**K-fold Cross Validataion:**

Here K is different from K of k-NN. So for not getting confusion let say k’- fold cross-validataion.

Earlier we were splitting data into 3 parts:

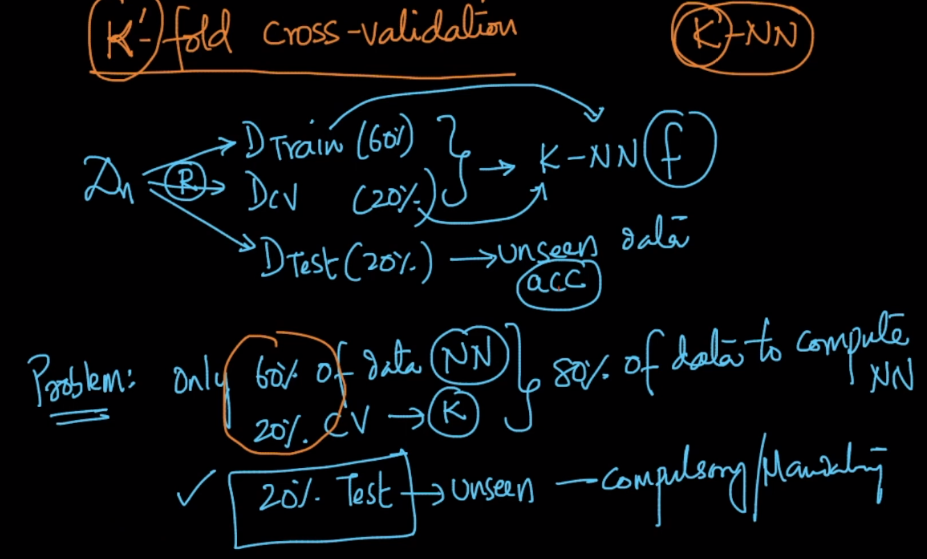
* Train: 60%
* Cross validation to identify k: 20%
* Test: 20%.

But in ML the more data we have the more good is for ML model.

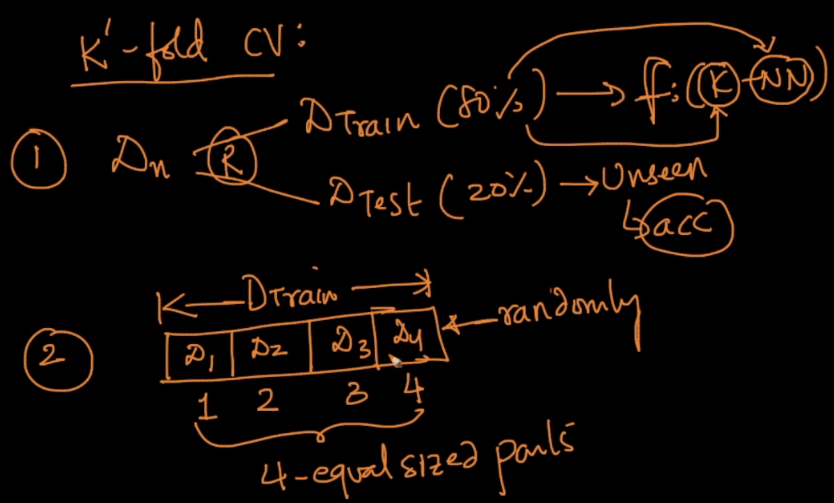
So using k’-fold cross validation we can combine Train and Cross Validation data which will do both training of k-NN and identifying K value.

Therefore now we have only two dataset.

* Train dataset
* Test dataset.



Now what we do is we randomly split Train dataset in 4 parts randomly (taking k’ = 4).



Now K is computed as given in below.

For each k, we do following things:

* Do training with d1, d2, d3 and use d4 as CV, you’ll get a4 as accuracy
* Do training with d1, d2, d4 and use d3 as CV, you’ll get a3 as accuracy
* Do training with d1, d3, d4 and use d2 as CV, you’ll get a2 as accuracy
* Do training with d2, d3, d4 and use d1 as CV, you’ll get a1 as accuracy
* Compute average of all 4 accuracy ak=1.
* Plot this accuracy.

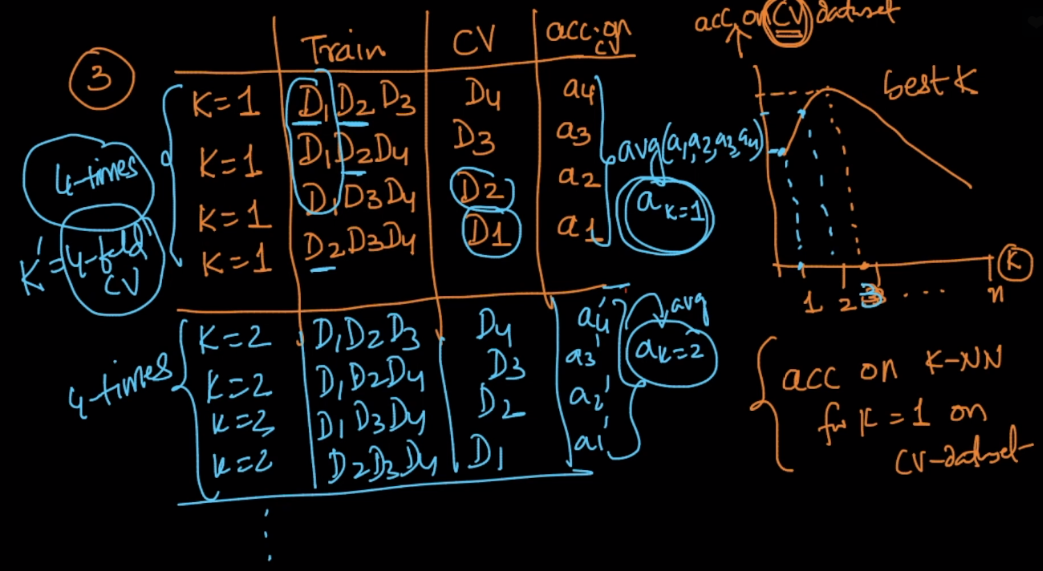
After doing this pick that K value whose accuracy is highest.

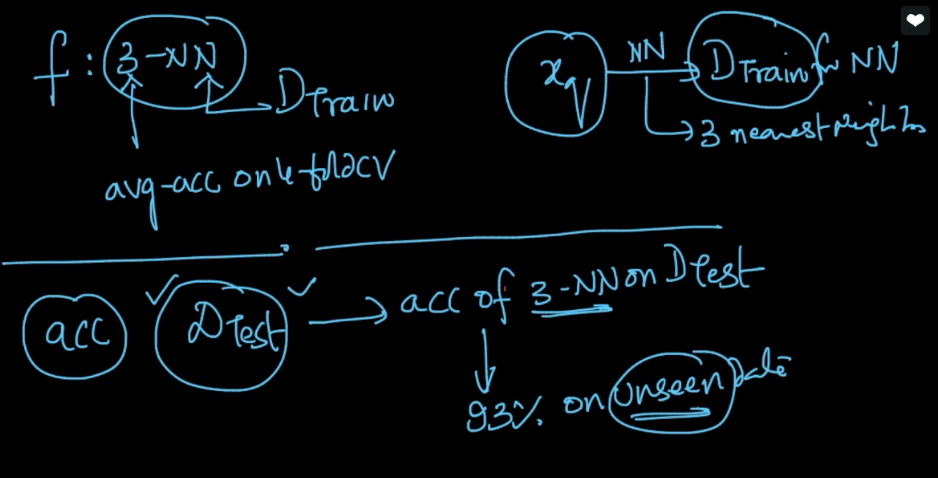
**Benefits of this:**

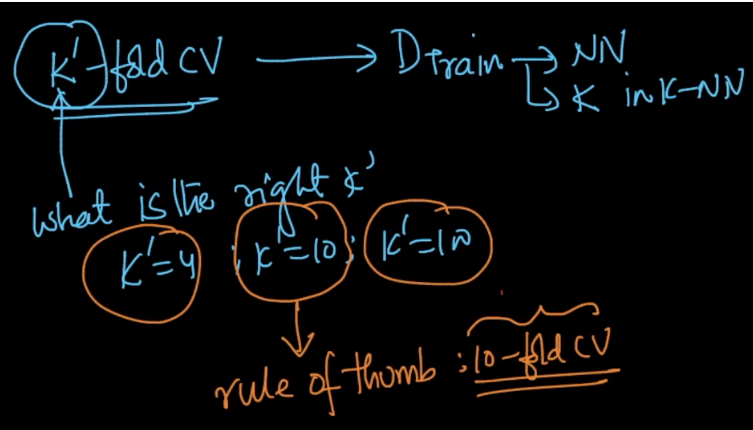
As you can see for computing k we are using 80% of our data, and not 60%, therefore our model will improve.

Here value of K’ can be anything.

**What Value k’ should have:** Mostly k’ = 10 is used in most of the cases.

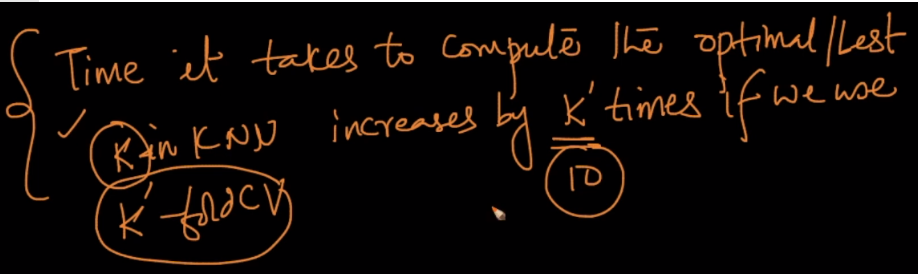






**Drawback of k’-fold CV:**

While hyperparameter tuning (finding value of k), it’s time complexity will be increases by k’ times.



**Comments:**

K-NN itself takes huge space and time complexity and hence we don't generally use it in the realtime deployment. then how K-fold CV can be used over it in terms of speed and space metrics?  
also, are there any real time use-cases or applications where K-fold CV is effective?

We donot use Crossvalidation in real time. See we use CV for hyperparameter tuning to get the best parameter. After we get the best parameter of the model, we train the model on whole data using that data and in test time we just calculate the output based on that model. In case of Knn, we find the best k using CV, then at testing time we use that best 'K' to test for the test query using the whole train dataset.

**Stratified KFold**

Kfold validation does not preserve the split of the output variable while splitting the data in k-folds. Imagine training a Naive Bayes classifier using KFold validation using 10 samples where 5 are positive and 5 are negative. Since KFold randomly selects the split imagine splitting it in an unfortunate way -- 1 split contains all positive samples and 1 contains all negative. Naive Bayes classifier will calculate the prior probabilities and find it to be 100% i.e. the model will think the output is always positive which is obviously wrong. To tackle this scenario we use Stratified split, what it would essentially do is preserve the split in the original dataset in training set, that is, if the original dataset has 50% positive and 50% negative outputs then the training set will also have 50% positive and 50% negative outputs.