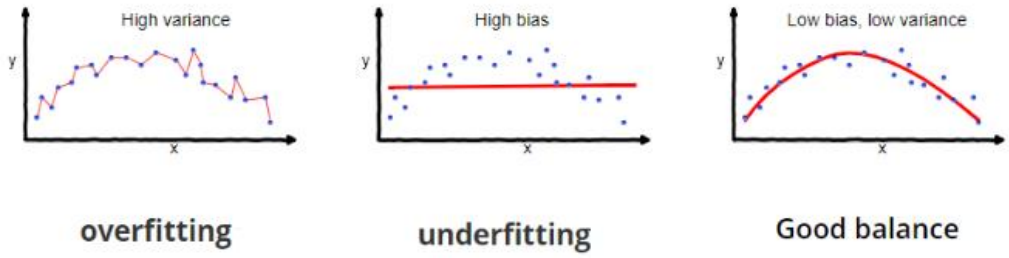
Bias Variance Tradeoff and Performance Evaluation and Improvement of an algorithm

Bias: Is the error due to wrong assumption implemented within the learning algorithm. In context of Machine learning signifies that the model is underfitting the training data. This occurs when the model does not adequately utilizes the inherent patterns discernable from the features to the predict the target class.

Variance: Is the error due to small fluctuations in the training data. In context to machine learning signifies that the model is overfitting the data. This occurs when the model is not only representing the pattern in the training data but also incorporates the noise in the data which is irrelevant.

Due to underfitting or overfitting the data generalization beyond the training set becomes a challenge.



Bias Variance Decomposition

Total Error =Reducible+ Irreducible(Noise)

Total Error $= \boldsymbol{bias^2 + Variance}$**+ Var(ϵ)**

MSE = Reducible

MSE Mean Squared error

D = random data

MSE of an estimator $\hat{\theta} = f(D)$ for the parameter θ is given by the formula

$MSE = E(\hat{\theta} - \theta)^2$ 1

$Bias = E(\hat{\theta}) - \theta$ and Let $\mu = E(\hat{\theta})$ μ is constant and θ is a constant.

Equation 1 can be written as

$MSE = E[(\hat{\theta} - \mu) + (\mu - \theta)]^2 = E[(\hat{\theta} - \mu)^2 + (\mu - \theta)^2 - 2(\hat{\theta} - \mu)(\mu - \theta)]$

$MSE = E[(\hat{\theta} - \mu)^2] + E[(\mu - \theta)^2] - 2E(\hat{\theta} - \mu)(\mu - \theta)]$

μ is constant therefore $E(\hat{\theta} - \mu) = [E(\hat{\theta}) - \mu] = 0$

$MSE = E[(\hat{\theta} - \mu)^2] + E[(\mu - \theta)^2]$

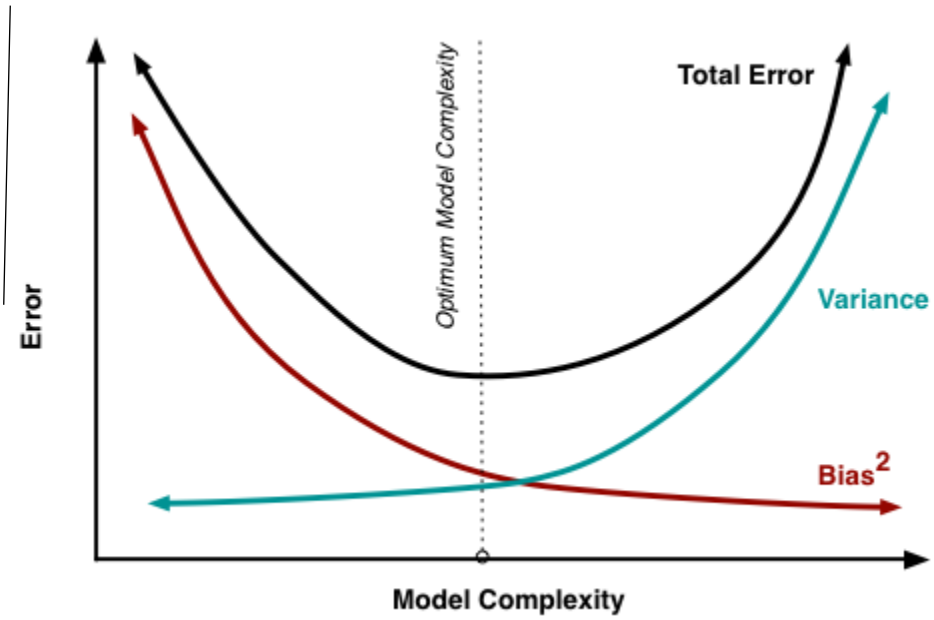
μ is constant and θ is a constant.

$E[(\mu - \theta)^2 = (\mu - \theta)^2$

$MSE = E[(\hat{\theta} - \mu)^2] + (\mu - \theta)^2$

$MSE = E[(\hat{\theta} - E(\hat{\theta}))^2] + (E(\hat{\theta}) - \theta)^2$

MSE = Variance + Bias^2 + **Var(ϵ)**



Complexity = Number of parameters, the model itself like linear equation versus polynomial

Red and blue in the training data black testing data(generalisation). Initially bias is high due to underfitting then it decreases once more details are incorporated in the algorithm. Once the model starts incorporating too much details the overfitting causes the variance to increase for the testing data . So total error can be minimized by the balancing bias variance tradeoff

The left sided of the dotted vertical line is representing **underfitting**

The right sided of the dotted vertical line is representing **overfitting**

The optimal complexity is obtained at the level where the test generalization achieves the minimum error. Model will always exhibit an inherent Irreducible error.

Cross Validation for Parameter Tuning

The testing and training set should be optimized for better prediction. To deal with this tradeoff, k fold cross validation can be implemented. The training data (usually 75% of the data) is divided into k parts, for instance if there are 200 examples, then these could potentially be divided into 10 parts with 20 examples each. The testing set usually comprises 25% of the data.

One of the parts is used as a validation set and the other nine parts are used as training sets. This is performed k=10 times by replacing the validation set every time. This way the whole data has gone through the training therefore all the inherent patterns have been explored.

Then we average the ten different testing performances. It is computationally intensive but the assessment of the algorithm is better. Cross-validation can be used to estimate the test error associated with a learning method in order to evaluate its performance, or to select the appropriate level of flexibility.

To improve model performance the sample size can be increased or some hyper parameters can be tuned but once the best optimal level has been reached a newer algorithm has to be explored.

Model Performance Evaluation

Confusion Matrix

		Predicted		
		No	**Yes**	
Actual	**No**	**True Negative(TN)**	**False Positive(FP)**	
	Yes	**False Negative(FN)**	**True Positive(TP)**	

$$Prediction\ Accuracy\ = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Error\ Rate\ = \frac{FP + FN}{TP + TN + FP + FN} = 1 - Prediction\ Accuracy$$

SENSTIVITY (True Positive rate) Measures the proportion of positive example that were correctly classified.

$$Senstivity\ = \frac{TP}{TP + FN}$$

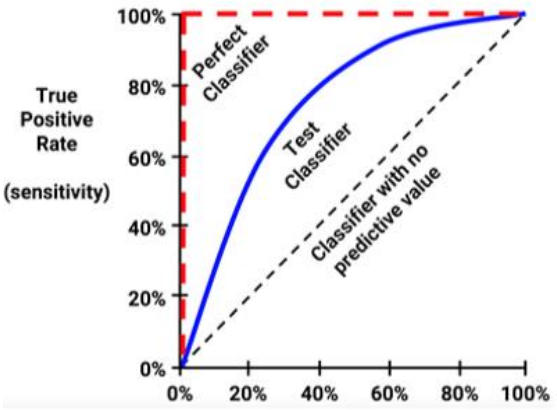
SPECIFICITY (True Negative Rate) Measures the proportion of negative examples that were correctly classified.

$$Specificity = \frac{TN}{TN + FP}$$

Precision PPV Positive Predictive Value: Proportion of positive examples that are truly positive.

$$Specificity = \frac{TP}{TP + FP}$$

Visualization of Performances ROC: Receiver Operating Characteristics curve helps examine tradeoff between detection of true positives (sensitivity on y axis) while avoiding false positives (1-specificity on y axis).



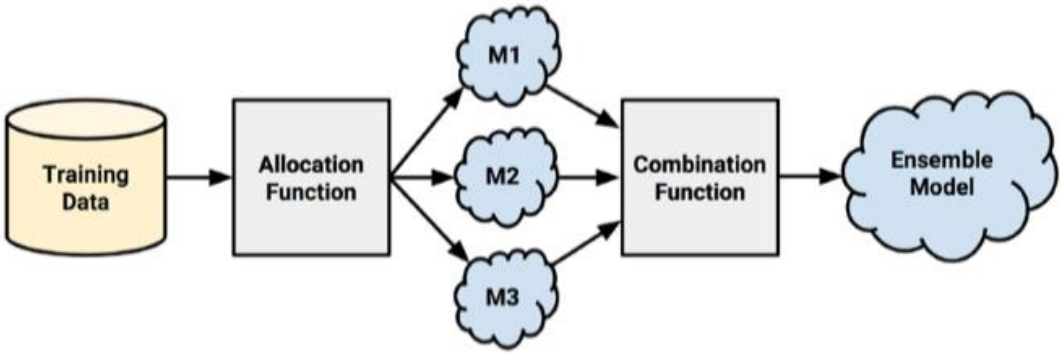
False Negative Rate(1-Specificity)

The red vertical line depicting the 100% accuracy. Area under ROC (AUC) ranges from .5 to 1 (Perfect classifier). This helps use determine how well the classes have achieved seperatibility.

Bootstrap Creating many samples from the initial sample whose size is usually small. This helps boost performance.

Performance Improvement by Meta Learning

If the performance of the algorithms is increased by combining various algorithm then this methodology is called Meta Learning or Ensembles, By using a number of weak learners in conjunction with each other causes the creation of a strong learner. Allocation functions assigns the data to each algorithm. Bagging , Boosting and Random Forest are important algorithms.



Ensembles are advantageous since they generalize well to new cases, no one bias can dominate the predicion therefore it offsets overfitting. Ensemble enables the usage of multiple learners which use data from different domain areas which enables better prediction. Real life scenarios are very complex embedded with intracies and facets consequently Ensemble is a good methodolgy for prediction.