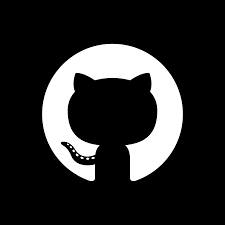
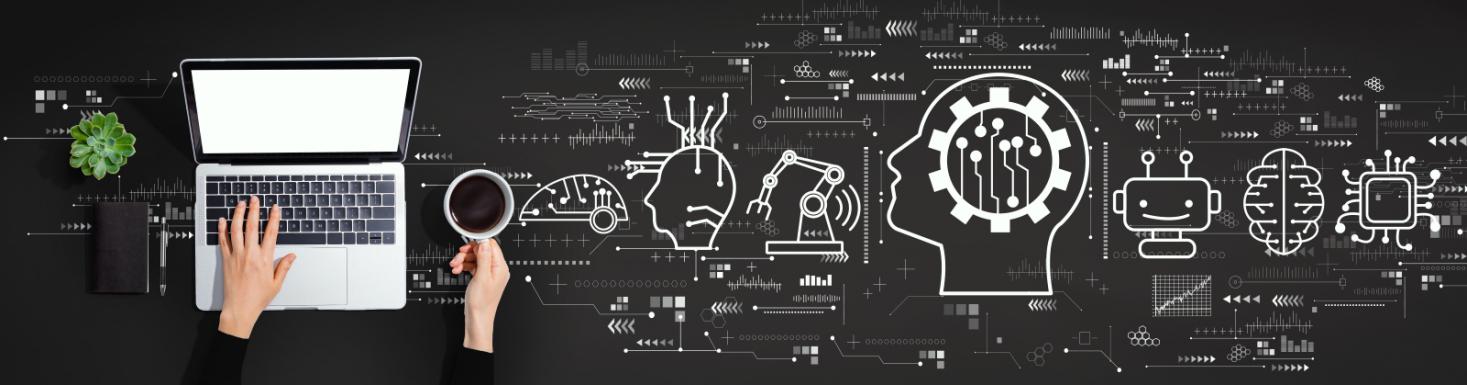
**BREAST CANCER MACHINE LEARNING CAPSTONE PROJECT REPORT**

**BY: UTPAL MISHRA**



**INTRODUCTION/ BUSINESS PROBLEM**

**BACKGROUND**

**Breast cancer** starts its activity in the breast cells of both men and women. Globally, it ranked as the second most common type of cancer after lung cancer and the fifth most common cause of death due to cancer. The **National Breast Cancer Foundation** has valued around 2,00,000 new breast cancer cases and 40,000 deaths annually in women. In men, these stats are 1700 and 450, correspondingly.

Breast cancer is highly major in today’s world. Cancer begins in cells and spreads to other parts of the human being. Additional cell growth develops a bulk of tissue named as a lump. Therefore, early detection of cancer is more important. Mammography is an exclusive screening test for breast cancer detection. According to the National Cancer Institute, an estimated 2,07,090 new cases and 39,840 deaths from breast cancer in only women, are predicted to occur in the United States, despite the recent breakthroughs in treatment. Given such conditions, early diagnosis of breast cancer is considered vivacious because statistics show a five-year survival rate of 96% for those who were diagnosed with cancer at early stages.

In the women's world, according to **Cancer Prevention and Control**(CPC), breast cancer is the second largest cause of death next to lung cancer, but if it is diagnosed at an early stage, it’s also one of the curable cancers. This is a tedious and confusing task, thus causing oversight errors that fail to detect cancer.

Among these, cancer is a term for a class of diseases identified by abnormal cells that grow and plunder healthy cells in the body. Breast cancer starts in the cells of the breast as a group of cancer cells that attack the surrounding tissues or spread to other areas of the body. Next to lung cancer, breast cancer is the second leading cause of cancer death among women. It occurs in both genders, though male breast cancer is rare. Recent statistics show that breast cancer is a serious disease with a high incidence rate and one of the leading causes of the early death of women.

**PROBLEM**

To build an efficient Machine Learning Classification model in order to analyse and detect Breast Cancer on the basis of lump dimensions and distribution available from the WDBC Data.

**DATASET**

**FOR MODELING:**

**SOURCE**

**Breast Cancer Wisconsin (Diagnostic) Data Set**

**Abstract**: Diagnostic Wisconsin Breast Cancer Database

**DATA SET INFORMATION**

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.  
  
Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.  
  
The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].  
  
This database is also available through the UW CS ftp server:  
ftp ftp.cs.wisc.edu  
cd math-prog/cpo-dataset/machine-learn/WDBC/

**ATTRIBUTE INFORMATION**

1. ID number  
   2) Diagnosis (M = malignant, B = benign)  
   3-32)  
     
   Ten real-valued features are computed for each cell nucleus:  
     
   a) radius (mean of distances from center to points on the perimeter)  
   b) texture (standard deviation of gray-scale values)  
   c) perimeter  
   d) area  
   e) smoothness (local variation in radius lengths)  
   f) compactness (perimeter^2 / area - 1.0)  
   g) concavity (severity of concave portions of the contour)  
   h) concave points (number of concave portions of the contour)  
   i) symmetry  
   j) fractal dimension ("coastline approximation" - 1)

**FOR COUNTRY WISE ANALYSIS USING FOURSQUARE (TO EXTRACT THE COORDINATES OF THE COUNTRIES):**

**SOURCE**

**Link: <https://www.wcrf.org/dietandcancer/cancer-trends/data-cancer-frequency-country>**

**ABOUT**

**T**he age-standardized rate for all cancers (including non-melanoma skin cancer) for men and women combined was 197.9 per 100,000 in 2018. The rate was higher for men (218.6 per 100,000) than women (182.6 per 100,000).

Age-standardized rates are used in the tables. This is a summary measure of the rate of disease that a population would have if it had a standard age structure. Standardization is necessary when comparing populations that differ concerning age because age has a powerful influence on the risk of dying from cancer.

**GLOBAL CANCER RATE:**

The highest cancer rate for men and women together was in Australia, at 468.0 people per 100,000. The age-standardized rate was at least 320 per 100,000 for 12 countries: Australia, New Zealand, Ireland, Hungary, the US, Belgium, France (metropolitan), Denmark, Norway, the Netherlands, Canada and New Caledonia (France). The countries in the top 12 come from Oceania, Europe and North America.

**CANCER RATE IN MENS:**

The highest cancer rate was found in Australia at 579.9 men per 100,000.

The age-standardized rate was at least 360 per 100,000 in 15 countries: Australia, New Zealand, Ireland, Hungary, France (metropolitan), the US, Latvia, Belgium, Norway, Slovenia, Estonia, Slovakia, Denmark, New Caledonia (France) and the Netherlands.

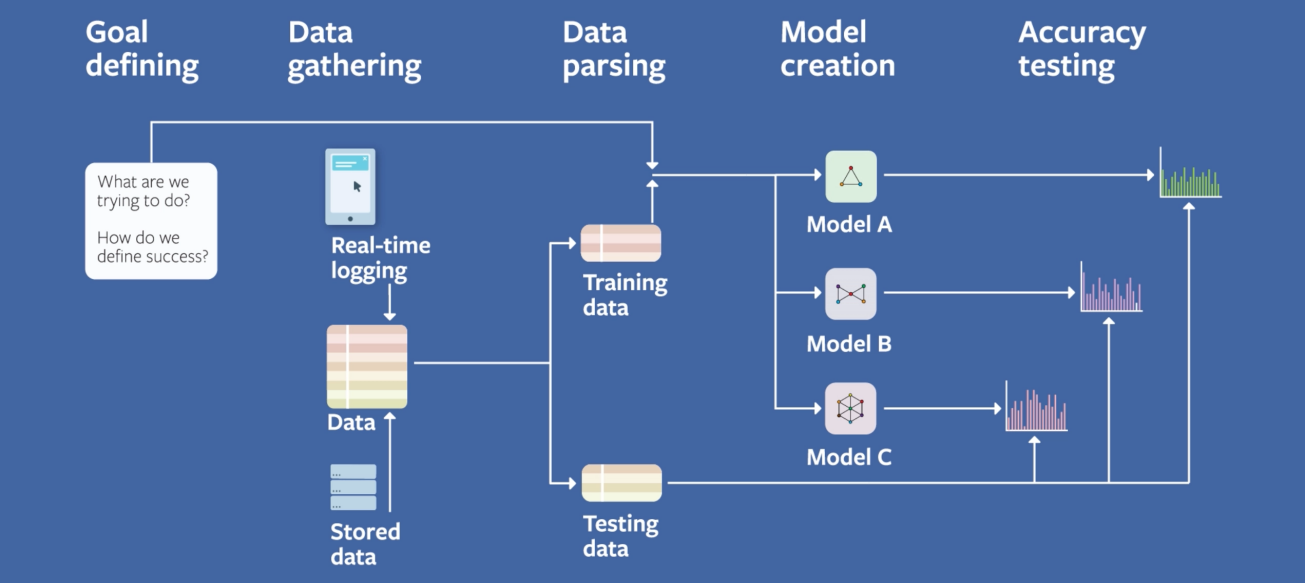
The countries in the top ten come from Europe, Oceania and the Americas.

**CANCER RATE IN WOMENS:**

The highest cancer rate was found in Australia at 363.0 women per 100,000.

The age-standardized rate was at least 300 per 100,000 in 11 countries (Australia, New Zealand, Hungary, Belgium, Canada, Denmark, Ireland, the US, the Netherlands, Norway and South Korea). The countries in the top ten come from Europe, Oceania and the Americas.

**METHODOLOGY**



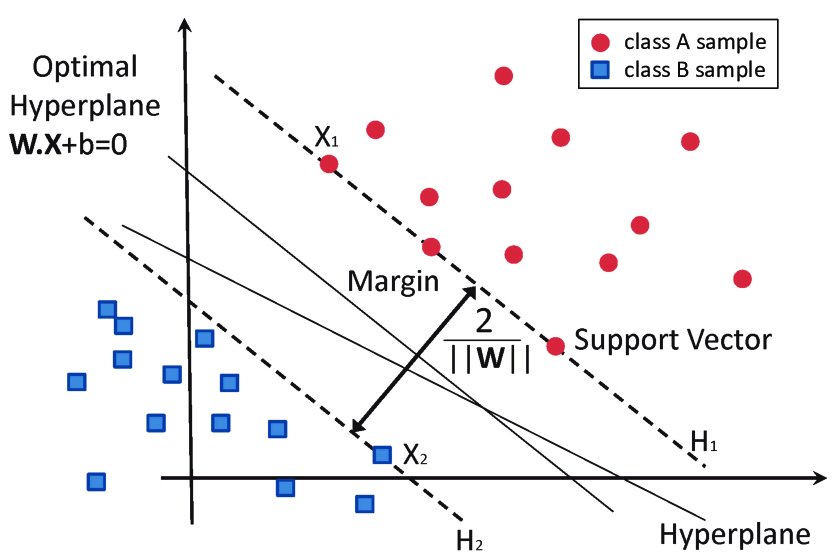
**ENSEMBLE LEARNING WITH CROSS VALIDATION**

In [statistics](https://en.wikipedia.org/wiki/Statistics" \o "Statistics) and [machine learning](https://en.wikipedia.org/wiki/Machine_learning" \o "Machine learning), **ensemble methods** use multiple learning algorithms to obtain better [predictive performance](https://en.wikipedia.org/wiki/Predictive_inference" \o "Predictive inference) than could be obtained from any of the constituent learning algorithms alone. Unlike a [statistical ensemble](https://en.wikipedia.org/wiki/Statistical_ensemble" \o "Statistical ensemble) in statistical mechanics, which is usually infinite, a machine learning ensemble consists of only a concrete finite set of alternative models, but typically allows for much more flexible structure to exist among those alternatives.

An ensemble is itself a supervised learning algorithm, because it can be trained and then used to make predictions. The trained ensemble, therefore, represents a single hypothesis. This hypothesis, however, is not necessarily contained within the hypothesis space of the models from which it is built. Thus, ensembles can be shown to have more flexibility in the functions they can represent. This flexibility can, in theory, enable them to [over-fit](https://en.wikipedia.org/wiki/Overfitting" \o "Overfitting) the training data more than a single model would, but in practice, some ensemble techniques (especially [bagging](https://en.wikipedia.org/wiki/Bootstrap_aggregating" \o "Bootstrap aggregating)) tend to reduce problems related to over-fitting of the training data.

Empirically, ensembles tend to yield better results when there is a significant diversity among the models. Many ensemble methods, therefore, seek to promote diversity among the models they combine. Although perhaps non-intuitive, more random algorithms (like random decision trees) can be used to produce a stronger ensemble than very deliberate algorithms (like entropy-reducing decision trees). Using a variety of strong learning algorithms, however, has been shown to be more effective than using techniques that attempt to *dumb-down* the models in order to promote diversity.

**SUPPORT VECTOR MACHINE**



**I**n [machine learning](https://en.wikipedia.org/wiki/Machine_learning" \o "Machine learning), **support-vector machines** (**SVMs**, also **support-vector networks**) are [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning" \o "Supervised learning) models with associated learning [algorithms](https://en.wikipedia.org/wiki/Algorithm" \o "Algorithm) that analyze data used for [classification](https://en.wikipedia.org/wiki/Statistical_classification" \o "Statistical classification) and [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis" \o "Regression analysis). Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-[probabilistic](https://en.wikipedia.org/wiki/Probabilistic_classification" \o "Probabilistic classification) [binary](https://en.wikipedia.org/wiki/Binary_classifier" \o "Binary classifier) [linear classifier](https://en.wikipedia.org/wiki/Linear_classifier" \o "Linear classifier). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

**I**n addition to performing [linear classification](https://en.wikipedia.org/wiki/Linear_classifier" \o "Linear classifier), SVMs can efficiently perform a non-linear classification using what is called the [kernel trick](https://en.wikipedia.org/wiki/Kernel_method" \l "Mathematics:_the_kernel_trick" \o "Kernel method), implicitly mapping their inputs into high-dimensional feature spaces.

More formally, a support-vector machine constructs a [hyperplane](https://en.wikipedia.org/wiki/Hyperplane" \o "Hyperplane) or set of hyperplanes in a [high-](https://en.wikipedia.org/wiki/High-dimensional_space" \o "High-dimensional space) or infinite-dimensional space, which can be used for [classification](https://en.wikipedia.org/wiki/Statistical_classification" \o "Statistical classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis" \o "Regression analysis), or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the [generalization error](https://en.wikipedia.org/wiki/Generalization_error" \o "Generalization error) of the classifier.

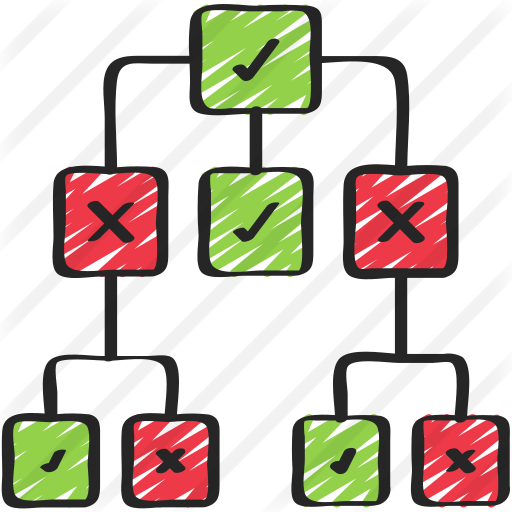
The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant, where such a set of vectors is an orthogonal (and thus minimal) set of vectors that defines a hyperplane. The sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated.

**NAIVE BAYES**

In [statistics](https://en.wikipedia.org/wiki/Statistics" \o "Statistics), **Naive Bayes classifiers** are a family of simple "[probabilistic classifiers](https://en.wikipedia.org/wiki/Probabilistic_classification" \o "Probabilistic classification)" based on applying [Bayes' theorem](https://en.wikipedia.org/wiki/Bayes'_theorem" \o "Bayes' theorem) with strong (naïve) [independence](https://en.wikipedia.org/wiki/Statistical_independence" \o "Statistical independence) assumptions between the features. They are among the simplest [Bayesian network](https://en.wikipedia.org/wiki/Bayesian_network" \o "Bayesian network) models. But they could be coupled with [Kernel density estimation](https://en.wikipedia.org/wiki/Kernel_density_estimation" \o "Kernel density estimation) and achieve higher accuracy levels.

Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. [Maximum-likelihood](https://en.wikipedia.org/wiki/Maximum-likelihood_estimation" \o "Maximum-likelihood estimation) training can be done by evaluating a [closed-form expression](https://en.wikipedia.org/wiki/Closed-form_expression" \o "Closed-form expression), which takes [linear time](https://en.wikipedia.org/wiki/Linear_time" \o "Linear time), rather than by expensive [iterative approximation](https://en.wikipedia.org/wiki/Iterative_method" \o "Iterative method) as used for many other types of classifiers.

**DECISION TREES**



A decision tree is a [flowchart](https://en.wikipedia.org/wiki/Flowchart" \o "Flowchart)-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules

A decision tree consists of three types of nodes

1. Decision nodes – typically represented by squares
2. Chance nodes – typically represented by circles
3. End nodes – typically represented by triangles

Decision trees are commonly used in [operations research](https://en.wikipedia.org/wiki/Operations_research" \o "Operations research) and [operations management](https://en.wikipedia.org/wiki/Operations_management" \o "Operations management). If, in practice, decisions have to be taken online with no recall under incomplete knowledge, a decision tree should be paralleled by a [probability](https://en.wikipedia.org/wiki/Probability" \o "Probability) model as a best choice model or online selection model [algorithm](https://en.wikipedia.org/wiki/Algorithm" \o "Algorithm). Another use of decision trees is as a descriptive means for calculating [conditional probabilities](https://en.wikipedia.org/wiki/Conditional_probability" \o "Conditional probability).

Decision trees, [influence diagrams](https://en.wikipedia.org/wiki/Influence_diagrams" \o "Influence diagrams), [utility functions](https://en.wikipedia.org/wiki/Utility_function" \o "Utility function), and other [decision analysis](https://en.wikipedia.org/wiki/Decision_analysis" \o "Decision analysis) tools and methods are taught to undergraduate students in schools of business, health economics, and public health, and are examples of operations research or [management science](https://en.wikipedia.org/wiki/Management_science" \o "Management science) methods

**DECISION TREE BAGGING**

**Bagging**(Bootstrap Aggregation) is used when our goal is to reduce the variance of a decision tree. Here idea is to create several subsets of data from training sample chosen randomly with replacement. Now, each collection of subset data is used to train their decision trees. As a result, we end up with an ensemble of different models. Average of all the predictions from different trees are used which is more robust than a single decision tree.

\*****Boosting**** is another ensemble technique to create a collection of predictors. In this technique, learners are learned sequentially with early learners fitting simple models to the data and then analyzing data for errors. In other words, we fit consecutive trees (random sample) and at every step, the goal is to solve for net error from the prior tree.

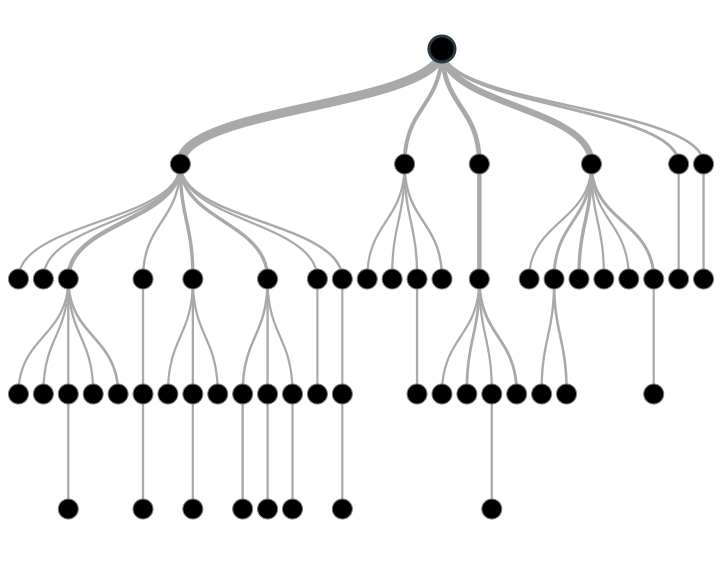
**EXTRA TREE**

**Extremely Randomized Trees Classifier(Extra Trees Classifier)** is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a “forest” to output it’s classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest.

**E**ach Decision Tree in the Extra Trees Forest is constructed from the original training sample. Then, at each test node, Each tree is provided with a random sample of k features from the feature-set from which each decision tree must select the best feature to split the data based on some mathematical criteria (typically the Gini Index). This random sample of features leads to the creation of multiple de-correlated decision trees.

**T**o perform feature selection using the above forest structure, during the construction of the forest, for each feature, the normalized total reduction in the mathematical criteria used in the decision of feature of split (Gini Index if the Gini Index is used in the construction of the forest) is computed. This value is called the Gini Importance of the feature. To perform feature selection, each feature is ordered in descending order according to the Gini Importance of each feature and the user selects the top k features according to his/her choice.

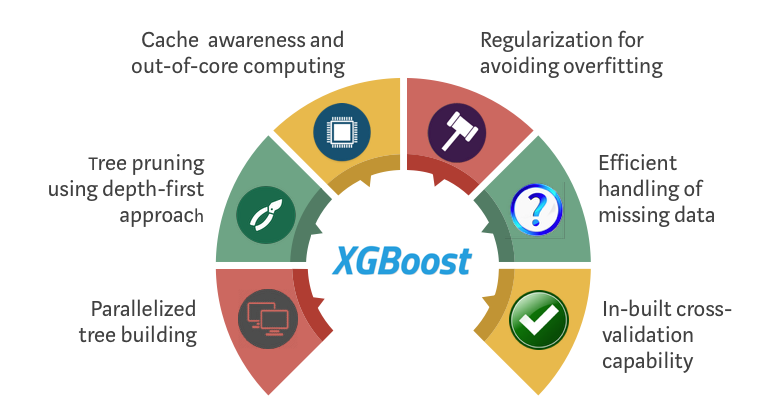
**RANDOM FOREST**



**Random forests** or **random decision forests** are an [ensemble learning](https://en.wikipedia.org/wiki/Ensemble_learning" \o "Ensemble learning) method for [classification](https://en.wikipedia.org/wiki/Statistical_classification" \o "Statistical classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis" \o "Regression analysis) and other tasks that operate by constructing a multitude of [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning" \o "Decision tree learning) at training time and outputting the class that is the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)" \o "Mode (statistics)) of the classes (classification) or mean prediction (regression) of the individual trees.[[1]](https://en.wikipedia.org/wiki/Random_forest" \l "cite_note-ho1995-1)[[2]](https://en.wikipedia.org/wiki/Random_forest" \l "cite_note-ho1998-2) Random decision forests correct for decision trees' habit of [overfitting](https://en.wikipedia.org/wiki/Overfitting" \o "Overfitting) to their [training set](https://en.wikipedia.org/wiki/Test_set" \o "Test set).

An extension of the algorithm was developed by [Leo Breiman](https://en.wikipedia.org/wiki/Leo_Breiman" \o "Leo Breiman)[[7]](https://en.wikipedia.org/wiki/Random_forest" \l "cite_note-breiman2001-7) and where the extension combines "[bagging](https://en.wikipedia.org/wiki/Bootstrap_aggregating" \o "Bootstrap aggregating)" idea and random selection of features.

**XGBOOST**



The library is laser focused on computational speed and model performance, as such there are few frills. Nevertheless, it does offer a number of advanced features.

**Model Features:**

Three main forms of gradient boosting are supported:

1. ****Gradient Boosting**** algorithm also called gradient boosting machine including the learning rate.
2. ****Stochastic Gradient Boosting**** with sub-sampling at the row, column and column per split levels.
3. ****Regularized Gradient Boosting**** with both L1 and L2 regularization.

**System Features:**

The library provides a system for use in a range of computing environments, not least:

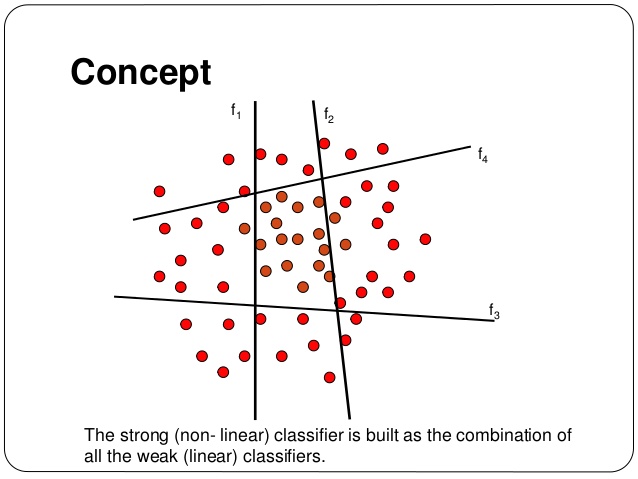
1. ****Parallelization**** of tree construction using all of your CPU cores during training.
2. ****Distributed Computing**** for training very large models using a cluster of machines.
3. ****Out-of-Core Computing**** for very large data sets that don’t fit into memory.
4. ****Cache Optimization**** of data structures and algorithm to make best use of hardware.

**Algorithm Features:**

The implementation of the algorithm was engineered for efficiency of compute time and memory resources. Some key algorithm implementation features include:

1. ****Sparse Aware**** implementation with automatic handling of missing data values.
2. ****Block Structure**** to support the parallelization of tree construction.
3. ****Continued Training**** so that you can further boost an already fitted model on new data.

**ADABOOST**



AdaBoost is best used to boost the performance of decision trees on binary classification problems. AdaBoost was originally called AdaBoost. AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. These are models that achieve accuracy just above random chance on a classification problem.

The most suited and therefore most common algorithm used with AdaBoost are decision trees with one level. Because these trees are so short and only contain one decision for classification, they are often called decision stumps.

Each instance in the training data set is weighted. The initial weight is set to:

**weight(xi) = 1/n**

Where xi is the ith training instance and n is the number of training instances

**CATBOOST**

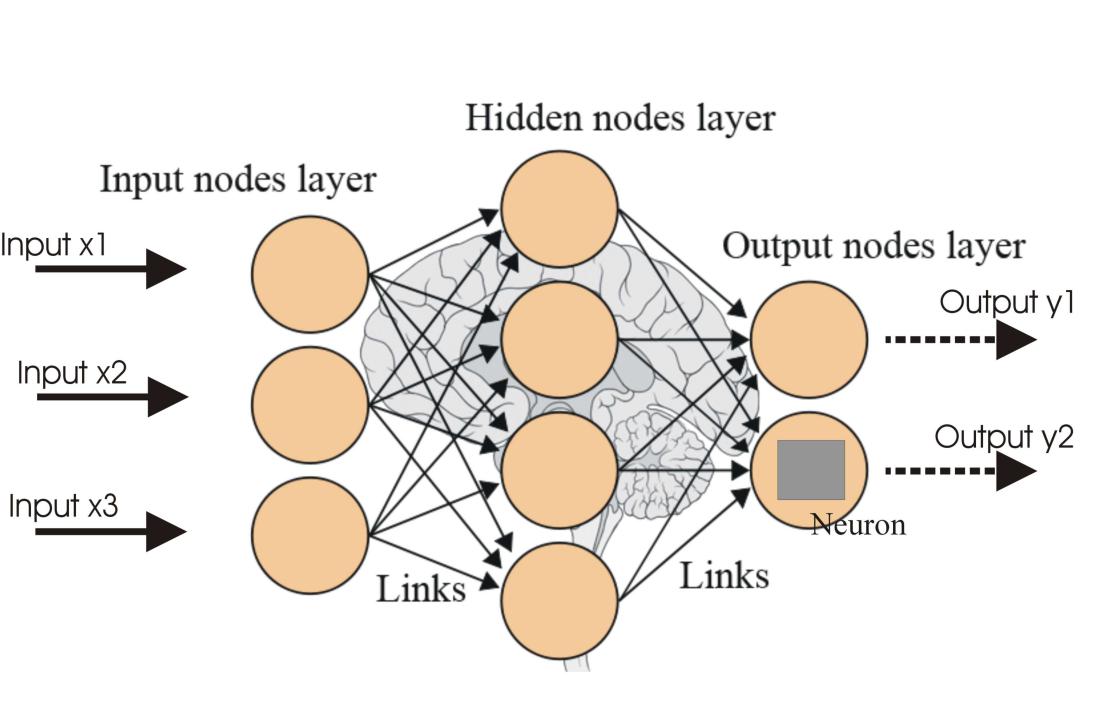
CatBoost is a recently open-sourced machine learning algorithm and work with diverse data types to help solve a wide range of problems that businesses face today. To top it up, it provides best-in-class accuracy.

It is especially powerful in two ways:

1. It yields state-of-the-art results without extensive data training typically required by other machine learning methods, and
2. Provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems.

“CatBoost” name comes from two words “****Cat****egory” and “****Boost****ing”. “****Boost****” comes from gradient boosting machine learning algorithm as this library is based on gradient boosting library. Gradient boosting is a powerful machine learning algorithm that is widely applied to multiple types of business challenges like fraud detection, recommendation items, forecasting and it performs well also. It can also return very good result with relatively less data, unlike DL models that need to learn from a massive amount of data.

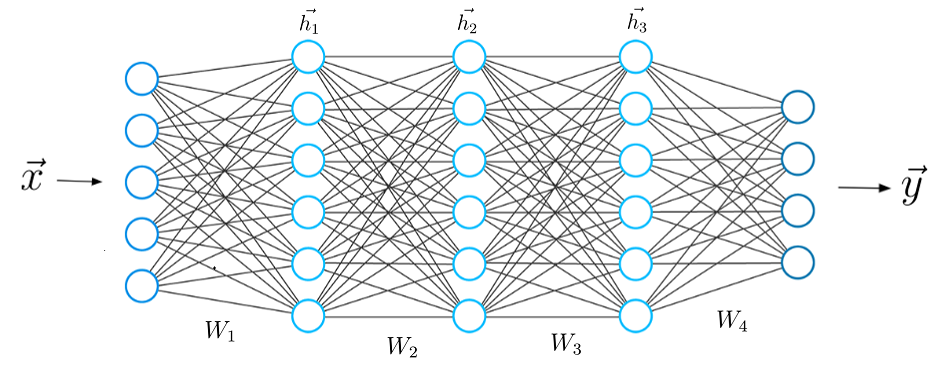
**ARTIFICIAL NEURAL NETOWORK**



**Artificial neural networks** (**ANNs**), usually simply called **neural networks** (**NNs**), are computing systems vaguely inspired by the [biological neural networks](https://en.wikipedia.org/wiki/Biological_neural_network" \o "Biological neural network) that constitute animal [brains](https://en.wikipedia.org/wiki/Brain" \o "Brain).

An ANN is based on a collection of connected units or nodes called [artificial neurons](https://en.wikipedia.org/wiki/Artificial_neuron" \o "Artificial neuron), which loosely model the [neurons](https://en.wikipedia.org/wiki/Neuron" \o "Neuron) in a biological brain. Each connection, like the [synapses](https://en.wikipedia.org/wiki/Synapse" \o "Synapse) in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a [real number](https://en.wikipedia.org/wiki/Real_number" \o "Real number), and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called *edges*. Neurons and edges typically have a *[weight](https://en.wikipedia.org/wiki/Weight_(mathematics)" \o "Weight (mathematics))* that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times

**DEEP LEARNING**



Deep learning is an [artificial intelligence (AI)](https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of [machine learning](https://www.investopedia.com/terms/m/machine-learning.asp) in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network

One of the most common AI techniques used for processing big data is machine learning, a self-adaptive algorithm that gets increasingly better analysis and patterns with experience or with newly added data.

If a digital payments company wanted to detect the occurrence or potential for fraud in its system, it could employ machine learning tools for this purpose. The computational algorithm built into a computer model will process all transactions happening on the digital platform, find patterns in the data set, and point out any anomaly detected by the pattern.

Deep learning, a subset of machine learning, utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning. The artificial neural networks are built like the human brain, with neuron nodes connected together like a web. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a nonlinear approach.

**PERFORMANCE METRICS**

While data preparation and training a machine learning model is a key step in the machine learning pipeline, it’s equally important to measure the performance of this trained model. How well the model generalizes on the unseen data is what defines adaptive vs non- adaptive machine learning models.

By using different metrics for performance evaluation, we should be in a position to improve the overall predictive power of our model before we roll it out for production on unseen data.

Without doing a proper evaluation of the ML model using different metrics, and depending only on accuracy, can lead to a problem when the respective model is deployed on unseen data and can result in poor predictions.

This happens because, in cases like these, our models don’t **learn** but instead **memorize**; hence, they cannot generalize well on unseen data. To get started, let’s define these three important terms:

1. **Learning:** ML model learning is concerned with the accurate prediction of future data, not necessarily the accurate prediction of training/available data.
2. **Memorization:** ML Model performance on limited data; in other words, [overfitting](https://heartbeat.fritz.ai/bias-variance-tradeoff-to-avoid-under-overfitting-d92f1fcff352) on the known training data set.
3. **Generalization:** Can be defined as the capability of the ML model to apply learning to previously unseen data. Without generalization there’s no learning, just memorization. But note that generalization is also goal specific —for instance, a well-trained image recognition model on zoo animal images may not generalize well on images of cars and buildings.

In the next section, we’ll discuss the different evaluation metrics available that could help in the generalization of the ML model.

## EVALUATION METRICS

The use of an evaluation matrix is one method of objectively evaluating a number of options against a number of criteria. These criteria are prioritized before the evaluation is made with greater weighting to those items of most importance. If there are criteria that absolutely must be met, two levels of evaluation matrices can be used. The first

level acts as a filter with each option evaluated against the mandatory criteria. Those options that meet every mandatory criterion go on to the second level to be evaluated against prioritized criteria.

After doing the usual Feature Engineering, Selection, and of course, implementing a model and getting some output in forms of a probability or a class, the next step is to find out how effective is the model based on some metric using test data sets.

### TYPES

When we talk about predictive models, we are talking either about a regression model (continuous output) or a classification model (nominal or binary output). The evaluation metrics used in each of these models are different.

### Class Output

Algorithms like SVM and KNN create a class output. For instance, in a binary classification problem, the outputs will be either 0 or 1. But these algorithms are not well accepted by the statistics community. However, today we have algorithms which can convert these class outputs to probability.

### Probabilistic Output

Algorithms like Logistic Regression, Random Forest, Gradient Boosting, Adaboost etc. give probability outputs. Converting probability outputs to class output is just a matter of creating a threshold probability.

Different performance metrics are used to evaluate different Machine Learning Algorithms and are discussed here.

## EVALUATION STRATEGIES

Metrics evaluates the quality of an engine by comparing engine's output (predicted result) with the original label (actual result). A engine serving better prediction should yield a higher metric score, the tuning module returns the engine parameter with the highest score. It is sometimes called [loss function](http://en.wikipedia.org/wiki/Loss_function) in literature, where the goal is to minimize the loss function.

During tuning, it is important for us to understand the definition of the metric, to make sure it is aligned with the prediction engine's goal.

In the classification template, we use Accuracy as our metric. Accuracy is defined as: the percentage of queries which the engine is able to predict the correct label.

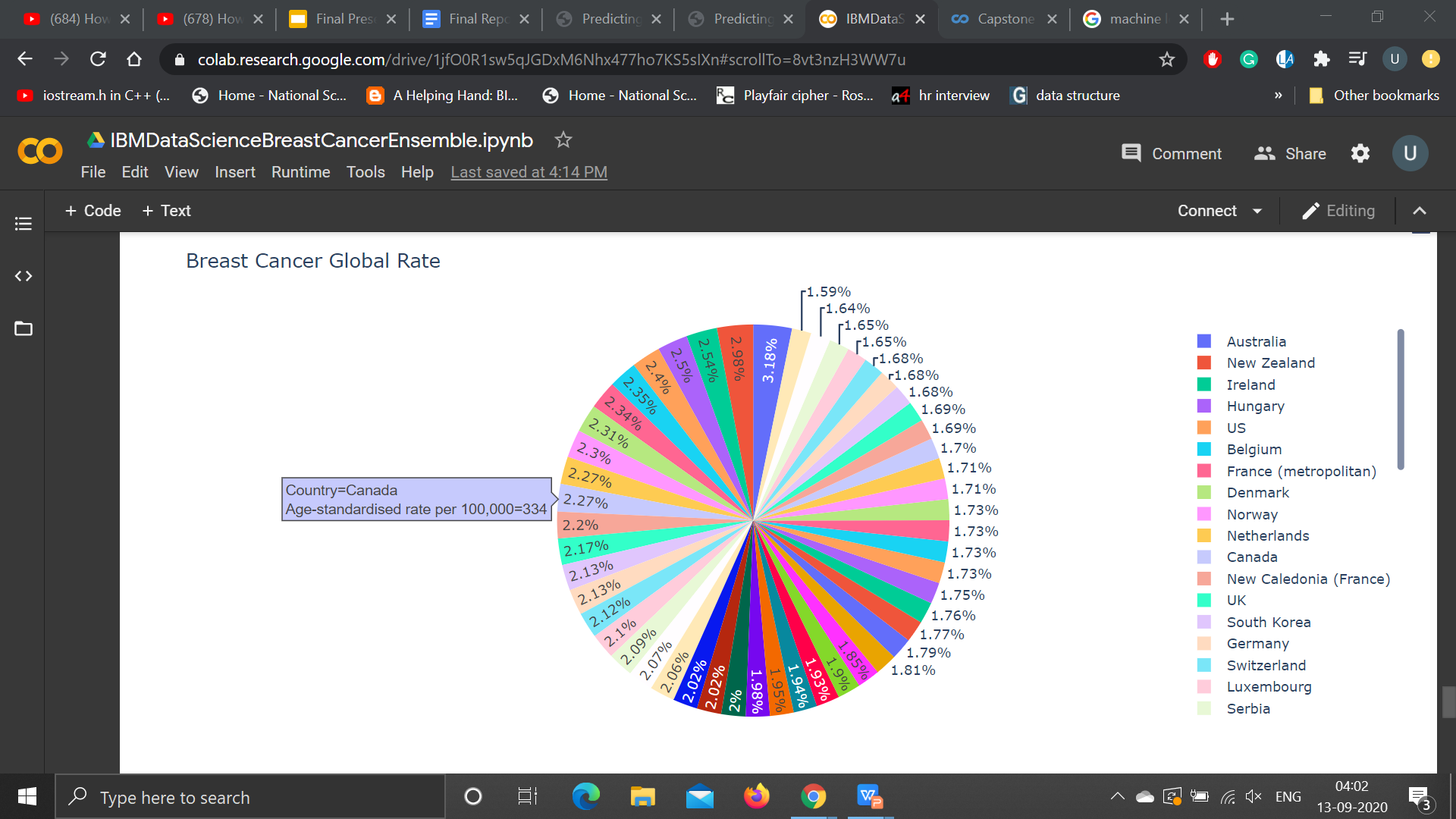
**Evaluation metrics** are used to measure the quality of the statistical or machine learning model. **Evaluating** machine learning models or algorithms is essential for any project.

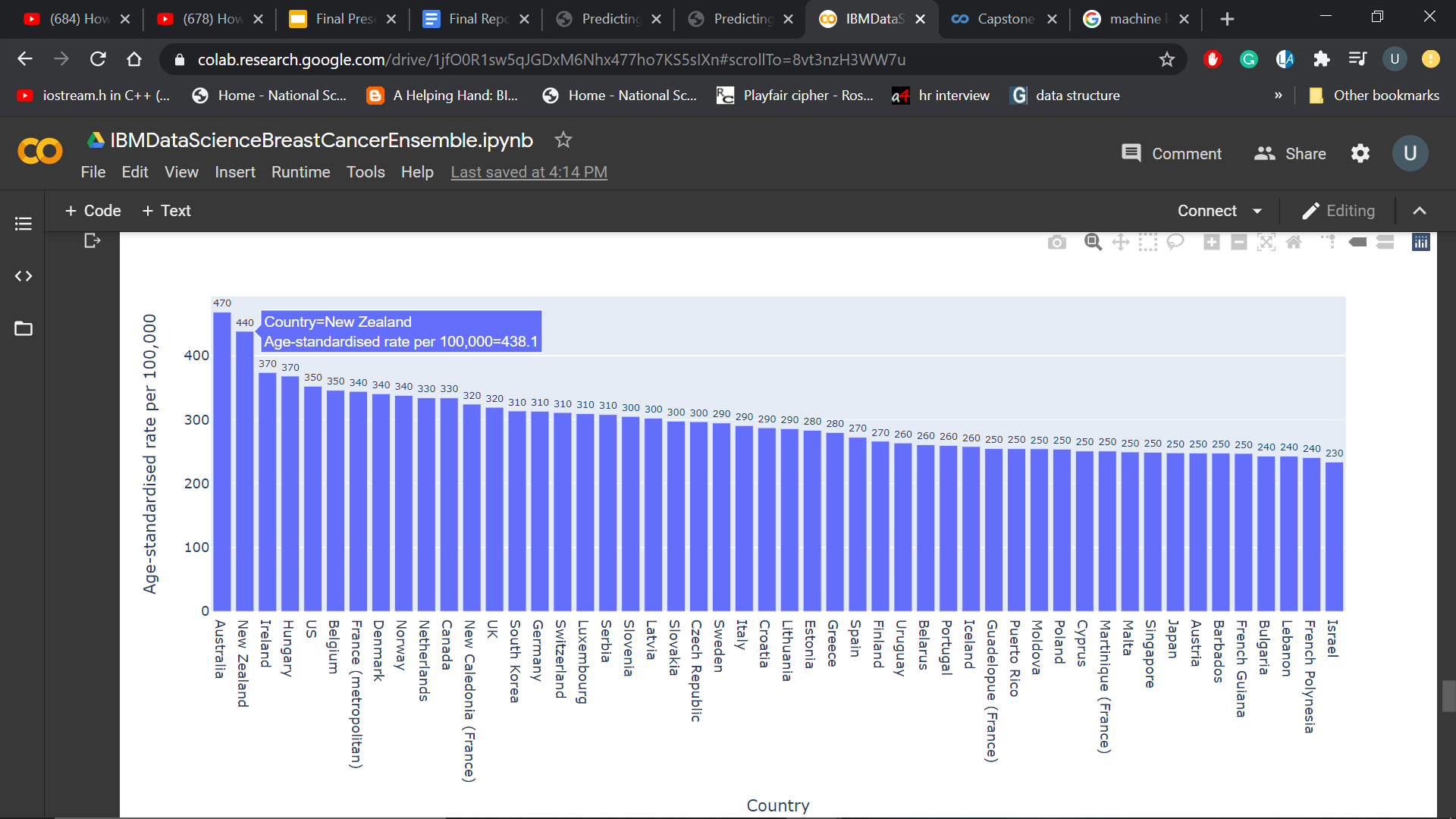
There are many different types of **evaluation metrics** available to test a model.

**APPLICATIONS**

If the evaluation matrix is being used to evaluate various design solutions, usually previously established requirements are used as the criteria. Since requirements are generally written in terms of “shall” or “should”, the mandatory criteria used in the first level of the evaluation are represented by the “shall” requirements while the “should” requirements are prioritized and used in the second level evaluation.

**EVALUATION METRICS**





## CROSS VALIDATION

Cross-validation is a statistical method used to estimate the skill of machine learning models. Cross Validation is a validation technique for the model to statistically examine the generalization pattern of the results on the independent data set.

It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

To ensure that the model has analyzed and understood the data pattern without noise or without being over-fitted/ under-fitting or with low bias, cross validation is required to statistically behold the stability of the model.

This model validation method provides a bit of flexibility over the splitting or groups or k- folds, which are as follow:

### k-Fold (k=2):

It means the data is grouped into two i.e. the training and the test data. This type of grouping is opted if, we have enough data to make the model learn the pattern on a randomly trained training data. Any duplicacy and overlapping of grouped data should be avoid and final model - after testing - should be retrained on the complete data set without any tuning in the hyper parameters.

### k-Fold (k=3):

This is comparatively a better approach then binary grouping the data set as the data set is separated into three, the training data, the validation data and the test data. To evaluate the quality of model fitted on trained data, model is validated (prior to testing) on a new sample (validation datas et). This pattern is chosen if the data size is sufficient enough to be grouped as such.

### k-Fold (k):

For splitting the data set, this is a prominent approach as the data available for to model the decipher the pattern is not never enough and model has to the problems of under-fitting and increased loss.

In this method, the data is grouped into k folds and model is trained into k-1 times. Each time k-1 portion is trained and is validated over the remaining portion.

Each time the model is trained is validated on a new piece of data which significantly reduces the under-fitting and the over-fitting problem. This method is chosen for a small sized data as the model is free from a high bias or a high variance.

## INFORMATION GAIN

Information gain is the reduction in entropy by transforming a data set and is often used in training decision trees. Information gain is calculated by comparing the entropy of the data set before and after a transformation.

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.

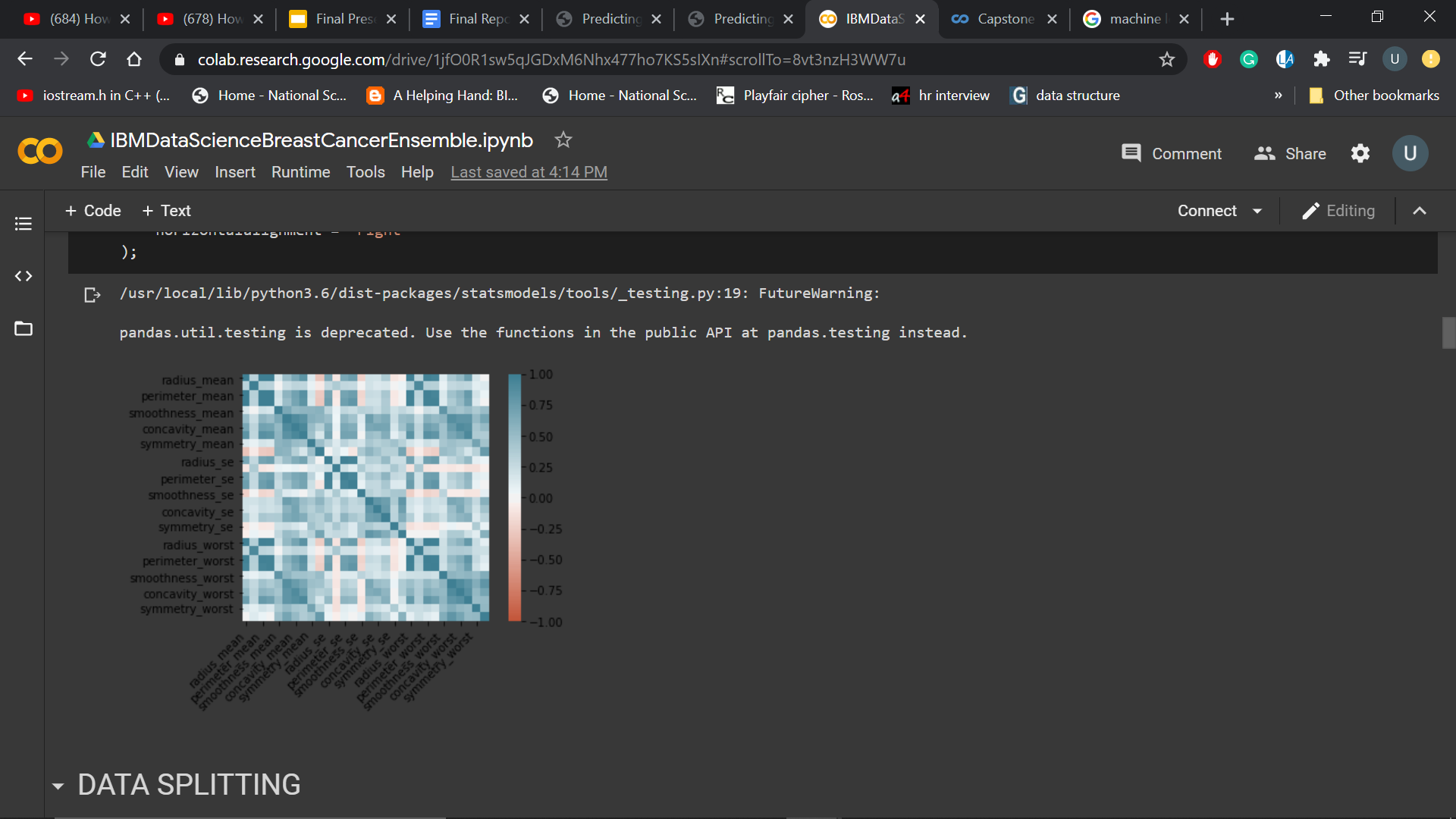
## ENTROPY

Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. Entropy is a measure of disorder or uncertainty and the goal of machine learning models. The higher the entropy more the information content.

## GINI INDEX

The Gini Index is calculated by subtracting the sum of the squared probabilities of each class from one. It favors larger partitions. Information Gain multiplies the probability of the class times the log (base=2) of that class probability. Information Gain favors smaller partitions with many distinct values.

## CONFUSION MATRIX



A **confusion matrix** is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.

Statistically, a confusion matrix is an N X N matrix, where N is the number of classes being predicted.

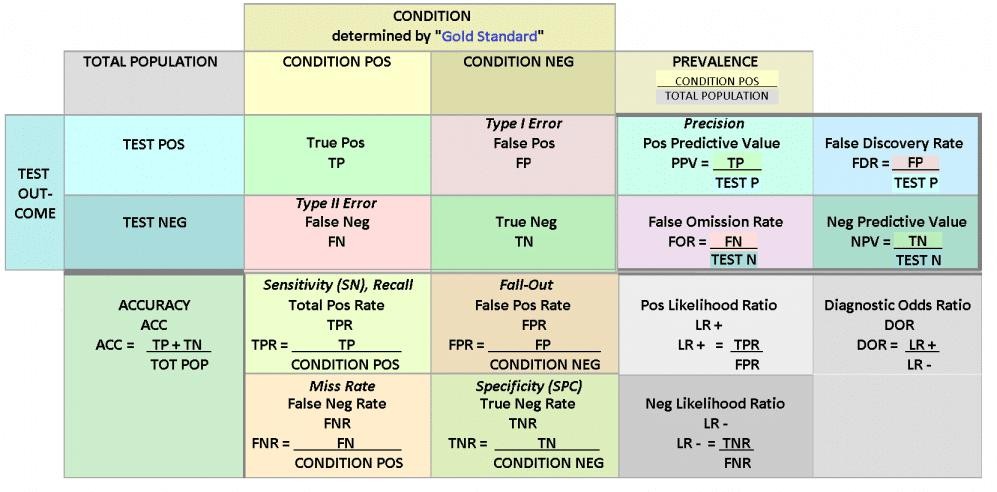
**Accuracy**: the proportion of the total number of predictions that were correct.

**Positive Predictive Value or Precision**: the proportion of positive cases that were correctly identified.

**Negative Predictive Value**: the proportion of negative cases that were correctly identified.

**Sensitivity or Recall**: the proportion of actual positive cases which are correctly identified.

**Specificity**: the proportion of actual negative cases which are correctly identified.



## F1-SCORE

In statistical analysis of binary classification, the F1 score (also F-score or F-measure) is a measure of a test's accuracy. The traditional F-measure or balanced F-score (F1 score) is the harmonic mean of precision and recall:

***F1= 2/ (recall-1 + precision-1) F1= 2\*(recall.precision) (recall + precision)***

F1 score conveys the balance between the precision and the recall.

## ROC CURVE

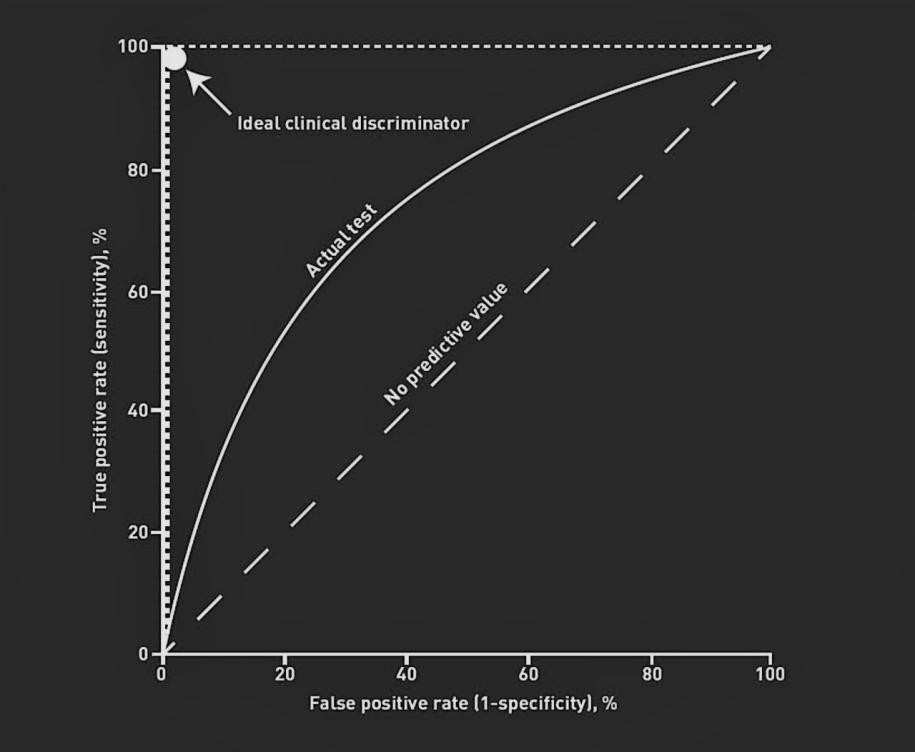
A useful tool when predicting the probability of a binary outcome is the Receiver Operating Characteristic curve, or ROC curve (Fig 2).

It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0.

The true positive rate is calculated as the number of true positives divided by the sum of the number of true positives and the number of false negatives. It describes how good the model is at predicting the positive class when the actual outcome is positive.

The ROC curve is a useful tool for a few reasons:

1. The curves of different models can be compared directly in general or for different thresholds.
2. The area under the curve (AUC) can be used as a summary of the model skill.



## ACTIVATION FUNCTIONS

These functions comes into play while playing with Network and hold its importance by imparting non-linearity to the model which would otherwise have been a Polynomial Regression Model with linear dependent relationship between feature variables and target variables.

With the functionality of being a complex computational, the model learn better through non-linearity also eligible to comprise and incorporate any non-linear functionality into the network of artificial neurons (Universal Function Approximator) and deal effectively with high-dimensional data. Activation functions are applied after evaluating the equation of hypothesis *h(x)* at each operational node.

***yi = Ø(h(x)) yi = Ø(w\*x + b)***

{ *yi = output at ith node Ø() = activation function*

*w = weight matrix (variable node) x = input matrix (constant node) b = bias (variable node) }*

1. **Sigmoid Function:**

#### *Ø(x) = 1 ; 0 < Ø(x)< 1*

***1 + e-ax***

*{ Ø() = activation function a = constant (usually =1)*

*x = input matrix(constant node) }*

Sigmoid functions is an easy and simple interpretable function but sticks to Vanishing Gradient problem due to slower convergence and a non-zero (0.5) centric function, resulting in harder optimization. The Sigmoid Formula is shown above.

### Hyperbolic Tangent (tanh(x)):

Formula for Hyperbolic Tangent is shown below:

#### *Ø(x) = tanh(x)* ; *0 < tanh(x) < 1*

*{ Ø() = activation function*

*x = input matrix(constant node) }*

A better performance is shown by Tan(h) function which ranges from -1 to 1 thus, making an easy optimization and zero centric function but still encounters Vanishing Gradient problem.

### Rectified Linear Unit (ReLu):

As the network goes through backpropagation while dealing with a large data set and using a complex computational function for mapping the hidden layers, the gradient diminishes to update the parameters as we move closer to the input as the error becomes smaller and smaller resulting in slower rate of convergence.

In a such a multi-layered network, ReLu stand out, resolving the Vanishing Gradient problem and thus, being the widely used activation.

Formula for ReLu function is represented below:

#### *Ø(x) = max(0,x) ; 0 < Ø(x) <* ∞

*{ Ø() = activation function*

*x = input matrix(constant node) }*

### Leaky ReLu:

This activation functions counters few drawbacks of ReLu which were of being limited to hidden layers and deactivating neurons or simply, resulting in dead neurons.

Leaky ReLu function is illustrated below:

#### *Ø(x) = max(0.1x,x) ; -*∞ *< Ø(x) <* ∞

*{ Ø() = activation function*

*x = input matrix(constant node) }*

### Randomized Leaky ReLu (RLReLu):

Randomized Leaky ReLu function is illustrated below:

#### *Ø(x) = max(0,x) + α\*min(0,x)) ; -*∞ *< Ø(x) <* ∞

*{ Ø() = activation function*

*x = input matrix(constant node) α = randomized variable }*

### Softmax

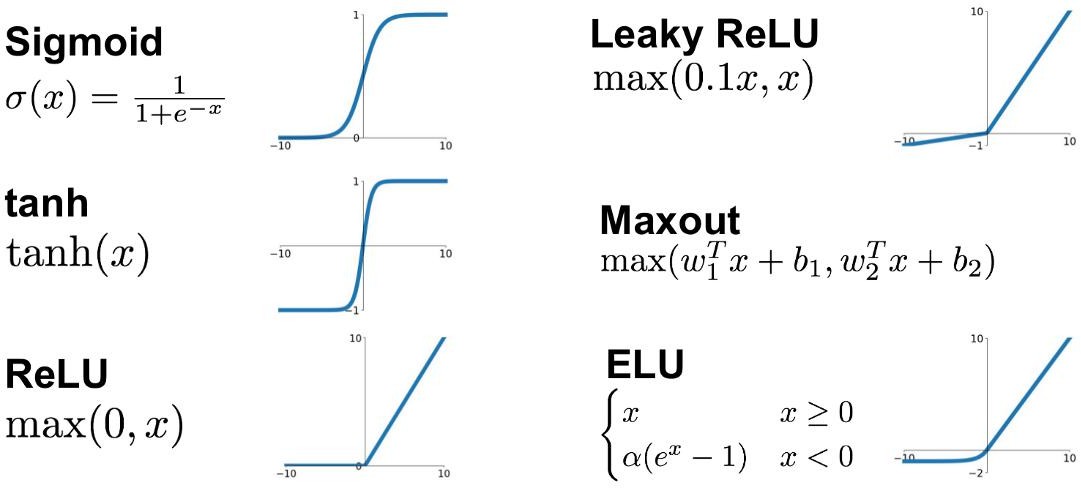
Softmax Function (usually at last layer) is used to find the probability of ‘n’ unique events to determine the class of the input. The outcome is based on the probability distribution ranging from 0 to 1 and determining the probability of each class with target class holding the maximum probability, comparatively in a multi-class classification problem.

Softmax Function formula is as shown:

##### *Ø(xi) = xi / ( kΣj=1 xj )*

*{ Ø(n) = activation function*

*x = input matrix(constant node) }*



**RESULTS AND CONCLUSION**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **PRECISION** | **RECALL** | **F1-SCORE** | **SUPPORT** | **ACCURACY** |
| **SVM** | **1.00** | **0.99** | **0.99** | **75** | **99.1228%** |
| **NAIVE BAYES** | **0.59** | **0.66** | **0.62** | **67** | 90.3509% |
| **DECISION TREE** | **0.59** | **0.66** | **0.62** | **67** | 92.9825% |
| **DT BAGGING** | **0.59** | **0.66** | **0.62** | **67** | 92.9825% |
| **EXTRA TREE** | **0.59** | **0.66** | **0.62** | **67** | 92.9825% |
| **RANDOM FOREST** | **0.59** | **0.66** | **0.62** | **67** | 97.3684% |
| **GRADIENT**  **DESCENT** | **0.59** | **0.66** | **0.62** | **67** | 97.3684% |
| **ADABOOST** | **0.97** | **0.99** | **0.98** | **75** | 97.3684% |
| **XGBOOST** | **0.97** | **1.00** | **0.99** | **75** | **98.2456%** |
| **ANN** | **0.96** | **1.00** | **0.98** | **75** | 97.3684% |
| **DEEP LEARNING** | **0.96** | **1.00** | **0.98** | **75** | 97.3684% |

***Table 1: Performance Matrix based on Breast Cancer Classification Under Various Classification Algorithms***

From the results (Table 1) obtained using various Machine Learning Classification Algorithms, it is observed that SVM gives the maximum accuracy of 99.1228% with 1.00 precision and XGBOOST Classifier with 98.2456% accuracy and 1.00 recall. Then, comes ADABOOST classifier, Artificial Neural Network (Layer 1 : 30 units, Layer 2: 15 units, Layer 3: 1 unit , Batch size: 30, Epoch: 1000) and Deep Neural Network (Layer 1 : 455

units, Layer 2: 200 units, Layer 3: 100 units, Layer 4: 50 units, Layer 5: 1 unit, Batch size: 30, Epoch: 1000) with 97.3684% of accuracy. Thus, it is clearly evident that Support Vector Classifier works the best for classification under conditions provided.

In this report, I have illustrated the performance of tabulated classification algorithm using various evaluations metrics. Understanding how well a machine learning model is going to perform on unseen data is the ultimate purpose behind working with these evaluation metrics. Metrics like accuracy, precision, recall are good ways to evaluate classification models for balanced data sets, but if the data is imbalanced and there’s class disparity, then other methods like ROC/AUC perform better in evaluating the model performance.

As we’ve seen, the ROC curve isn’t just a single number; it’s a whole curve. It provides nuanced details about the behavior of the classifier, but it’s also hard to quickly compare many ROC curves to each other. The AUC is one way to summarize the ROC curve into a

single number so that it can be compared easily and automatically. A good ROC curve has a lot of space under it (because the true positive rate shoots up to 100% very quickly). A bad ROC curve covers very little area. So high AUC is good, and low AUC is not so good.

single number so that it can be compared easily and automatically. A good ROC curve has a lot of space under it (because the true positive rate shoots up to 100% very quickly). A bad ROC curve covers very little area. So high AUC is good, and low AUC is not so good.

One last important point to keep in mind, since we focused largely on classification tasks in this guide: Any machine learning model should be optimized and evaluated according to the task it’s built to address.