**What is Cross Validation?**

* Cross validation is a statistical method used to estimate the performance (or accuracy) of machine learning models. **For cross validation, we’ll be splitting the training set again into two sets, one will remain the training set, and the other will be known as the validation set.**
* you have trained the model with the dataset available and now you want to know how well the model can perform.
* One approach can be that you are going to test the model on the dataset you have trained it on, but this may not be a good practice.
* Our main objective is that the model should be able to work well on the real-world data, although the training dataset is also real-world data, it represents a small set of all the possible data points(examples) out there.
* So to know the real score of the model, it should be tested on the data that it has never seen before and this set of data is usually called testing set.
* It is *also of use in determining the hyper parameters of your model*.

## ****Types of Cross Validation :****

* There are different types of cross validation methods, and they could be classified into two broad categories
  1. Non-exhaustive Methods :

1. Holdout Method
2. K fold cross validation
3. Stratified K Fold cross validation
   1. Exhaustive Methods.
4. Leave-P-Out cross validation
5. Leave One Out Cross Validation (LOOCV)

https://www.ritchieng.com/machine-learning-cross-validation/

**Holdout method :**

1. In this method, we split our training set and take out a small part as the validation set. We train our model with the new and smaller training set, and validate the accuracy of the model on the validation set, which is still unseen by the model.
2. But there’s a problem here. Because we don’t know what data will be in the training set and what data will be in the validation set, we might end up with high variance.
3. For different sets of training and validation sets, we might end up with different results. To avoid this, we’ll be using a variation of the holdout method.

**K- Fold Cross Validation :**

1. When we split our training dataset to get a validation set, there’s always a risk of losing some crucial data from the training set, or of losing patterns which might go unnoticed by the model. This will in turn lead to overfitting or underfitting.
2. To avoid this we need enough amount of data in both the training set and the validation set. And for this, we use K-Fold Cross Validation.
3. In this method, the original training set is divided into k subsets. The holdout method is now repeated k times with different datasets. In each fold, one of the k subsets is taken as the validation set, and the remaining k – 1 subsets are used as the training set. The error estimations from all the folds are taken and averaged to give us the final error estimation of the model.

**Let us go through this in steps:**

1. **Randomly split your entire dataset into k number of folds (subsets)**
2. **For each fold in your dataset, build your model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for kth fold**
3. **Repeat this until each of the k-folds has served as the test set**
4. **The average of your k recorded accuracy is called the cross-validation accuracy and will serve as your performance metric for the model.**
5. Because we’re using all the k sets for validation, each datapoint appears in the validation set exactly once. And each point of data appears in the training set exactly k – 1 times. This greatly improves the accuracy of the model.
6. This also **reduces** **bias** as most of the data is being used for **fitting**, and **reduces variance** as most of the data is also being used for **validation**. And because we’re interchanging the data in each fold, it improves the overall efficiency of the model.
7. Depending on your dataset, you can select a k value of your own. But in most cases, k = 5 or k = 10 is preferred.

**Stratified K-Fold Cross Validation :**

1. In some cases, data might not have been divided properly betwen the training and validation sets. For example, in a **classification problem**, there may be a large number of negative outcomes in a validation set, and in another validation set, it could be the opposite. This will again lead to bias and high variance in the outcome.
2. To avoid this, we make a slight modfification in the K-Fold Cross Validation method, such that in each set, we make sure there are equal or close to equal results of all categories.
3. In case of continuous values, we make sure the means of all the outcomes are comparable. This variation of K-Fold is known as the Stratified K-Fold Cross Validation.

**Leave-P-Out Cross Validation :**

1. In this method, if there are n data points, n – p data points are taken in one iteration and the remaining p data points are used for validation. This kind of iteration goes on for all possible combinations of p from the original dataset.
2. The errors from all these iterations are averaged to get the final efficiency figure.
3. This is classified as an exhuastive method because the model has to be trained for all combinations of the dataset. If you choose a high value for p, this wil be even more exhaustive.

**Leave-One-Out Cross Validation :**

1. This is a variation of the Leave-P-Out cross validtion method, where the value of p is 1. This is much less exhaustive as the value of p is very low. This means the number of possible combinations is n, where n is number of data points.

**Rolling Cross Validation :**

For [time-series](https://www.mygreatlearning.com/blog/time-series-analysis-and-forecasting/) data the above-mentioned methods are not the best ways to evaluate the models. Here are two reasons as to why this is not an ideal way to go:

1. Shuffling the data messes up the time section of the data as it will disrupt the order of events
2. Using cross-validation, there is a chance that we train the model on future data and test on past data which will break the golden rule in time series i.e. “peaking in the future is not allowed”.

Keeping these points in mind we perform cross validation in this manner

1. We create the fold (or subsets) in a forward-chaining fashion.
2. Suppose we have a time series for stock prices for a period of n years and we divide the data yearly into n number of folds. The folds would be created like:

iteration 1: training [1], test [2]

iteration 2: training [1 2], test [3]

iteration 3: training [1 2 3], test [4]

iteration 4: training [1 2 3 4], test [5]

iteration 5: training [1 2 3 4 5], test [6]

.

.

.

iteration n: training [1 2 3 ….. n-1], test [n]

Here as we can see in the first iteration, we train on the data of the first year and then test it on 2nd year. Similarly in the next iteration, we train the on the data of first and second year and then test on the third year of data.

Note:  It is not necessary to divide the data into years, I simply took this example to make it more understandable and easy.

|  |  |
| --- | --- |
| **KFold** | **ShuffleSplit** |
| * KFold will divide your data set into prespecified number of **folds**, and every sample must be in one and only one fold. A fold is a subset of your dataset. * KFold just divides the dataset into k folds. * In KFold, during each round you will use one fold as the test set and all the remaining folds as your training set. | * ShuffleSplit will randomly sample your entire dataset during each **iteration** to generate a training set and a test set. The test\_size and train\_size parameters control how large the test and training test set should be for each iteration. * Since you are sampling from the entire dataset during each iteration, values selected during one iteration, could be selected again during another iteration. * ShuffleSplit works iteratively.In ShuffleSplit, during each round n you should only use the training and test set from iteration n. As your data set grows, cross validation time increases, making shufflesplits a more attractive alternate. |

**Variance and Bias in Machine Learning :**

|  |  |
| --- | --- |
| **Variance** | **Bias** |
| * Variance, in the context of Machine Learning, is a type of error that occurs due to a model's sensitivity to small fluctuations in the training set. * **High variance** would cause an algorithm to model the noise in the training set. This is most commonly referred to as [**overfitting**](https://en.wikipedia.org/wiki/Overfitting). * A model with low variance means sampled data is close to where the model predicted it would be. * A model with high variance will result in significant changes to the projections of the target function. * Machine learning algorithms with low variance include linear regression, logistics regression, and linear discriminant analysis. * Those with high variance include decision trees, support vector machines and k-nearest neighbors. | * Bias, in the context of Machine Learning, is a type of error that occurs due to erroneous assumptions in the learning algorithm. * **High bias** would cause an algorithm to **miss** relevant relations between the input features and the target outputs. This is sometimes referred to as **underfitting**. * Bias is the amount that a model’s prediction differs from the target value, compared to the training data. * Bias error results from simplifying the assumptions used in a model so the target functions are easier to approximate. Bias can be introduced by model selection. * A linear algorithm often has high bias, which makes them learn fast. * In contrast, nonlinear algorithms often have low bias. |

**How can we achieve both low bias and low variance?**

In practice the most methodology is:

1. Select an algorithm with a high enough capacity to sufficiently model the problem. In this stage we want to **minimize the bias**, so we aren't concerned about the variance yet.
2. Regularize the model above, to **minimize its variance**.

**Resampling data :**

Resampling data is the process of extracting new samples from a data set in order to get more accurate results. There are a variety of ways to resample data including:

* **K fold resampling**, in which a given data set is split into a K number of sections, or folds, where each fold is used as a testing set.
* **Bootstrapping**, which involves iteratively resampling a dataset with replacement.

Resampling can affect bias. If the average prediction values are significantly different from the true value based on the sample data, the model has a high level of bias.

**Cause of high bias/variance in ML:**

The most common factor that determines the bias/variance of a model is its **capacity** (think of this as how complex the model is).

* **Low capacity** models (e.g. linear regression), might miss relevant relations between the features and targets, causing them to have high bias. This is evident in the left figure above.
* On the other hand, **high capacity** models (e.g. high-degree polynomial regression, neural networks with many parameters) might model some of the noise, along with any relevant relations in the training set, causing them to have high variance, as seen in the right figure above.

**How to reduce the variance in a model?**

* The easiest and most common way of reducing the variance in a ML model is by applying techniques that limit its effective capacity, i.e. [**regularization**](https://en.wikipedia.org/wiki/Regularization_(mathematics)).
* The most common forms of regularization are [parameter norm penalties](https://pdfs.semanticscholar.org/presentation/be32/622ae17839111e0fc58dc2f66b680618f8a8.pdf), which limit the parameter updates during the training phase; [early stopping](https://en.wikipedia.org/wiki/Early_stopping), which cuts the training short; [pruning](https://en.wikipedia.org/wiki/Pruning_(decision_trees)) for tree-based algorithms; [dropout](https://en.wikipedia.org/wiki/Dropout_(neural_networks)) for neural networks, etc.

**The Bias-Variance Trade-Off :**

* A model with high variance may represent the data set accurately but could lead to overfitting to noisy or otherwise unrepresentative training data.
* A model with high bias may underfit the training data due to a simpler model that overlooks regularities in the data.
* A linear machine-learning algorithm will exhibit high bias but low variance. On the other hand, a non-linear algorithm will exhibit low bias but high variance.
* Using a linear model with a data set that is non-linear will introduce bias into the model. The model will underfit the target functions compared to the training data set.
* Using a non-linear model on a linear dataset, the non-linear model will overfit the target function.
* The Mean Square Error (MSE) can be used in a linear regression model with the training set to train the model with a large portion of the available data and act as a test set to analyze the accuracy of the model with a smaller sample of the data.

<https://www.youtube.com/watch?v=gJo0uNL-5Qw&list=PLeo1K3hjS3uvCeTYTeyfe0-rN5r8zn9rw&index=13>

<https://www.youtube.com/watch?v=7062skdX05Y&list=PLZoTAELRMXVPBTrWtJkn3wWQxZkmTXGwe&index=73>

<https://www.mygreatlearning.com/blog/cross-validation/>

<https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f>

<https://blog.contactsunny.com/data-science/different-types-of-validations-in-machine-learning-cross-validation>

<https://www.mastersindatascience.org/learning/difference-between-bias-and-variance/>