Classification Methods for Breast Cancer Recognition Based on the Breast Cancer Wisconsin (Diagnostic) Data Set Using Machine Learning

Mir Mohammad Khaleghi
Department of Electrical and Computer Engineering.
Isfahan University of Technology
Isfahan, Iran

***Abstract*—In this article, our focus is on the recognition of benign and malignant tumors in breast cancer. We aim to compare the performance of eight different mainstream machine learning classification methods using a dataset specifically curated for this purpose. The dataset comprises components extracted from medical images of breast tumors.**

Our primary objective is to accurately distinguish between benign and malignant tumors based on these extracted components. To accomplish this, we will first provide a detailed description of the dataset and its components. Subsequently, we will discuss and provide a brief review of the methods employed for comparison in this study.

Following the methodological discussion, we will present the results of the comparison and draw conclusions based on our findings.

Breast Cancer, Machine Learning, Classification, Feature extraction, PCA.

I. Introduction

In this study, our objective is to compare the performance of multiple powerful machine learning algorithms on a shared dataset. We will conduct a comparative analysis both before and after feature extraction using PCA to observe the extent of variation in the classification scores. As a measure of comparison, we have chosen Mean Accuracy.

The dataset was obtained from Kaggle [1] and consists of 30 components extracted from combined 569 benign and malignant samples."Each sample is accompanied by its respective label, with the classification task focusing on determining whether the sample is benign or malignant.

Correlation analysis revealed highly correlated pairs of features, and as a preprocessing step, three pairs with a correlation exceeding 95% were removed. This resulted in 23 features with relatively low correlation. Additionally, the dataset was normalized across all features to enhance classifier stability and convergence properties. Finally, the dataset was divided into an 80-20 split for training and testing purposes respectively.

II. Summary and Setup of Experimental Methods

A total of eight distinct Machine Learning methods were evaluated using the same dataset, both before and after feature extraction, in order to determine the optimal performing method. In this study, a concise review of the methods will be provided, including the hyperparameters utilized for the analysis.

A. MLP

MLPs (Multilayer Perceptrons) are recognized as one of the earliest and robust classification techniques. They excel in creating non-convex decision boundaries and possess the capability to approximate any function with precision. MLPs belong to the category of fully connected feedforward Artificial Neural Networks (ANNs). Figure 1 illustrates the architecture of a 3-layer MLP. Each layer is associated with

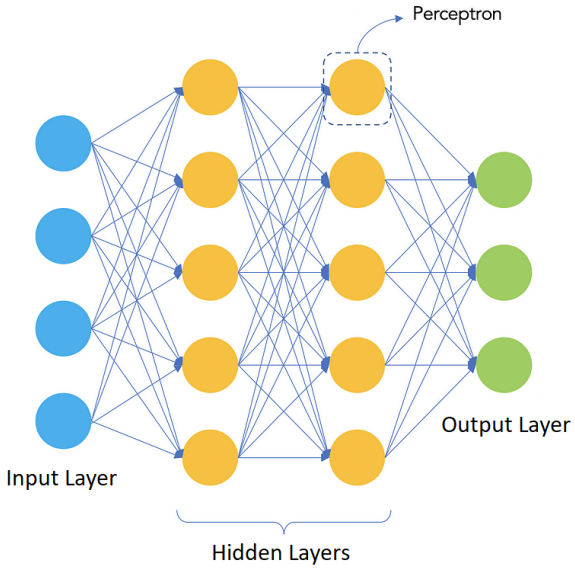

Fig. 1. Structure of an MLP

a weight matrix, and the output of layer "l" can be computed as follows:

$$\begin{aligned} a_l &= f\left(n_l\right) \\ n_l &= w^T a_{l-1} + b_l \end{aligned} \tag{1}$$

The activation function $f(.)$ introduces non-linearity to the neural network, such as tanh, relu, or sigmoid. The weights of the network are updated through backpropagation using Stochastic Gradient Descent (SGD). SGD calculates the gradient of the cost function, which measures the deviation between the network's output and the expected output, with respect to each weight in every layer. This gradient is then subtracted from the weights of each parameter, aiming to guide the network towards a minimum.

To address the convergence stability issues of early SGD versions, various enhancements were proposed. One of these is

momentum SGD, which is utilized in this context. Momentum SGD improves both the convergence speed and stability of the learning process.[2]

B. Naive Bayes

A probabilistic machine learning model known as the Naive Bayes classifier is employed for classification tasks. The essence of this classifier lies in its utilization of the well-known Bayes theorem, as described in [3].

$$P(y \mid X) = \frac{P(X \mid y)P(y)}{P(X)} \tag{2}$$

Where X is an n dimensional feature vector defined as $X = [x_1, x_2, \cdots, x_n]^T$. Therefore, by substituting for X in (2) and expanding using the chain rule we get:

$$P(y \mid x_1, x_2, \cdots, x_n) = \frac{P(x_1 \mid y) \cdots P(x_n \mid y) P(y)}{P(x_1) \cdots P(x_n)} \tag{3}$$

Nonetheless, as the denominator of equation (3) remains constant across the entire dataset, we can conclude that:

$$P(y \mid x_1, x_2, \cdots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \tag{4}$$

By employing equation (4), the Naive Bayes classifier determines the most probable class y based on the predictor variable X, thereby predicting the class.

$$y = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y) \tag{5}$$

C. Logistic Regression

Logistic Regression, in essence, calculates the likelihood of an event taking place using a provided dataset of independent variables [4]. Specifically, Logistic Regression applies a logit transformation to the odds, which represents the probability of success relative to the probability of failure (represented by 1 and 0, respectively). This transformation, commonly referred to as the 'log odds' is modeled using the logistic function, expressed as:

$$\ln\left(\frac{p_k}{1-p_k}\right) = \theta_0 + \theta_1 X_1 + \cdots + \theta_n X_n \tag{6}$$

$$\text{logit}(p_k) = \frac{1}{1+e^{-p_k}} \tag{7}$$

the dependent variable $\text{logit}(p)$ is considered, assuming that the log odds exhibit linearity with respect to the feature vector X. By utilizing equations (6) and (7), it can be demonstrated that we obtain the following relationship:

$$\text{logit}(p_k) = \ln\left(\frac{p_k}{1-p_k}\right) = \theta_0 + \theta_1 X_1 + \cdots + \theta_n X_n \tag{8}$$

The parameter vector θ, represented by an $n+1$ dimensional vector, is commonly estimated using Maximum Likelihood Estimation (MLE). Once the parameter vector is obtained, we can estimate the likelihood of success for an input vector by applying it to equation (8). Additionally, classification can be accomplished by using a threshold on the output of the logit transform, typically set to ≥ 0.5 for a value of 1, and 0 otherwise.[5]

D. k-NN

Considered as one of the simplest classifiers, the k-NN classifier categorizes a given data point based on its proximity to previously observed samples within a predefined neighborhood. Specifically, the k nearest samples to a given sample are determined using a distance metric (typically the Minkowski distance of order 2, $d(p,q) = \|p-q\|$, and then the class of the target sample is determined through majority voting among these k samples. The underlying concept is that if a given sample is closest to samples of a certain class ω within a fixed neighborhood, it is likely to belong to the same class. [6]

Additionally, there exists a trade-off between overfitting and underfitting concerning the hyperparameters k. However, a statistical analysis conducted in [7] suggests that an optimal value for k is generally $\sqrt{N}$, where N represents the number of available samples.

Although simple, the k-NN classifier exhibits strong consistency results. As the number of samples (N) tends to infinity, the two-class k-NN is guaranteed to achieve an error rate no worse than twice the Bayes optimal error rate.

$$R^* \leq R_{kNN} \leq R^*\left(2 - \frac{MR^*}{M-1}\right) \tag{9}$$

Given that R^* represents the Bayes optimal error rate, R_{kNN} denotes the error rate associated with the k-NN classifier, and M represents the number of classes, we can express the following relationship. In the case of a two-class scenario and employing equation (9), as the Bayes optimal error rate R^* approaches 0, we can observe that:

$$R^* \leq R_{kNN} \leq 2R^* \tag{10}$$

E. SVM

SVM is a classification technique designed to maximize the margin of the decision boundary, which is represented by a hyperplane. The objective is to achieve the best possible generalization [8]. The approach involves defining the margin as twice the distance between the hyperplane and the nearest samples from each class. If we define the hyperplane as:

$$w^T x + b = 0 \tag{11}$$

It can be demonstrated, without sacrificing generality, that the margin corresponds to the distance between the two hyperplanes.

$$\begin{aligned} w^T x + b &= 1 \\ w^T x + b &= -1 \end{aligned} \tag{12}$$

Under the assumption that the hyperplanes touch the closest samples from each class, it can be shown that the margin is represented by $m = \frac{2}{\|w\|}$. To maximize this margin, a loss function is introduced and minimized using a sequential optimization technique.[9] A specific type of SVM, known as

"soft margin SVM," permits samples to fall within the region between the two hyperplanes outlined in (12), but imposes a penalty on the loss function for each deviating sample. Specifically, the objective of the SVM is to

$$\begin{aligned} &\text{minimize } \frac{1}{n}\sum_{i=1}^{n}\xi_i + \lambda\|w\|^2 \\ &\text{s.t. } y_i\left(w^T x_i + b\right) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, \text{ for all } i \end{aligned} \tag{13}$$

In this study, the C-SVM variant of the soft margin SVM was utilized. The C-SVM variant incorporates a slack variable, denoted as ξ_i, which is the smallest non-negative value satisfying the condition $y_i\left(w^T x_i + b\right) \geq 1 - \xi_i$.[10]

F. Decision trees(DTs)

Decision trees serve as decision support tools that employ a tree-like structure to model decisions and their potential outcomes, encompassing chance events, resource costs, and utility. In the context of machine learning, decision trees are commonly used for classification tasks. Starting from a root node that encompasses the entire dataset, the tree splits based on specific criteria (such as the Gini index or entropy), gradually progressing towards leaves that contain sufficiently homogeneous subsets of the original data. Figure 2 provides an illustration of a decision tree.

There are multiple algorithms available to train decision trees, with CART and ID3 being two of the most widely used ones. Both approaches employ a criterion for assessing the quality of splits, enabling the dataset to be divided into partitions at each node in a greedy manner that maximizes the purity of the resulting subsets. However, constructing the optimal decision tree has been proven to be NP-complete, necessitating the adoption of greedy techniques to practically build decision trees.[11]

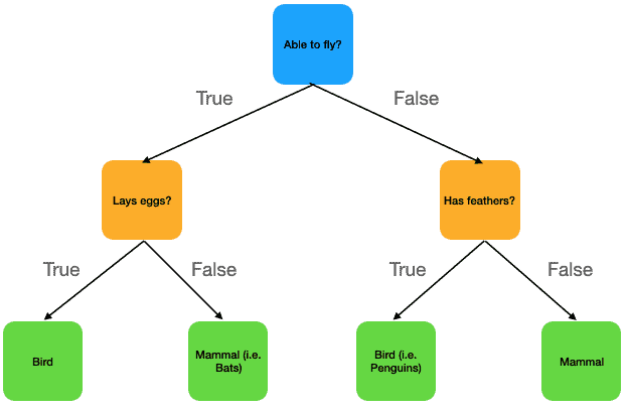

Fig. 2. An instance of a decision tree

G. Random Forest

Random Forest is an extension of decision trees and a specific category of ensemble machine learning models. Its goal is to enhance the generalization and stability of individual decision trees. This is achieved by generating a multitude of decision trees, each trained on a subset of the dataset and a subset of available features. The final prediction is made by aggregating the outputs of all trained decision trees through majority voting [12].

H. AdaBoost

AdaBoost, introduced by Yoav Freund and Robert Schapire in 1997 [13], is a statistical classification meta-algorithm designed to enhance the performance of various learning algorithms. The concept involves initially forming an ensemble of weak classifiers that exhibit slightly better performance than random guessing. Subsequently, at each iteration, the misclassified samples within the ensemble are assigned higher weights, thereby strengthening the overall classifier's resilience against classification errors. To be more specific, a boosted classifier can be represented as:

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \tag{14}$$

In each iteration t, a weak learner f_t is chosen, which takes a vector x as input and predicts the class of that input. The weak learner is assigned a weight α_t, and the objective is to minimize the total training error E_t of the boosting classifier with t stages, as shown in equation (15).

$$E_t = \sum_i E\left[w_{i,t}F_{t-1}\left(x_i\right) + \alpha_t h\left(x_i\right)\right] \tag{15}$$

Here, $F_{t-1}(x)$ represents the boosted classifier that has been constructed up to the previous stage of training, and $f_t(x) = \alpha_t h(x)$ denotes a weak learner that is being evaluated for inclusion in the final ensemble of classifiers. Additionally, in each iteration, a weight $w_{i,t}$ is assigned to every sample in the training set, which is equal to the current error $E\left(F_{t-1}\left(x_i\right)\right)$. This weight assignment process aims to enhance the classifier's resilience to classification errors as the training progresses.

III. Experiment Setup

The renowned Python machine learning library, Scikit-learn, was utilized to implement the aforementioned classifiers. The implementation can be found in the accompanying Python notebook file named 'breast-cancer-classification.ipynb'. The following are the hyperparameters that were used for each of the classifiers:

A. MLP

– Two Hidden Layers with 100 and 50 neurons each to ensure non-convex decision surface.
– Activation of Hidden Layers: relu
– Maximum Iterations: 3000
– Initial Learning Rate: 0.0001
– Last Layer Activation: sigmoid
– Optimizer: Variable Learning Rate Momentum SGD
- Momentum (γ) : 0.9

B. Naive Bayes

- Variance Smoothing Epsilon Value: 10^{-9}

C. Logistic Regression

- Regularization: $L2$ norm of weights added to the loss function

D. KNN

- Metric: Minkowski distance
- k: $\sqrt{N}$ where N is the number of available samples.

E. C-SVM

- C: 1
- Kernel: Linear

F. Decision Tree

- Splitting Criterion: Gini index
- Min. Samples to Split: 2
- Min. Impurity Decrease: 0

G. Decision Tree

- Decision Tree Properties: Same as above.
- Max. Number of Features Used for an Individual Tree: $\sqrt{N}$ where N is the number of features.
- Number of Decision Trees Built: 100

H. AdaBoost

- Number of Weak Learners: 250
- Weight applied to each Weak Classifier at every Iteration (Learning Rate): 1

IV. Results

To ensure the reproducibility of the experiments and obtain consistent results, a fixed random state seed was set globally. This seed allows for the replication of the presented results in this paper. For the binary classification task of voice gender recognition, each of the classifiers mentioned in the preceding sections was trained on the dataset. The experiments were conducted twice: once without applying Principal Component Analysis (PCA) to reduce the dimensionality of the feature space, and once with PCA applied.

A. before implementing PCA

Figure 3 provides an overview of the experimental results before applying dimensionality reduction. It is evident that Random Forest and AdaBoost outperform the other classifiers by a significant margin, while the Naive Bayes classifier exhibits noticeably poorer performance. It is worth noting that both Random Forest and AdaBoost in Scikit-learn utilize Decision Trees as their weak learners. Consequently, their mean accuracy is similar. However, a closer examination of the confusion matrices reveals that the two classifiers exhibit slightly different behavior.

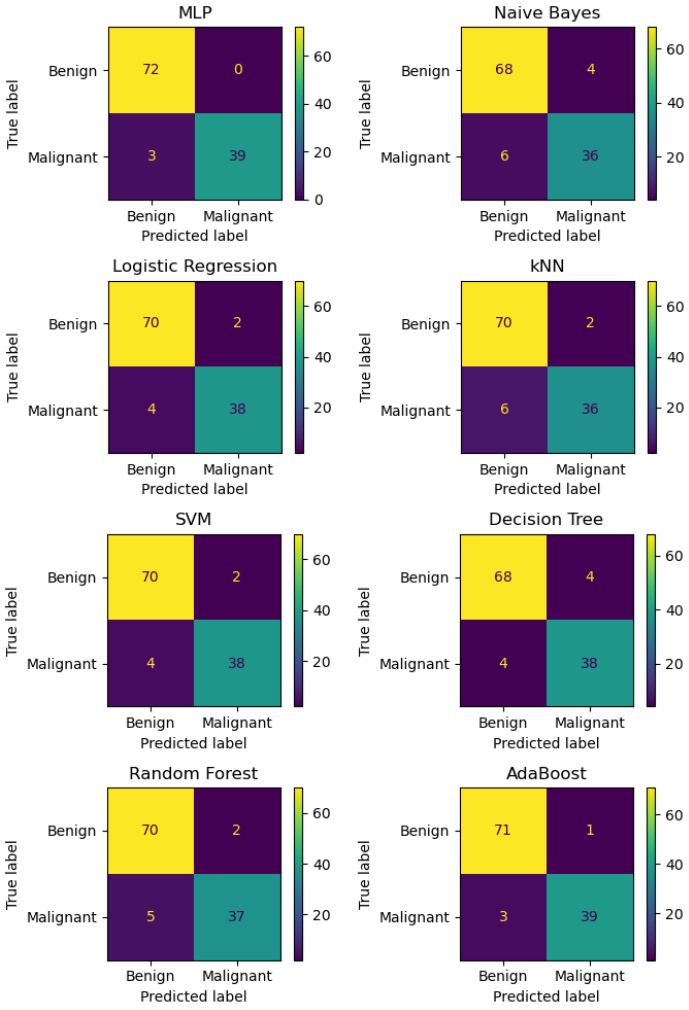

Fig. 3. Confusion Matrices of Each Classifier on the Dataset Before PCA

B. after implementing PCA

Before we move on to the experiments, it is not without merit to take a look at the distribution of the classes over the two principal axes to see how separable they are if we only chose to take the two components corresponding to the two biggest eigenvalues. As is evident in figure 4, even the two most informative axes alone are already enough to give a reasonable representation of the dataset. In our experiments however, we have use all the principal axes up to the point where 90% of the information is preserved. This results in a dimensionality reduction from the original 23-d feature space, to only 7 features, while still giving comparable results to the experiments performed before PCA as we can see in figure 5.

In the reduced feature space, the performance rankings of the classifiers are as follows: MLP achieves the highest performance, followed by Random Forest, while Decision Tree performs the poorest, even falling below Naive Bayes. Interestingly, performing PCA on the dataset has the least impact on the mean accuracy of the Naive Bayes classifier. Additionally, it is worth noting that in both comparisons,

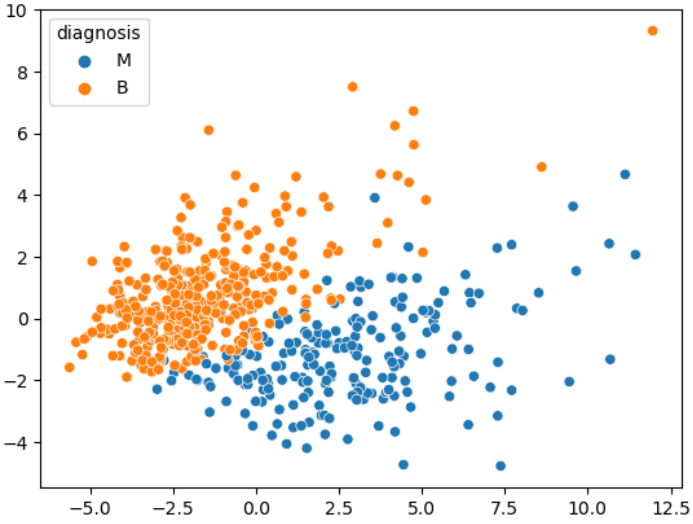

Fig. 4. visualization of the Dataset over the Two Principal Axes

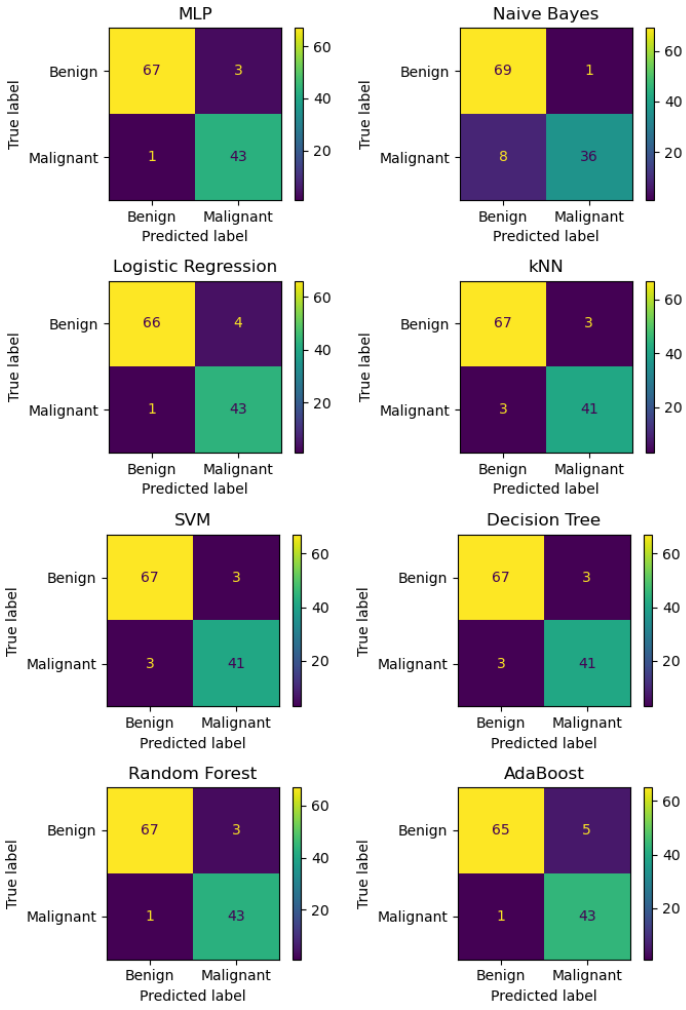

Fig. 5. Confusion Matrices of Each Classifier on the Dataset after PCA

Random Forest consistently outperforms a single Decision Tree.

The performance accuracy of the examined classifiers before and after PCA is presented in the table 1.

TABLE I
COMPARISON BETWEEN ACCURACY PERCENTAGE OF DIFFERENT CLASSIFIERS BEFORE AND AFTER PCA

classifier	before PCA	after PCA
MLP	97.37	96.49
$NaiveBayes$	91.23	92.1
$LogisticRegression$	94.74	95.61
kNN	92.98	94.74
SVM	94.74	94.74
$DecisionTree$	92.98	94.74
$RandomForest$	93.86	96.49
$AdaBoost$	96.49	94.74

V. CONCLUSION

This study involved a comparison of the performance of eight popular machine learning classification frameworks on a tabular dataset, both before and after dimensionality reduction. Additionally, we demonstrated the significant impact of dimensionality reduction on certain datasets, where reducing the feature space led to negligible changes in accuracy of some classifiers.

Across all comparisons, the more complex methods such as AdaBoost, MLP, and Random Forests exhibited the best classification results.

REFERENCES

[1] "Breast Cancer Wisconsin (Diagnostic) Data Set." [Online]. Available: https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data
[2] Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.
[3] Rish, I. (2001). An empirical study of the naive Bayes classifier. In Proceedings of the 2001 International Conference on Machine Learning (ICML) (pp. 41-48).
[4] Hosmer, D. W., Lemeshow, S. (2000). Applied logistic regression (2nd ed.). Wiley.
[5] Cox, D. R. (1958). The regression analysis of binary sequences (with discussion). Journal of the Royal Statistical Society. Series B (Methodological), 20(2), 215-242.
[6] Cover, T., Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), 21-27.
[7] Hastie, T., Tibshirani, R., Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). Springer.
[8] Cortes, C., Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297.
[9] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2), 121-167.
[10] Cristianini, N., Shawe-Taylor, J. (2000). An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press.
[11] Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1), 81-106.
[12] Liaw, A., Wiener, M. (2002). Classification and regression by randomForest. R News, 2(3), 18-22.
[13] Freund, Y., Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1), 119-139.