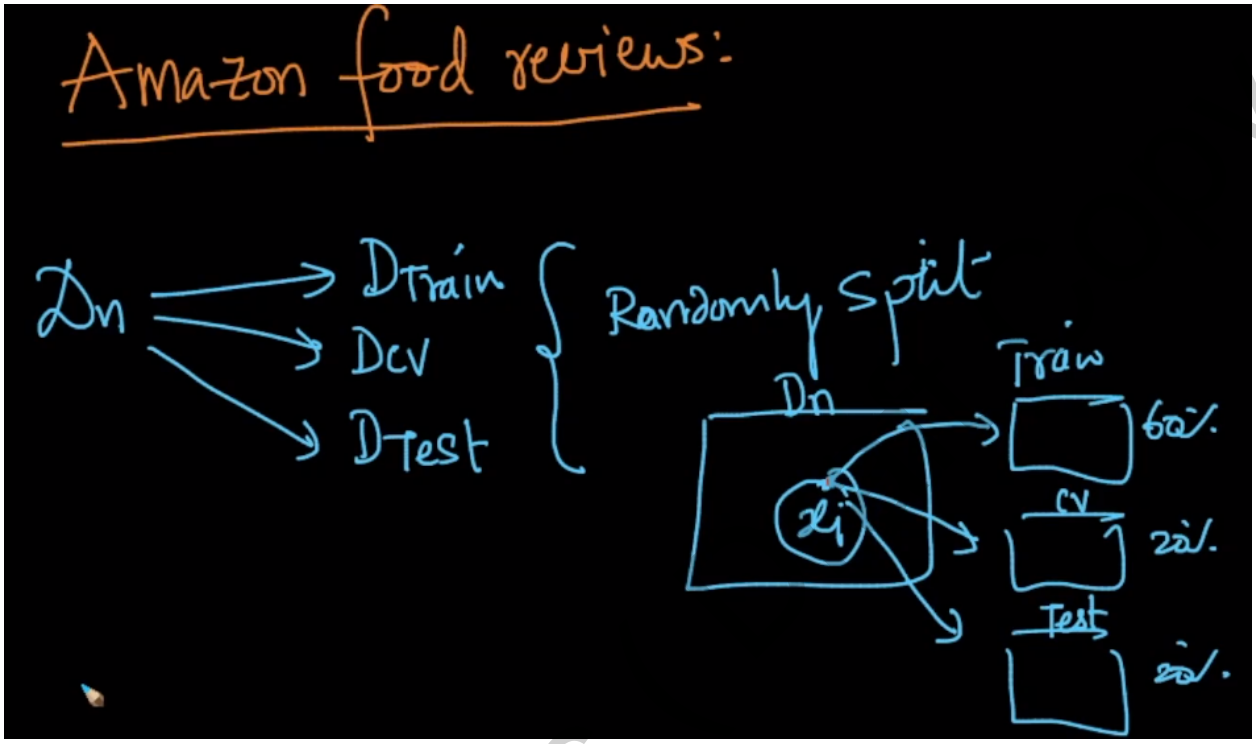
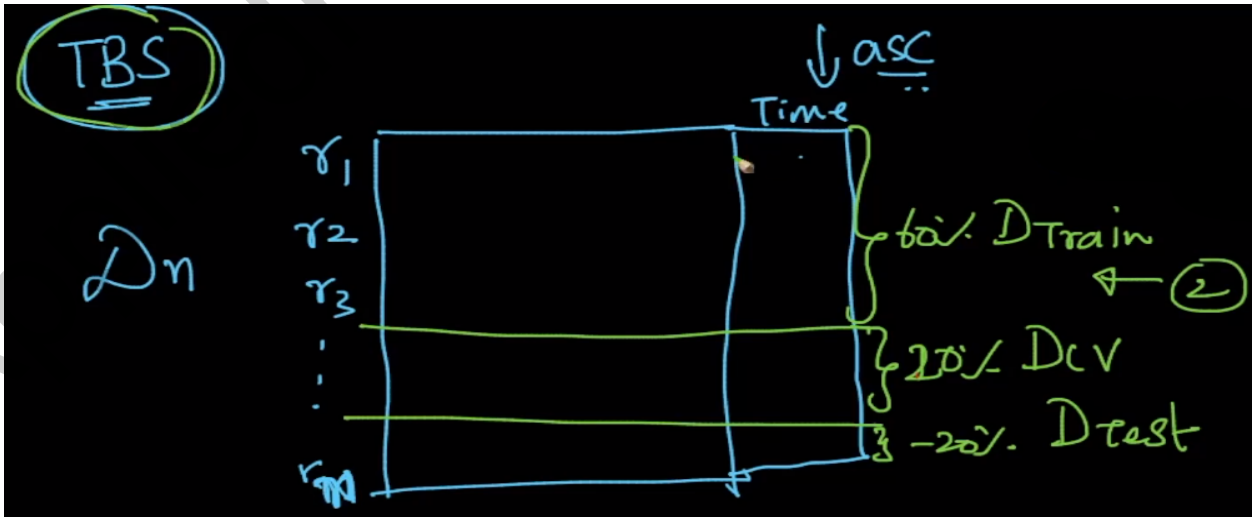
29.17 Time Based Splitting

So far we have split the dataset into D_{Train}, D_{cv} and D_{Test} using random splitting.



We also do have another strategy of splitting the dataset which is known as Time Based Splitting. There are certain problems where Random Splitting is better than Time Based Splitting, and in some problems, Time Based Splitting is better than Random Splitting.

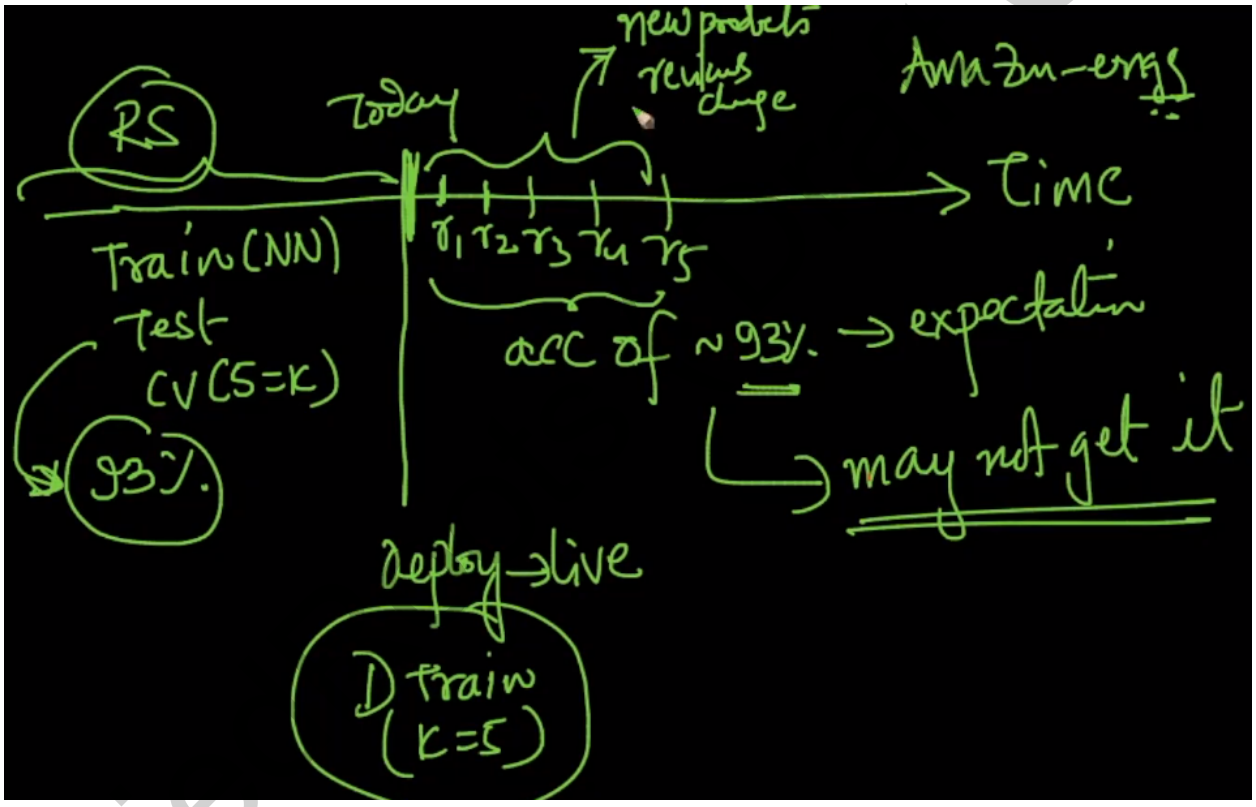
In order to perform Time Based Splitting, we need to have the timestamp column in the dataset.



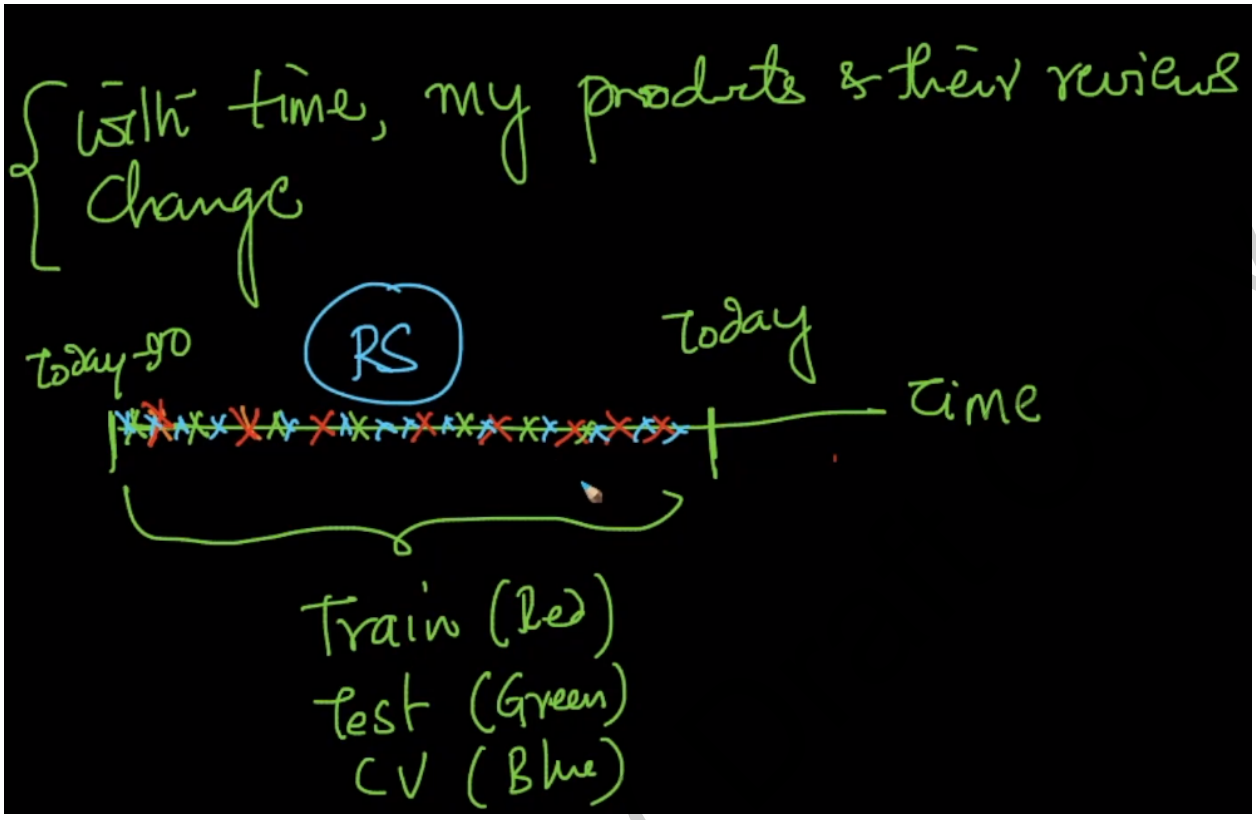
Procedure to perform Time Based Splitting

1) Sort the dataset 'D_n' in the ascending order of the time column values.
2) Now we split the dataset with the first 60% of the points into 'D_{Train}', the next 20% of the points into 'D_{cv}', and the last 20% of the points into 'D_{Test}'.

In case of Random Splitting, we had
D_{Train} → For finding the nearest neighbors
D_{cv} → For choosing the optimal 'K' value
D_{Test} → For measuring the accuracy



The above figure shows applying random splitting on a real-time dataset. Here for the products, the reviews would change as the time keeps progressing. It doesn't mean the already existing reviews would change, but as the time keeps progressing, we do get many more reviews on the similar products purchased, and the reviews/feedback would change. In such a case, if we perform a random splitting, there are chances for a few of the latest reviews to go into the D_{Train} and D_{cv}, and a few of the older reviews to go into D_{Test}. Due to this, the model once trained in the past and gave a good score on D_{Test}, if it is deployed, might not give a similar score on the latest data. It is because as the patterns in the data keep changing, the model performance also keeps changing.

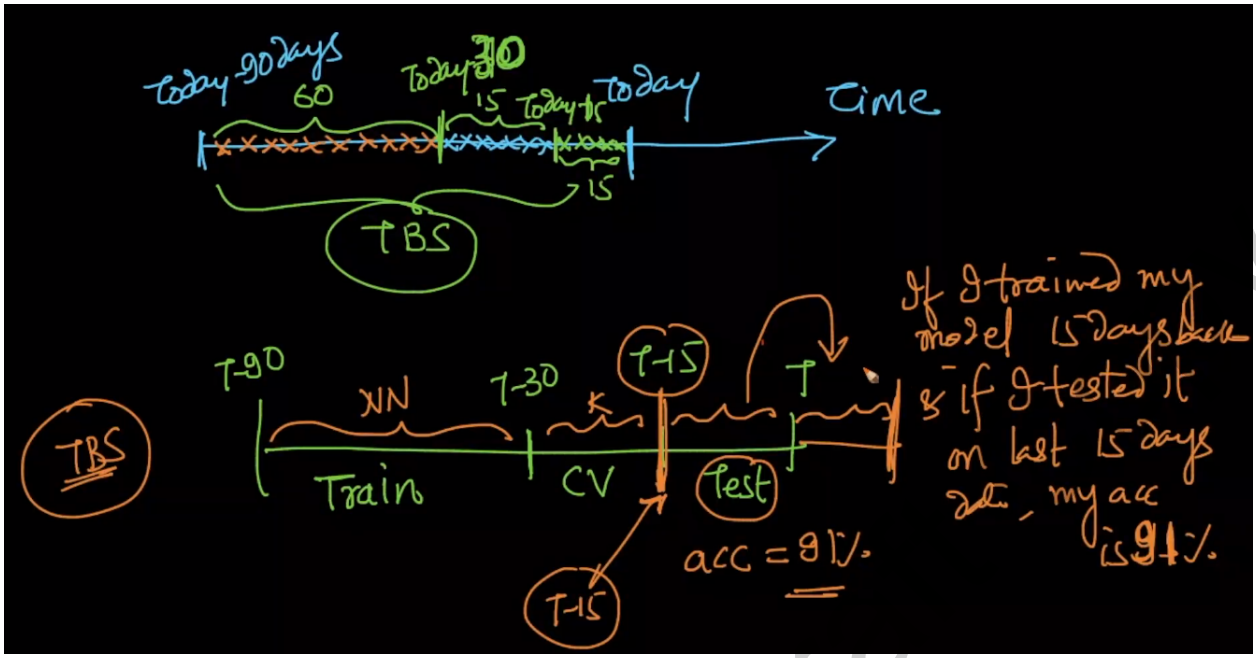


Also there are chances for new products to get added, and also the older products to disappear. In case, if a new product gets added today on the website, and if there is no data about similar products previously, then we couldn't make a valid prediction. Similarly, if we have a product that is no more available for sale, then it is always better to keep it in D_{Train} because, if it comes in D_{Test}, then making predictions again on it, is just a waste of time and meaningless, as that product is never going to be added again for sale. Hence in order to get rid of all these kinds of problems, we go for Time Based Splitting.

In case of the time based splitting, we split the older data into D_{Train} and D_{cv}, and then the future data would be D_{Test}.

Whenever we apply Random Splitting, if we train, test and deploy the model, and if it gives an accuracy of say x%, then in future the same model might not give similar accuracy for the future data, whereas whenever we apply time-based splitting, if we train a model with a particular set of data belonging to a particular interval, cross-validate on a particular interval and test the model on a particular interval of data and finally deploy it, then the accuracy obtained on the test data will be the almost the same for the future data as well.

Note: People wrongly understand that the consistency in the accuracy as mentioned above is due to the presence of the 'Time' column in our dataset. It is not because of the presence of the 'Time' column, but due to the way in which we split the dataset.



Note: Whenever the 'Time' column is available in the dataset, and if things/behavior of the data changes over the time, then time-based splitting is preferred over random splitting.

In case, if the 'Time' column is not present in the dataset, then we have to go for Random Splitting, as we have no other option.