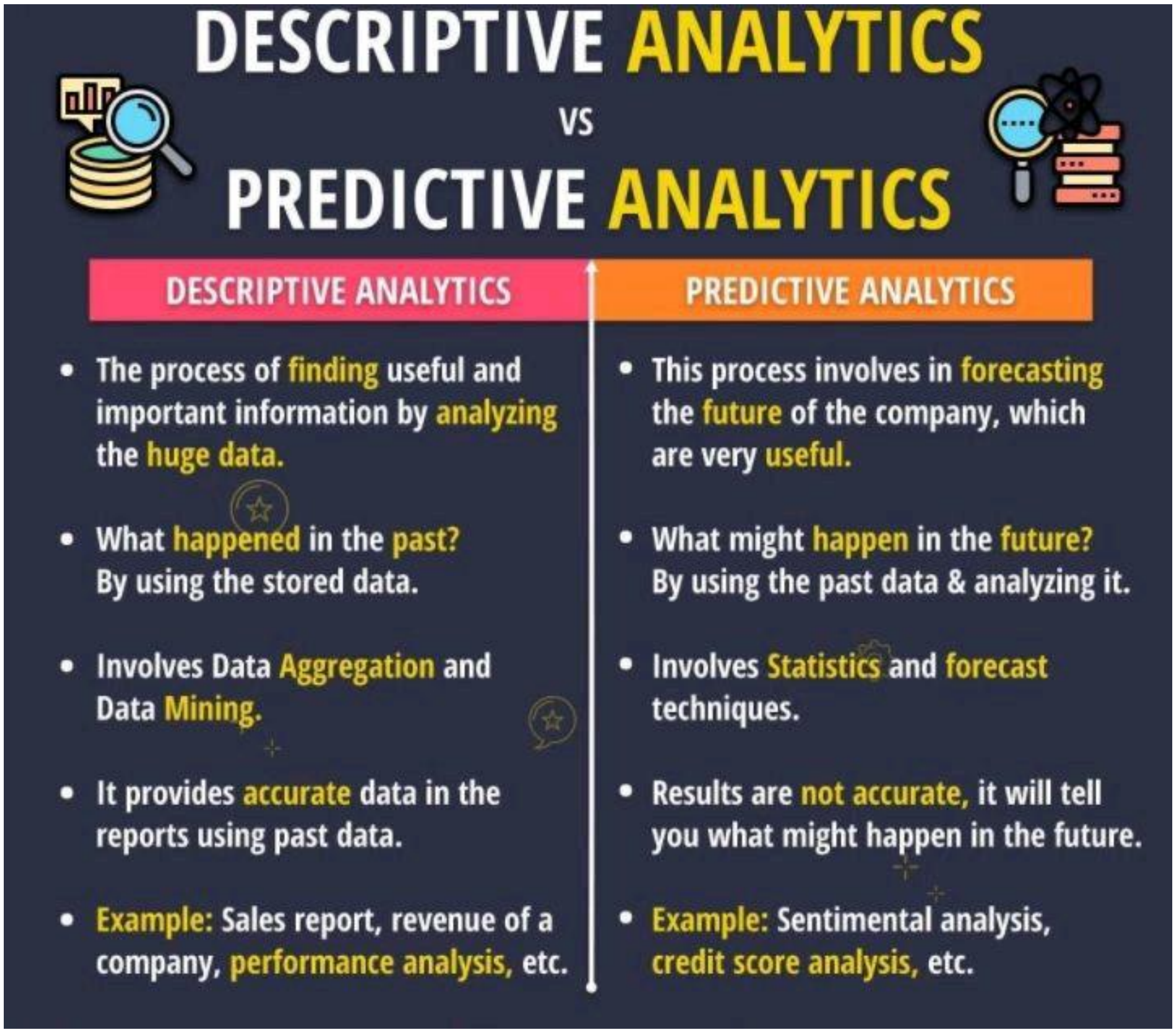
# Fundamentals of Machine Learning

# Chapter 3

## Selecting a Model

**Model selection** is the process of choosing the best ml model for a given task. It is done by comparing various model candidates on chosen evaluation metrics calculated on a designed evaluation schema. Choosing the **correct evaluation schema**, whether a simple train test split or a complex cross-validation strategy, is the **crucial first step** of building any machine learning solution.



DESCRIPTIVE ANALYTICS VS PREDICTIVE ANALYTICS

| DESCRIPTIVE ANALYTICS | PREDICTIVE ANALYTICS |
|---|---|
| The process of finding useful and important information by analyzing the huge data. | This process involves in forecasting the future of the company, which are very useful. |
| What happened in the past? By using the stored data. | What might happen in the future? By using the past data & analyzing it. |
| Involves Data Aggregation and Data Mining. | Involves Statistics and forecast techniques. |
| It provides accurate data in the reports using past data. | Results are not accurate, it will tell you what might happen in the future. |
| Example: Sales report, revenue of a company, performance analysis, etc. | Example: Sentimental analysis, credit score analysis, etc. |

**How to evaluate machine learning models and select the best one?**
We'll dive into this deeper, but let me give you a quick step-by-step:

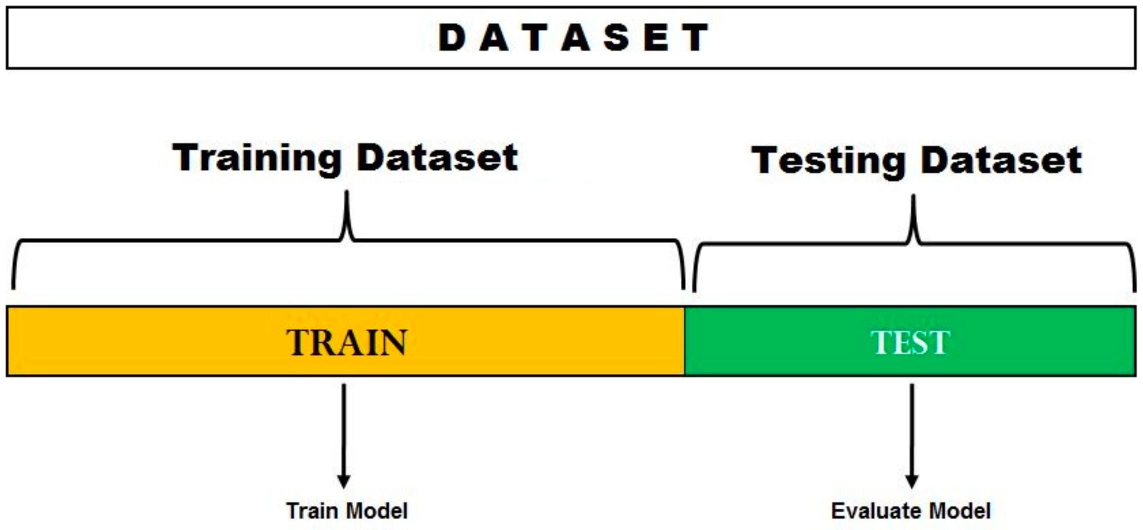
**Step 1: Choose a proper validation strategy**. Can't stress this enough, without a reliable way to validate your model performance, no amount of hyperparameter tuning and state-of-the-art models will help you.

**Step 2: Choose the right evaluation metric.** Figure out the business case behind your model and try to use the machine learning metric that correlates with that. Typically no one metric is ideal for the problem.
So calculate multiple metrics and make your decisions based on that. Sometimes you need to combine classic ML metrics with a subject matter expert evaluation. And that is ok.

**Step 3: Keep track of your experiment results**. Whether you use a spreadsheet or a dedicated experiment tracker, make sure to log all the important metrics, learning curves, dataset versions, and configurations. You will thank yourself later.

**Step 4: Compare experiments and pick a winner.** Regardless of the metrics and validation strategy you choose, at the end of the day, you want to find the best model. But no model is really best, but some are good enough.
So make sure to understand what is good enough for your problem, and once you hit that, move on to other parts of the project, like model deployment or pipeline orchestration.

**Training a Model for supervised learning:**



**Holdout Method:** is the simplest sort of method to evaluate a classifier. In this method, the data set (a collection of data items or examples) is separated into two sets, called the **Training set and Test set**.
A classifier performs function of assigning data items in a given collection to a target category or class.

**Example** –
E-mails in our inbox being classified into spam and non-spam.
Classifier should be evaluated to find out, it's accuracy, error rate, and error estimates. It can be done using various methods. One of most primitive methods in evaluation of classifier is **'Holdout Method'**.

In the holdout method, data set is partitioned, such that – maximum data belongs to training set and remaining data belongs to test set.

**Example** –
If there are 20 data items present, 12 are placed in training set and remaining 8 are placed in test set.

- After partitioning data set into two sets, training set is used to build a model/classifier.
- After construction of classifier, we use data items in test set, to test accuracy, error rate and error estimate of model/classifier.

However, it is vital to remember two statements with regard to holdout method. These are :
If maximum possible data items are placed in training set for construction of model/classifier, classifier's error rates and estimates would be very low and accuracy would be high. This is sign of a good classifier/model.

**Example** –
A student 'gfg' is coached by a teacher. Teacher teaches her all possible topics which might appear for exam. Hence, she tends to commit very less mistakes in exam, thus performing well.
If more training data are used to construct a classifier, it qualifies any data used from test set, to test it (classifier).
If more number of data items are present in test set, such that they are used to test classifier built using training set. We can observe more accurate evaluation of classifier with respect to it's accuracy, error rate and estimation.

**Example** –
A student 'gfg' is coached by a teacher. Teacher teaches her some topics, which

might appear for the exam. If the student 'gfg' is given a number of exams on basis of this coaching, an accurate determination of student's weak and strong points can be found out.
If more test data are used to evaluate constructed classifier, it's error rate, error estimate and accuracy can be accurately determined.

**<u>Problem</u> :**
During partitioning of whole data set into 2 parts i.e., training set and test set, if all data items belonging to class – GFG1, are placed in test set entirely, such that none of data items of class GFG1 are in training set. It is evident, that model/classifier built, is not trained using data items of class – GFG1.

**<u>Solution</u> :**
Stratification is a technique, using which data items belonging to class – GFG1 are divided and placed into two data sets i.e training set and test set, equally. Such that, model/classifier is trained by data items belonging to class -GFG1.

**Example –**
All the four data items belonging to class – GFG1, here, are divided equally and placed, two data items each, into two data sets – training set and test set.

**Cross-Validation in Machine Learning (k-fold method)**

Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. ***We can also say that it is a technique to check how a statistical model generalizes to an independent dataset***.

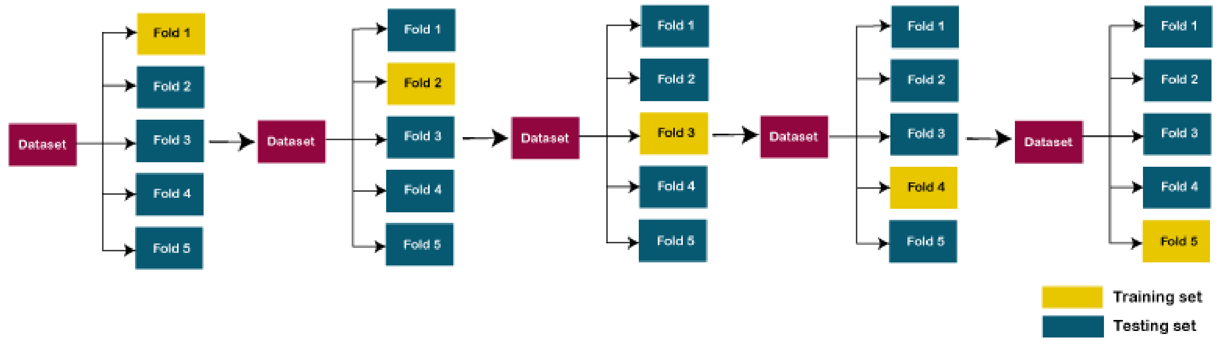
**K-Fold Cross-Validation**
K-fold cross-validation approach divides the input dataset into K groups of samples of equal sizes. These samples are called **folds**. For each learning set, the prediction function uses k-1 folds, and the rest of the folds are used for the test set. This approach is a very popular CV approach because it is easy to understand, and the output is less biased than other methods.
The steps for k-fold cross-validation are:

- Split the input dataset into K groups
- For each group:
    - Take one group as the reserve or test data set.
    - Use remaining groups as the training dataset

- Fit the model on the training set and evaluate the performance of the model using the test set.

Let's take an example of 5-folds cross-validation. So, the dataset is grouped into 5 folds. On 1st iteration, the first fold is reserved for test the model, and rest are used to train the model. On 2nd iteration, the second fold is used to test the model, and rest are used to train the model. This process will continue until each fold is not used for the test fold.

Consider the below diagram:



**Stratified k-fold cross-validation**

This technique is similar to k-fold cross-validation with some little changes. This approach works on stratification concept, it is a process of rearranging the data to ensure that each fold or group is a good representative of the complete dataset. To deal with the bias and variance, it is one of the best approaches.

It can be understood with an example of housing prices, such that the price of some houses can be much high than other houses. To tackle such situations, a stratified k-fold cross-validation technique is useful.

**Holdout Method**

This method is the simplest cross-validation technique among all. In this method, we need to remove a subset of the training data and use it to get prediction results by training it on the rest part of the dataset.

The error that occurs in this process tells how well our model will perform with the unknown dataset. Although this approach is simple to perform, it still faces the issue of high variance, and it also produces misleading results sometimes.

**Limitations of Cross-Validation**

There are some limitations of the cross-validation technique, which are given below:

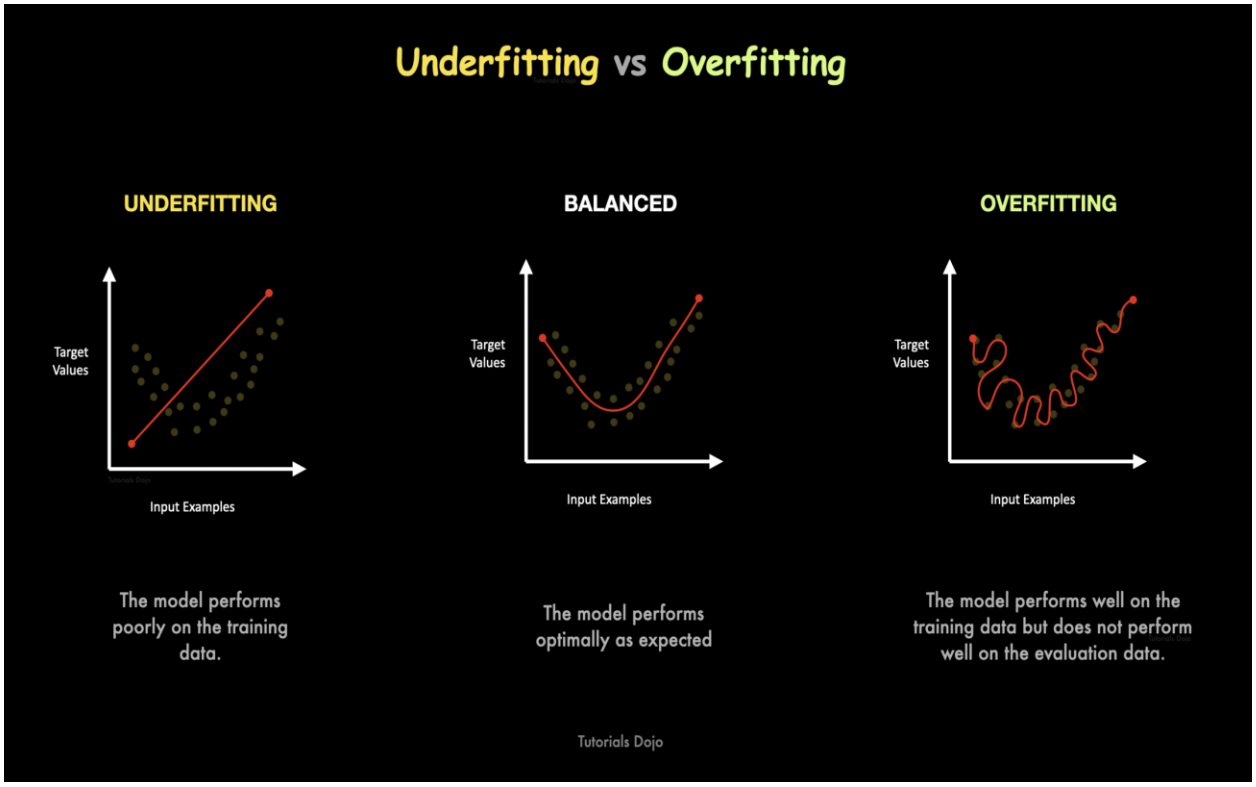
- For the ideal conditions, it provides the optimum output. But for the inconsistent data, it may produce a drastic result. So, it is one of the big disadvantages of cross-validation, as there is no certainty of the type of data in machine learning.
- In predictive modeling, the data evolves over a period, due to which, it may face the differences between the training set and validation sets. Such as if we create a model for the prediction of stock market values, and the data is trained on the previous 5 years stock values, but the realistic future values for the next 5 years may drastically different, so it is difficult to expect the correct output for such situations.

**Applications of Cross-Validation**

- This technique can be used to compare the performance of different predictive modeling methods.
- It has great scope in the medical research field.
- It can also be used for the meta-analysis, as it is already being used by the data scientists in the field of medical statistics.

**Model representation and interpretability**

- The main goal of each machine learning model is **to generalize well**.
- Here **generalization** defines the ability of an ML model to provide a suitable output by adapting the given set of unknown input.
- It means after providing training on the dataset, it can produce reliable and accurate output.
- Hence, the underfitting and overfitting are the two terms that need to be checked for the performance of the model and whether the model is generalizing well or not.


Underfitting vs Overfitting
UNDERFITTING
BALANCED
OVERFITTING
Target Values
Input Examples
The model performs poorly on the training data.
The model performs optimally as expected
The model performs well on the training data but does not perform well on the evaluation data.
Tutorials Dojo

**Underfitting**
A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data, i.e., it only performs well on training data but performs poorly on testing data.

**Reasons for Underfitting:**
- High bias and low variance
- The size of the training dataset used is not enough.
- The model is too simple.
- Training data is not cleaned and also contains noise in it.
-

**Techniques to reduce underfitting:**
- Increase model complexity
- Increase the number of features, performing feature engineering
- Remove noise from the data.
- Increase the number of epochs or increase the duration of training to get better results.
-

**Overfitting:**

A statistical model is said to be overfitted when the model does not make accurate predictions on testing data. When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set.

**Reasons for Overfitting are as follows:**

- High variance and low bias
- The model is too complex
- The size of the training data
-

**Techniques to reduce overfitting:**

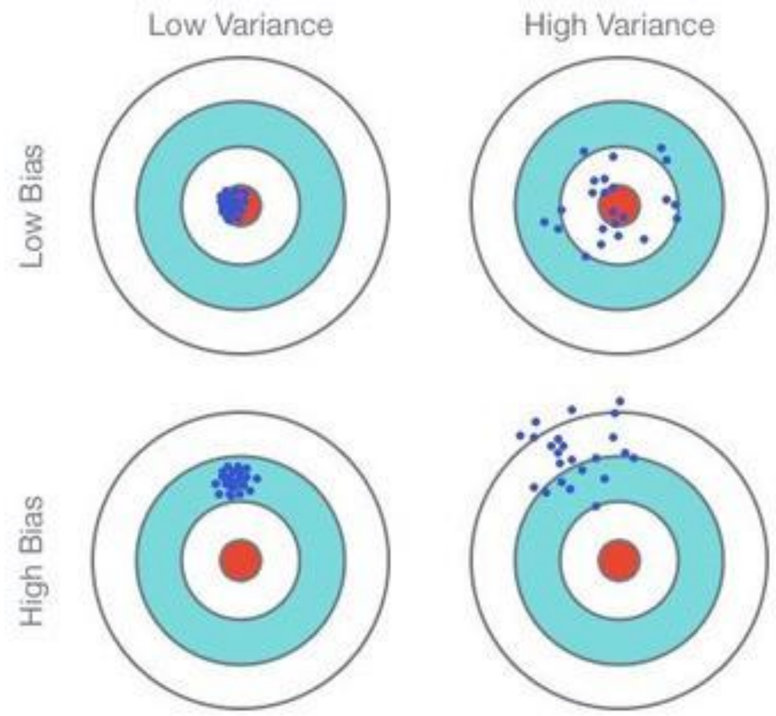
- Increase training data.
- Reduce model complexity.
- Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
- Ridge Regularization and Lasso Regularization

Use dropout for neural networks to tackle overfitting.

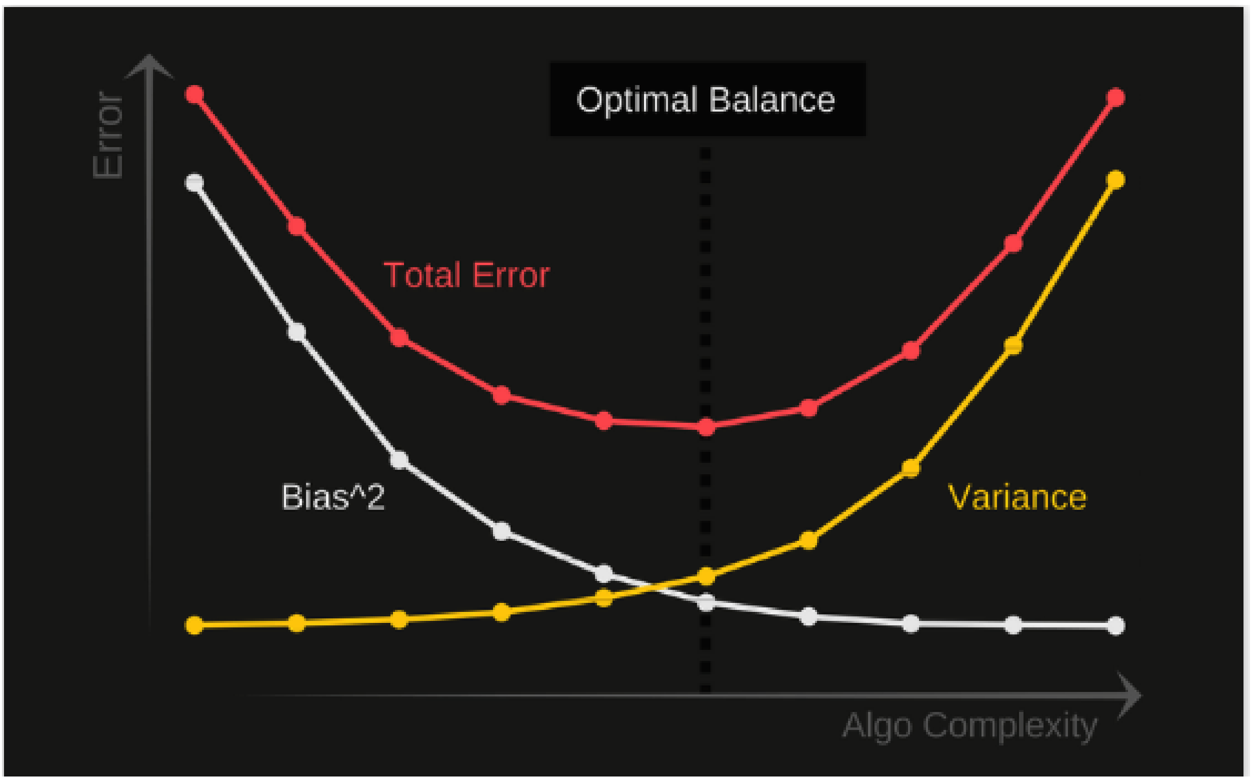
**Good Fit in a Statistical Model:**
Ideally, the case when the model makes the predictions with 0 error, is said to have a good fit on the data. This situation is achievable at a spot between overfitting and underfitting. In order to understand it, we will have to look at the performance of our model with the passage of time, while it is learning from the training dataset.

- **Bias**: Assumptions made by a model to make a function easier to learn. It is actually the error rate of the training data. When the error rate has a high value, we call it High Bias and when the error rate has a low value, we call it low Bias.
- **Variance**: The difference between the error rate of training data and testing data is called variance. If the difference is high then it’s called high variance and when the difference of errors is low then it’s called low variance. Usually, we want to make a low variance for generalized our model.

**Bias Variance Trade-off**



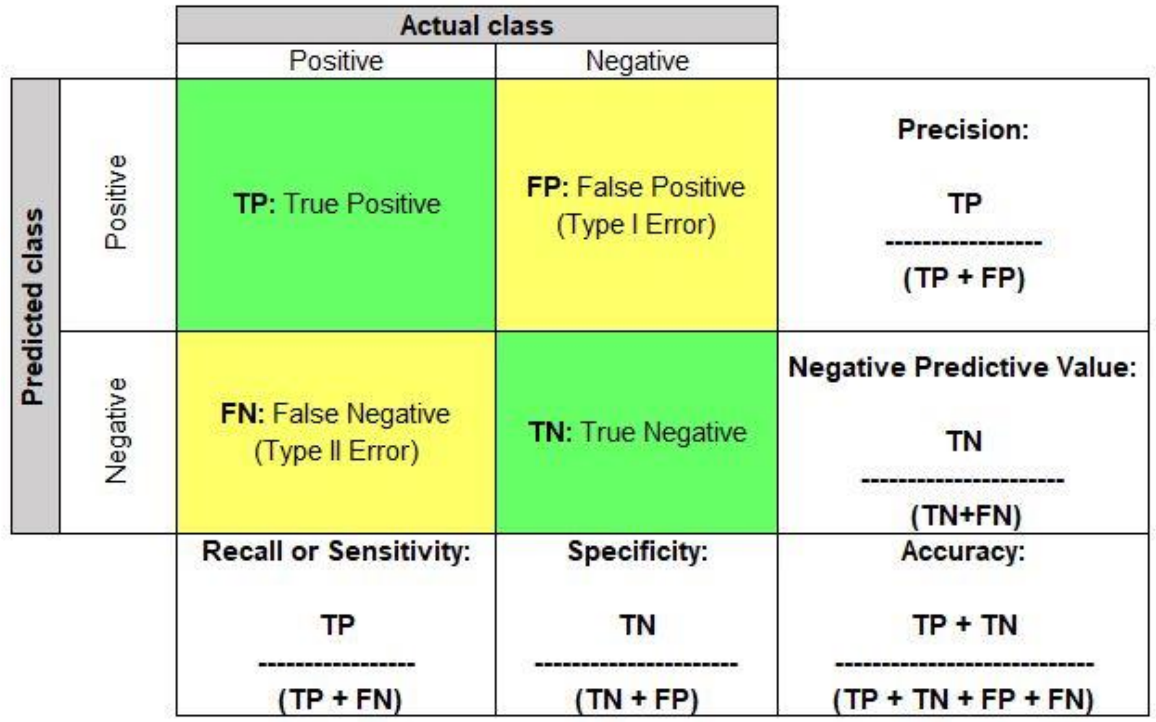
## Bias Variance Trade-off



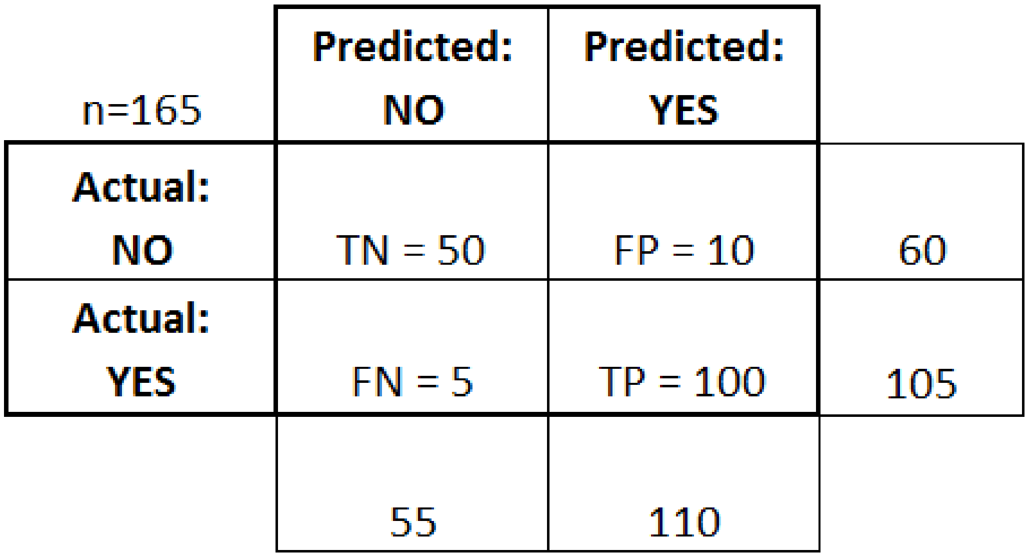
## Evaluating performance of a model

## What is a Confusion Matrix?

- A confusion matrix, also known as an **error matrix**, is a **summarized table** used to assess the performance of a classification model. The number of correct and incorrect predictions are summarized with count values and broken down by each class.
- For Example, let's say that there were ten instances where a classification model predicted 'Yes' in which the actual value was 'Yes'. Then the number ten would go in the top left corner in the True Positive quadrant. This leads us to some key terms:
- **Positive (P)**: Observation is positive (eg. **is** a dog).
- **Negative (N)**: Observation is not positive (eg. **is not** a dog).
- **True Positive (TP)**: Outcome where the model correctly predicts the positive class.
- **True Negative (TN)**: Outcome where the model correctly predicts the negative class.
- **False Positive (FP)**: Also called a **type 1 error**, an outcome where the model incorrectly predicts the positive class when it is actually negative.
- **False Negative (FN)**: Also called a **type 2 error**, an outcome where the model incorrectly predicts the negative class when it is actually positive



| | | Actual class | | |
|---|---|---|---|---|
| | | Positive | Negative | |
| Predicted class | Positive | **TP:** True Positive | **FP:** False Positive (Type I Error) | **Precision:** $\frac{TP}{(TP + FP)}$ |
| | Negative | **FN:** False Negative (Type II Error) | **TN:** True Negative | **Negative Predictive Value:** $\frac{TN}{(TN+FN)}$ |
| | | **Recall or Sensitivity:** $\frac{TP}{(TP + FN)}$ | **Specificity:** $\frac{TN}{(TN + FP)}$ | **Accuracy:** $\frac{TP + TN}{(TP + TN + FP + FN)}$ |



| n=165 | Predicted: NO | Predicted: YES | |
|---|---|---|---|
| Actual: NO | TN = 50 | FP = 10 | 60 |
| Actual: YES | FN = 5 | TP = 100 | 105 |
| | 55 | 110 | |

- **Accuracy:**

Overall, how often is the classifier correct?
(TP+TN)/total = (100+50)/165 = 0.91
**Misclassification Rate:**
Overall, how often is it wrong?
(FP+FN)/total = (10+5)/165 = 0.09
equivalent to 1 minus Accuracy
also known as "Error Rate"

- **True Positive Rate:**

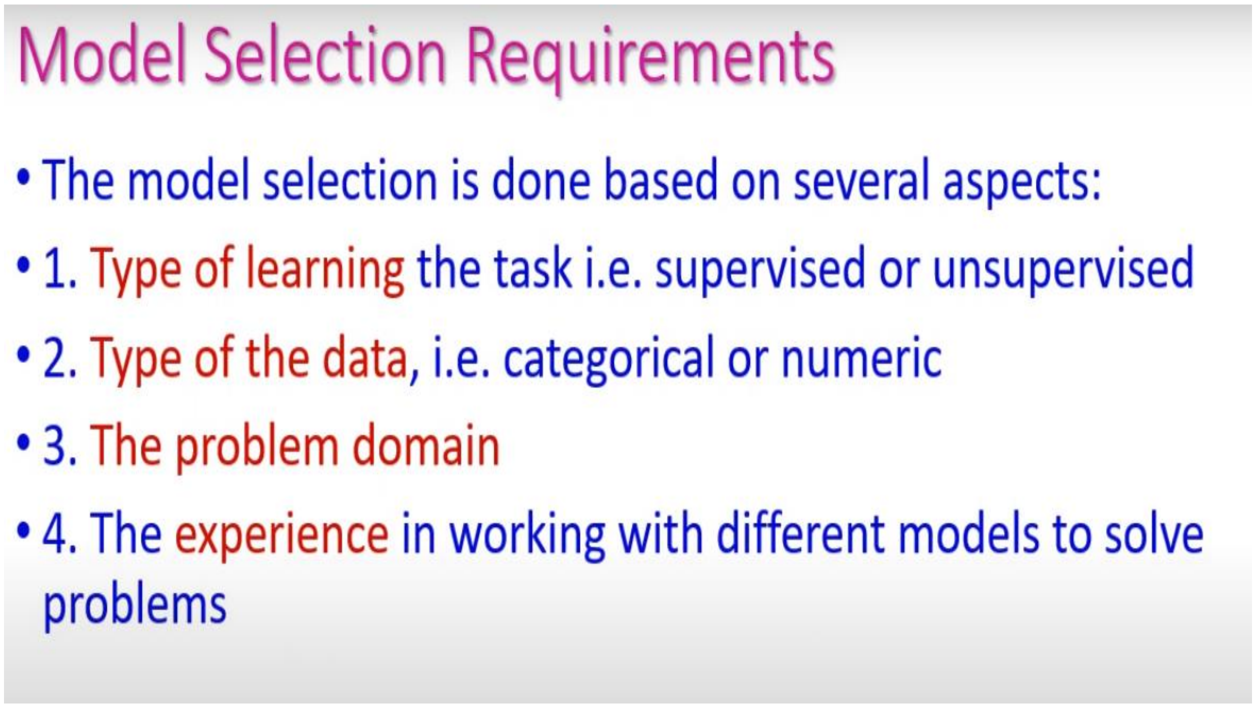
When it's actually yes, how often does it predict yes?
TP/actual yes = 100/105 = 0.95
also known as "Sensitivity" or "Recall“

  - **Precision**:

When it predicts yes, how often is it correct?
TP/predicted yes = 100/110 = 0.91

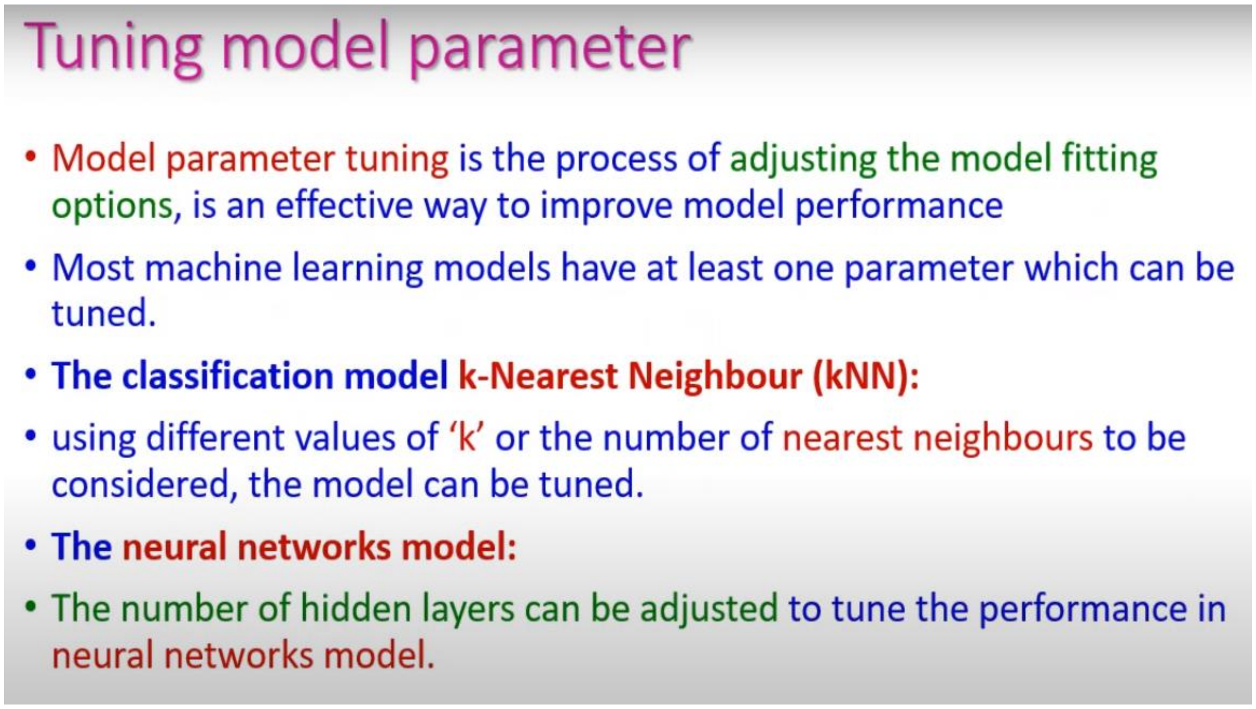
**Improving Performance of a model**

- Model Selection Requirement
- Tuning Model Parameter
  - kNN
- Combining several models
  - Ensemble
  - Bootstrap Aggregating/ Bagging
  - Adaptive boosting or AdaBoost



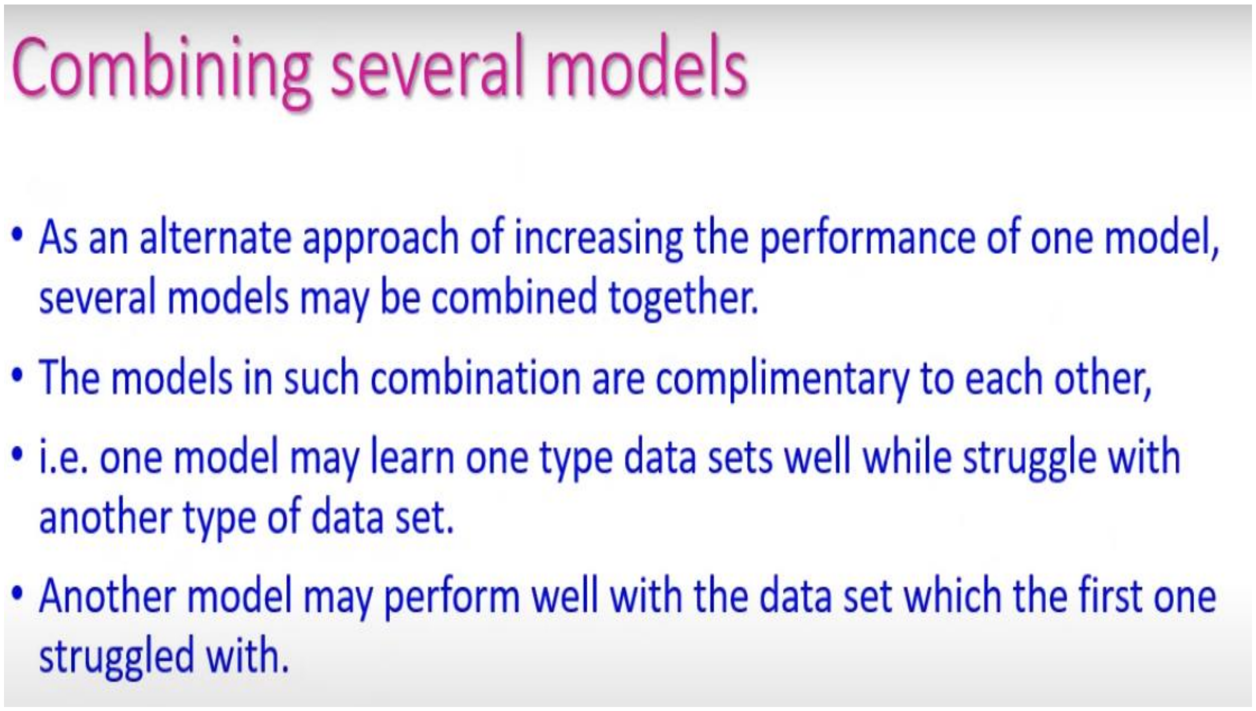
## Model Selection Requirements

- The model selection is done based on several aspects:
- 1. Type of learning the task i.e. supervised or unsupervised
- 2. Type of the data, i.e. categorical or numeric
- 3. The problem domain
- 4. The experience in working with different models to solve problems



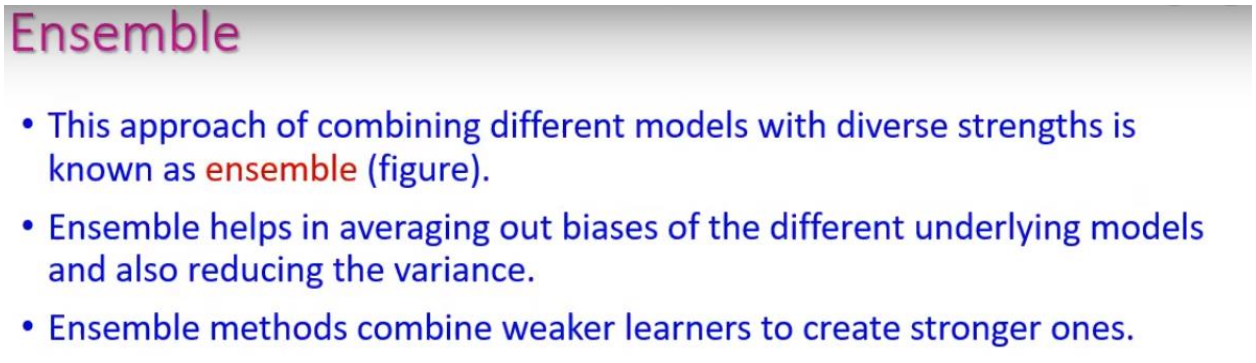
## Tuning model parameter

- Model parameter tuning is the process of adjusting the model fitting options, is an effective way to improve model performance
- Most machine learning models have at least one parameter which can be tuned.
- **The classification model k-Nearest Neighbour (kNN):**
- using different values of 'k' or the number of nearest neighbours to be considered, the model can be tuned.
- **The neural networks model:**
- The number of hidden layers can be adjusted to tune the performance in neural networks model.



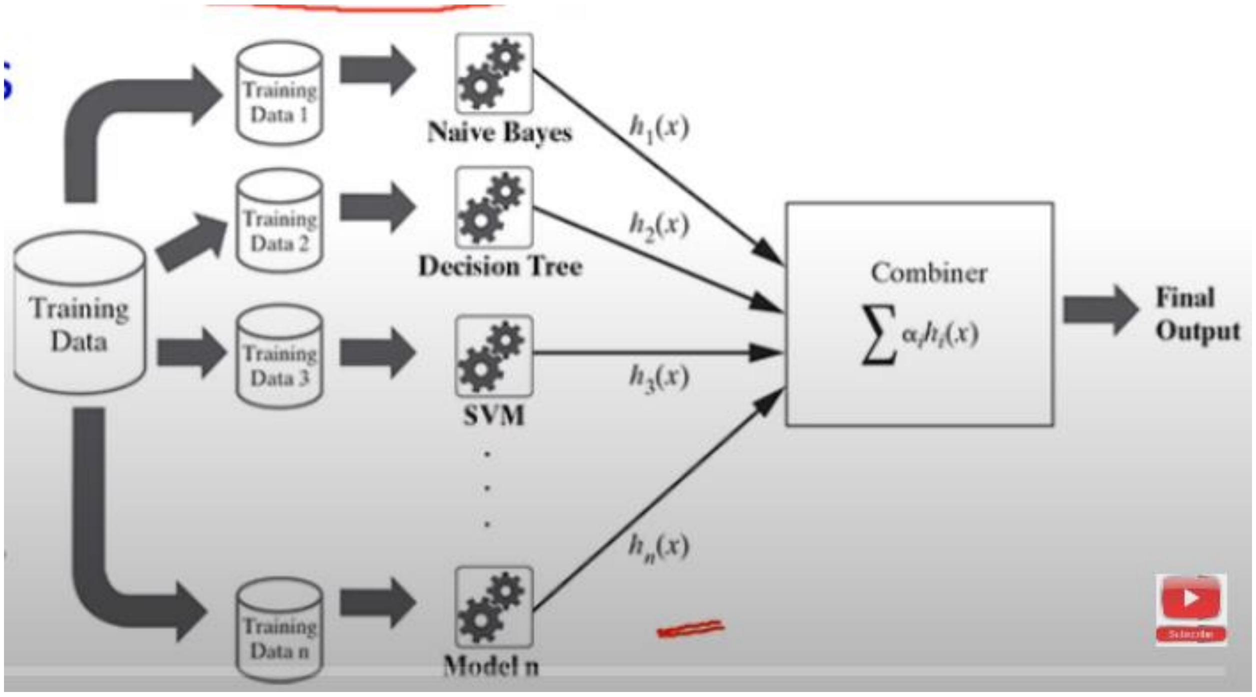
## Combining several models

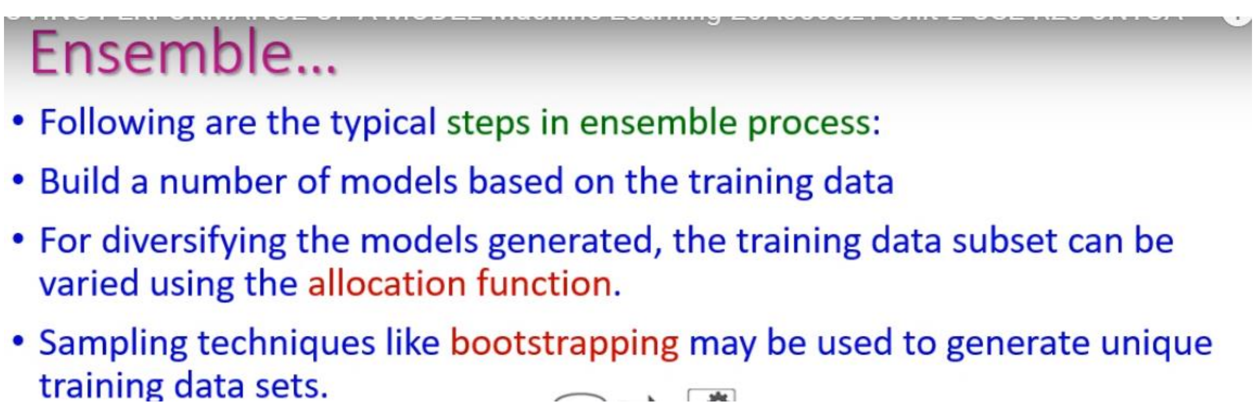
- As an alternate approach of increasing the performance of one model, several models may be combined together.
- The models in such combination are complimentary to each other,
- i.e. one model may learn one type data sets well while struggle with another type of data set.
- Another model may perform well with the data set which the first one struggled with.



## Ensemble

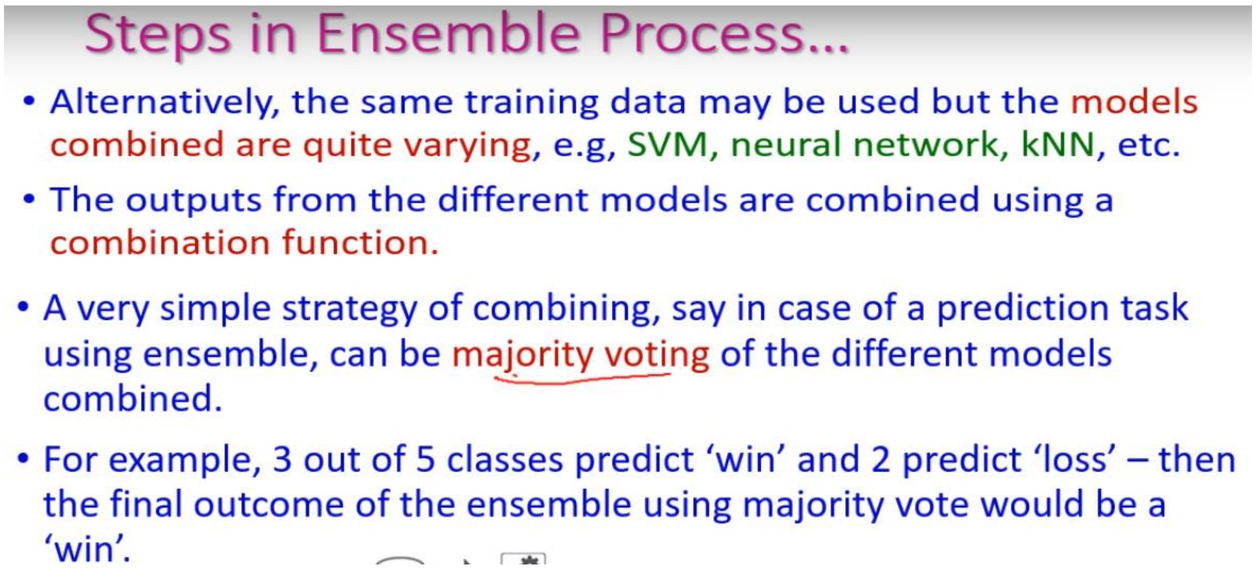
- This approach of combining different models with diverse strengths is known as ensemble (figure).
- Ensemble helps in averaging out biases of the different underlying models and also reducing the variance.
- Ensemble methods combine weaker learners to create stronger ones.





## Ensemble…

- Following are the typical steps in ensemble process:
- Build a number of models based on the training data
- For diversifying the models generated, the training data subset can be varied using the allocation function.
- Sampling techniques like bootstrapping may be used to generate unique training data sets.



## Steps in Ensemble Process…

- Alternatively, the same training data may be used but the models combined are quite varying, e.g, SVM, neural network, kNN, etc.
- The outputs from the different models are combined using a combination function.
- A very simple strategy of combining, say in case of a prediction task using ensemble, can be majority voting of the different models combined.
- For example, 3 out of 5 classes predict 'win' and 2 predict 'loss' – then the final outcome of the ensemble using majority vote would be a 'win'.

## Bootstrap Aggregating or Bagging

- One of the earliest and most popular ensemble models is bootstrap aggregating or bagging.
- Bagging uses bootstrap sampling method to generate multiple training data sets.
- These training data sets are used to generate (or train) a set of models using the same learning algorithm.
- Then the outcomes of the models are combined by majority voting (classification) or by average (regression).
- Bagging suitable for unstable learners like a decision tree, in which a slight change in data can impact the outcome of a model significantly.

## Adaptive boosting or AdaBoost

- Just like bagging, boosting is another key ensemble based technique.
- **The weaker learning models are trained on resampled data and the outcomes are combined using a weighted voting approach based on the performance of different models.**
- Adaptive boosting or AdaBoost is a special variant of boosting algorithm.
- It is based on the idea of generating weak learners and slowly learning.
- Random forest is another ensemble-based technique. It is an ensemble of decision trees – hence the name random forest to indicate a forest of decision trees.