## Modal Validation

How accurate is our modal.
Which algo is best for dataset.

<u>Cross validation technique</u>

## WHAT IS CROSS VALIDATION

Cross Validation is a technique which involves ***reserving a particular sample of a dataset*** on which you do not train the model.
Later, you test your model on this sample before finalizing it.
Here are the steps involved in cross validation:

- You *reserve* a sample data set , called as VALIDATION SET or Dataset
- Train the model using the remaining part of the dataset
- Use the reserve sample of the test (validation) set. This will help you in gauging the effectiveness of your model's performance.
- If your model delivers a positive result on validation data, go ahead with the current model. It rocks!

## Validation DataSet Approach

In this approach, we reserve 50% of the dataset for validation and the remaining 50% for model training. However, a major disadvantage of this approach is that since we are training a model on only 50% of the dataset, there is a huge possibility that we might **miss out** on some interesting information about the data which will lead to a **higher bias**.

```
train, validation = train_test_split(data, test_size=0.50, random_state = 5)
```

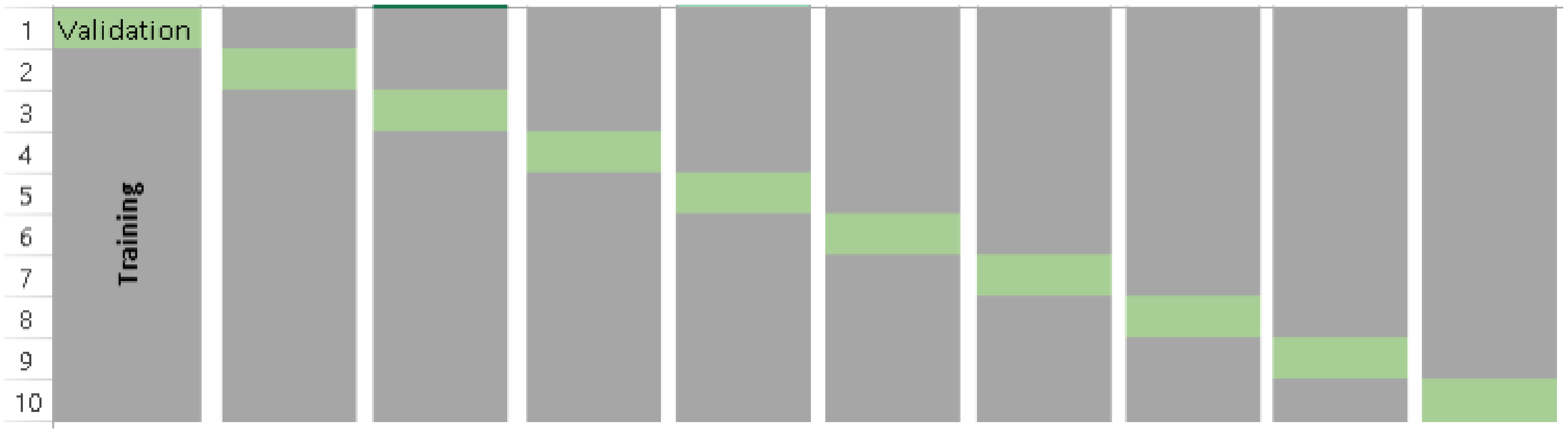
# K-FOLD Cross Validation

- Randomly split your entire dataset into k”folds”
- For each k-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
- Record the error you see on each of the predictions
- Repeat this until each of the k-folds has served as the test set
- The **average of your k recorded errors is called the cross-validation error** and will serve as your performance metric for the model

IT Bodhi
Nurturing Talent

# K-FOLD Visual

- K-fold validation when k=10
- General values 5 or 10 are used for ‘k’



## Leave one out cross validation (LOOCV)

- We make use of all data points, hence the bias will be low
- We repeat the cross validation process n times (where n is number of data points) which results in a higher execution time
- This approach leads to higher variation in testing model effectiveness because we test against one data point. So, our estimation gets highly influenced by the data point. If the data point turns out to be an outlier, it can lead to a higher variation

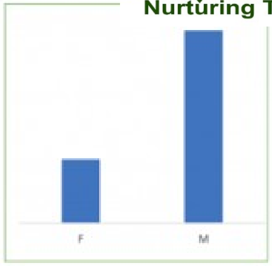
  - If there are ‘n’ rows in dataset and ‘k’ is set to ‘n’ then K-fold becomes LOOCV.

# Stratified k-fold cross validation

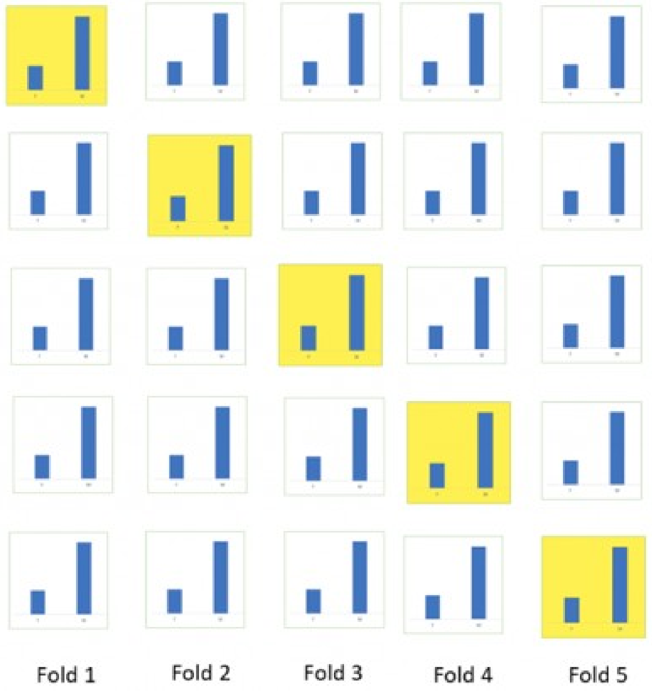
- It is generally a better approach when dealing with both bias and variance. A randomly selected fold might not adequately represent the minor class, particularly in cases where there is a huge class imbalance.
- For example, in a binary classification problem where each class comprises of 50% of the data, it is best to arrange the data such that in every fold, each class comprises of about half the instances

```
from sklearn.model_selection import StratifiedKFold

skf = StratifiedKFold(n_splits=5, random_state=None)
```



Class Distributions



## How to measure the model’s bias-variance?

After k-fold cross validation, we’ll get *k* different model estimation errors (e1, e2 …..ek). IIn an ideal scenario, these error values should sum up to zero.
To return the model’s bias, we take the average of all the errors. **Lower the average value of errors,** better the model.
Similarly for calculating the model variance, we take standard deviation of all the errors.
**A low value of standard deviation** suggests our model does not vary a lot with different subsets of training data.

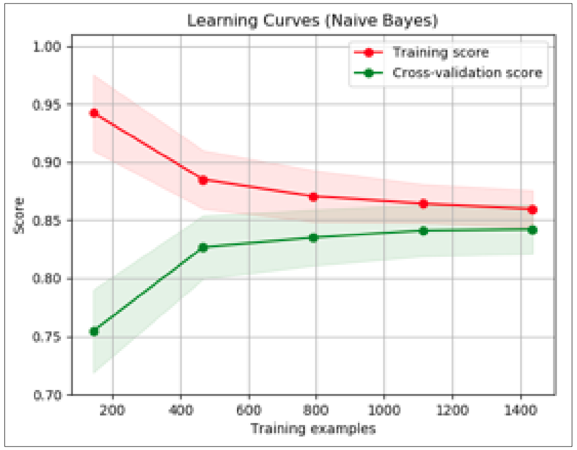
*Example : Titatnic example*

## LEARNING CURVE

- `estimator` — indicates the learning algorithm we use to estimate the true model;
- `X` — the data containing the features;
- `y` — the data containing the target;
- `train_sizes` — specifies the training set sizes to be used;
- `cv` — determines the cross-validation splitting strategy (we'll discuss this immediately);
- `scoring` — indicates the error metric to use; the intention is to use the mean squared error (MSE) metric, but that's not a possible parameter for `scoring`; we'll use the nearest proxy, negative MSE, and we'll just have to flip signs later on.

# LEARNING CURVE

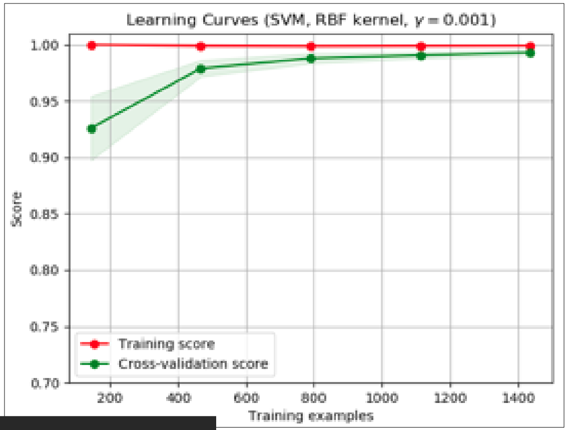
- A learning curve shows the validation and training score of an estimator for varying numbers of training samples.
- It is a tool to find out how much we benefit from adding more training data and whether the estimator suffers more from a variance error or a bias error.
- If both the validation score and the training score converge to a value that is too low with increasing size of the training set, we will not benefit much from more training data.
- In the following plot you can see an example: naive Bayes roughly converges to a low score.



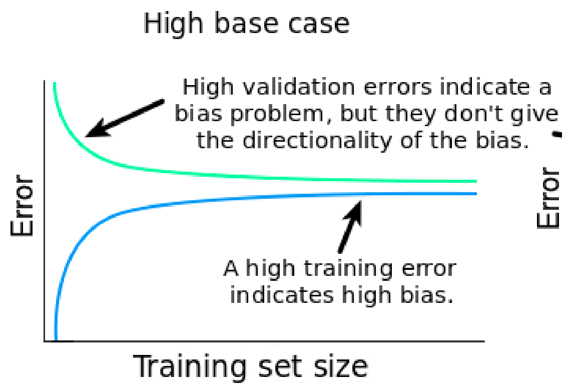
- We will probably have to use an estimator or a parametrization of the current estimator that can learn more complex concepts (i.e. has a lower bias).

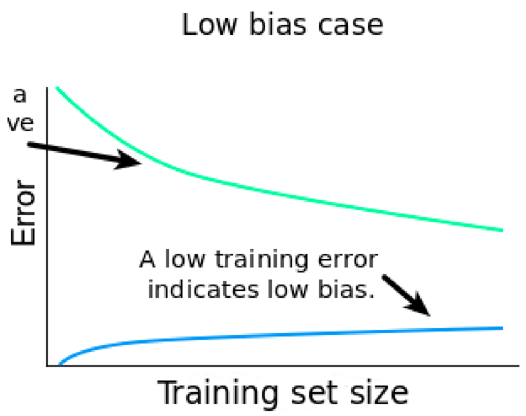
- If the training score is much greater than the validation score for the maximum number of training samples, adding more training samples will most likely increase generalization.

- In the following plot you can see that the SVM could benefit from more training examples.



# What info is here





- if the training error is very low, it means that the training data is fitted very well by the estimated model. If the model fits the training data very well, it means it has low bias with respect to that set of data.
- If the training error is high, it means that the training data is not fitted well enough by the estimated model. If the model fails to fit the training data well, it means it has high bias with respect to that set of data.

A narrow gap indicates low variance. Generally, the more narrow the gap, the lower the variance. The opposite is also true: the wider the gap, the greater the variance. Let's now explain why this is the case.

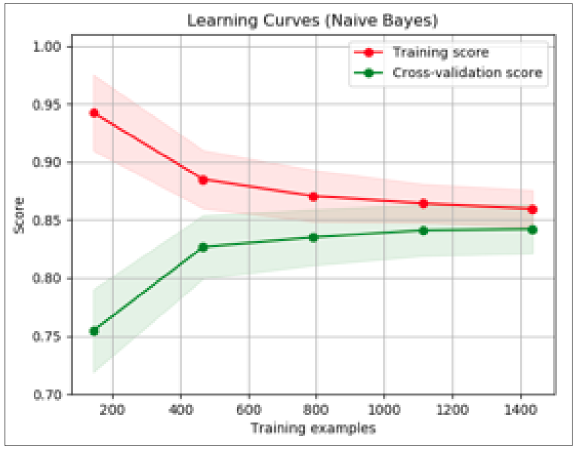
Gap = validation error - training error

**If your model has high bias, you should:**

- Try adding/creating more features
- Try complex model
- Try decreasing the regularisation parameter

**If your model has high variance, you should:**

- Get more data
- Try a smaller set of features
- Try increasing the regularisation parameter



| Scoring | Function | Comment |
|---|---|---|
| **Classification** | | |
| ‘accuracy’ | metrics.accuracy_score | |
| ‘balanced_accuracy’ | metrics.balanced_accuracy_score | for binary targets |
| ‘average_precision’ | metrics.average_precision_score | |
| ‘brier_score_loss’ | metrics.brier_score_loss | |
| ‘f1’ | metrics.f1_score | for binary targets |
| ‘f1_micro’ | metrics.f1_score | micro-averaged |
| ‘f1_macro’ | metrics.f1_score | macro-averaged |
| ‘f1_weighted’ | metrics.f1_score | weighted average |
| ‘f1_samples’ | metrics.f1_score | by multilabel sample |
| ‘neg_log_loss’ | metrics.log_loss | requires `predict_proba` support |
| ‘precision’ etc. | metrics.precision_score | suffixes apply as with ‘f1’ |
| ‘recall’ etc. | metrics.recall_score | suffixes apply as with ‘f1’ |
| ‘roc_auc’ | metrics.roc_auc_score | |

| Scoring | Function | Comment |
|---|---|---|
| **Clustering** | | |
| ‘adjusted_mutual_info_score’ | metrics.adjusted_mutual_info_score | |
| ‘adjusted_rand_score’ | metrics.adjusted_rand_score | |
| ‘completeness_score’ | metrics.completeness_score | |
| ‘fowlkes_mallows_score’ | metrics.fowlkes_mallows_score | |
| ‘homogeneity_score’ | metrics.homogeneity_score | |
| ‘mutual_info_score’ | metrics.mutual_info_score | |
| ‘normalized_mutual_info_score’ | metrics.normalized_mutual_info_score | |
| ‘v_measure_score’ | metrics.v_measure_score | |
| **Regression** | | |
| ‘explained_variance’ | metrics.explained_variance_score | |
| ‘neg_mean_absolute_error’ | metrics.mean_absolute_error | |
| ‘neg_mean_squared_error’ | metrics.mean_squared_error | |
| ‘neg_mean_squared_log_error’ | metrics.mean_squared_log_error | |
| ‘neg_median_absolute_error’ | metrics.median_absolute_error | |
| ‘r2’ | metrics.r2_score | |