29.13 Need for Cross-Validation

The main purpose of performing cross-validation is to find out the optimal hyperparameter 'K' value for the K-NN model. Let our given dataset be represented mathematically in the form of a set as

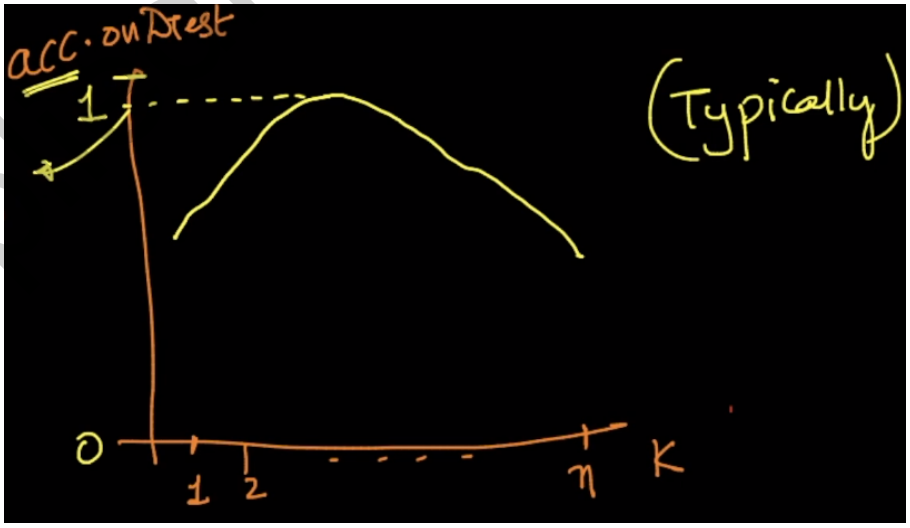
$$\mathbf{D = \{(x_i, y_i) | x_i \in R^d, y_i \in \{0,1\}\}}$$

How to choose the 'K' value?

1) Divide the dataset 'D' into the training set (D_{Train}) and the test set (D_{Test}).
2) For different values of 'K', fit the K-NN model on 'D_{Train}' and make predictions on 'D_{Test}' and then compute the test accuracy.

K	Accuracy on D_{Test}
K=1 K=2 . . K=5 . . .	$accuracy_1$ $accuracy_2$ . . $accuracy_5$ . . .

3) Build a 2D line plot with all the 'K' values on the 'X' axis and the corresponding Test 'Accuracy' scores on the 'Y' axis, as shown below. Whichever value of 'K' yields the highest test accuracy score, that would be chosen as the optimal value.



But here we have an issue. The main goal of machine learning is to learn a function 'f' and give the best accuracy on the unseen data. If the model doesn't give the best accuracy on the unseen data, then the model itself is useless.

Here we are using 'D_{Train}' to learn the function of finding out the nearest neighbors, 'D_{Test}' to find out the optimal 'K' value. There needs to be an unseen dataset on which we can measure the accuracy(Here 'unseen' in the sense, it is not used either for finding out the nearest neighbors or for finding out the optimal 'K'). So this concept of splitting the dataset into two parts(ie., D_{Train} and D_{Test}) fails, as this is not serving the actual purpose of Machine Learning.

For example, if we got an accuracy of 0.96 for K=6 on 'D_{Test}', it means that our model predicts the results correctly on 'D_{Test}' in 96% of the cases, when K=6. But as 'D_{Test}' has been used/seen by the model for finding out the optimal 'K', we couldn't say for sure that the same model could give the same accuracy on some unseen data.

When an algorithm works well for unseen future data, we call it **generalization**. The accuracy computed from the predictions made on such unseen data is called Generalization Accuracy. So we should split the dataset in such a way that we also get an unseen set of data points, to compute the generalization accuracy.

So we split the dataset into 3 parts. They are the training dataset (D_{Train}), cross-validation dataset (D_{cv}) and test dataset(D_{Test}).

D_{Train} → Used for learning the function 'f' to obtain the nearest neighbors

D_{cv} → Used for finding out the optimal 'K' value

D_{Test} → The unseen data used for computing the generalization accuracy.

Procedure for Cross-Validation (Simple Cross-Validation) on K-NN using D_{Train}, D_{cv} and D_{Test}

1) For different values of 'K', fit the K-NN model on 'D_{Train}'.
2) For every value of 'K', after fitting the model on 'D_{Train}', make predictions on 'D_{cv}' and compute the accuracy score for the predictions made on 'D_{cv}'.
3) Build a 2D line plot with all the 'K' values on the 'X' axis, and their corresponding cross-validation scores(ie., accuracies computed on 'D_{cv}') on the 'Y' axis.
4) Pick the value of 'K' which yields the highest cross-validation accuracy, and that will be your optimal 'K' value.
5) After obtaining the optimal 'K' value, we have to fit the K-NN model on 'D_{Train}' using the optimal 'K' value, and then make predictions on 'D_{Test}'(unseen data). Compute the accuracy for the predictions made on 'D_{Test}' and this accuracy is called 'Generalization Accuracy.'

Q) **Why do we need cross-validation?**

Ans) Cross-Validation is used to assess the predictive performance of the models and to judge how well they perform on unseen data.

The motivation to use the cross-validation mechanism is that when we fit a model, we are fitting it on the training dataset(D_{Train}). Without cross-validation, we only have the information on how our model performs on the seen data.

Ideally we would like to see how does the model perform when we have new data(ie., unseen data), in terms of accuracy of its predictions.

Q) **Which value of 'K' should be chosen, if multiple 'K' values give the highest cross-validation accuracy?**

Ans) If we get the same highest accuracy for multiple values of 'K', then if we need the test time to obtain the nearest neighbors to be low, we should go for the least value of 'K' among those which give the highest cross-validation accuracy.

If we have lots of data, and.test time is of no concern, then picking the largest value of 'K' is preferable, as we can have more trust in the decision, as there are more points supporting our decision/class label.