Cross Validation

Any dataset sample may fall in any of the two types:

Primary Data - the dataset is sourced and managed directly from the company/client

Secondary Data - the dataset is sourced from third party providers.

SL 🡺 Significance Level (alpha value)

* SL is all about understanding error tolerance of the given project.
* Since in AI, we deal with SAMPLES, there must be a scope of ERROR TOLERANCE in every project.
* That error tolerance in the world of Stats is SL value.
* Statisticians suggest to use SL values in any project @ 0.05, 0.01, 0.1.
* he above values suggested by Statisticians assume the fact that the dataset that is used for AI shall be highly compatible with the statistical formulae that is used for convergence or analysis.

Using Cross Validation helps us to define the SL value for the dataset.

**Cross Validation Technique**

Goal:

1. To get the minimum score threshold

2. To understand what optimal score I can achieve from the dataset

3. To extract the best training sample that can give the best score

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | data = pd.read\_csv("iris.csv")  data.head()   |  | **sepal\_length** | **sepal\_width** | **petal\_length** | **petal\_width** | **species** | | --- | --- | --- | --- | --- | --- | | 0 | 5.1 | 3.5 | 1.4 | 0.2 | setosa | | 1 | 4.9 | 3.0 | 1.4 | 0.2 | setosa | | 2 | 4.7 | 3.2 | 1.3 | 0.2 | setosa | | 3 | 4.6 | 3.1 | 1.5 | 0.2 | setosa | | 4 | 5.0 | 3.6 | 1.4 | 0.2 | setosa | |
|  |  | features = data.iloc[:,:-1].values  label = data.iloc[:,-1].values |
|  | Demonstrate the score threshold with LogisticRegression | from sklearn.linear\_model import LogisticRegression  modelAlgo = LogisticRegression() |

Working of Cross Validation

-----------------------------

dataset ( 100 records )

cv = 5 ---- How many splits i must create to perform cross validation?

How many iterations I must perform for model building to get scores?

5 samples

S1 (1-20) S2(21-40) S3(41-60) S4(61-80) S5(81-100)

Iteration1: S1(testSet) S2-S5(trainSet) ---> Create Model ---> Get the Score

Iteration2: S2(testSet) S1,S3,S4,S5(TrainSet)--> CreateModel--> Get the Score

Iteration3: S3(testSet) S1,S2,S4,S5(TrainSet)--> CreateModel--> Get the Score

Iteration4: S4(testSet) S1,S3,S2,S5(TrainSet)--> CreateModel--> Get the Score

Iteration5: S5(testSet) S1,S3,S4,S2(TrainSet)--> CreateModel--> Get the Score

|  |  |  |
| --- | --- | --- |
|  |  | from sklearn.model\_selection import cross\_val\_score  # Supress warnings  import warnings  warnings.filterwarnings('ignore')  scores = cross\_val\_score(modelAlgo,  features,  label,  cv = 10) #5 or 10  scores  array([1. , 0.93333333, 1. , 1. , 0.93333333,  0.93333333, 0.93333333, 1. , 1. , 1. ]) |
|  | # What is the minimum score threshold for this dataset?  scores.mean = 0.9733 is the CL value.  If I know CL value, we can get the SL value too.  SL = 1-scores.mean()) = 0.026 | print("Minimum Score Threshold is : ",scores.mean())  print("Suggested SL value to commit: ", 1-scores.mean())  Minimum Score Threshold is : 0.9733333333333334  Suggested SL value to commit: 0.026666666666666616 |
|  | # What is the optimal score I can achieve for this dataset using LogisticRegression? | scores.max()  1.0 |
|  | 3. To extract the best training sample that gives the best score for LogisticRegression  Step1: Initialize the algo  Step2: Initialize K-Fold Cross Validation function  # 3. initialize for loop to identify which sample gives the best score and which sample is the best #. training sample | # Step1:  from sklearn.linear\_model import LogisticRegression  modelAlgo = LogisticRegression()  # Step2:  from sklearn.model\_selection import KFold  kfold = KFold(n\_splits=10, #Use the same CV values that was applied in cross\_val\_score  shuffle=True,  random\_state = 1) # To ensure the data is not randomized at every iteration  # 3. initialize for loop to identify which sample gives the best score and which sample is the best  #. training sample  counter = 0  for train,test in kfold.split(features):    #Counter will help you track the sample split  counter += 1    #Extract the training set and testing set  X\_train,X\_test = features[train],features[test]  y\_train,y\_test = label[train] , label[test]    #Fit the model  modelAlgo.fit(X\_train,y\_train)    if modelAlgo.score(X\_test,y\_test) >= 1.0:  print("Test Score {} Train Score {} for Sample Split {}".format(modelAlgo.score(X\_test,y\_test),modelAlgo.score(X\_train,y\_train),counter))  Test Score 1.0 Train Score 0.9777777777777777 for Sample Split 1  Test Score 1.0 Train Score 0.9777777777777777 for Sample Split 4  Test Score 1.0 Train Score 0.9703703703703703 for Sample Split 7  Test Score 1.0 Train Score 0.9703703703703703 for Sample Split 9 |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  | # Extract the samples  # Step1: Initialize the algo  from sklearn.linear\_model import LogisticRegression  modelAlgo = LogisticRegression()  # Step2: Initialize K-Fold Cross Validation function  from sklearn.model\_selection import KFold  kfold = KFold(n\_splits=10, #Use the same CV values that was applied in cross\_val\_score  shuffle=True,  random\_state = 1) # To ensure the data is not randomized at every iteration  # 3. initialize for loop to identify which sample gives the best score and which sample is the best  #. training sample  counter = 0  for train,test in kfold.split(features):    #Counter will help you track the sample split  counter += 1    if counter == 1:  X\_train,X\_test,y\_train,y\_test = features[train],features[test],label[train] , label[test] |
|  |  | kfold.split(features) |
|  |  | from sklearn.linear\_model import LogisticRegression  finalModel = LogisticRegression()  finalModel.fit(X\_train,y\_train)  finalModel.score(X\_test,y\_test)  1.0 |

Another method to extract best sample (Optimized Way --> Dealing with Large Data in less time)

StraifiedShuffleSplit

|  |  |
| --- | --- |
|  | # 3. To extract the best training sample that gives the best score for LogisticRegression  # Step1: Initialize the algo  from sklearn.linear\_model import LogisticRegression  modelAlgo = LogisticRegression()  # Step2: Initialize StratifiedShuffleSplit Cross Validation function  from sklearn.model\_selection import StratifiedShuffleSplit  ss = StratifiedShuffleSplit(n\_splits=10, #Use the same CV values that was applied in cross\_val\_score  test\_size=0.2,  random\_state = 1) # To ensure the data is not randomized at every iteration  # 3. initialize for loop to identify which sample gives the best score and which sample is the best  #. training sample  counter = 0  for train,test in ss.split(features,label):    #Counter will help you track the sample split  counter += 1    #Extract the training set and testing set  X\_train,X\_test = features[train],features[test]  y\_train,y\_test = label[train] , label[test]    #Fit the model  modelAlgo.fit(X\_train,y\_train)    if modelAlgo.score(X\_test,y\_test) >= 1.0:  print("Test Score {} Train Score {} for Sample Split {}".format(modelAlgo.score(X\_test,y\_test),modelAlgo.score(X\_train,y\_train),counter))  Test Score 1.0 Train Score 0.9666666666666667 for Sample Split 3  Test Score 1.0 Train Score 0.975 for Sample Split 7  Test Score 1.0 Train Score 0.9583333333333334 for Sample Split 10 |

|  |  |
| --- | --- |
|  | # Extract the samples  # Step1: Initialize the algo  from sklearn.linear\_model import LogisticRegression  modelAlgo = LogisticRegression()  # Step2: Initialize K-Fold Cross Validation function  from sklearn.model\_selection import StratifiedShuffleSplit  ss = StratifiedShuffleSplit(n\_splits=10, #Use the same CV values that was applied in cross\_val\_score  test\_size=0.2,  random\_state = 1) # To ensure the data is not randomized at every iteration  # 3. initialize for loop to identify which sample gives the best score and which sample is the best  #. training sample  counter = 0  for train,test in ss.split(features,label):    #Counter will help you track the sample split  counter += 1    if counter == 7:  X\_trainSS,X\_testSS,y\_trainSS,y\_testSS = features[train],features[test],label[train] , label[test] |
|  | from sklearn.linear\_model import LogisticRegression  finalModel = LogisticRegression()  finalModel.fit(X\_trainSS,y\_trainSS)  finalModel.score(X\_testSS,y\_testSS)  1.0 |