**Measuring Bias and Variance of Machine Learning Algorithms**

**Submitted by:**

**Aman Kumar**

**Hitesh H S**

**Batch 52 – INSOFE, Bengaluru**

**Guide: Dr. Soumen Manna**



**International School of Engineering, Bengaluru**

**Contents**

**Abstract----------------------------------------------------------------------------------------------------4**

**1. Introduction--------------------------------------------------------------------------------------------4**

**1.1 Bias--------------------------------------------------------------------------------------------4**

**1.2 Variance--------------------------------------------------------------------------------------4**

**1.3 Graphical Definition-----------------------------------------------------------------------6**

**1.4 Bias-Variance Tradeoff-------------------------------------------------------------------7**

**1.5 Total Error----------------------------------------------------------------------------------7**

**1.6 Assumptions---------------------------------------------------------------------------------8**

**1.7 Measuring Bias and Variance for Regression----------------------------------------8**

**1.8 Measuring Bias and Variance for Classification-----------------------------------11**

**1.9 Decomposition-----------------------------------------------------------------------------12**

**2. Models Built------------------------------------------------------------------------------------------16**

**2.1 Linear Regression------------------------------------------------------------------------16**

**2.2 Multinomial Logistic Regression------------------------------------------------------17**

**2.3 Decision Tree------------------------------------------------------------------------------18**

**2.4 Random Forest----------------------------------------------------------------------------20**

**2.5 Support Vector Machine----------------------------------------------------------------20**

**2.6 K-Nearest Neighbors---------------------------------------------------------------------21**

**3. Analysis and Evaluation---------------------------------------------------------------------------23**

**3.1 Regression----------------------------------------------------------------------------------23**

**3.2 Problem Statement-----------------------------------------------------------------------23**

**3.3 Data Description--------------------------------------------------------------------------23**

**3.4 Exploratory Data Analysis--------------------------------------------------------------24**

**3.5 Data Cleaning-----------------------------------------------------------------------------24**

**3.6 Bootstrap Sampling----------------------------------------------------------------------24**

**3.7 Correlation Plot---------------------------------------------------------------------------25**

**3.8 Model Evaluation-------------------------------------------------------------------------26**

**3.9 Measuring Bias Square------------------------------------------------------------------26**

**3.10 Evaluation--------------------------------------------------------------------------------26**

**3.11 Measuring Variance--------------------------------------------------------------------27**

**3.12 Evaluation--------------------------------------------------------------------------------27**

**3.13 Mean Squared Error-------------------------------------------------------------------28**

**3.14 Classification-----------------------------------------------------------------------------30**

**3.15 Problem Statement----------------------------------------------------------------------30**

**3.16 Data Description-------------------------------------------------------------------------30**

**3.17 Exploratory Data Analysis------------------------------------------------------------30**

**3.18 Data Cleaning----------------------------------------------------------------------------30**

**3.19 Handling Missing Data-----------------------------------------------------------------31**

**3.21 Feature Selection------------------------------------------------------------------------31**

**3.22 Data Splitting----------------------------------------------------------------------------31**

**3.23 Model Evaluation-----------------------------------------------------------------------32**

**3.24 Measuring Bias Square----------------------------------------------------------------32**

**3.25 Measuring Variance--------------------------------------------------------------------33**

**3.26 Findings-----------------------------------------------------------------------------------33**

**4. Future Analysis--------------------------------------------------------------------------------------34**

**5. References--------------------------------------------------------------------------------------------35**

**Abstract**

The purpose of the internship is to measure Bias and Variance for various regression and classification models and find out which models are Bias sensitive and Variance sensitive. For regression, we are using Bootstrap sampling and creating 100 different samples from the same data.

1. **Introduction**
   1. **Bias**

Bias are the simplifying assumptions made by a model to make the target function easier to learn.

Generally, parametric algorithms have a high bias making them fast to learn and easier to understand but generally less flexible. In turn, they have lower predictive performance on complex problems that fail to meet the simplifying assumptions of the algorithm’s bias.

* Low Bias: Suggests less assumptions about the form of the target function.
* High-Bias: Suggests more assumptions about the form of the target function.
* Examples of low-bias machine learning algorithms include: Decision Trees, k-Nearest Neighbors and [Support Vector Machines](https://machinelearningmastery.com/support-vector-machines-for-machine-learning/).
* Examples of high-bias machine learning algorithms include: Linear Regression, Linear Discriminant Analysis and Logistic Regression.
  1. **Variance**

Variance is the amount that the estimate of the target function will change if different training data is used.

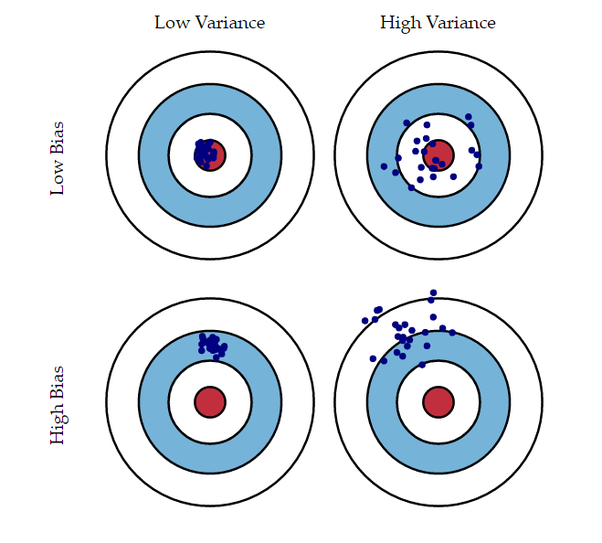
The target function is estimated from the training data by a machine learning algorithm, so we should expect the algorithm to have some variance. Ideally, it should not change too much from one training dataset to the next, meaning that the algorithm is good at picking out the hidden underlying mapping between the inputs and the output variables.

Machine learning algorithms that have a high variance are strongly influenced by the specifics of the training data. This means that the specifics of the training have influences on the number and types of parameters used to characterize the mapping function.

* Low Variance: Suggests small changes to the estimate of the target function with changes to the training dataset.
* High Variance: Suggests large changes to the estimate of the target function with changes to the training dataset.

Generally, nonparametric machine learning algorithms that have a lot of flexibility have a high variance. For example, decision trees have a high variance, that is even higher if the trees are not pruned before use.

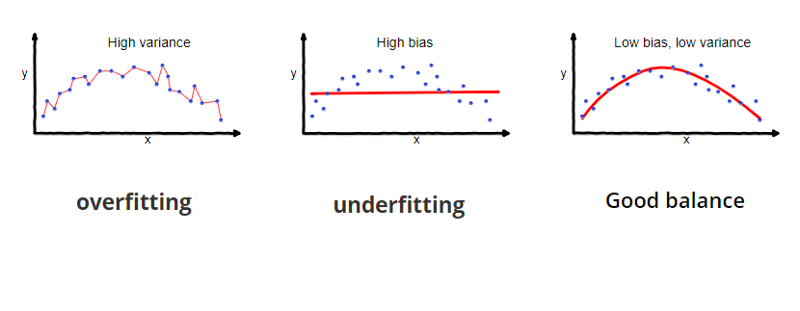
* Examples of low-variance machine learning algorithms include: Linear Regression, Linear Discriminant Analysis and Logistic Regression.
* Examples of high-variance machine learning algorithms include: Decision Trees, k-Nearest Neighbors and Support Vector Machines.
  1. **Graphical Definition**



In the above diagram, center of the target is a model that perfectly predicts correct values. As we move away from the bulls-eye our predictions become get worse and worse. We can repeat our process of model building to get separate hits on the target.

In supervised learning, **underfitting** happens when a model unable to capture the underlying pattern of the data. These models usually have high bias and low variance. It happens when we have very less amount of data to build an accurate model or when we try to build a linear model with a nonlinear data. Also, this kind of models are very simple to capture the complex patterns in data like Linear and logistic regression.

In supervised learning, **overfitting** happens when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over noisy dataset. These models have low bias and high variance. These models are very complex like Decision trees which are prone to overfitting.



* 1. **Bias Variance Tradeoff**

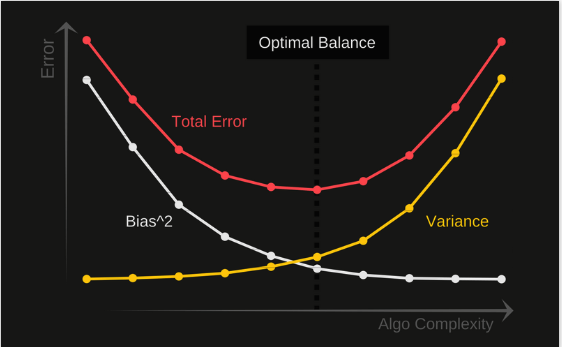
If our model is too simple and has very few parameters, then it may have high bias and low variance. On the other hand, if our model has large number of parameters then it’s going to have high variance and low bias. So, we need to find the right/good balance without overfitting and underfitting the data.

This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can’t be more complex and less complex at the same time.

* 1. **Total Error**

To build a good model, we need to find a good balance between bias and variance such that it minimizes the total error.

**Total Error = Bias^2 + Variance + Irreducible Error**



An optimal balance of bias and variance would never overfit or underfit the model.

Therefore, understanding bias and variance is critical for understanding the behavior of prediction models.

* 1. **Assumptions**

Bias and variance are statistical terms and can be used in varied contexts. However, in this article, they will be discussed in terms of an estimator which is trying to fit/explain/estimate some unknown data distribution.

Before we delve into the bias and variance of an estimator, let us assume the following: -

1. There is a data generator, Y = f(X) + ϵ, which is generating Data(X,Y), where ϵ is the added random gaussian noise, centered at origin with some standard deviation σ i.e. E[ϵ] = 0 and Var(ϵ) = σ² . Note that data can be sampled repetitively from the generator yielding different sample sets say Xᵢ, Yᵢ on iᵗʰ iteration.
2. We are trying to estimate (fit a curve) to the sample set we have available from the generator, using an estimator. An estimator usually is a class of models like Ridge regressor, Decision Tree or Support Vector Regressor etc. A class of models can be represented as g(X/θ) where θ are the parameters. For different values of θ, we get different models within that particular class of models and we try vary θ to find the best fitting model for our sample set.
   1. **Measuring Bias and Variance for Regression**

**Bias** of an estimator is the “expected” difference between its estimates and the true values in the data. Intuitively, it is a measure of how “close” (or far) is the estimator to the actual data points which the estimator is trying to estimate. Notice that I have used the word “expected” which implies that the difference is being thought over keeping in mind that we will be doing this model training experiment (ideally)infinite number of times.

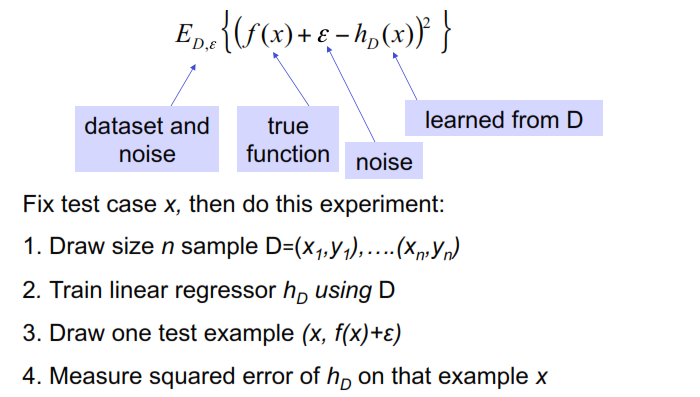
Naturally, an estimator will have **high bias** at a test point(and hence overall too, in the limit) if it does NOT wiggle or change too much when a different sample set of the data is thrown at it. This will usually be the case when an estimator does not have enough “capacity” to adequately fit the inherent data generating function. Therefore, simpler models have a higher bias compared to more sophisticated models.

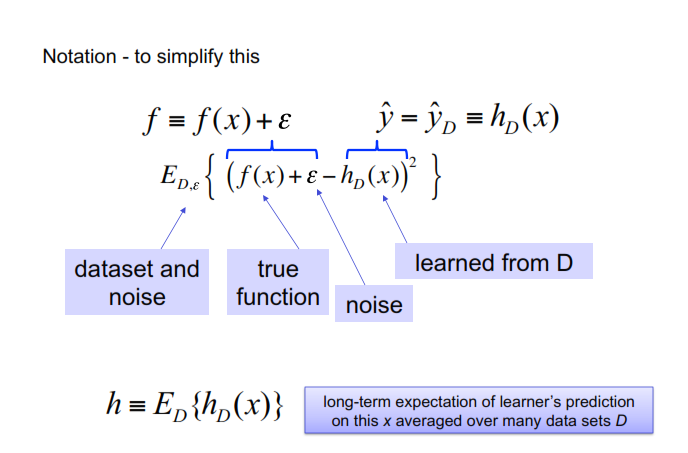
**Variance**of an estimator is the “expected” value of the squared difference between the estimate of a model and the “expected” value of the estimate (over all the models in the estimator).

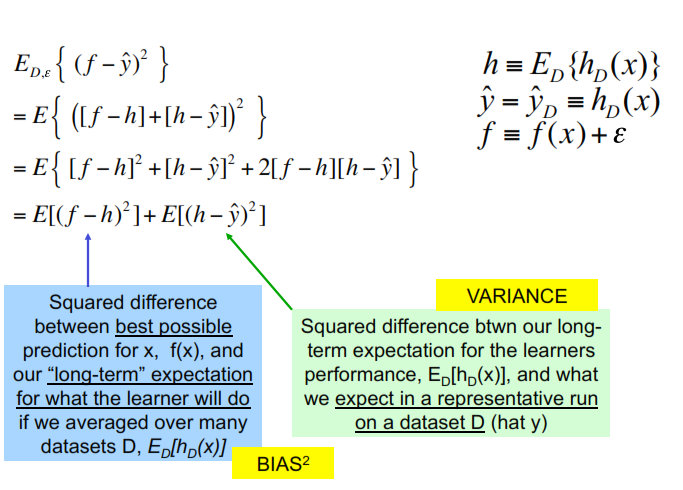
**Intrinsic “target noise”** This quantity is a lower bound on the expected cost of any learning algorithm. It is the expected cost of the Bayes-optimal classier.

**Squared “bias”** This quantity measures how closely the learning algorithm's average guess (over all possible training sets of the given training set size) matches the target.

**“Variance”** This quantity measures how much the learning algorithm's guess “bounces around” for the different training sets of the given size.







In addition to the intuitive insight the bias and variance decomposition provides, it has several other useful attributes. Chief among these is the fact that there is often a “bias-variance trade-off.” Often as one modifies some aspect of the learning algorithm, it will have opposite effects on the bias and the variance. For example, usually as one increases the number of degrees of freedom in the algorithm, the bias shrinks but the variance increases. The optimal number of degrees of freedom (as far as expected loss is concerned) is the number of degrees of freedom that optimizes this trade-off between bias and variance.

* 1. **Measuring Bias and Variance for Classification**

For classification, the quadratic loss function is often inappropriate because the class labels are not numeric. In practice, an overwhelming majority of researchers in the Machine Learning community instead use expected misclassification rate, which is equivalent to the zero-one loss.

We present the general result involving the expected zero-one loss, E(C), where the (implicit) conditioning event is arbitrary. Then we specialize to the standard conditioning used in conventional sampling theory statistics: a single test point, target, and training set size.

Let X and Y be the input and output spaces respectively, with cardinalities |X| and |Y| and elements x and y, respectively. We assume that both X and Y are countable. However, this assumption is not needed for this paper, provided all sums are re-placed by integrals. In classification problems, Y is usually a small finite set.

The “target” f is a conditional probability distribution P (YF = yF | x), where YF is a Y -valued random variable. Unless explicitly stated otherwise, we assume that the target is fixed. As an example, if the target is a noise-free function from X to Y , for any fixed x we have P (YF = yF | x) = 1 for one value of yF , and 0 for all others.

The “hypothesis” h generated by a learning algorithm is a similar distribution P (YH = yH | x), where YH is a Y -valued random variable. As an example, if the hypothesis is a single-valued function from X to Y , as it is for many classifiers (e.g., decision trees, nearest-neighbours), then P (YH = yH | x) = 1 for one value of yH , and 0 for all others.

We will drop the explicitly delineated random variables from the probabilities when the context is clear. For example, P (yH ) will be used instead of P (YH = yH ).

**Proposition 1** *YF and YH are conditionally independent given f and a test point x.*

Proof : P (yF , yH | f , x) = P (yF | yH, f , x)P (yH , f , x) = P (yF | f , x)P (yH | f , x).

The last equality is true because (by definition) yF depends only on the target f and the test point x.

The training set d is a set of m pairs of x-y values. We do not make any assumptions about the distribution of pairs. In particular, our mathematical results do not require them to be generated in an i.i.d. (in- dependently and identically distributed) manner, as commonly assumed.

To assign a penalty to a pair of values yF and yH, we use the loss function l : Y \*Y -> R. In this paper we consider the zero-one loss function defined as

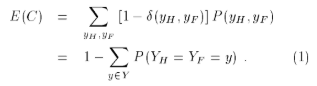




The cost, C, is a real-valued random variable defined as the loss over the random variables YF and YH . So, the expected cost is



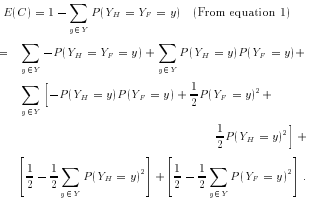
For zero-one loss, the cost is usually referred to as misclassification rate and is derived as follows:



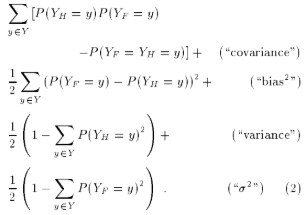
The notation used here is a simplified version of the extended Bayesian formalism (EBF) described in Wolpert (1994). In particular, the results of this paper do not depend on how the X-values in the test set are determined, so there is no need to define a random variable for those X-values as is done in the full EBF.

* 1. **The Decomposition**

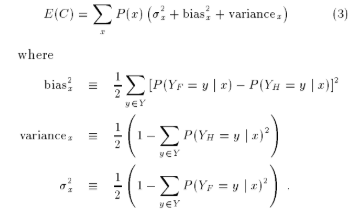
We present the general result involving the expected zero-one loss, E(C), where the (implicit) conditioning event is arbitrary. Then we specialize to the standard conditioning used in conventional sampling theory statistics: a single test point, target, and training set size.



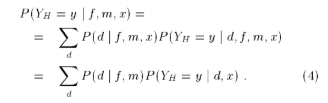
Rearranging the terms, we have E(C) =



In this paper we are interested in E(C | f, m), the expected cost where the target is fixed, and one averages over training sets of size m. One way to evaluate this quantity is to write it as ∑x P (x)E(C | f, m, x) and then use Equation 2 to get E(C | f, m, x). By Proposition 1, yH and yF are independent when one conditions on f and x, hence the \covariance" term vanishes. So



(To simplify the exposition, the f and m in the conditioning events are still implicit even though x needs to be explicit.) To better understand these formulas, note that P (YF = y | x) is the probability (after any noise is taken into account) that the fixed target takes on the value y at point x. To understand the quantity P (YH = y | x), one must write it in full as



In this expression, P (d | f, m) is the probability of generating training set d from the target f , and P (YH = y | d, x) is the probability that the learning algorithm makes guess y for point x in response to the training set d. So, P (YH = y | x) is the average (over training sets generated from f ) Y value guessed by the learning algorithm for point x.

Note that while the quadratic loss decomposition involves quadratic terms, and the log loss decomposition involves logarithmic terms (Wolpert submitted), our zero-one loss decomposition does not involve Kronecker delta terms, but rather involves quadratic terms.

Our definitions of “bias2,” “variance,” and “noise” obey some appropriate desiderata, including:

The “bias2” term measures the squared difference between the target's average output and the algorithm's average output. It is a real-valued non-negative quantity and equals zero only if P (YF = y | x) = P (YH = y | x) for all x and y. These properties are shared by bias2 for quadratic loss.

The variance term measures the “variability” (over YH ) of P (YH | x). It is a real-valued non-negative quantity and equals zero for an algorithm that always makes the same guess regardless of the training set (e.g., the Bayes optimal classifier). As the algorithm becomes more sensitive to changes in the training set, the variance increases. Moreover, given a distribution over training sets, the variance only measures the sensitivity of the learning algorithm to changes in the training set and is independent of the underlying target. This property is shared by variance for quadratic loss.

The noise measures the “variance” of the target in that the definitions of variance and noise are identical except for the interchange of YF and YH. In addition, the noise is independent of the learning algorithm. This property is shared by noise for quadratic loss.

In contrast to our definition of bias2, the definitions of bias (for a fixed target and a given instance) suggested in Kong & Dietterich (1995), Dietterich & Kong (1995), and Breiman (1996) are only two-valued for binary classification; they cannot quantify subtler levels of mismatch between a learning algorithm and a target. However, their decompositions have the advantage that their bias is zero for the Bayes optimal classifier, while ours may not be.

The major distinction between the decompositions arises in the variance term. All of the desiderata for variance listed above are violated by the decompositions proposed by Kong & Dietterich (1995), Dietterich & Kong (1995), and Breiman (1996). Specifically, in their definitions the variance can be negative and is not minimized by a classifier that ignores the training set. Kong & Dietterich (1995) note this shortcoming explicitly. The following examples illustrates the phenomenon for the decomposition suggested by Breiman (1996). Assume a noise free target with 51% heads and 49% tails. Consider an x for which the target has the value tails; the average error of a majority classifier will be slightly above 50%, yet the probability of error for the \aggregate" majority classifier will be one. This causes the variance to be negative.

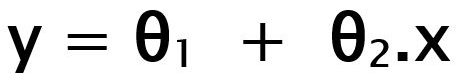
Another advantage of our decomposition is that its terms are a continuous function of the target. An Infinitesimal change in the target, which changes the class most commonly predicted by the learning algorithm for a given x, will not cause a large change in our bias, variance, or noise terms. In contrast, the other definitions of bias and variance do not share this property.

1. **Models Built**
   1. **LINEAR REGRESSION:**

**Linear Regression** is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering, and the number of independent variables being used.

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

**Hypothesis function for linear regression:**



Here,

‘x’ is independent variable or input variable

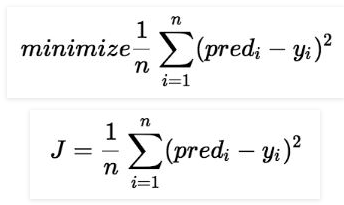
‘y’ is labels to data (output)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.

**θ1:** intercept  
**θ2:** coefficient of x

Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

**Cost Function(J):**

By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ1 and θ2 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y).

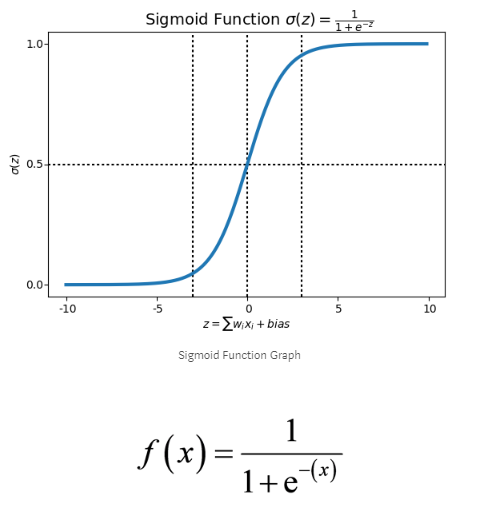
Cost function(J) of Linear Regression is the **Root Mean Squared Error (RMSE) between** predicted ‘y’ value (pred) and true ‘y’ value (y).

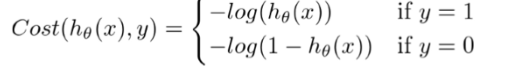
* 1. **MULTINOMIAL LOGISTIC REGRESSION:**

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability.

The hypothesis of logistic regression tends it to limit the cost function between 0 and 1. Therefore linear functions fail to represent it as it can have a value greater than 1 or less than 0 which is not possible as per the hypothesis of logistic regression.

In order to map predicted values to probabilities, we use the Sigmoid function. The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.



**** For logistic regression, the Cost function is defined as:

* 1. **DECISION TREE:**

Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems. Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

We can represent any Boolean function on discrete attributes using the decision tree.

In Decision Tree the major challenge is to identification of the attribute for the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures:

* Information Gain
* Gini Index

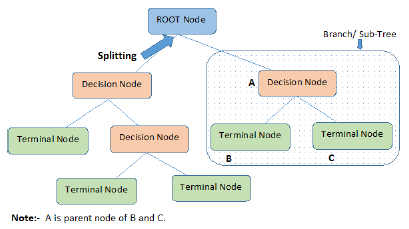
**Entropy**: It is a measure of uncertainty of a random variable.

**Information gain**: When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy.

**Gini Index:** It is a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with lower Gini index should be preferred.

Let’s look at the basic terminology used with decision tree:

1. **Root Node:** It represents entire population or sample and this further
2. gets divided into two or more homogeneous sets.
3. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
4. **Decision Node:** When a sub-node splits into further sub-nodes, then it
5. is called decision node.
6. **Leaf/ Terminal Node:** Nodes do not split is called Leaf or Terminal
7. node.
8. **Pruning:** When we remove sub-nodes of a decision node, this process
9. Is called pruning.
10. **Branch / Sub-Tree:** A sub section of entire tree is called branch or
11. Subtree.
12. **Parent and Child Node:** A node, which is divided into sub-nodes is
13. Called parent node of sub-nodes whereas sub-nodes are the children of
14. Parent node.

.

* 1. **RANDOM FOREST:**

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as **bagging**. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

**Approach:**

* Pick at random K data points from the training set
* Build the decision tree associated with those K data points.
* Choose the number N-tree of trees you want to build and repeat step 1 & 2.
* For a new data point, make each one of your N-tree trees predict the value of Y for the data point, and assign the new data point the average across all of the predicted Y values.

* 1. **SUPPORT VECTOR MACHINE:**

SVM can be used to solve both classification and regression problems.

The idea of SVM is to find nonlinear boundaries by constructing a linear boundary in a large, transformed version of the feature space.

SVM’s are classified into two categories:

1. Linear SVM’s – In linear SVM’s the training data i.e. classifiers are separated by a hyper-plane.

2.Non-Linear SVM’s- In non-linear SVM’s it is not possible to separate the training data using a hyper-plane. Under such conditions, the training data is too complex that it is impossible to find a representation for every feature vector.

Let’s look at the basic terminology used with SVM:

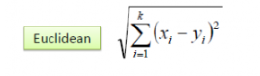
* **Kernel**: The function used to map a lower dimensional data into higher dimensional data.
* **Hyper** **Plane**: In SVM this is basically a separation line between the data classes. In SVR, it is the line that will help us to predict the continuous value or target value.
* **Boundary** **Line**: In SVM there are two lines other than hyper plane which creates a margin. The support vectors can be on the boundary line or outside.
* **Support** **Vectors**: This are the data points which are closest to the boundary.
  1. **K – NEAREST NEIGHBORS**

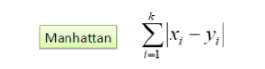
One of the simplest algorithms in machine learning is KNN, that is used for classification and regression. KNN is non-parametric and instance-based learning algorithm. It is a lazy algorithm. KNN algorithm is based on feature similarity.

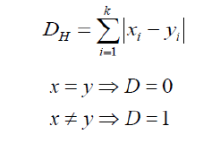
**Approach:**

1. Input: K (number of clusters), set of points x1 … xn
2. Place centroids c1 … ck at random points
3. Repeat until convergence or a certain amount of iterations
4. For point Xi find nearest centroid Cj
5. Assign the point Xi to cluster J
6. For each cluster J:
7. Calculate new centroid Cj = mean of all points Xi assigned to cluster J in previous step.

There are various methods for calculating this distance, of which the most commonly known methods are Euclidean, Manhattan (for continuous) and Hamming distance (for categorical).

**Euclidean Distance:** It’s calculated as the square root of the sum of the squared differences between the two point of interest.

**Manhattan Distance:** Calculate the distance between real vectors using the sum of their absolute difference. Also called **City Block Distance**.

**Hamming Distance:** It is used for categoricalvariables**.** If the predicted value(x) and the real value(y) are same, the distance D will be equal to 0, otherwise D will be 1.

1. **Analysis and Evaluation**
   1. **Regression**
   2. **Problem Statement:**

Using the train data to obtain expected estimates of test data. This will be done in the following ways:-

* Forming 100 bootstrap samples of the data.
* Building same model family for example linear regression on all the samples. Now we have 100 trained linear regression models. We’ll then obtain 100 different predictions/estimates for the test data.
* Now we’ll calculate the expected estimate of linear regression by averaging out test estimates.
* These will be used to calculate Bias and Variance according to the formula explained in the Bias and Variance Decomposition section.
* This procedure will be repeated for other model families.
  1. **Data Description:**

This dataset contains results of NASA airfoil testing in 1989 and is published in the UCI Machine Learning Repository [reference]. It is about self-induced or better self-caused noise due to airflow over an airfoil. The data consists of 1,503 rows, with 6 features obtained from a series of aerodynamic and acoustic experiments on airfoil blade sections conducted in an anechoic wind tunnel. The data set comprises different size NACA 0012 airfoils at various wind tunnel speeds and angles of attack. The span of the airfoil and the observer position were the same in all of the experiments.

The features are as follows:

1. Frequency, in Hertzs.

2. Angle of attack, in degrees.

3. Chord length, in meters.

4. Free-stream velocity, in meters per second.

5. Suction side displacement thickness, in meters.

The target variable/feature is:

6. Scaled sound pressure level, in decibels.

* 1. **Exploratory Data Analysis:**

The dataset consists of 1503 observations. Each of these observations are being described by 5 features and the target is to measure the sound pressure level.

* 1. **Data Cleaning:**

1128 samples will be used for training, 374 as test data. The data had no NA values and was otherwise clean. So, no pre-processing was required apart from using the correct datatypes for each feature.

* 1. **Bootstrap Sampling Method for Regression**

The basic idea of bootstrap is making inference about an estimate(such as sample mean) for a population parameter θ (such as population mean) on sample data. It is a resampling method by independently sampling with replacement from an existing sample data with same sample size n and performing inference among these resampled data. Bootstrap Sampling involves the following steps:

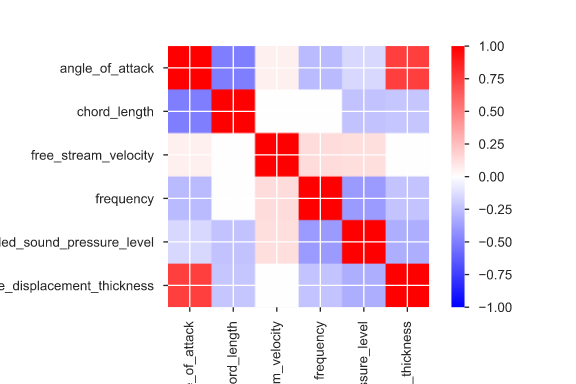
1. A sample from population with sample size n.
2. Draw a sample from the original sample data **with replacement**with size n and replicate the sampling process multiple times. Each re-sampled sample is called a **Bootstrap Sample**.
3. Evaluate the required statistic for each Bootstrap Sample.
4. Construct a sampling distribution with these B Bootstrap statistics and use it to make further statistical inference.

In our work, we shall be using this method to arrive at the **Expected Estimates** (which are required for calculating bias and variance) of each model family type like Linear Regression, Decision Trees to name a few.

**Why Bootstrap?**

Bootstrapping does not make assumptions about the distribution of the data. Traditional methods often assume that the data follow the normal distribution or some other distribution. For example, K-fold method gives equal weightage to every observation, irrespective of what the actual data distribution would be. Bootstrap makes no such assumptions. Given a reasonably large number of samples, those observations which have a higher value in the original data’s distribution; will automatically be represented more number of times (or more accurately, the correct number of times). So, we will be left with samples which follow the data’s original distribution and not something pre-decided.

* 1. **Correlation Plot**



* 1. **Model Evaluation:**
  2. **Bias Square**

|  |  |
| --- | --- |
| MODELS | BIAS\_SQUARE |
| RANDOM FOREST | 13.67256 |
| DECISION TREE | 13.71496 |
| LINEAR REGRESSION | 22.57787 |
| SUPPORT VECTOR MACHINE | 23.70495 |

* 1. **Evaluation:**
* From the above analysis, we found out that Linear Regression and Support Vector Machines are Bias sensitive models.
  1. **Variance**

|  |  |
| --- | --- |
| MODELS | VARIANCE |
| LINEAR REGRESSION | 0.154464 |
| SUPPORT VECTOR MACHINE | 0.1729648 |
| RANDOM FOREST | 0.6364176 |
| DECISION TREE | 7.048892 |

**Graphical representation of Variance**

* 1. **Evaluation**
* Decision Tree is Variance sensitive model.
  1. **Mean Squared Error**

|  |  |
| --- | --- |
| MODELS | MEAN SQUARED ERROR |
| RANDOM FOREST | 14.30262 |
| DECISION TREE | 20.69336 |
| LINEAR REGRESSION | 22.73079 |
| SUPPORT VECTOR MACHINE | 23.87619 |

**Graphical representation of Mean Squared Error Model Wise**

As shown above, Random Forest has the least error on test and SVM has the highest error.

**Graphical representation of bias and variance percentage wise**

From the above graphs we found out that, the percentage of Bias error is more in linear regression, support vector machines and Random forest. And the percentage of Variance error is more in Decision tree and random forest. So, we can infer that Decision tree is high variance model as more of its error comes from variance and Linear Regression is high bias model as more of its error comes from bias.

* 1. **Classification**
  2. **Problem Statement:**

We have to build various models on this dataset and then measure bias and variance for each of the machine learning models and then find out which models/algorithms are **bias sensitive** and **variance sensitive.**

* 1. **Data Description:**

Tennis, one of the most popular professional sports around the world, still uses manual coding of point outcomes.

Every tennis match is made up of a sequence of points. A point begins with a serve and players exchange shots until a player makes an error or is unable to return a shot in play.

Traditionally, the shot ending a point in tennis has been had been described in one of three mutually exclusive ways: a winner, an unforced error, or a forced error. A winner is a shot that was in play, not touched by the opponent, and ends with the point going to the player who made the shot. The other two categories are two distinct types of errors where both end with the point going to the player who did not make the shot.

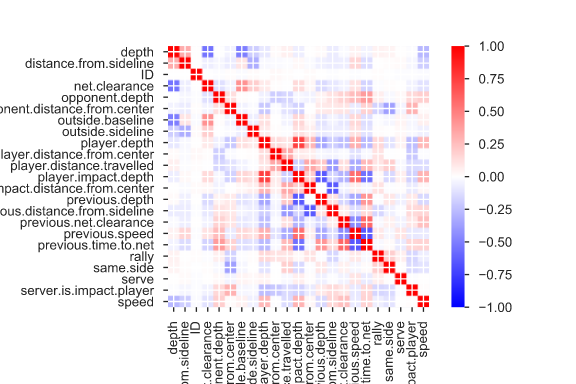
* 1. **Exploratory Data Analysis**

The data consists of 8001 observations. Every observation is described by 26 features and 1 class column which identifies it to be either a winner, unforced error or forced error.

* 1. **Data Cleaning:**
* **Correct the datatypes:**  By seeing the data structure we found that ‘serve’ was not in correct format. So, we converted it to categorical from integer.
* **Label Encoding:** we had to do label encoding for the following categorical features ['serve','hitpoint','outside.sideline','outside.baseline','same.side','previous.hitpoint','server.is.impact.player','outcome','gender']
* **Target:** “outcome” 3 categorical levels: winner, unforced error and forced error.
  1. **Handling Missing Data:**

Since there were no missing values in our dataset, there was no need for any imputation to be done.

* 1. **Correlation Plot**



* 1. **Feature Selection:**

Our data consists of 27 features including target class. Broadly we have photometric data and spectral data. As ID is unique, we can drop the ID column for analysis.

* 1. **Data Splitting:**

Dataset consists of 8001 observations and 26 features. We are splitting the data into 80:20 ratio. We are using 80 % of the data as train data and 20% of the data as test data by using a random seed number as 123.

* 1. **Model Evaluation**
  2. **Bias Square**

|  |  |
| --- | --- |
| MODELS | AVG\_BIAS\_SQUARE |
| RANDOM FOREST | 0.12065584 |
| DECISION TREE | 0.137531094 |
| LOGISTIC REGRESSION | 0.145459962 |
| SUPPORT VECTOR MACHINE | 0.149493311 |
| K-NEAREST NEIGHBORS | 0.215915053 |

**Graphical Representation of Bias Square**

* 1. **Variance**

|  |  |
| --- | --- |
| MODELS | AVG\_VARIANCE |
| DECISION TREE | 0.07097063 |
| RANDOM FOREST | 0.140993129 |
| SUPPORT VECTOR MACHINE | 0.151166528 |
| K-NEAREST NEIGHBORS | 0.164222361 |
| LOGISTIC REGRESSION | 0.177570198 |

**Graphical Representation of Variance**

* 1. **Findings:**
* From the above analysis, we found out that, K-Nearest Neighbors, Support Vector Machines and Logistic Regression are bias sensitive models.
* Logistic Regression, K-Nearest Neighbors, Support Vector Machines and Random Forest are variance sensitive models

1. **Future Analysis** 
   1. **Future Analysis:**

* For future analysis, we can try implementing different neural network architectures and also Naïve Bayes. This might provide better results.
* Try different sampling methods.
* Use regularization terms.

1. **References:**

* Bias Plus Variance Decomposition for Zero-One Loss Functions Ron Kohavi Data Mining and Visualization Silicon Graphics, Inc. 2011 N. Shoreline Blvd Mountain View, CA 94043-1389 ronnyk@sgi.com David H. Wolpert The Santa Fe Institute 1399 Hyde Park Rd. Santa Fe, NM 87501 [dhw@santafe.edu](mailto:dhw@santafe.edu)

**Websites:**

* <https://towardsdatascience.com/the-bias-variance-trade-off-explanation-and-demo-8f462f8d6326>