Predictive Modeling using Supervised Machine Learning approach

Ravindra Kumar, Megha Kumar, Medhavi Pandey
Assistant Professor, Computer Science & Engineering Department, Delhi Technical Campus

Abstract

At the intersection between artificial intelligence and statistics, supervised learning allows algorithms to automatically build predictive models from just observations of a system. During the last twenty years, supervised learning has been a tool of choice to analyze the increasing and complexifying data generated in the context of molecular biology, with successful applications in genome annotation, function prediction, or biomarker discovery. Among supervised learning there are various methods stand out for linear and non-linear classification of data. The goal of this paper is to provide an accessible and comprehensive introduction of the workflow of the supervised learning . The first part of the review is devoted to an intuitive description of various kind of machine learning methods. The second part of the review provides the workflow and different compositions of supervised learning.

1. Machine Learning and Pattern Classification

Predictive modeling is the general concept of building a model that is capable of making predictions. Typically, such a model includes a machine learning algorithm that learns certain properties from a training dataset in order to make those predictions. Predictive modeling can be divided further into two sub areas: Regression and pattern classification. Regression models are based on the analysis of relationships between variables and trends in order to make predictions about continuous variables, e.g., the prediction of the maximum temperature for the upcoming days in weather forecasting. In contrast to regression models, the task of pattern classification is to assign discrete class labels to particular observations as outcomes of a prediction. To go back to the above example: A pattern classification task in weather forecasting could be the prediction of a sunny, rainy, or snowy day [1].

2. Supervised, unsupervised, and reinforcement learning workflow

2.1 Supervised learning

Supervised learning is the Data mining task of inferring a function from **labeled training data**. The training data consist of a set of training examples.

In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the (**supervisory signal**)[2].

Classification of patterns tasks can be grouped into two main sub-categories: Supervised and unsupervised learning. In supervised learning, the class labels in the dataset, which is used to build the classification model, are known. For example, a dataset for a collections of spam filtering can be composed of different types of messages which can be comprise of the "ham" (=not spam) messages. In a supervised learning problem, we would know which message in the training set is spam or ham, and we'd use this information to train our model in order to classify new unseen messages [3].

2.2 Unsupervised Learning Concept

In Data mining, the problem of unsupervised learning is that of trying to find hidden structure in unlabeled data. Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution[4].

In unsupervised learning task deal with unlabeled instances, and the classes have to be inferred from the unstructured dataset. Typically, unsupervised learning employs a clustering technique in order to group the unlabeled samples based on certain similarity (or distance) measures.

2.3 Reinforcement learning Concept

A third class of learning algorithms is described by the term "reinforcement learning". Here, the model is learned from a series of actions by maximizing a "reward function". The reward function can either be maximized by penalizing "bad actions" and/or rewarding "good actions". A popular example of reinforcement learning would be the training of self-driving car using feedback from the environment [5].

3. Workflow of Supervised Learning

As of today, the famous "Iris" flower dataset is probably one of the most commonly used examples when in comes to introducing various concepts in the field of "data science". The Iris dataset was created and used by R. A. Fisher in context of his discriminant analysis in 1936, and it is freely available at the UCI machine learning repository.

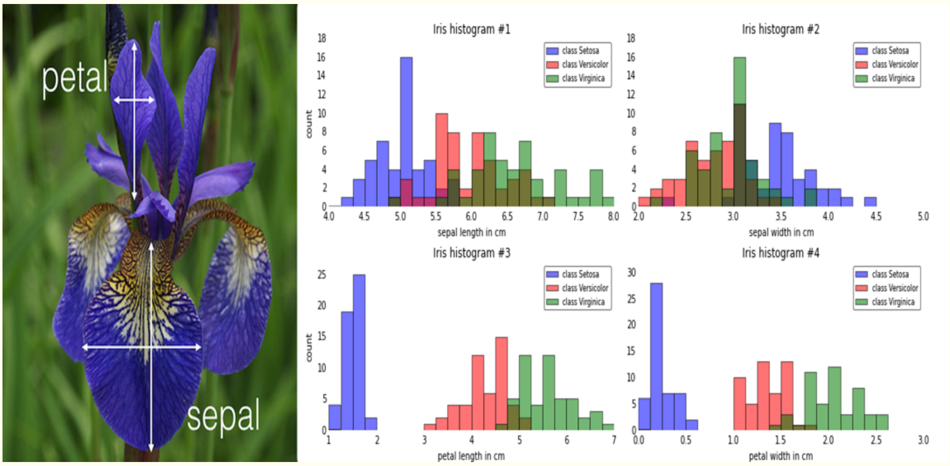


Figure 1: Iris Flower

Here, it serves as a perfect example of a supervised classification task, where the class labels are the three flower species: Setosa, Virginica, and Versicolor. And every of the 150 instances (individual flowers) consists of four features:

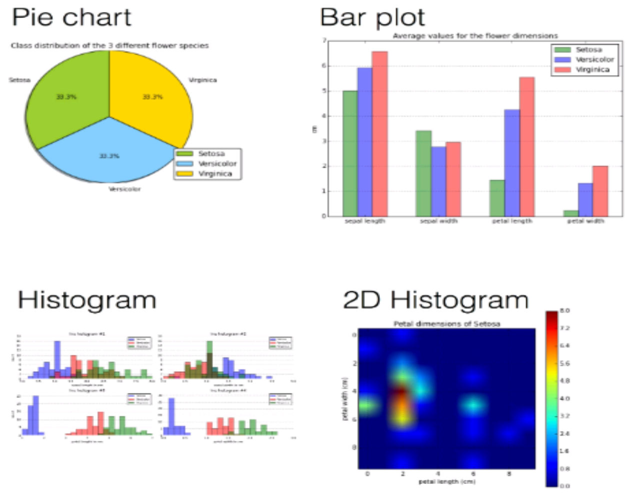
- Sepal width
- Sepal length
- Petal width
- Petal height

(all measured in centimeters)[6,7]

3.1 Visualization

When we are dealing with a new dataset, it is often useful to employ simple visualization techniques for explanatory data analysis, since the human eye is very powerful at discovering patterns. However, sometimes we have to deal with data that consists of more than three dimensions and cannot be captured in a single plot: One way to overcome such limitations could be to break down the attribute set into pairs and create a scatter plot matrix. In practice, the choice of a "good and useful" visualization technique highly depends on the type of data, the dimensionality of the feature space, and the question at hand[8].

Below are just a few examples of more or less useful visualizations of the Iris dataset.



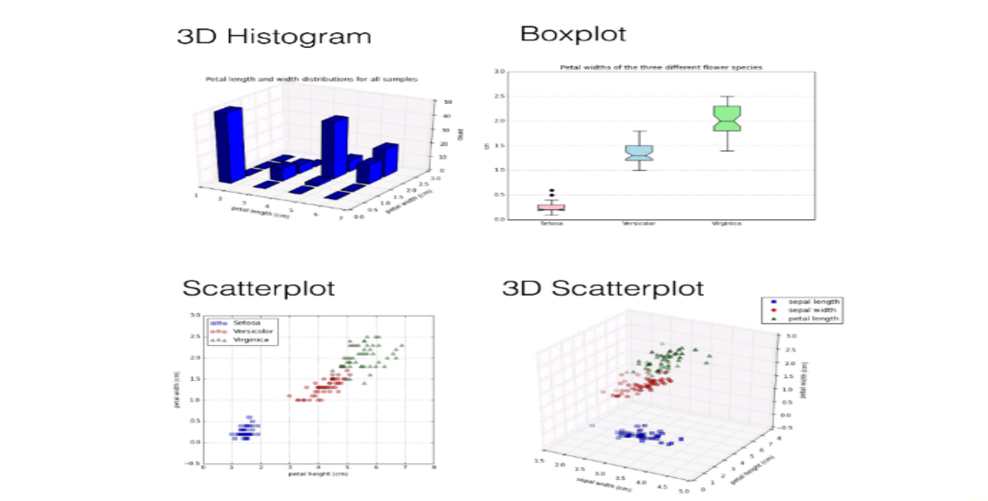


Figure 2: Graph Plotting

Looking at those plots above, the scatter plots and (1D) histograms in particular, we can already see that the petal dimensions contain more discriminatory information than the sepal widths and lengths based on the smaller overlap between the three different flower classes. This information could, for example, be used for feature selection in order to remove noise and reduce the size of our dataset.

3.2 Workflow diagram

In the following section, we will have a look at some of the main steps of a typical supervised learning task, and the diagram below should give us an intuitive understanding of how they are connected [9].

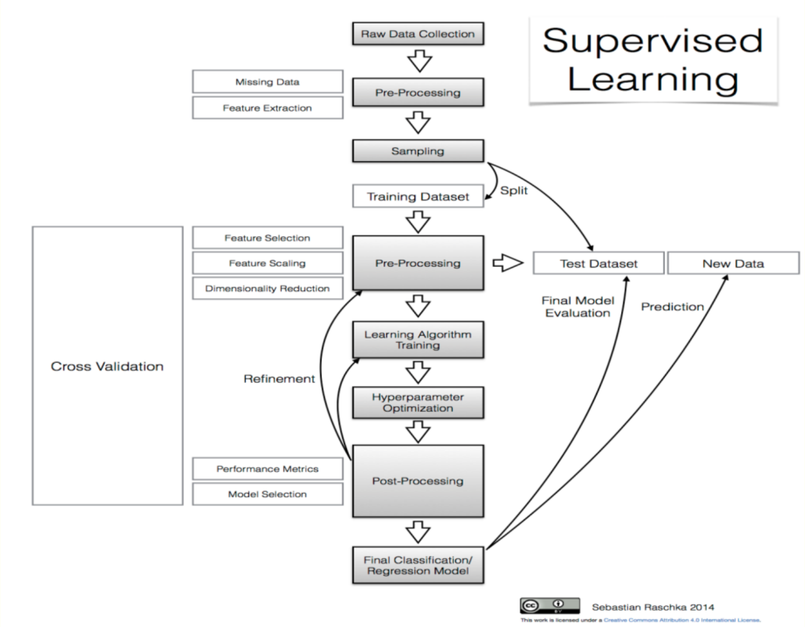


Figure 3: Workflow Diagram

3.3 Raw data collection and feature extraction

When we'd download the Iris dataset, we noticed that it is already in "good shape", and it seems that R. A. Fisher has already done some initial "pre-processing" for us: No missing data and numeric features that can be used by a learning algorithm.

However, let us assume that the raw data of the Iris dataset consisted of a series of images. In this case, a first pre-processing step (feature extraction) could involve the scaling, translation, and rotation of those images in order to obtain the dimensions of the sepals and petals in centimeters.

Occlusion of the leaves could be a problem that might lead to missing data: Many machine learning algorithms won't work correctly if data is missing in a dataset so that "ignoring" missing data might not be an option. If the sparsity (i.e., the amount of empty cells in the dataset) is not too high, it is often recommended to remove either the samples rows that contain missing values, or the attribute columns for which data is missing. Another strategy for dealing with missing data would be imputation: Replacement of missing values using certain statistics rather than complete removal. For categorical data, the missing value can be interpolated from the most frequent category, and the sample average can be used to interpolate missing values for numerical attributes. In general, resubstitution via k-nearest neighbor imputation is considered to be superior over resubstitution of missing data by the overall sample mean[10]

3.3 Sampling

Assuming that we extracted certain features (here: sepal widths, sepal lengths, petal widths, and petal lengths) from our raw data, we would now randomly split our dataset into a training and a test dataset.

The training dataset will be used to train the model, and the purpose of the test dataset is to evaluate the performance of the final model at the very end[11]

It is important that we use the test dataset only once in order to avoid overfitting when we compute the prediction-error metrics. Overfitting leads to classifiers that perform well on training data but do not generalize well so that the prediction-error on novel patterns is relatively high. Thus, techniques such as cross-validation are used in the model creation and refinement steps to evaluate the classification performance. An alternative strategy to re-use a test dataset for the model evaluation would be to create a third dataset, the so-called validation dataset[11].

3.4 Cross-Validation

Cross-validation is one of the most useful techniques to evaluate different combinations of feature selection, dimensionality reduction, and learning algorithms. There are multiple flavors of cross-validation, and the most common one would probably be k-fold cross-validation. In k-fold cross-validation, the original training dataset is split into *k* different subsets (the so-called "folds") where 1 fold is retained as test set, and the other k-1 folds are used for training the model. E.g., if we set *k* equal to 4 (i.e., 4 folds), 3 different subsets of the original training set would be used to train the model, and the 4th fold would be used for evaluation. After 4 iteration, we can eventually calculate the average error rate (and standard deviation) of the model, which gives us an idea of how well our model generalizes[12].

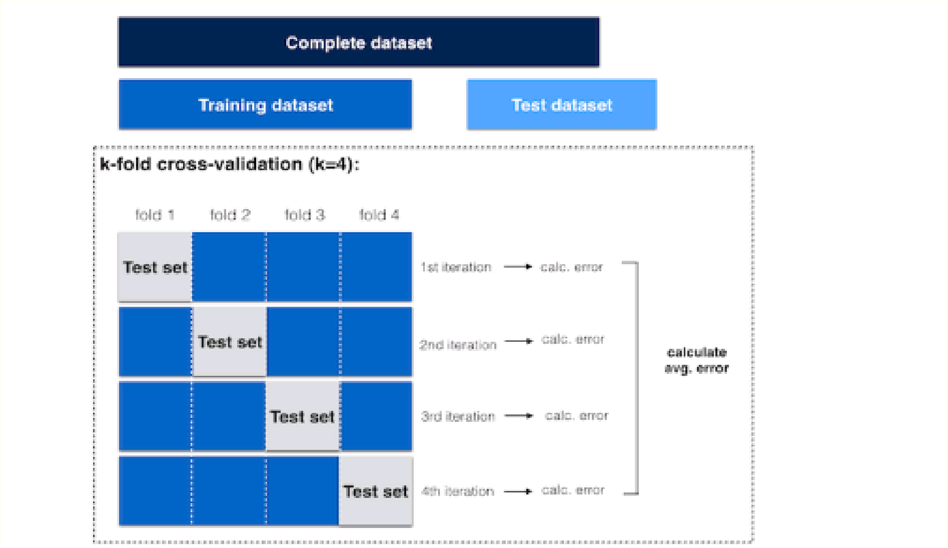

Complete dataset
Training dataset
Test dataset
k-fold cross-validation (k=4):
fold 1
fold 2
fold 3
fold 4
Test set
1st iteration → calc. error
Test set
2nd iteration → calc. error
Test set
3rd iteration → calc. error
Test set
4th iteration → calc. error
calculate avg. error

Figure 4: Cross Validation

3.5 Normalization

Normalization and other feature scaling techniques are often mandatory in order to make comparisons between different attributes (e.g., to compute distances or similarities in cluster analysis), especially, if the attributes were measured on different scales (e.g., temperatures in Kelvin and Celsius); proper scaling of features is a requirement for most machine learning algorithms.

The term “normalization” is often used synonymous to “Min-Max scaling”: The scaling of attributes in a certain range, e.g., 0 to 1.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Another common approach is the process of (z-score) “standardization” or “scaling to unit-variance”: Every sample is subtracted by the attribute’s mean and divided by the standard deviation so that the attribute will have the properties of a standard normal distribution (μ=0, σ=1).

$$z = \frac{x - \mu}{\sigma}$$

One important point that we have to keep in mind is that if we used any normalization or transformation technique on our training dataset, we’d have to use the same parameters on the test dataset and new unseen data[13].

3.6 Feature Selection and Dimensionality Reduction

Often, feature selection and dimensionality reduction are grouped together (like here in this article). While both methods are used for reducing the number of features in a dataset, there is an important difference. Feature selection is simply selecting and excluding given features **without changing** them. Dimensionality reduction **transforms** features into a lower dimension. Distinguishing between feature selection and dimensionality reduction might seem counter-intuitive at first, since feature selection will eventually lead (reduce dimensionality) to a smaller space. In practice, the key difference between the terms “feature selection” and “dimensionality reduction” is that in feature selection, we keep the “original feature axis”, whereas dimensionality reduction usually involves a transformation technique[14]. The main purpose of those two approaches is to remove noise, increase computational efficiency by retaining only “useful” (discriminatory) information, and to avoid overfitting (“curse of dimensionality”).

Commonly used dimensionality reduction techniques are linear transformations such as Principal Component Analyses (PCA) and Linear Discriminant Analysis (LDA). PCA can be described as an “unsupervised” algorithm, since it “ignores” class labels and its goal is to find the directions (the so-called principal components) that maximize the variance in a dataset. In contrast to PCA, LDA is “supervised” and computes the directions (“linear discriminants”) that will represent the axes that maximize the separation between multiple classes.

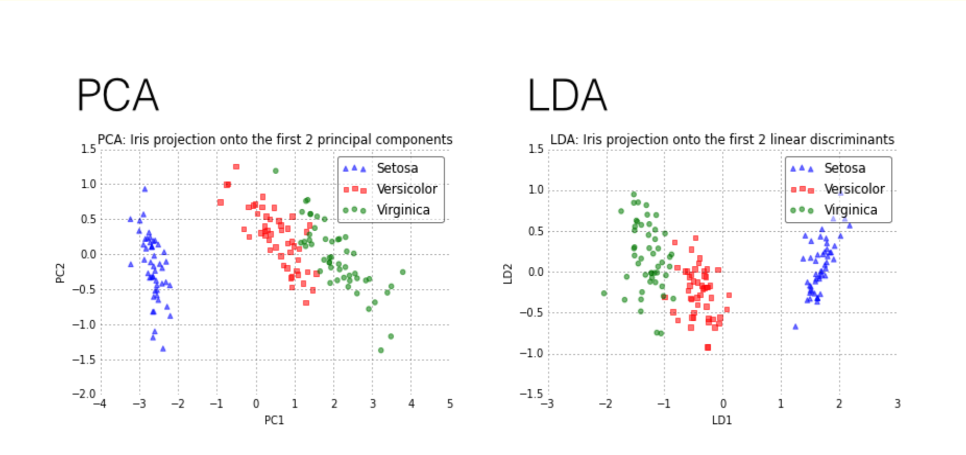


Figure 5 : Projection

The image below shows the iris data plotted on a 2-dimensional feature subspace after transformation via Linear Discriminant Analysis (LDA). The black lines denote exemplary, linear decision boundaries that divide the feature space into three decision regions (R1, R2, R3). Based on these decision regions, new observations can be classified among the three different flower species: R1 → Virginica, R2 → Versicolor, and R3 → Setosa.

3.7 Learning algorithms and hyperparameter tuning

There are a enormous number of different learning algorithms, and the details about the most popular ones are perfect topics for separate articles and applications. Here is just a very brief summary of four commonly used supervised learning algorithms [15]

Support Vector Machine (SVM) is a classification method that samples hyperplanes which separate between two or multiple classes. Eventually, the hyperplane with the highest margin is retained, where “margin” is defined as the minimum distance from sample points to the hyperplane. The sample point(s) that form margin are called support vectors and establish the final SVM model.

- **Bayes classifiers** are based on a statistical model (i.e., Bayes theorem: calculating posterior probabilities based on the prior probability and the so-called likelihood). A Naive Bayes classifier assumes that all attributes are conditionally independent, thereby, computing the likelihood is simplified to the product of the conditional probabilities of observing individual attributes given a particular class label.

- **Artificial Neural Networks (ANN)** are graph-like classifiers that mimic the structure of a human or animal "brain" where the interconnected nodes represent the neurons.

- **Decision tree classifiers** are tree like graphs, where nodes in the graph test certain conditions on a particular set of features, and branches split the decision towards the leaf nodes. Leaves represent lowest level in the graph and determine the class labels. Optimal tree are trained by minimizing Gini impurity, or maximizing information gain.

A very simple decision tree for the iris dataset could be drawn like this:

t

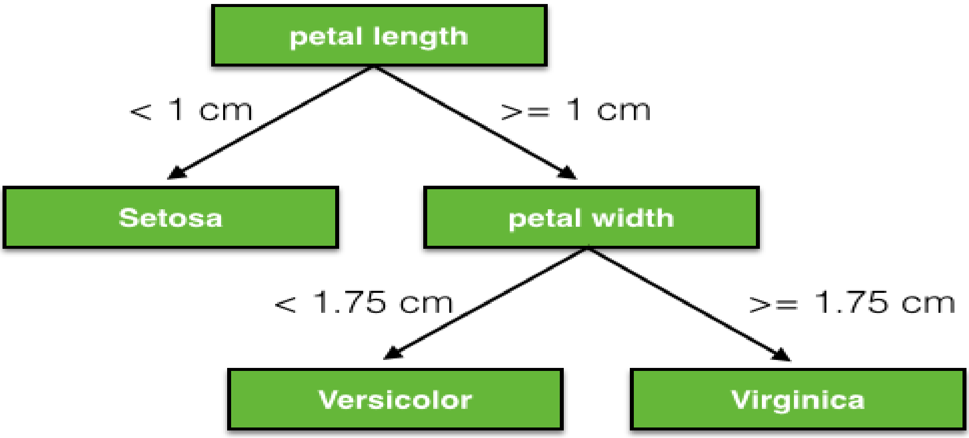
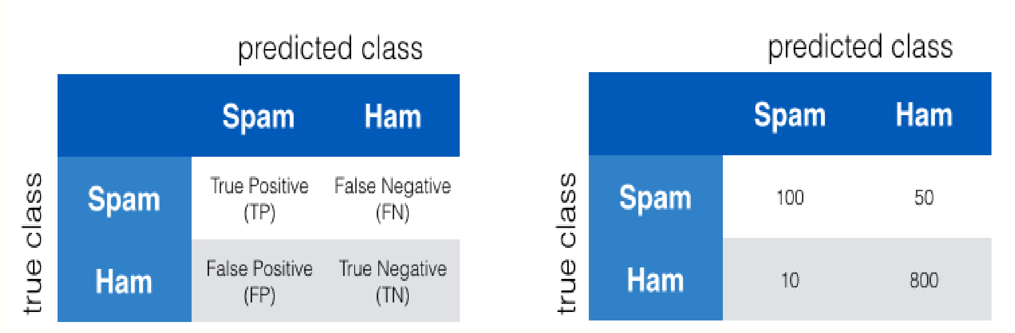


Figure 6 : Decision Tree

Hyper-parameters are the parameters of a classifier or estimator that are not directly learned in the machine learning step from the training data but are optimized separately. The goals of hyper-parameter optimization are to improve the performance of a classifier and to achieve good generalization of a learning algorithm. A popular method for hyper-parameter optimization is Grid Search. Typically, Grid Search is implemented as an exhaustive search (in contrast to randomized parameter optimization) of candidate parameter values. After all possible parameter combination for a model are evaluated, the best combination will be retained[16].

3.8 Prediction-error metrics and model selection

A convenient tool for performance evaluation is the so-called confusion matrix, which is a square matrix that consists of columns and rows that list the number of instances as “actual class” vs. “predicted class” ratios.A confusion matrix for a simple “spam vs. ham” classification could look like:



		predicted class	
		Spam	Ham
true class	Spam	True Positive (TP)	False Negative (FN)
	Ham	False Positive (FP)	True Negative (TN)

		predicted class	
		Spam	Ham
true class	Spam	100	50
	Ham	10	800

Figure 7 : Prediction Errors

Often, the prediction “accuracy” or “error” is used to report classification performance. Accuracy is defined as the fraction of correct classifications out of the total number of samples; it is often used synonymous to specificity/precision although it is calculated differently. Accuracy is calculated as :

$$\frac{TP+TN}{P+N}$$

where TP=True Positives, TN=True Negatives, P=Positives, N=Negatives.

The empirical error of a classification model can be calculated by 1-Accuracy.

However, the choice of an appropriate prediction-error metric is highly task-specific. In context of an “email spam” classification, we would especially be interested in a low false positive rate. Of course, a spam email that was classified as ham is certainly annoying, but not as bad as missing any important information by mis-classifying “ham” as “spam”.

One convenient way to tweak a classifier in context of a binary classification problem such as “spam” classification is the Receiver Operating Characteristic (ROC, or ROC curve)[17].

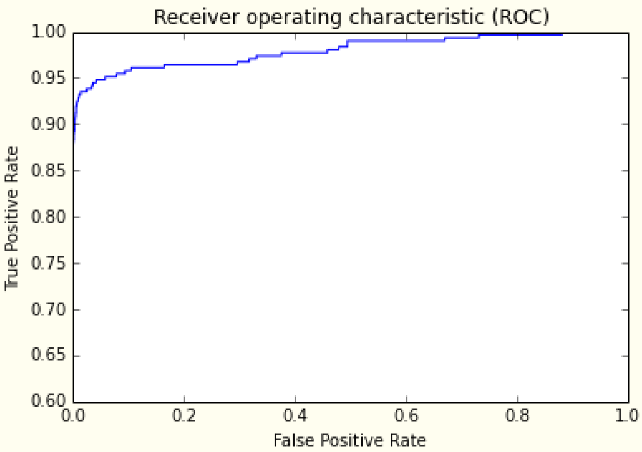


Figure 8 : Prediction Curve

Other indicators for classification performances are **Sensitivity**, **Specificity**, **Recall**, and **Precision**.

Sensitivity (synonymous to recall) and precision are assessing the "True Positive Rate" for a binary classification problem: The probability to make a correct prediction for a "positive/true" case (e.g., in an attempt to predict a disease, the disease is correctly predicted for a patient who truly has this disease).

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

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

Specificity describes the "True Negative Rate" for a binary classification problem: The probability to make a correct prediction for a "false/negative" case (e.g., in an attempt to predict a disease, no disease is predicted for a healthy patient).

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

In a typical supervised learning workflow, we would evaluate various different combinations of feature subspaces, learning algorithms, and hyper parameters before we select the model that has a satisfactory performance. As mentioned above, cross-validation is a good way for such an assessment in order to avoid over-fitting to our training data.

4. Conclusion

Predictive Analytics is the amalgamation of human expertise and proficiency with technology – people, tools and algorithms are the core of the predictive analytics. Learning the patterns from the historical and current data and the application of algorithms not only to analyse the trends but also to predict the future outcomes is possible because of the above factors [10].

The recent upraise in the field of predictive analytics is mainly due to the BigData, huge volume and abundant data available for research and its application irrespective of the field. But an organization should be well versed in case of why they would require predictive analytics. Imminent step after this is to explain the business requirement and the sort of questions the organizations need to find the answer. Establishment of the technology is the initial step and then comes the important part of testing the applied constraints so that they meet the confined requirements. Another vital motive is to deal with all the challenges and fulfill them, thus extending the output to next scale of advancement.

REFERENCES

[1] https://sebastianraschka.com/Articles/2014_intro_supervised_learning.html
[2] Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval* (Vol. 463). New York: ACM press
[3] Baeza-Yates, R. and Ribeiro-Neto, B., 1999. *Modern information retrieval* (Vol. 463). New York: ACM press.
[4] Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Machine Learning Proceedings 1995* (pp. 194-202). Morgan Kaufmann.
[5] Sutton RS, Barto AG. Introduction to reinforcement learning. Cambridge: MIT press; 1998 Mar 1.
[6] Guillen, Pablo, et al. "Supervised learning to detect salt body." *SEG Technical Program Expanded Abstracts 2015*. Society of Exploration Geophysicists, 2015. 1826-1829.

[7]Alvarellos-González, A., Pazos, A., & Porto-Pazos, A. B. (2012). Computational models of neuron-astrocyte interactions lead to improved efficacy in the performance of neural networks. *Computational and mathematical methods in medicine*, *2012*.

[8] Leban, G., Zupan, B., Vidmar, G., & Bratko, I. (2006). Vizrank: Data visualization guided by machine learning. *Data Mining and Knowledge Discovery*, *13*(2), 119-136.

[9] Wang, M., Cui, Y., Wang, X., Xiao, S., & Jiang, J. (2017). Machine learning for networking: Workflow, advances and opportunities. *IEEE Network*, *32*(2), 92-99.

[10] Fayyad, U., Haussler, D., & Stolorz, P. (1996). Mining scientific data. *Communications of the ACM*, *39*(11), 51-57.

[11] Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994* (pp. 148-156). Morgan Kaufmann.

[12] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, *12*(Oct), 2825-2830.

[13] Jayalakshmi, T., and A. Santhakumaran. "Statistical normalization and back propagation for classification." *International Journal of Computer Theory and Engineering* 3.1 (2011): 1793-8201.

[14]https://towardsdatascience.com/feature-selection-and-dimensionality-reduction-f488d1a035de

[15] Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems* (pp. 2951-2959.

[16] Bardenet, R., Brendel, M., Kégl, B., & Sebag, M. (2013, February). Collaborative hyperparameter tuning. In *International conference on machine learning* (pp. 199-207).

[17] González-Recio, Oscar, Guilherme JM Rosa, and Daniel Gianola. "Machine learning methods and predictive ability metrics for genome-wide prediction of complex traits." *Livestock Science* 166 (2014): 217-231.