greatlearning
Learning for Life

Cross Validation

greatlearning
Learning for Life

Need for cross validation?

1. We wish to know how well a ML model is likely to perform in production.

2. Model's performance in training is no guarantee production performance

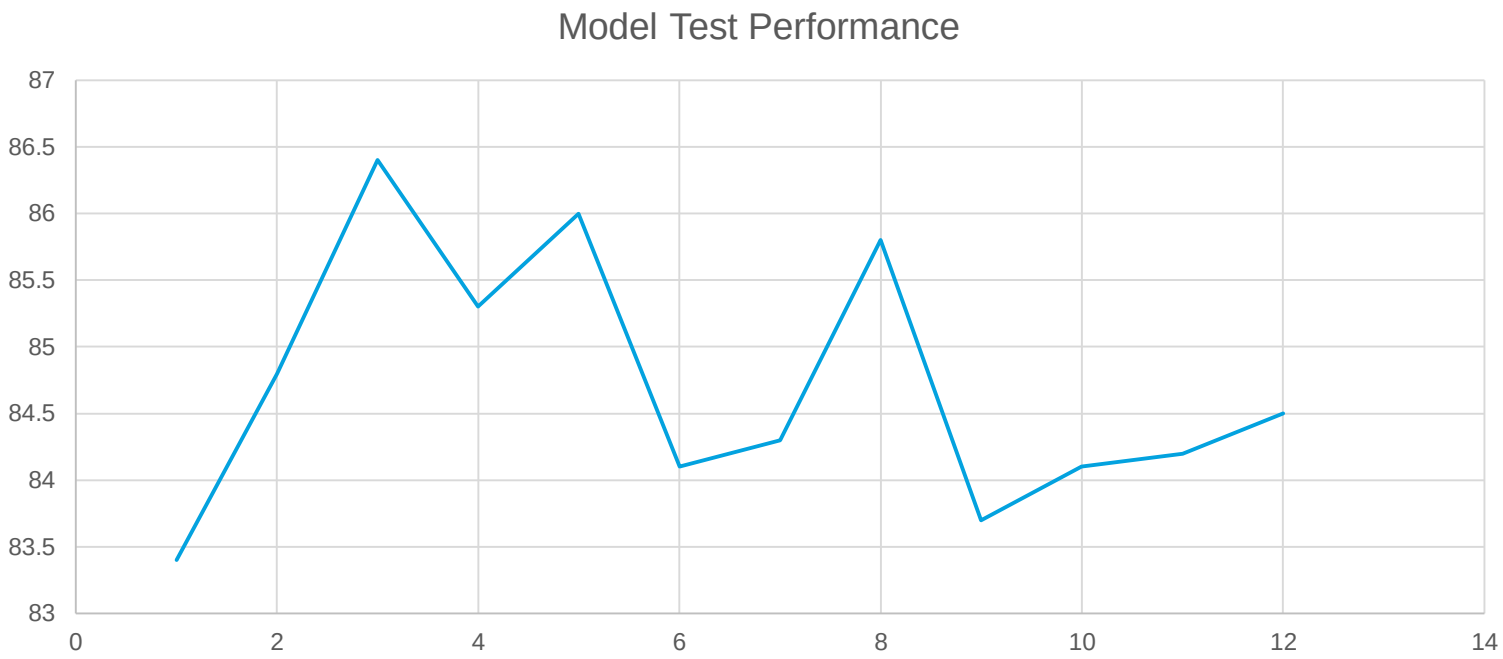
3. To estimate the model production score, hold a part of the sample data out of training phase. We call it test data which represents the universe

4. Usually the available data is not sufficient to split into training and test set and expect the two to represent the universe

5. Hence the model error on test data may not be good estimate of the model error in the universe

6. In the absence of large data sets, a number of techniques can be employed to estimate the model error in production

7. One of the techniques is cross validation



1. Car_MPG prediction accuracy trained on 70% and tested on 30%
2. For each run, we get different accuracy scores
3. Simple training/test a.k.a validation approach gives varying results with every run (random state not set)
4. Thus we cannot rely on one round of testing as we would have by chance got the split that gives max score first time
5. To get a more realistic estimate, we have to use more reliable techniques such as cross validation

greatlearning
Learning for Life

What is cross validation?

1. Cross-validation is a technique to evaluate / validate a machine learning model and estimate its performance on unseen data

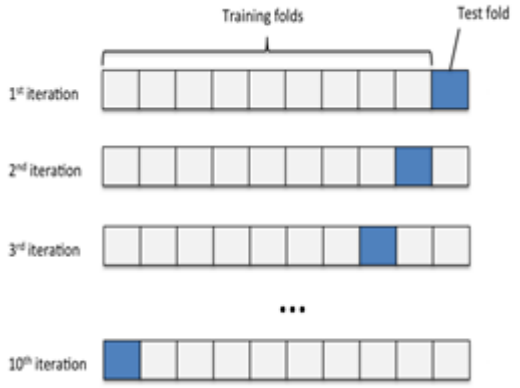
2. The techniques creates and validates given model multiple times

3. The number of times it does so, is dependent on the value selected by the user of the technique. Usually expressed as “K” which is always an integer

4. The sequence of steps used is iterated through as many times as K

5. The process begins by dividing the original data into K parts / folds using random function

Cross validation procedure



1. Shuffle the dataset randomly
2. Split the dataset into k folds
3. For each distinct fold:
   a. Keep the fold data separate / hold out data set
   b. Use the remaining folds as a single training data set
   c. Fit the model on the training set and evaluate it on the test set
   d. Retain the evaluation score and discard the model
   e. Loop back
4. The steps 3.a to 3.e will be executed K times
5. Summarize the scores and average it by dividing the sum by K.
6. Analyze the average score, the dispersion to assess the likely performance of the model in the unseen data (production data / universe)

Implementing K Fold cross validation

Visual understanding (example based on scikitlearn guide)

a) from numpy import array
b) from sklearn.model_selection import KFold
c) data = array([10,20,30,40,50,60,70,80,90,100])
d) kfold = KFold(5, True)
e) for train, test in kfold.split(data):
f) print('train: %s, test: %s' % (data[train], data[test]))

Training Data	Test Data
[10 20 30 40 50 60 80 90]	[70 100]
[10 40 50 60 70 80 90 100]	[20 30]
[10 20 30 40 50 70 90 100]	[60 80]
[10 20 30 50 60 70 80 100]	[40 90]
[20 30 40 60 70 80 90 100]	[10 50]

Note : We cannot have K > number of data points…. Why?

Ref: Kfold_introduction.ipynb

Configuring the K

1. K is an integral number. Minimum value of K has to be 2. There will be two iterations in this case

2. Max value of K can be the number of data points. This is also known as Leave One Out Cross Validation or LOOCV

3. Whatever the value of K chosen, the resulting training and test data should be representative of the unseen data as much as possible

4. There is not formula to decide the K but K = 10 is usually considered good

5. Too large a K, means less variance across the training sets thus limit the model differences across iterations

6. For a sample size (N) of n, and K = k, number of records (r) per fold = n/k.

Evaluating the model in an iteration

1. In each iteration, the model is trained on K -1 number of folds and evaluated on the left out fold.

2. The MSE or Mean Squared Error is thus calculated on the left out fold

3. Since the procedure is repeated K times, we will have K MSEs. Total up all the MSE and divide by K to get the overall expected MSE

$$CVk = (\text{sum}(MSEi) \text{ for } I = 1 \text{ to } K) / K$$

Some salient features of K-fold

1. Each record / data point in the sample data before creating the Kfolds, is assigned to a single fold and stays in that fold for the duration of the procedure.

2. This means that each data point is used once in hold-out set and K-1 times in training

3. When hyper parameters are to be tweaked, split the original data into two. Keep one part aside. Use the other to do the Kfold validation. Once the optimal hyperparameters are found, assess the model on the test data

4. Any data transformation done on the whole set outside the loop, may lead to data leakage and overfitting

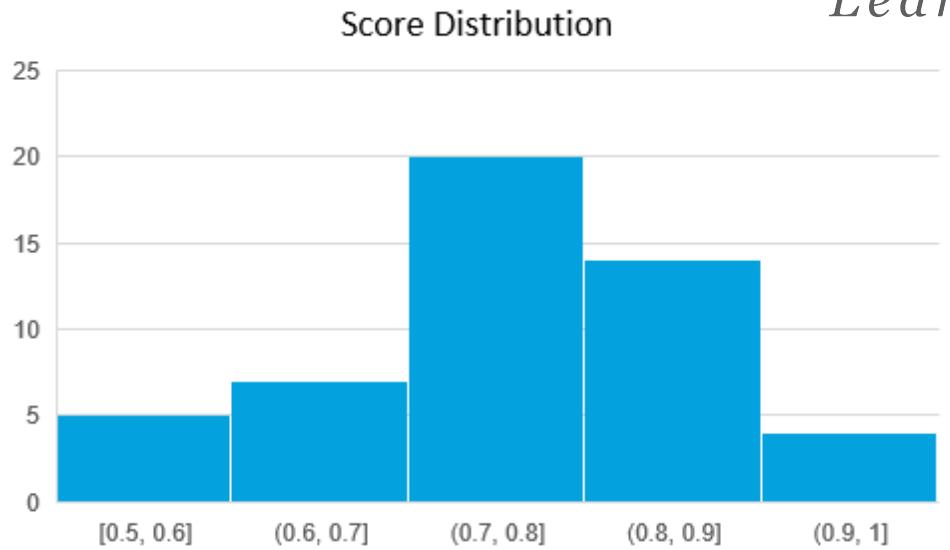
Implementing K Fold cross validation

K fold in Pima Indian Classification

```
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
import numpy as np

filename = 'pima-indians-diabetes.data'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)

array = dataframe.values
X = array[:,0:8]
Y = array[:,8]

num_folds = 50
seed = 7

kfold = KFold(n_splits=num_folds, random_state=seed)
model = LogisticRegression()
results = cross_val_score(model, X, Y, cv=kfold)
print(results)
print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```


Kfold_logistic.ipynb

Mean	0.770166667
Standard Error	0.015172553
Median	0.8
Mode	0.8
Standard Deviation	0.107286148
Sample Variance	0.011510317
Kurtosis	0.2246995
Skewness	-0.267638887
Range	0.5
Minimum	0.5
Maximum	1
Sum	38.50833335
Count	50
Confidence Level(95.0%)	0.030490386



1. Distribution of the scores on 50 iterations
2. Model accuracy is likely to be in 0.77 – 0.03 to 0.77 + 0.03 i.e. 0.74 – 0.80 at 95% confidence level

greatlearning
Learning for Life

Leave One Out Cross validation (LOOCV) procedure

1. In this method, a single observation (x1, y1) is used for the validation set and the remaining (x2,y2)…. (xn, yn) make up the training set

2. The statistical model is fit on the n-1 training examples

3. The statistical model prediction yhat is made for the excluded observation using x1.

4. MSE1 = (yhat – y)^2 for the excluded point

5. The MSE is unbiased but is poor estimate because it is highly variable

6. We can repeat this by keeping every data point for test one at a time and using rest for training

Ref: LOOCV_Introduction.ipynb