Short Notes On Lasso & Ridge :-

* lasso Regularizer containing absolute value function, this values ie (absolute) are not diffrentiable and it contain more computational resources ie taking more time to compute.
* Ridge Regularizers containing Square value function, this values are diffrentiable and it doesn't contain more time to compute because it is a simple resource.
* We have a advantage in Lasso regr is it contain feature selection. (ie what ever feature is not important it contain its coefficent as zero for Unwanted features)
* Ridge regr has not this advantage of feature selection, both the Regularizers perform same task to resolve the issue of "Overfitting".

General Termonologies :-

① In K'NN. Regression | classifier we have to choose 'K' value but how to know which 'K' we have to use, by expeormentally only we can say that we have to choose this K. value, In other terms we can't deside 'K' value.

② Eqn - $m_1 = m_0 - \eta \left[\frac{\partial f}{\partial m}\right]_{m_0}$ is update function of gradient - descent. where (η) is learning rate, what 'η' value we have to choose in eqn again we have to do experiments with diffrent values. by expernmental only we can deside the values.

from the above ($\underline{K}$ - in KNN, $\underline{\eta}$ - GD, $\underline{\lambda}$ - Lasso, Ridge) all are Hyper parameters which are tune the algorithms.

Scanned with CamScanner

difference b/w parameter & Hyper parameter :-

Data → Algorithm → Model → m^*, c^* ⇒ Linear Regression / Logistic Regression

Algorithm → Hyper parameters

m^*, c^* → parameters → Best-fitted / separated value.

Parameter is nothing but output of our learning Algorithm (ie model)

in the above case (m, c) are parameters.

Hyperparameter is nothing but which are tune the algorithm / model with this we solve the issues of "overfitting". by this tuning the hyperparameters we will achive the Low bias and low Variance (not too low var).

Parameters Comes from the model. (output), where Hyper parameters are part of an Algorithm

Linear Regression

$(y_{act} - y_{pred})^2$

$$\arg_{m,c}\left\{\text{Min} \sum_{i=1}^{n} (y_i - (w^T x + c))^2 + \text{Reg}\right\}^0$$

Lasso: $\lambda * \sum_{i=1}^{d} |w_i|$

Ridge: $\lambda * \sum_{i=1}^{d} (w_i)^2$

Logistic Regression

$(y_{act} * y_{pred})$

$$\arg_{w}\left\{\text{Min} \sum_{i=1}^{n} (1 + \exp\{-y_i * (w^T x)\}) + \text{Reg}\right\}^0$$

Lasso: $\lambda * \sum_{i=1}^{d} |w_i|$

Ridge: $\lambda * \sum_{i=1}^{d} (w_i)^2$

d → dimentionality

for Suppose if we Consider Regularizer-term in either linear Regression or logistic regression where we use Either Lasso or Ridge Regularizer

If we Consider λ value as zero (ie ⇒ $\lambda = 0$)

Scanned with CamScanner

Then the total term of regularizer will be zero. as Shown in eqn., then we get only linear reg-term or logistic regr-term ie our model having "overfitting" problem., if we have to Improve lambda (λ) value then we get Underfitting problem. because there only regularization term value increases then no impact of linear and logistic

$\lambda \uparrow \rightarrow$ Underfit $\quad \lambda \downarrow \rightarrow$ Overfit

This means there is not maintain high lamda value and low lambda value. keep change the values of λ ie we have to tune every time with λ value in Order to get best model.

Q - what if Overfit and Underfit a problem ?

A - when ever we have a overfit or Underfit issue, we have to use Regularizer term., this Introduce the 'λ', which tune our entire algorithm, ie it helps to our model from Overfitting / underfitting Issue.

Hyper parameter tuning in S.V.M :-

In S.V.M our task is to minimize the Margin,

S.V.M eqn - $\text{Min}\left\{\frac{1}{2} \times \|w\| + C * \frac{1}{n}\sum_{i=1}^{n} \xi_i\right\}$ ($C \rightarrow 0$)

In the above eqn $\rightarrow$ C is Hyperparameter., if "C=0" then

$\text{Min}\left\{\frac{1}{2} \times \|w\|\right\} \Rightarrow$ which simply means we maximize the margins. then $\text{Max}\left\{\frac{2}{\|w\|}\right\} \Rightarrow$ So this eqn is nothing but "Hard Margin" S.V.M equation. the main problem with this eqn

Scanned with CamScanner

Is to the data is Linearly Separable with out any misclassifications (ie there is misclassification but we assumed it).

So this hard Margin S.V.M eqn gives us Underfit Issue because Initially we Consider there is no misclassifier point but In real world we didn't get like this data., Some data points Overlapping with other points.

Now we Consider $C = 0$ then the entair term will be get '0' and we left with only Hard margin eqn and thats gives us Underfitting Issue.

$C \downarrow$ — Underfitting $\quad C \uparrow$ — Overfitting.

from this we can Say if Our 'C' value decreases we face Underfitting issues, if we increase C value we get Overfitting Issues.

In SVM we have kernel's also if we Consider R.B.F there is kernel value will be — $\exp\left\{-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right\}$

So here 'σ' is hyperparameter, So we can represent it with 'γ' also $\Rightarrow$ $\exp\{-\gamma \|x_1 - x_2\|^2\}$

where — $\gamma = \frac{1}{2\sigma^2}$.

Note: ① if we Consider linear SVM eqn → we have only One hyper parameter that is 'C'

② if we Consider kernel SVM with R.B.F → we get two hyper-parameters are — C, γ So In this Case we have to tune two values where In Simple S.V.M eqn we have to tune only 'C' value.

Scanned with CamScanner

Hyperparameters with different algorithms :-

① linear or logistic Regression → $\lambda \downarrow$ – overfit problem
$\lambda \uparrow$ – Underfit problem

② Support vector Machine → $C \downarrow$ – Underfitting ; $\gamma \downarrow$ – Underfitting
$C \uparrow$ – Overfitting ; $\gamma \uparrow$ – Overfitting

③ k-Nearest Neighbour → $k \downarrow$ – overfitting
$k \uparrow$ – Underfitting

In kNN we have two hyper parametrs are – k, P.

wher 'p' is a distance Matrix when $p = 1 \Rightarrow$ Manhatten distance
$p = 2 \Rightarrow$ Euclidian distance
$p = 3 \Rightarrow$ Minkowski distance

In Order to get best model in kNN we have to tune both (k, P) parameters by experiments like more hit & trial methods.

④ Decision Tree → Depth ↑ – Overfitting ; No of points in leaf
Depth ↓ – Underfitting ;

In D.T we are memorize the entire data. where we mention each and every condition that gets more complex and increases depth of the decisions ie deeply going to conditions. So that is consider as a Overfitting Issue.

⑤ Random forest → Depth + No of D.T + No of points in leaf

In Random forest also same thing in Decision tree because in it also we have decision makings while teaining data.

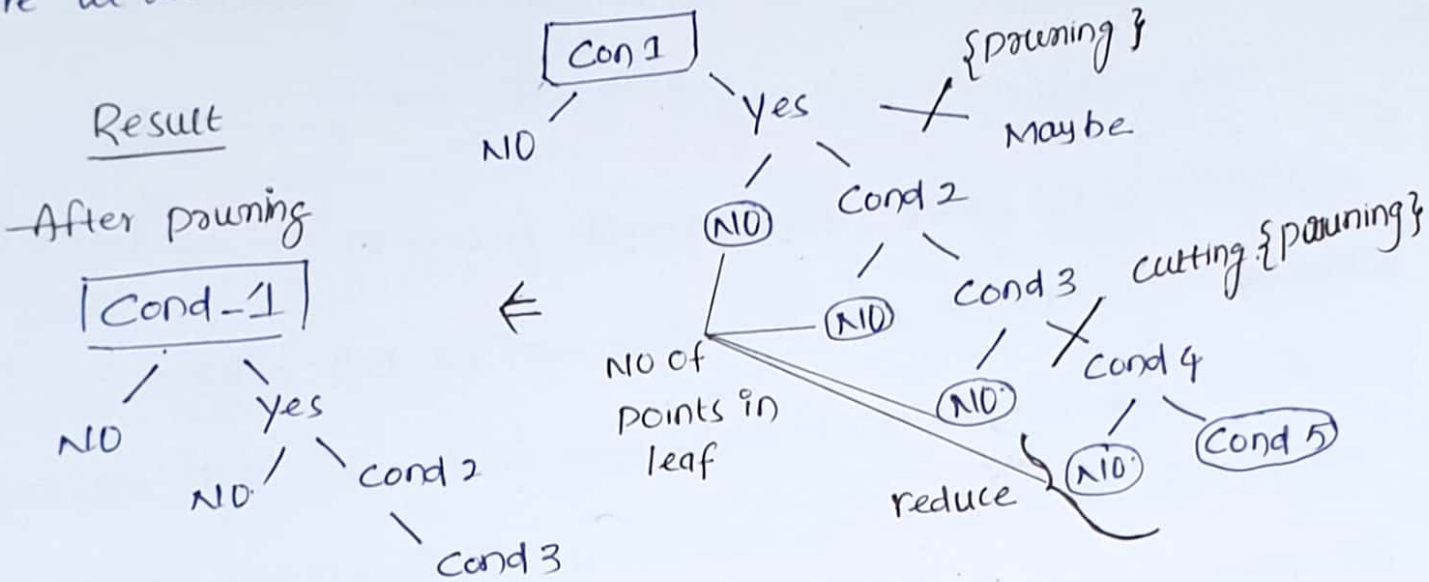
In Random forest → Base learners (D.T) + Row & Column Sampling + Assrigators
↳ Over-fitting → Depth (↑)

that gives best Model.

Scanned with CamScanner

Q - How Do you treat Overfitting in DT?

A - When we create a Decision tree for Suppose it has issue of overfitting (ie we have high depth) it contains more decisions making after getting deep tree, we have a technique called as pruning., by this we cut down the depth of DT, then the remaining part will be lost- ie we are reduce the depth of decision tree.



Q - How do you treat the overfit issue in DT?

In Decision Tree we have to use Regularizer which are

① Pruning

② changing the hyperparameter → H.P for DT is → Depth & # of points in leafs.

③ Random forest

Note :- NO of points in leaf's means - terminal end points in DT

In case of linear Reg - we use Hp → λ

logistic Reg - λ

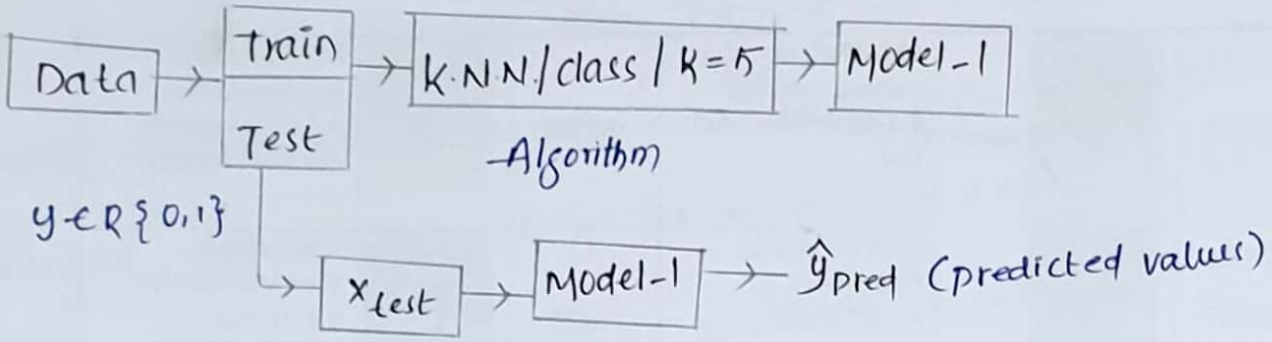
SVM - C, γ (where RBF kernel)

} Gradient Descent - η ↓ Hyperparameter

In this all algorithms after tuning we have to again tune the values with Gradient descent 'η'

Scanned with CamScanner

Hyper parameter tuning process in K.N.N. Algorithm :-



Accuracy → Train data → y_{trn}, $\hat{y}_{trn\text{-}pred}$ – 90%.
↳ Test data → y_{tst}, $\hat{y}_{tst\text{-}pred}$ – 60%. } Overfitting

Step-1 :- In process of Hyperparameter tuning Let us Consider a dataset after that we have to Split it in-to two parts as train data, test data. then take train data which is classification type ie the 'y' target variable is categorical type So we have to do classificatron task. So here we applying K.N.N. Classification algorithm to train data and Consider hyperparameter "K" as 5 and that gives a model which we Consider as "model-1", after making a model, we take X_{test} data value and put it in model that gives us to predicted values of "$\hat{y}$"

Step-2 :- Now we have to use Accuracy matrices to calculate the accuracy of a test data aswell train data. for suppose Consider train data (y_{trn}, $\hat{y}_{trn\text{-}prd}$) that gives us 90% accuracy. Similarly consider tst data (y_{tst}, $\hat{y}_{tst\text{-}prd}$) that gives 60%. then we can Understood Our model has Overfitting Issue.

Scanned with CamScanner

Step-3 :- for Suppose if we change the values of 'k' we will get the diffrent accuray values, because the 'k' takes the nearest points consideration, thats why entair values of our model will be changed., here now we change the algorithm ie k value, So let "k = 7" then do entair process of Step-1 we will get accuray values are —

Accuray-2 → Train data → 95% } Overfitting issue.
↳ Test data → 80%

Now look at to Acuuray of Second model-2 in train data perform well, now test data also performed well but as compared to train set low result, which indicates "Overfitting" Issue again.

Note :- Here our goal is to make a best model for predictions which has no issues of Overfitting and Under-fitting. but what ever values we get from model-1 & Model-2 both are having Overfitting issue So here we have to reduce that issue.

In Ordus to reduce this issues we have to keep change the values of 'k' Hyperparameter ie tuning the hyperparameter that change the model and values of accuray. by this we get best values of Accuray In Order to do by hit & trail or every time do experiment Untill Unless we reach the best model.

Now we Consider k = 101 for Suppose. apply the Same data and do Step-1 we get model-3 that shows accuray as below —

Scanned with CamScanner

Accuracy-3 → Train-data — 50% } Underfitting
↳ Test-data — 60%

Now we get train data accuracy as 50% and test data as 60%. In this test data has more accurate values but in train data we didn't have good accuracy this gives us result as Underfitting.

If we observe accuracy result of train and test, test has more Accuracy as Compare train but with 60% result we can't predict well. So we have to improve our accuracy In order to get 90% result in test prediction. then we can say that is good model.

So again we have to do trials in Order to get best model, let Consider k = 13 and do Step-1 we get model-4. If it's shows values as-

Accuracy-4 → Train-data → 90% } Best model
↳ Test-data → 91%

So here "k=13" where model-4 gives values of accuracy is predicted well in train data as well test data. So we can say that our model-4 is "Best model" which has no issues of Overfitting and Underfitting.

The result "Best model" what ever we get by change of k value ie here every time we tune the model by hyperparameter 'k'

So here we get best model in order to change 'k' value by every time we Observe train and test data. which is Unseened but in real time there is no use of test data. (ie we have to check train data accuracy values) only.

Scanned with CamScanner

There we use the Concept of validation. in this we Split train data into two parts as validation data and train data.

Validation set :- it is a set of data used to train model (AI) with the goal of finding and optimizing the best model to solve a given problem. Validation sets also known as dev sets.

Step-1 :- first of all we have to choose a data and split the data into two parts as train & test- data after that keep test data as it is. Consider trained data. Split this trained into two parts as trained and validation data. after this we have to choose algorithm in order to do this we have to check our target variable if it is a numerical value Contain do regression task., for Suppose if we Consider Categorical then do classification task. So here i choose algorithm. K.N.N. and apply our data to this algo. we get model after it we have to take "X_validated-data", apply to a model that gives $\hat{y}$ pred- values.

Step-2 :- Now here we Consider Accuracy matrices to get Accuracy of a model in Order to do it we Consider both datas and apply

Scanned with CamScanner

matrices to that for suppose initially i considered $k = 5$

Accuracy → Train data $(y_{trn}, \hat{y}_{trn})$ → 95%.
↳ Validated data $(y_{val}, \hat{y}_{val})$ → 85%. } Overfit problem.

Now here we calculate accuracy of both train and validated data. the get good result but as compare to train, valid data has low accuracy this will get the issue of overfit.

Step-3 :- In order to reduce it we have to do hyperparameter tuning So in this algorithm 'k' is a hyperparameter So every time we have to change the value of 'k' ie we have to keep trials on this data. Until unless we get "good model" from ($k = 1, 2, 3, \ldots \hat{e}$) we get models as- (Model-1, Model-2 .... model_$\hat{e}$) & Accuracies are also respectively. So from this models we pick easily "best model". as similar as hyperparameter tuning on train and test data.

But here one biggest dropback of validation process is we take train data and further devided as validation, train data. so here we decrease the train samples from train data that will be effect on result like accuracy ie Overfitting (or) Underfitting issues majorly., for suppose we have 1000 data points.

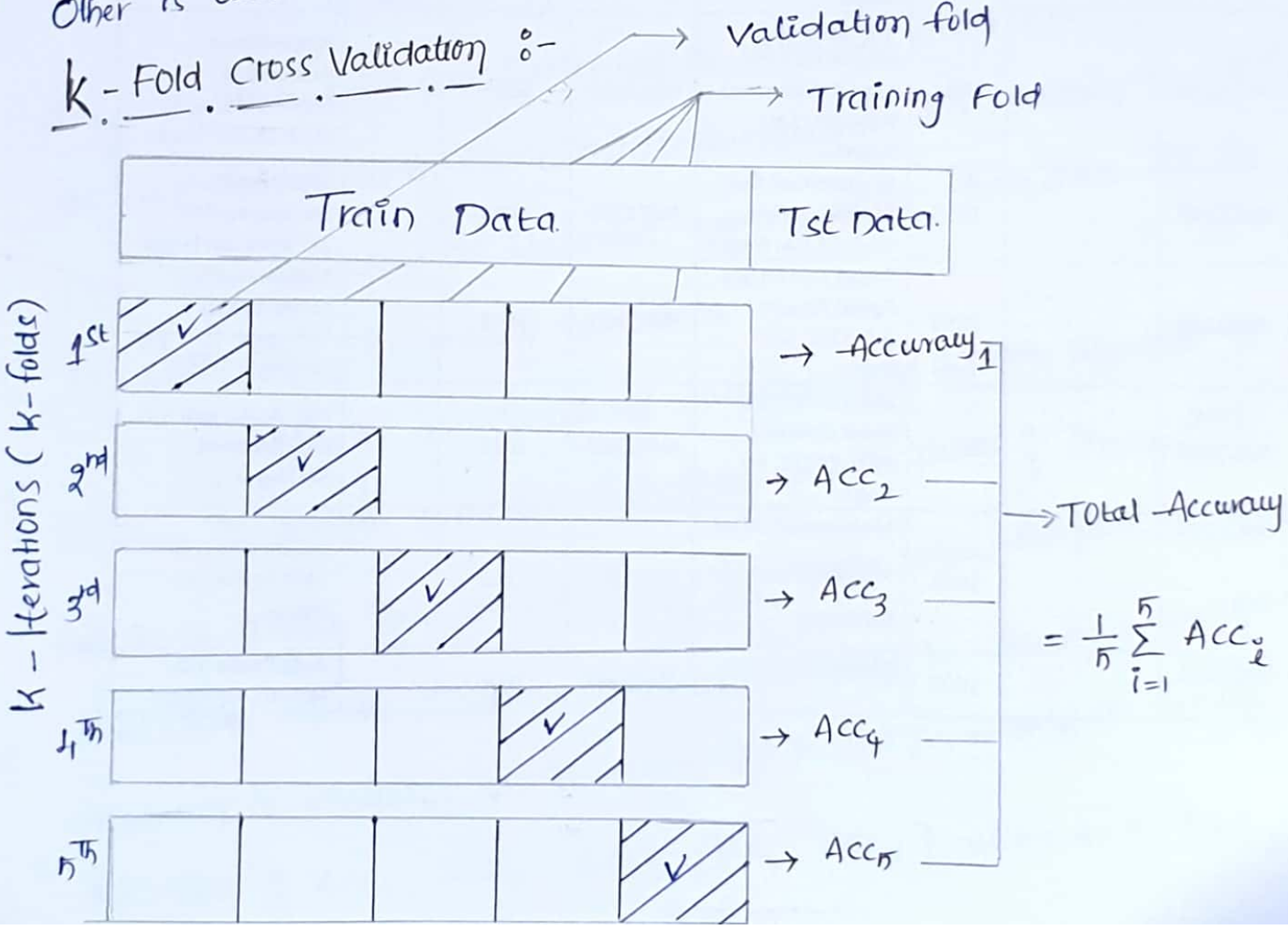
1000 (Data points) → 700 (trn) → 500 (trn) ✓
→ 200 (valid)
→ 300 (tst)

So if we look in to train data points initially they are 1000 data points after validation split we are left with 500 data points only ie half of the data points, but we know that Inorder to get a best model we need to train more data points, then only we get best performances like Accuracy, the resultant tells us wether the model is best or not.

"In Order to prevent this problem we have to use atechnique called as Cross validation."

it is a statistical method of evaluating and comparing learning algos. by deviding data into two segments One is used to learn or train model Other is used to validate the model.

K-Fold Cross Validation :-



Scanned with CamScanner

In this validation process firstly we take a dataset and devide it in-to two parts as train data and test data now Consider train data and Split it further k-folds in this we have to take k-value as {5, 7, 10 -- etc} any real value but in generally Consider "k=10" here this k is different from KNN., So this 'k' will be devided that part in-to Such many kinds., here we Considered k=5 now we will get "5-folds" ie train data Split in-to 5 many-times with 5 parts in train data. So far Suppose if we devided Split Intially data as 80% train and 20% test, this 80% devided equally to all train parts as shown in figure ie 16% equally devided each part.

So if we looking in-to the diagram in 1st iteration of train data. the first part will Considered as validated data and remaining all are Considered as train datas, Similarly in 2nd iteration the 2nd part will Considered as validated remining all are train data. lets do it Untill the 5th iteration train data.

After it we Considered train data and applying algorithm to it then it makes a model, then after Consider values of X_{train} and put in-to model that gives us $\hat{y}_{pred-valid}$ value Compare it with y_{val} then finally applying performance matrix ie Accuray we get Accuray of model-1 (or) iteration 1st result, Similarly do for all we get $\{Acc_1, Acc_2, Acc_3, Acc_4, Acc_5\}$ where $k=5$, $P=1$

Scanned with CamScanner

Then finally we do Summation of all the values we get total Accuracy at $\{k=5; p=1\}$

So if we observe here we consider $k=5$ (which means KNN points) and $p=1$ (Manhatten distance), which are hyperparameters, and 'k' is a fold in Cross Validation. here $k=5$ we decided So we get '5' models with 5 iterations. So if we change the values of ($k=5,7,9$; KNN) and $p=(1,2)$ Intially Consider $\{k=5; p=1\}$. Now we take $\{k=5; p=2\}$ again we get 5 iterations with 5 models. So now look in to how many pairs we have.

there are — $k = \{5, 7, 9\}$, $p = \{1, 2\}$ } ⇒ Totally we got '6' pairs.

for each pair we have '5' models because of our k-fold is '5'

from this we got total '30' models internally. when we apply "k-fold".

In practice it is very difficult to train the '30' models, In Order to Overcome it we have to use Some techniques in k-fold cross validation. those are—

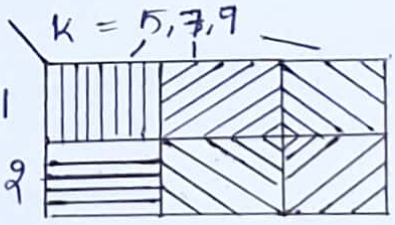
① Grid Search Cross Validation

② Random Search Cross Validation.

Grid Search C.V :- In this process we have to grid all the parameters what ever we have $\{k, p\}$

⇒ $k = \{5, 7, 9\}$

$p = \{1, 2\}$



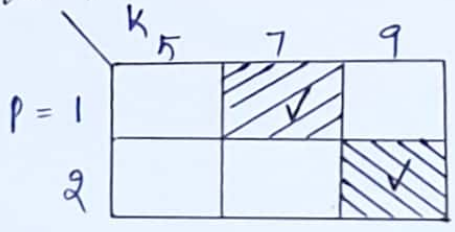
Scanned with CamScanner

So in Grid Search Cross Validation firstly we have to make a Structure of Grid Containing all values of hyper parameters (K, P) In Case of K-NN Algorithm as shown in figure, then we have to train model as- K=5 and P=1, then train another model where {K=7, P=1} Similarly we have to do it till {9,2} as shown in figure., where we train total data by taking different hyper parameters, every time this will be shuffle data where you mention random state or not.

Randomized Search Cross Validation :-

R.S.C.V is better than the Grid search C.V. interms of Computational Complexity. In Grid Search CV it computes each and every pair that means in above example we have '30' models all this models Compute by Grid Search C.V but in Randomized Search C.V its don't do all models Computation. just randomly Select the models to train then it will pick '5' randomly Selected models, in Case-II Select 10 models, In Case-III Select 8 models for Suppose, all this Selection is done by randomly. and do Cross Validation

∴ Both the techniques do same Operation but in Grid do all models validation (ie train), in case of R.S.C.V it pick randomly the model as Compare to Grid, random.S.C.V has less Computational Complexity, So from this we can say R.S.C.V is better than G.C.V., Generally if we have low models pick G.S.C.V



K = {5, 7, 9} ; P = {1, 2}

out of this pick Randomly '2' models only.

Scanned with CamScanner