Machine Learning Techniques in Bioinformatics

In the post-genome era, research in bioinformatics has been overwhelmed by the experimental data. Due to continued research, there is a continuous growth in the amount of biological data available and current biological databases are populated by vast experimental data. The exponential growth of these data available, raises two problems: on one hand, efficient information storage and management and, on the other hand, the extraction of useful information from these data. The second problem is one of the main challenges in computational biology, which requires the development of tools and methods capable of transforming all these heterogeneous data into biological knowledge about the underlying mechanism. The complexity of biological data ranges from simple strings (nucleotides and amino acids sequences) to complex graphs (biochemical networks); from 1-Dimentional (sequence data) to 3-Dimentional (protein and RNA structures). Considering the amount and complexity of the data, it is becoming impossible for an expert to compute and compare the entries within the current databases. Machine learning techniques are increasingly being used to address problems in computational biology and bioinformatics. Novel computational techniques to analyze high throughput data in the form of sequences, gene and protein expressions, pathways, and images are becoming vital for understanding diseases and future drug discovery. Machine learning techniques such as Markov models, support vector machines, neural networks, and graphical models have been successful in analyzing life science data because of their capabilities in handling randomness and uncertainty of data noise and in generalization. Thus, machine learning and artificial intelligence techniques have been widely applied in this domain to discover and mine the knowledge in the databases and has indeed gained a lot of success in this research area. At present, with various learning algorithms available in the literature, researchers are facing difficulties in choosing the best method that can apply to their data. Machine Learning in Bioinformatics is an indispensable resource for computer scientists, engineers, biologists, mathematicians, researchers, clinicians, physicians, and medical informaticists.

Machine learning is by and large a direct descendant of an older discipline, statistical model fitting. Like its predecessor, the goal in machine learning is to extract useful information from a corpus of data by building good probabilistic models. Machine learning consists in programming computers to optimize a performance criterion by using example data or past experience. The optimized criterion can be the accuracy provided by a predictive model—in a modelling problem, and the value of a fitness or evaluation function—in an optimization problem. In a modelling problem, the ‘learning’ term refers to running a computer program to induce a model by using training data or past experience. Machine learning uses statistical theory when building computational models since the objective is to make inferences from a sample. The two main steps in this process are to induce the model by processing the huge amount of data and to represent the model and making inferences efficiently.

A machine learning algorithm is one that can learn from experience (observed examples) with respect to some class of tasks and a performance measure. These are suitable for molecular biology data due to the learning algorithm’s ability to construct classifiers/hypotheses that can explain complex relationships in the data. The classifiers or hypotheses can then be interpreted by a domain expert who suggests some wet-lab experiments to validate or refute the hypotheses. This feedback loop between in silico and
in vivo / in vitro experiments accelerates the knowledge discovery process over the biological data. This feedback is an important characteristic of machine learning in bioinformatics. Generally, there are two types of learning schemes in machine learning:

- **Supervised Learning** where the output has been given a priori labelled or the learner has some prior knowledge of the data; and

- **Unsupervised Learning**, where no prior information is given to the learner regarding the data or the output.

**Supervised Classification**

In a classification problem, set of elements are divided into classes. Given an element (or instance) of the set, a class is assigned according to some of the element’s features and a set of classification rules. In many real-life situations, this set of rules is not known, and the only information available is a set of labelled examples (i.e. a set of instances associated with a class). Supervised classification paradigms are algorithms that induce the classification rules from the data.

For example, in order to tackle splice site prediction as a supervised classification problem, the instances to be classified would be DNA sequences of a given size. The attributes of a given instance would be the nucleotide at each position in the sequence. Assuming that we are looking for donor sites, so the possible values for the class would be true donor site or false donor site. Since, it is the case of supervised classification; set of labelled examples, i.e. a set of sequences of true and false donor sites along with their label is required. At this point, this training set may be used to build up a classifier. Once the classifier has been trained, it may be used to label new sequences, using the nucleotide present at each position as an input to the classifier and getting the assigned label (true or false donor site) as an output.

In two-group supervised classification, there is a feature vector \( X \in \mathbb{R}^n \) whose components are called *predictor variables* and a label or class variable \( C \in \{0,1\} \). Hence, the task is to induce classifiers from *training data*, which consists of a set of \( N \) independent observations \( D_N = \{ (x^{(1)}, c^{(1)}), ..., (x^{(N)}, c^{(N)}) \} \) drawn from the joint probability distribution \( p(x, c) \) as shown in Table 1. The classification model will be used to assign labels to new instances according to the value of its predictor variables.

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( \ldots )</th>
<th>( X_n )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (x^{(1)}, c^{(1)}) )</td>
<td>( x^{(1)}_1 )</td>
<td>( \ldots )</td>
<td>( x^{(1)}_n )</td>
</tr>
<tr>
<td>( (x^{(2)}, c^{(2)}) )</td>
<td>( x^{(2)}_1 )</td>
<td>( \ldots )</td>
<td>( x^{(2)}_n )</td>
</tr>
<tr>
<td>( (x^{(N)}, c^{(N)}) )</td>
<td>( x^{(N)}_1 )</td>
<td>( \ldots )</td>
<td>( x^{(N)}_n )</td>
</tr>
<tr>
<td>( x^{(N+1)} )</td>
<td>( x^{(N+1)}_1 )</td>
<td>( \ldots )</td>
<td>( x^{(N+1)}_n )</td>
</tr>
</tbody>
</table>

**Table 1.** Raw data in a supervised classification problem.

Supervised classification technique is based on the principles of machine learning techniques in which of parameters of inferring a function is estimated based on training data such that a set of input vector, which consists of realized values of explanatory factors
Machine Learning Techniques in Bioinformatics

is being used to get desired output values of dependant factors with desired accuracy. This function is also called as classifier. This inferred function is expected to predict correct output value for any valid input vector. This means, it requires the learning algorithm to generalize from the training data to unseen situations with desired accuracy. In order to develop reliable inferred function following steps needs to be followed:

- Selection of appropriate training data set which is to be representative of real world of the problem under consideration along with representative sets of output values.
- Selection of input features (factors) which can able to predict the output with desired accuracy but should not be too large in numbers.
- Determination of structure of the function and corresponding learning algorithms based on optimized performance through cross validation techniques on a sub-set of training data set which is also known as validation set. Certain control parameters are used for this purpose.
- Evaluation of the accuracy of the learned function after parameter adjustments on a test data set which is different from the training set.

Large numbers of supervised learning algorithms are available in literature with their advantages and disadvantages but there is no single algorithms which can be used on all types of data sets. There are four major issues which needs special consideration in supervised learning:

**Tradeoff between bias and variance:** The prediction error is sum of bias and variance of the learning algorithms. Generally it is desirable that a learning algorithm with low bias should be flexible such that it can fit the data set but it should not be that flexible that it fit differently to each training data set due to its high variance. Therefore, it is necessary to adjust this tradeoff between bias and variance.

**Availability of dataset and complexity of function:** In case, simple true function, learning algorithm with high bias and low variance will results reliable inferred function with the help of small amount of dataset. But in case of highly complex true function resulting from interactions within different components needs large amount of training dataset to build learning algorithm with low bias and high variance. Therefore, it is desirable for good learning algorithms to automatically adjust the bias/variance tradeoff based on the amount of data available and the apparent complexity of the function to be learned.

**Dimensions of input dataset:** Large dimension of the dataset may create confusion and it may become difficult learning problem even if the true function depends on only small number of features. This will results in large variance. Hence, high input dimensionality typically requires tuning the classifier to have low variance and high bias. It is always desirable to apply feature selection procedures or dimensionality reduction techniques to get desirable output.

**Noisy output values:** In case output values are incorrect beyond a limit due to response errors then the learning algorithm is expected to lead to undesirable inferred function. This is case where it is usually best to employ a high bias, low variance classifier.

The selection of learning algorithms depends on number of other factors such as (i) heterogeneity of the data, (ii) redundancy of data and (iii) linear and non-linear
relationships among the factors etc. In case the input feature dataset is heterogeneous such as discrete, discrete ordered, counts, continuous values then decision tree based methods are easy to apply whereas learning algorithms, including support vector machines, linear regression, logistic regression, neural networks and nearest neighbor methods can be applied on numerical feature which are scaled to similar ranges (e.g., to the [-1,1] interval). If the input features contain redundant information in terms of multi-collinearity number of learning algorithms such as linear regression, logistic regression and distance based methods performs poorly because of numerical instabilities. The best way to solve this problem is through imposing regularization conditions. Again, if there are complex interactions among features then algorithms based on decision trees and neural networks work better due to their inherent capabilities to deal with this situation. The selection of the best algorithm can be done using cross validation techniques to the given dataset and problem at hand. Some of the most widely used learning algorithms are support vector machines, linear regression, logistic regression, naive Bayes, linear discriminant analysis, decision trees, k-nearest neighbor algorithm and Neural Networks (multilayer perception).

Classification and Prediction Techniques

Let the total sample size of the biological data set is “n” and there are “m” features on which data is available. Let, the data matrix of size m x n is represented by $X = (x_{ij})$ where, $x_{ij}$ represent data on $j$-th feature of $i$-th observation. Further, let $y = (y_1, y_2, \ldots, y_n)$ is a matrix of response variable which may be categorical in nature depending on number of classes for classification problem.

**Naive Bayes Classifier:** It is a simple probabilistic classifier based on Bayes theorem. It is also known as independent feature model due to its strong assumptions of independence. It based on the assumption that inclusion or exclusion of a particular feature in the model is independent of inclusion or exclusion of any other feature and their contributions towards probability is independent of each other. In many practical applications maximum likelihood method can be used in estimation of parameters of this model instead of Bayesian probability. Depending on the probability model it can be trained efficiently using supervised learning techniques. It has been found in literature that, in spite of its oversimplified assumptions it out performed number of current approaches such as boosted trees or random forest under specific situations. One of the major advantages of this classifier is that it requires limited amount of training data set for estimation of parameters for classification. Due to the assumption of independence of variables only variances are required for estimation instead of full covariance matrix.

This classifier is best suited when there is high dimensional feature data set. Let C represents a class. The probabilistic model for this class is conditional model i.e. $P(C|X_1, X_2, \ldots, X_m)$ over dependent response variable. From application of Bayes theorem we get:

$$P(C|X_1, X_2, \ldots, X_m) \propto P(C) P(X_1, X_2, \ldots, X_m / C)$$

The prior probability of $j$-th class is: $P(C=j) = \frac{\text{Number of class } j \text{ samples}}{\text{Total numbers of samples}}$
The likelihood function \( p(X_1, X_2, \ldots, X_m | C) \) can be written as \( \prod_{i=1}^{m} p(X_i | C) \) under the assumption of conditional independence of features. Now a new instance can be classified with maximum posterior probability obtained as:

\[
\text{Arg max}_{c_j \in C} P(c_j) \pi_c p(X_i | C_j)
\]

This technique was further extended by Demichelis et al. (2006).

**Logistic Regression:** Let \( Y \) denotes the levels of sub-functions within each function of the genes. The number of level may varies for each function. This is also known as the response variable, which is nominal in nature i.e. denoting any sub-function with any level does not change the interpretation and analysis technique. Now, the columns of \( X \) matrix represent statistics related to each gene which are explanatory variable for a given response i.e. sub-functions of a gene. Let there are \( K+1 \) possible response levels, then multinomial logistic regression model can be written as

\[
\log \left( \frac{P_r(Y=1/X)}{P_r(Y=K+1/X)} \right) = \beta_{o_i} + \beta_1 X_1, \; i=1,2,\ldots, K
\]

where, \( \beta_{o_i} \), \( i=1,2,\ldots,K \) are intercept parameter and \( \beta_1, \beta_2, \ldots, \beta_K \) are \( K \) vectors of \( G \times 1 \) dimension for slope parameters. This model can be fitted using method of maximum likelihood using either Fisher scoring algorithm or New-Raphson algorithm. The likelihood that \( g \)-th gene will have response level \( y_j \) can be obtained as

\[
P_r(Y=y_j | x_j) = \begin{cases} 
\frac{\exp(\beta_{o_1} + \beta_1 x_j)}{1 + \sum_{m=2}^{K} \exp(\beta_{o_m} + \beta_m x_j)}, & 1 \leq y_j = i \leq k \\
1 & y_j = k + 1 
\end{cases}
\]

The model fitting information for the reliability of estimated probability, following criterion may be calculated for \( j \)-th observation.

\[-2\log L = -2 \sum_{j} \frac{w_j}{f_j} \log P_r(Y=y_j|x_j)\]

where, \( w_j \) and \( f_j \) are weight and frequency of \( j \)-th observation and \( \sigma^2 \) is the dispersion parameter. Further, Akaike Information Criterion (AIC) can also be obtained as

\[\text{AIC} = -2\log L + 2p\]

where, \( p \) is the number of parameters in the model. Cox and Snell (1989) proposed following coefficient of determinant for model fitting information

\[R^2 = 1 - \frac{L(O)}{L(\hat{\beta})}^{2/n}\]
where \( L(O) \) is the likelihood of the interrupt model only and \( L(\beta) \) is the likelihood of the specified model for sample size \( n \). The maximum value of \( R^2 \) is \( R^2_{\text{max}} = 1 - \frac{L(O)}{L(\beta)} \). The adjusted coefficient of determinant (Magelkerke 1991) can be written as

\[
R^2_{\text{adj}} = \frac{R^2}{R^2_{\text{max}}}
\]

The values of \( R^2 \) and \( R^2_{\text{adj}} \) are found to be the best criterion for indicating better models while fitting. The functional prediction re-substitutions accuracy has been also estimated from misclassification errors matrices. Further, sensitivity and specificity has also calculated.

**Linear Discriminant Analysis (LDA):** The Linear Discriminant Analysis (LDA) or Fisher’s linear discriminant is a techniques which is based on the principles of finding weighted linear combination of the features in such a way that it should be able to characterize or distinguish two or more than two classes of objects or events. It can also be used as dimensionality reduction techniques in case of high dimensional datasets. It has close relationship with ANOVA or regression analysis technique which also expresses dependent variable as a linear combination of features. However, in case of LDA dependent variable is categorical instead of continuous. Logistic regression and probit regression are more similar to LDA, as they also explain a categorical variable. These methods are preferable in applications where it is not reasonable to assume that the independent variables are normally distributed, which is a fundamental assumption of the LDA method. Further, LDA is also related to Principal Component Analysis (PCA) and factor analysis in which linear combination of the features is being generated in such a way that explains maximum variability of the data. PCA does not take in to account the any differences of the class whereas factor analysis is based on the principles of taking in to account of differences and ignoring similarities. Factor analysis does not distinguish between dependent and independent factors but in case of LDA this distinction is very much clear. LDA works when the measurements made on independent features when they are continuous quantities. However, in case of categorical independent features its equivalent technique discriminant correspondence analysis can be applied.

This technique is based upon the principle of separation of classes through maximization of between class seperability and minimization of with-in-class variability. The most common form of LDA is Fisher’s LDA (FLDA) or Diagonal LDA (DLDA) which are based on diagonal co-variance matrix. Any new observation can be classified based on the distance of each group mean vector using Mahalanobis distance. Let the \( \mu \) represents the mean vector and \( \mathbf{V} \) is Covariance matrix for a feature set \( \mathbf{X} \), then Mahalanobis distance can be defined as

\[
D = \sqrt{(\mathbf{X} - \mu) \mathbf{V}^{-1} (\mathbf{X} - \mu)}
\]

DLDA ignores correlation among features. It was shown by Dudoit et al. (2002) that generalized performance of DLDA is better than any other discriminant analysis techniques. The initial feature selection exercise improves the performance of DLDA. The major assumption of LDA is about normality of data, whereas, there is no such assumption in case of logistic regression technique.
**K-Nearest-Neighbor classifiers (K-NN):** This classifier is based on the distances among closest K neighbors to a particular unit. If, sample i and j are represented by \( \mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im}) \) and \( \mathbf{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jm}) \) then distances can be calculated based on Euclidean distance technique on standardized data set. In order to classify new observation, majority vote among K - neighbors is being considered for its classification in a particular class. It has been observed that as K increases, the variance of the classifier decreases but its bias increases. It has been shown by Friedman 1997; Dudoit et al. (2002), that this classifier is highly consistent. Best results can be obtained when feature selection step is employed before application of the classifier.

**Artificial Neural Networks:** This technique was proposal by Barnard Widrow (1950). This is a data driven and non-parametric model based approach. In this, a network of nodes (neurons) is being generated through assigning different weights. This includes may be obtained through both supervised and unsupervised learning. The basic principle of learning in this case is modifications in synaptic weights which are determinant through learning algorithm. For example in correlation learning rule; weights are adjusted according to Hebb’s rule. \( \Delta(W_{ij}) = O_i O_j \), where \( \Delta W_{ij} \) is change in weights of i-th node, which is connected to j-th node and \( O_i \) is output of i-th unit. In case of learning rule via error correction, weights are adjusted by minimizing output errors with respect to weights i.e. \( \Delta(W_{ij}) = E(O_i O_j) \). This algorithm is being used in Perceptron, MADALALINE and back propagation models. This technique is capable of solving number of complex problems but it is complex and computationally extensive.

**Support Vector Machine (SVM):** This technique was introduced by Vapnik (1995). This technique is based on finding linear hyperplanes in input space and kernel space for avoiding over fitting. Let training sample data consist of n pairs \( (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \) with \( x_i \in \mathbb{R}^p \) and \( y_i \in \{-1, 1\} \) then SVM classifier finds hyperplane \( (P_0) \) bisecting closest points of the data which is linearly separable. The \( P_0 \) is defined as

\[
\{\mathbf{x}: f(\mathbf{x}) = \mathbf{x}^T \mathbf{\beta} + \beta_0 = 0\} \text{ and } \| \mathbf{\beta} \| = 1
\]

Classifier creates a parallel hyperplane \( P_1 \) such defined as

\[
(P_1) \{\mathbf{x}: f(\mathbf{x}) = \mathbf{x}^T \mathbf{\beta} + \beta_0 = -1\}
\]

On a point in class -1 closet to \( P_0 \) and second hyperplane \( P_2 \) as

\[
(P_2) = \{\mathbf{x} : f(\mathbf{x}) = \mathbf{x}^T \mathbf{\beta} + \beta_0 = 1\} \text{ on a point in class closet to } P_0
\]

The optimum hyperplane for separating the data can be formed by maximizing the perpendicular distance between two parallel supporting planes \( P_1 \) and \( P_2 \) i.e. M. The resulting classifier can be given by

\[
\hat{y} = \text{sign} (\mathbf{x}^T \mathbf{\beta} + \beta_0)
\]

As we know that classes are separable. So, \( m = 2/\| \mathbf{\beta} \| \) there maximization of M leads to minimization of \( \| \mathbf{\beta} \| / 2 \)
Therefore, this problem can be reduced to minimization of

$$\mathcal{O}(\beta) = \| \beta \| / 2$$

Subject to \( y_i ( \mathbf{x}_i^T \beta + \beta_0 ) \geq 1 \) for all \( \{ (\mathbf{x}_i, y_i) \}, i = 1, 2, \ldots, n \}

In case data set is not separable then this technique maps the data into higher dimensional space where training set is separable via some transformation.

\( K : \mathbf{x} \rightarrow \Phi(\mathbf{x}) \).

A kernel function \( K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \) computes inner product in some expanded feature space. Linear or Gaussian kernels are widely used.

**Methods of Error Estimation**

Cross-validation is primarily a way of measuring the predictive performance of a statistical model. Every statistician knows that the model fit statistics are not a good guide to how well a model will predict: high \( R^2 \) does not necessarily mean a good model. It is easy to over-fit the data by including too many degrees of freedom and so inflate \( R^2 \) and other fit statistics. One way to measure the predictive ability of a model is to test it on a set of data not used in estimation. Data miners call this a “test set” and the data used for estimation is the “training set”.

**K-Fold Cross-validation**: Create a K-fold partition of the dataset. For each of the K experiments, use K-1 folds for training and the remaining for testing (as shown below).

<table>
<thead>
<tr>
<th>Experiment 1</th>
<th>Experiment 2</th>
<th>Experiment 3</th>
<th>Experiment 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Test examples</td>
</tr>
</tbody>
</table>

The advantage of K-Fold Cross validation is that all the examples in the dataset are eventually used for both training and testing. The true error is estimated as the average error rate.

**Leave-one-out Cross Validation**: Leave-one-out is the degenerate case of K-Fold Cross Validation, where K is chosen as the total number of examples. For a dataset with N examples, perform N experiments. For each experiment use N-1 examples for training and the remaining example for testing. As usual, the true error is estimated as the average error rate on test examples.
Ordinary Bootstrap: Ordinary Bootstrap method given by Efron (1998) has the problem that the learning and test sets overlap. In this method, bootstrap samples of size $n$ are repeatedly drawn from the original data $x$ by simple random sampling with replacement. In this, a prediction rule is built on a bootstrap sample and tested on the original sample, averaging the misclassification rates across all bootstrap replications gives the ordinary bootstrap estimate. This method seriously underestimates the prediction error since a subset of data is used both in building and in assessing the prediction model.

Bootstrap Cross-Validation: This method is proposed by Fu et. al. (2005) to handle small sample problems. The procedure generates $B$ bootstrap samples of size $n$ from the observed sample and then calculates a leave-one-out cross-validation estimate on each bootstrap sample. Averaging the $B$ cross-validation estimates gives the bootstrap cross-validation estimate for the prediction error. The paper of Fu et. al. (2005) did not carefully address the issue of feature selection. When the method is applied to high dimensional gene expression data, it is to be noted that feature selection must be conducted in this method on every leave-one-out learning set derived from every bootstrap sample. Since an original observation can appear more than once in a bootstrap sample, a leave-one-out learning set may overlap with the left out item when the cross-validation procedure is applied on a bootstrap sample. Consequently, the bootstrap cross-validation method tends to underestimate the true prediction error.

MEASURES OF PERFORMANCE

Sensitivity, specificity, and accuracy are the terms which are most commonly associated with a binary classification test and these statistically measure the performance of the test. In a binary classification, a given data set is divided into two categories on the basis of whether they have common properties or not by identifying their significance (error calculations are based on the confusion matrix (Table 2)).

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>TP</td>
</tr>
<tr>
<td></td>
<td>FP</td>
</tr>
<tr>
<td>Negative</td>
<td>FN</td>
</tr>
</tbody>
</table>
|           | NPV    | Specificity

Table 2. Confusion matrix in a two class problem

Sensitivity and specificity are statistical measures of the performance of a binary classification test, also known in statistics as classification function. Sensitivity measures the proportion of actual positives which are correctly identified as such (e.g. the percentage of sick people who are correctly identified as having the condition). Specificity measures
the proportion of negatives which are correctly identified (e.g. the percentage of healthy people who are correctly identified as not having the condition).

For example, if we test a group of people for the presence of a disease. Some of these people have the disease, and our test says they are positive, then these are called **true positives** (TP). Some have the disease, but the test claims they don't. They are called **false negatives** (FN). Some don't have the disease, but the test says they don't. These are **true negatives** (TN). Finally, we might have healthy people who have a positive test result which is categorised as **false positives** (FP). Thus, the number of true positives, false negatives, true negatives, and false positives add up to 100% of the set.

**Specificity** is the proportion of people that tested negative (TN) of all the people that actually are negative (TN+FP). As with sensitivity, it can be looked at as the probability that the test result is negative given that the patient is not sick. With higher specificity, fewer healthy people are labeled as sick.

\[
\text{Specificity} = \frac{TN}{TN + FP} \times 100
\]

**Sensitivity** is the proportion of people that tested positive (TP) of all the people that actually are positive (TP+FN). It can be seen as the probability that the test is positive given that the patient is sick. With higher sensitivity, fewer actual cases of disease go undetected.

\[
\text{Sensitivity} = \frac{TP}{TP + FN} \times 100
\]

The relationship between sensitivity and specificity, as well as the performance of the classifier, can be visualized and studied using the ROC curve. In addition to sensitivity and specificity, the performance of a binary classification test can be measured with positive predictive values (PPV) and negative predictive values (NPV). The positive prediction value answers the question "If the test result is positive, how well does that predict an actual presence of disease?". It is calculated as \((\text{true positives}) / (\text{true positives} + \text{false positives})\); that is, it is the proportion of true positives out of all positive results. The negative prediction value is the same, but for negatives.

\[
\text{PPV} = \frac{TP}{TP + FP} \times 100
\]

\[
\text{NPV} = \frac{TN}{TP + FN} \times 100
\]

\[
\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \times 100
\]

The area under the receiver operating characteristic curve is used as a performance measure for machine learning algorithms.

Hence, a good binary classification test always results with high values for all the three factors, sensitivity, specificity and accuracy, whereas a poor binary classification test results with low values for all. If sensitivity is high and specificity is low then, there is no need to bother about the positive candidates but the negative candidates must be reexamined to eliminate false positives (negative candidates mistakenly selected). But if sensitivity is low and specificity is high then, there is no need to bother about the negative candidates but the positive candidates must be reexamined to eliminate false negatives (positive candidates mistakenly rejected). An average binary classification test always results with average values which are almost similar for all the three factors.