to remove a part from your training set and use it to get predictions from the model that is trained on the rest of the data. This method suffers from high variance since it takes only a single run to execute all this. It may also give misleading results.

### ****3.Leave-p-out Cross-Validation****

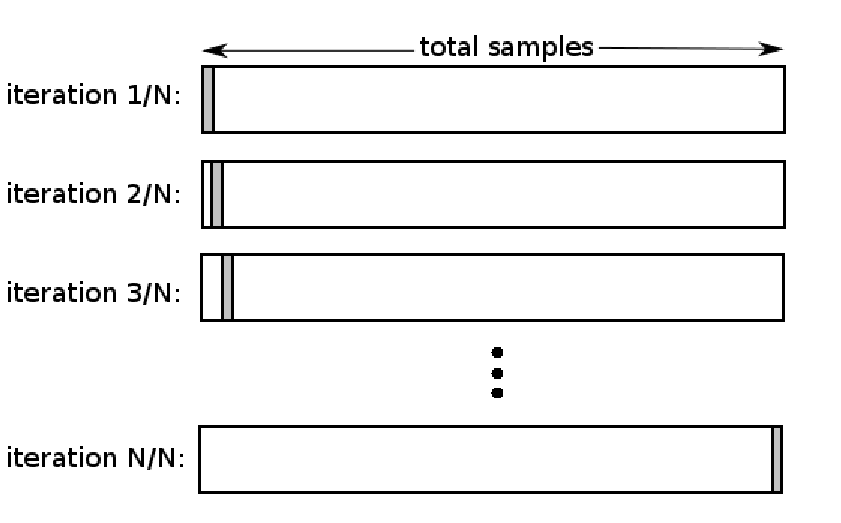
In this approach, **p** data points are left out of the training data. Let’s say there are **m**data points in the data set, then **m-p**data points are used for the training phase. And the **p**data points are kept as the validation set.

This technique is rather exhaustive because the above process is repeated for all the possible combinations in the original data set. To check the overall effectiveness of the model, the error is averaged for all the trials.

It becomes computationally infeasible since the model needs to train and validate for all possible combinations and for a considerably large **p**.

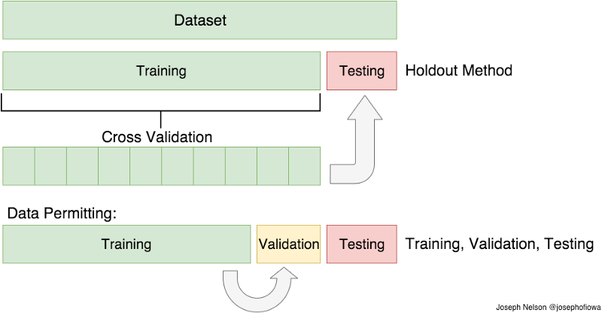
### ****4.Leave-one-out Cross-Validation****

This method of Cross-validation is similar to Leave-p-out Cross-validation but the only difference is that in this case **p = 1**. It actually saves a lot of time which is a big advantage.Although If the sample data is too large, it can still take a lot of time. But it would still be quicker than the Leave-p-out cross-validation method.The number of possible combinations is equal to the number of data points in the original sample n.



**Why is Cross-Validation Important?**

If the original validation partition is not representative of the overall population, then the resulting model may appear to have a high accuracy when in reality it just happens to fit the unusual validation set well, causing you to implement a model that actually has poor accuracy when applied to future data. With cross-validation, you can double-check how accurate your model is on multiple different subsets of data, ensuring it will generalize well to data you collect in the future.



## ****Implementation of Cross Validation In Python:****

We do not need to call the fit method separately while using cross validation, the cross\_val\_score method fits the data itself while implementing the cross-validation on data. Below is the example for using k-fold cross validation.

*import pandas as pd  
import numpy as np  
from sklearn.metrics import accuracy\_score, confusion\_matrix  
from sklearn.ensemble import RandomForestClassifier  
from sklearn import svm  
from sklearn.model\_selection import cross\_val\_score  
#read csv filedata = pd.read\_csv("D://RAhil//Kaggle//Data//Iris.csv")#Create Dependent and Independent Datasets based on our Dependent #and Independent featuresX = data[['SepalLengthCm','SepalWidthCm','PetalLengthCm']]  
y= data['Species']model = svm.SVC()accuracy = cross\_val\_score(model, X, y, scoring='accuracy', cv = 10)  
print(accuracy)#get the mean of each fold   
print("Accuracy of Model with Cross Validation is:",accuracy.mean() \* 100)*

**Output :**

****

The Accuracy of the model is the average of the accuracy of each fold.

* That cross validation is a procedure used to avoid overfitting and estimate the skill of the model on new data.
* There are common tactics that you can use to select the value of k for your dataset.
* There are commonly used variations on cross-validation, such as stratified and repeated, that are available in scikit-learn.

**Problem Statement and Solution :**

**Cross validation of time series data**

Time series data is characterised by the correlation between observations that are near in time (autocorrelation). However, classical cross-validation techniques such as **[KFold](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html" \l "sklearn.model_selection.KFold" \o "sklearn.model_selection.KFold)** and **[ShuffleSplit](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.ShuffleSplit.html" \l "sklearn.model_selection.ShuffleSplit" \o "sklearn.model_selection.ShuffleSplit)** assume the samples are independent and identically distributed, and would result in unreasonable correlation between training and testing instances (yielding poor estimates of generalisation error) on time series data. Therefore, it is very important to evaluate our model for time series data on the “future” observations least like those that are used to train the model. To achieve this, one solution is provided by **TimeSeriesSplit**.

**Time Series Split**

Time Series Split is a variation of k-fold which returns first k folds as train set and the (k+1) th fold as test set. Note that unlike standard cross-validation methods, successive training sets are supersets of those that come before them. Also, it adds all surplus data to the first training partition, which is always used to train the model.

This class can be used to cross-validate time series data samples that are observed at fixed time intervals.

Example of 3-split time series cross-validation on a dataset with 6 samples:

**>>> from** **sklearn.model\_selection** **import** TimeSeriesSplit

**>>>** X = np.array([[1, 2], [3, 4], [1, 2], [3, 4], [1, 2], [3, 4]])

**>>>** y = np.array([1, 2, 3, 4, 5, 6])

**>>>** tscv = TimeSeriesSplit(n\_splits=3)

**>>>** print(tscv)

TimeSeriesSplit(max\_train\_size=None, n\_splits=3)

**>>> for** train, test **in** tscv.split(X):

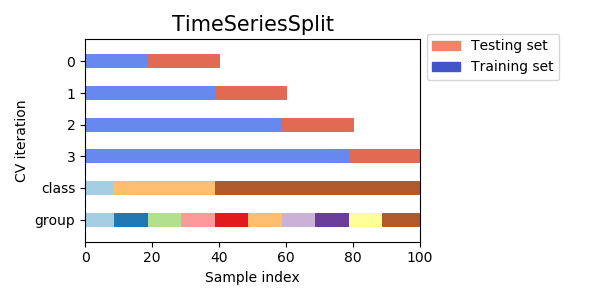
**...**  print("*%s* *%s*" % (train, test))

[0 1 2] [3]

[0 1 2 3] [4]

[0 1 2 3 4] [5]

Here is a visualization of the cross-validation behavior.



**Advantages and Disadvantages of Cross Validation :**

Cross Validation in Machine Learning is a great technique to deal with overfitting problem in various algorithms. Instead of training our model on one training dataset, we train our model on many datasets. Below are some of the advantages and disadvantages of Cross Validation in Machine Learning:  
  
**Advantages of Cross Validation**

**1. Reduces Overfitting:** In Cross Validation, we split the dataset into multiple folds and train the algorithm on different folds. This prevents our model from overfitting the training dataset. So, in this way, the model attains the generalization capabilities which is a good sign of a robust algorithm.  
  
**Note:**Chances of overfitting are less if the dataset is large. So, Cross Validation may not be required at all in the situation where we have sufficient data available.  
  
**2. Hyperparameter Tuning:** Cross Validation helps in finding the optimal value of hyperparameters to increase the efficiency of the algorithm.  
  
**Disadvantages of Cross Validation**  
  
**1. Increases Training Time:** Cross Validation drastically increases the training time. Earlier you had to train your model only on one training set, but with Cross Validation you have to train your model on multiple training sets.   
  
For example, if you go with 5 Fold Cross Validation, you need to do 5 rounds of training each on different 4/5 of available data. And this is for only one choice of hyperparameters. If you have multiple choice of parameters, then the training period will shoot too high.  
  
**2. Needs Expensive Computation:** Cross Validation is computationally very expensive in terms of processing power required.

## ****Limitations Of Cross-Validation****

The following are a few limitations faced by Cross-Validation:

1. In an ideal situation, Cross-Validation will produce optimum results. But in case of **inconsistent data**, the results may vary drastically. It is quite uncertain what kind of data will be encountered by the model.
2. Predictive modeling often requires an **evolution in terms of data**, this can pretty much change the training and the validation sets drastically.
3. The results may **vary depending upon the features of the data set**. Let us say we make a predictive model to detect an ailment in a person and we train it with a specific set of population. It may vary with the general population causing inconsistency and reduced efficiency.

## ****Cross-Validation Applications****

With the overpowering applications to prevent a Machine Learning model from [Overfitting and Underfitting,](https://www.edureka.co/blog/overfitting-in-machine-learning/) there are several other applications of Cross-Validation listed below:

1. We can use it to compare the performances of a set of predictive modeling procedures.
2. Cross-Validation excels in the field of medical research.
3. It can be used in the meta-analysis since a lot of data analysts are already using cross-validation.