Predicting Shopper's Intentions on Online platforms using Machine Learning Algorithms

By-Anshh Chaturvedi (102003665) and Harshit Sharma (102003653)
[1]Computer Science and Engineering Department (CSED), Thapar University, Patiala, Punjab,India.
[2]These authors contributed equally: Anshh Chaturvedi and Harshit Sharma
email:achaturvedi_be20@thapar.edu , hsharma_be20@thapar.edu

1. INTRODUCTION

The increase in e-commerce usage over the past few years has created potential in the market, but the fact that the conversion rates have not increased at the same rate leads to the need for solutions that present customized promotions to the online shoppers. The recent outburst of online shopping has added a new dimension in the business sector. People tend to explore online to find the items they need and buy through online transactions. It has made their life easier and more comfortable. At the same time, it has become a great need for sellers to know the patterns and intentions of different types of online customers. The customer's purchase intention can be predicted by analyzing the history of the customers. Online shopping behavior's data has been analyzed to build a classification model to predict their purchase intention. In physical retailing, a salesperson can offer a range of customized alternatives to shoppers based on the experience he or she has gained over time. This experience has an important influence on the effective use of time, purchase conversion rates, and sales figures. Many e-commerce and information technology companies invest in early detection and behavioral prediction systems which imitate the behavior of a salesperson in a virtual shopping environment. In parallel with these efforts, some academic studies addressing the problem from different perspectives using machine learning methods have been proposed. While some of these studies deal with categorization of visits based on the user's navigational patterns, others aim to predict the behavior of users in real time and take actions accordingly to improve the shopping cart abandonment and purchase conversion rates.

Machine learning can help suppliers make better sense of the exponentially increasing data (both transactional and behavioral) and more accurately predict spending behavior of the buyer. Ultimately, this can lead to better customized experiences and drive top line and bottom line growth. To predict the purchasing intention of the visitor we aggregated pageview data kept track during the visit along with some session and user information as input to machine learning algorithms. We apply oversampling and feature selection preprocessing techniques to improve the success rates and scalability of the algorithms. In our study, the purchasing intention model is designed as a binary classification problem measuring the user's intention to finalize the
transaction.
Thus, we aim to offer content only to those who intend to purchase and not to offer content to the other users. Data mining techniques have been exploited to help enterprises for knowledge discovery and decision making by analyzing the past data. It is also very challenging to predict individual customer intentions in real-time. Here, we analyzed a few data mining techniques, it also involved some classification algorithms and clustering.

Some of the articles related to the same problem are referred to and their research work is described in the following section. The third section contains a description of the data, how it is collected, how it is preprocessed to make it suitable for further use in analysis. The results of the additional analysis have been analyzed and summarized in the conclusion section. The architecture diagram of the framework used in the project is included in the fourth section. This diagram depicts the order in which the various processes were carried out. The fifth section contains the algorithms that were used to create this project. There are numerous machine learning algorithms available, which have been compared in a table in the results section. It also includes how different machine learning algorithms and their precision, accuracy, recall, F1 score, sensitivity, and so on are . The final section discusses the sources used to create this project. Many eminent researchers have worked in this field, as evidenced by the references section.

2. BACKGROUND

Carmona CJ et al. deals with categorization of visits based on the user's navigational patterns.

A first study of Mobasher et al. assessed two different clustering techniques based on user transactions and page views in order to find useful aggregate profiles that can be used by recommendation systems to achieve effective personalisation at early stages of user's visits in an online store.

Ding et al. used HMM to model the clickstream data of the visitors and showed that predicting the intention of the user in real time and taking customized actions in this context helps to increase the conversion rates and decrease the shopping cart abandonment rates.

Ghose S et al. proposed a system which handles the loss of throughput in Web servers due to overloading, by assigning priorities to sessions on an e-commerce website according to the revenue that will generate. Data were formed of clickstream and session information and Markov chains, logistic linear regression, decision trees and naïve Bayes were investigated in order to measure the probability of users' purchasing intention.

3. DATASET DESCRIPTION

It was sourced from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset#)

Feature	**Explanation**	**Measurement**	**Range**
Administrative	Number of pages visited by the visitor about account management		
Administrative_Duration	Total amount of time (in seconds) spent by the visitor on account management related pages		
Informational	Number of pages visited by the visitor about the Web site, communication and address information of the shopping		

site

Informational_Duration Total amount of time (in seconds) spent by the visitor on informational pages

Numeric [0,…,705] Seconds [0,…,63973]

ProductRelated Number of pages visited by visitor about product related pages

Percentage [0,…,0.2]
ExitRates Average exit rate value of the pages visited by the visitor

ProductRelated_Duration Total amount of time (in seconds) spent by the visitor on product related pages

PageValues Average page value of the pages visited by the visitor

BounceRates Average bounce rate value of the pages visited by the visitor
Numeric [0,…,27] Seconds [0,…,3398]

SpecialDay Closeness of the site visiting time to a special day
Percentage [0,…,0.2] Float [0,…,361] Float [0,..,1]

Numeric [0,…,24]

Seconds [0,…,2549.]

Month Month value of the visit date String [January,…,December] OperatingSystems Operating system of the visitor Numeric

[1,…,8] Browser Browser of the visitor Numeric [1,…,13]

Region Geographic region from which the session has been started by the visitor

VisitorType Visitor type as ‘‘New Visitor,’’ ‘‘Returning Visitor,’’ and ‘‘Other’’

TrafficTypes Traffic source by which the visitor has arrived at the Web site (e.g., banner,
SMS, direct)

Weekend Boolean value indicating whether the date of the visit is weekend

Revenue Class label indicating whether the visit has been finalized with a transaction

String Non-Returning, Returning Boolean TRUE, FALSE

Numeric [1,…,9]

Numeric [1,…,20]

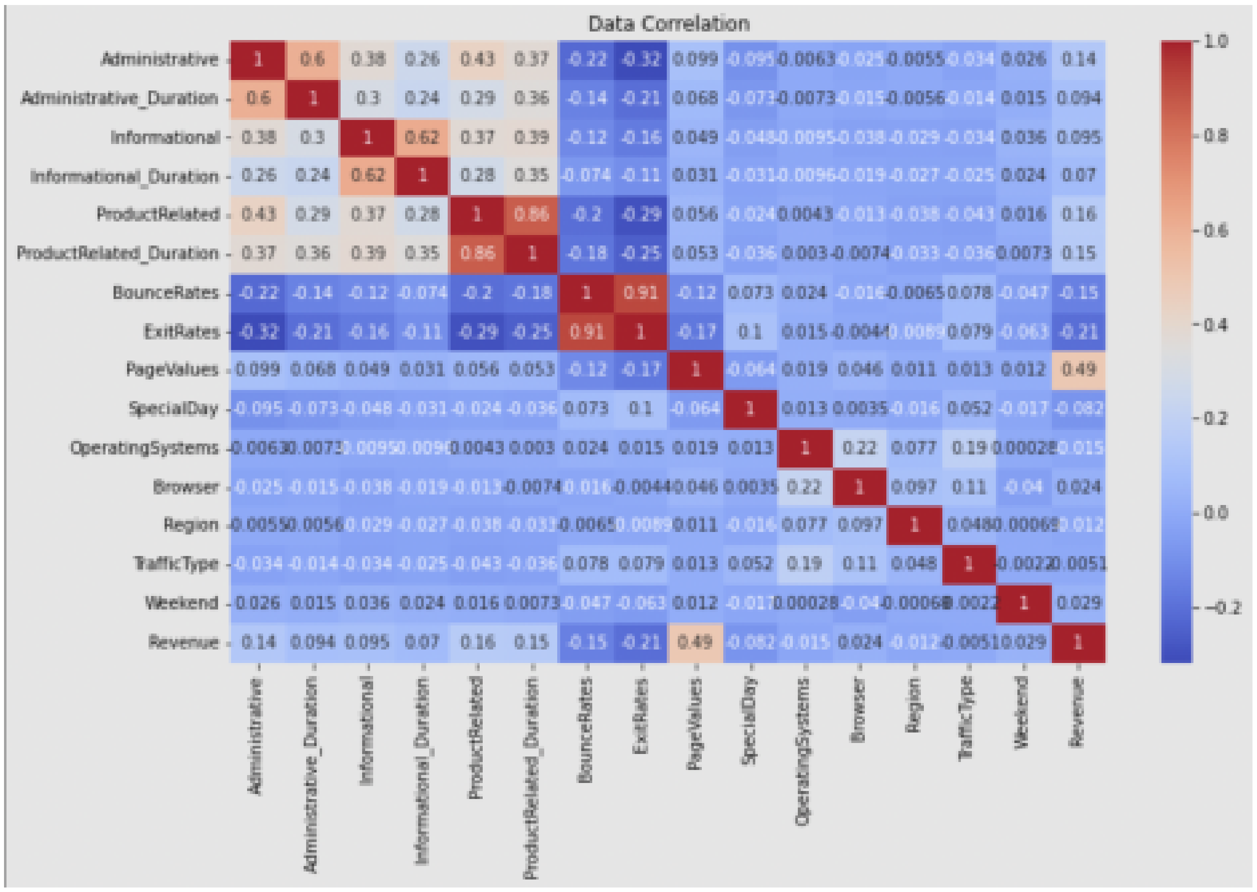
Boolean `TRUE, FALSE

Correlation matrix:

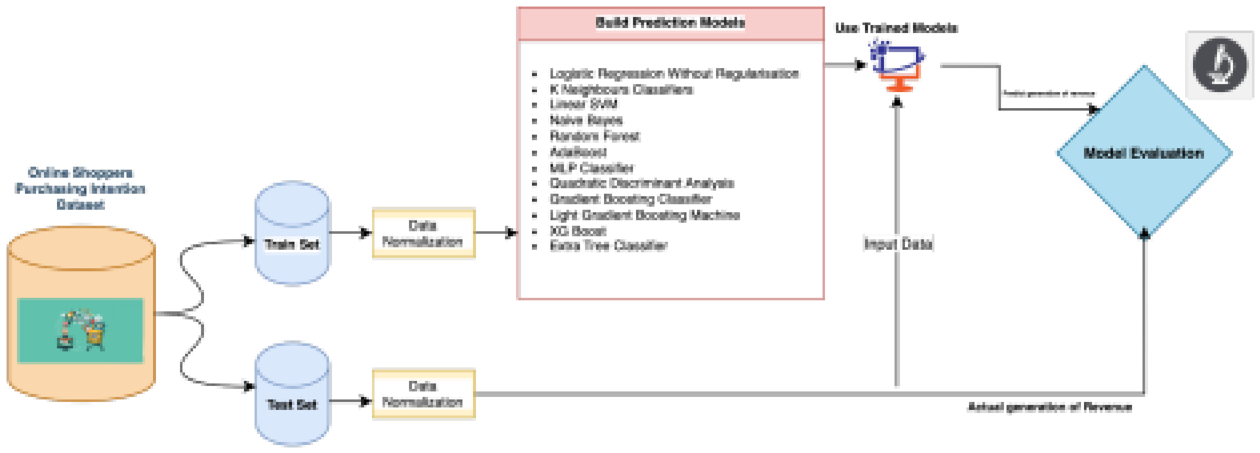
In total, the dataset contains 18 attributes and 12330 entries. The following figure shows the correlation amongst various attributes of the dataset. From this, we can conclude that the attribute that has the most direct relationship with whether revenue is going to be generated or not, is the PageValue i.e. the number of pages the user visits in the session.

Correlation between most of the attributes are low and ranging between 0.3-0.7.

The exceptional cases are the correlation between Bounce Rate and Exit Rates, i.e. 0.91 and between ProductRelated and ProductRelated_Duration, i.e. 0.86.



4. ARCHITECTURE



5. ALGORITHM USED

Logistic regression

Logistic regression is very similar to linear regression as a concept and it can be thought of as a "maximum likelihood estimation" problem where we are trying to find statistical parameters that maximize the likelihood of the observed data being sampled from the statistical distribution of interest.

$$p_i = \frac{e^{(\beta_0+\beta_1 x_i)}}{1+e^{(\beta_0+\beta_1 x_i)}}$$

This equation can be linearized by the following transformation

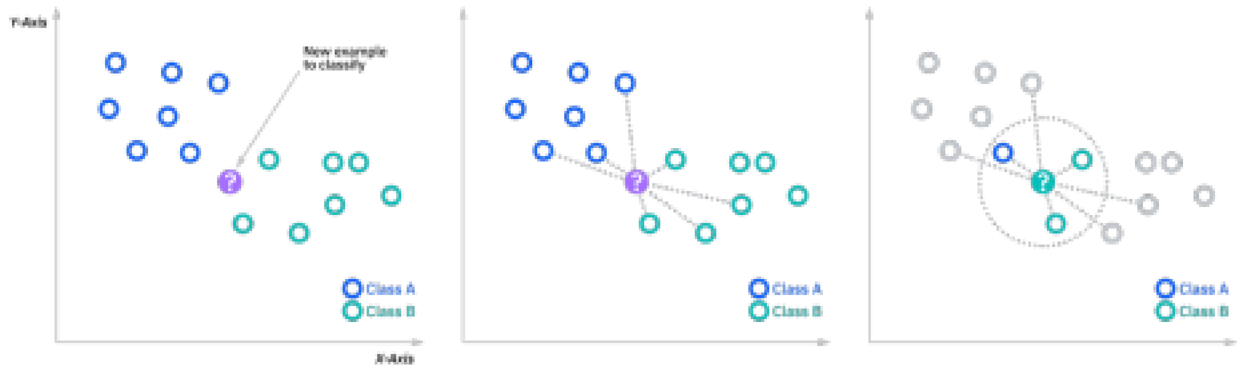
$$\text{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_i$$

In logistic regression, the response variable is modeled with a binomial distribution or its special case Bernoulli distribution. The value of each response variable, yi, is 0 or 1, and we need to figure out parameter pi values that could generate such a distribution of 0s and 1s.

$$\text{L}_{log} = -\ln(L) = -\sum_{i=1}^{N}\left[-\ln(1+e^{(\beta_0+\beta_1 x_i)}) + y_i\,(\beta_0+\beta_1 x_i)\right]$$

K neigbours classifiers

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.



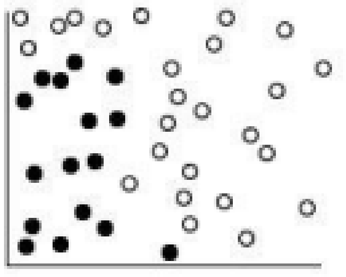
The k value in the k-NN algorithm defines how many neighbors will be checked to determine the classification of a specific query point. For example, if k=1, the instance will be assigned to the same class as its single nearest neighbor. Defining k can be a balancing act as different values can lead to overfitting or underfitting. Lower values of k can have high variance, but low bias, and larger values of k may lead to high bias and lower variance. The choice of k will largely depend on the input data as data with more outliers or noise will likely perform better with higher values of k. Overall, it is recommended to have an odd number for k to avoid ties in classification, and cross-validation tactics can help you choose the optimal k for your dataset.

Linear SVM

SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable. A separator between the categories is found, then the data are transformed in such a way that the separator could be drawn as a hyperplane. Following this, characteristics of new data can be used to predict the group to which a new record should belong.

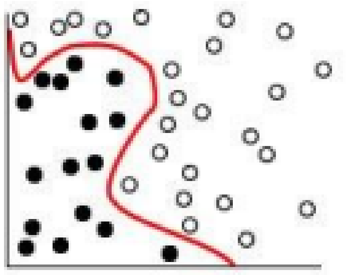
For example, consider the following figure, in which the data points fall into two different categories.

Figure 1. Original dataset



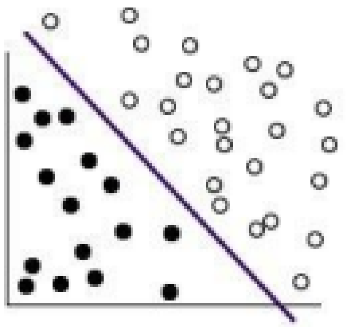
The two categories can be separated with a curve, as shown in the following figure.

Figure 2. Data with separator added



After the transformation, the boundary between the two categories can be defined by a hyperplane, as shown in the following figure.

Figure 3. Transformed data



The mathematical function used for the transformation is known as the **kernel** function. SVM in IBM® SPSS® Modeler supports the following kernel types:

- Linear
- Polynomial
- Radial basis function (RBF)
- Sigmoid

Naive Bayes

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Steps to implement:

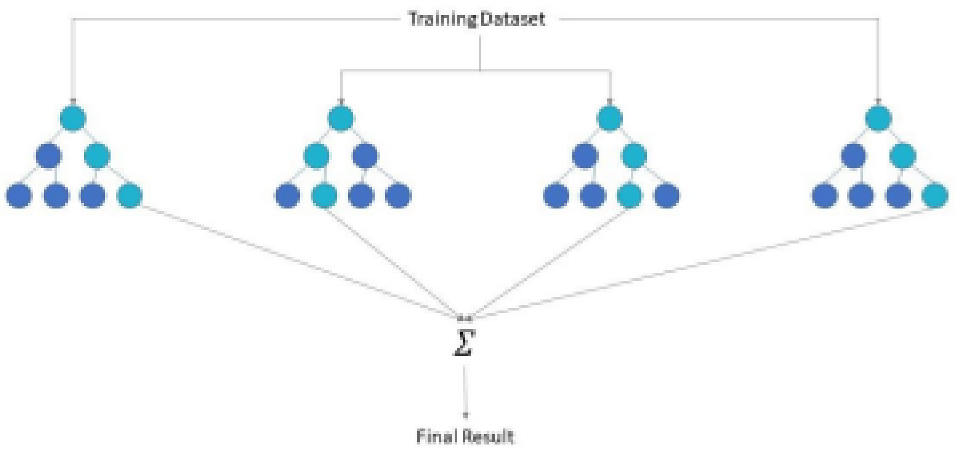
- Data Pre-processing step
- Fitting Naive Bayes to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.

Random Forest

The random forest algorithm is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees. Feature randomness, also known as feature bagging or "the random subspace method", generates a random subset of features, which ensures low correlation among decision trees. This is a key difference between decision trees and random forests. While decision trees consider all the possible feature splits, random forests only select a subset of those features.

Random forest algorithms have three main hyperparameters, which need to be set before training. These include node size, the number of trees, and the number of features sampled. From there, the random forest classifier can be used to solve for regression or classification problems.

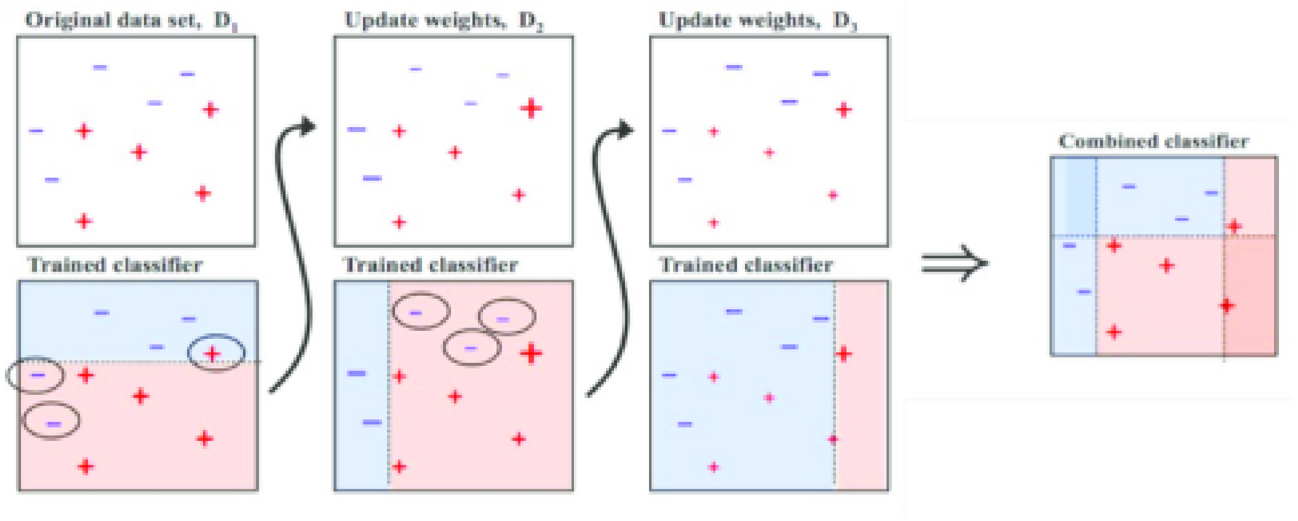
The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the bootstrap sample. Of that training sample, one-third of it is set aside as test data, known as the out-of-bag (oob) sample, which we'll come back to later. Another instance of randomness is then injected through feature bagging, adding more diversity to the dataset and reducing the correlation among decision trees. Depending on the type of problem, the determination of the prediction will vary. For a regression task, the individual decision trees will be averaged, and for a classification task, a majority vote—i.e. the most frequent categorical variable—will yield the predicted class. Finally, the oob sample is then used for cross-validation, finalizing that prediction.



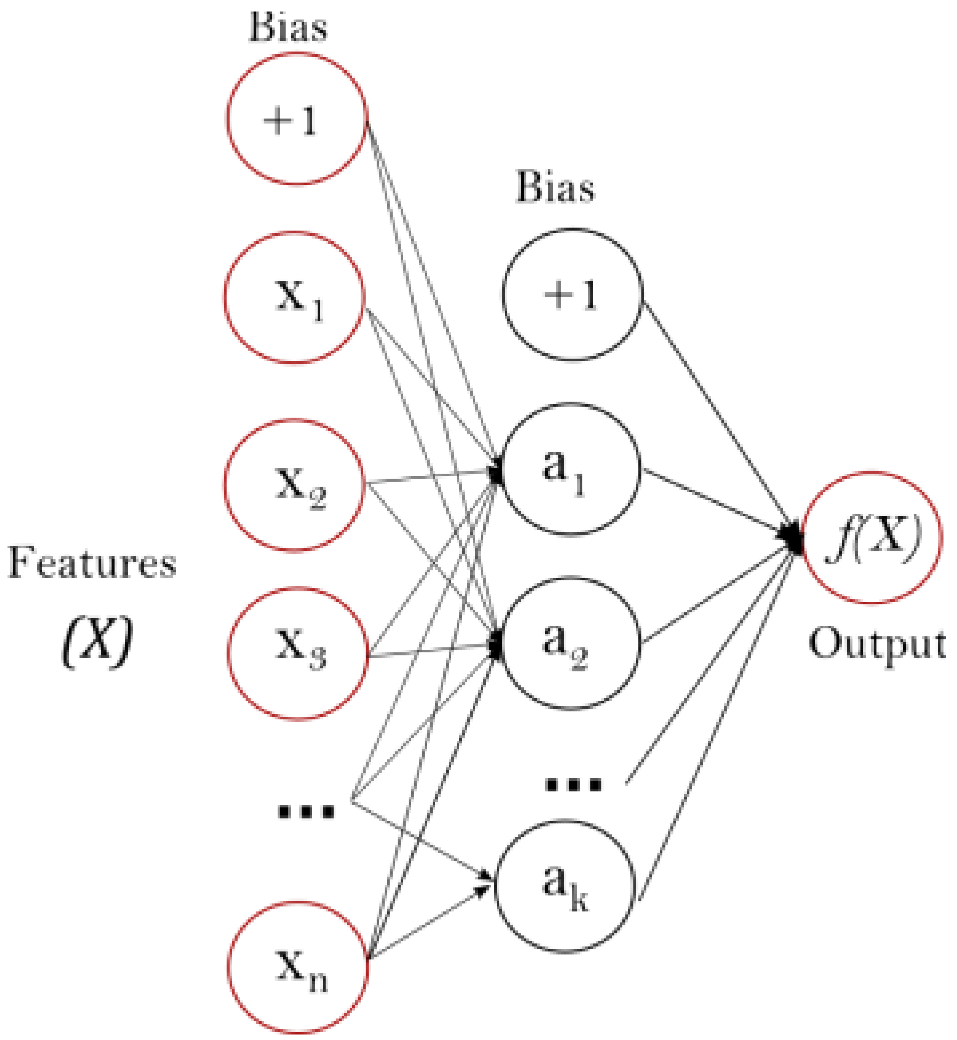
Adaboost

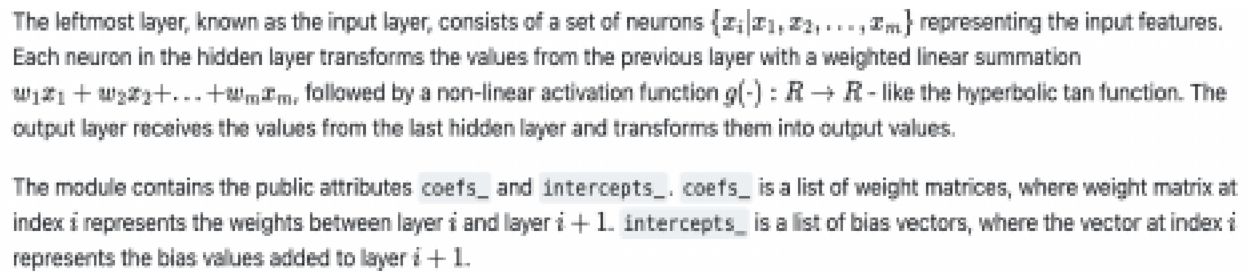
Adaptive Boosting is a good ensemble technique and can be used for both Classification and Regression problems. In most cases, it is used for classification problems. It is better than any other model as it improves model accuracy.

Adaboosting makes ‘n’ number of decision trees during the data training period.With AdaBoost, however, the algorithm only makes a node with two leaves, known as Stump. As the first decision tree/model is made, the incorrectly classified record in the first model is given priority. Only these records are sent as input for the second model. It will keep training models until and unless a lower error is received.The process goes on until we specify a number of base learners we want to create.



Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function f(·):Rm→Ro by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features X=x1,x2,...,x and a target y, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers.




The leftmost layer, known as the input layer, consists of a set of neurons $\{x_i | x_1, x_2, \ldots, x_m\}$ representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation $w_1 x_1 + w_2 x_2 + \ldots + w_m x_m$, followed by a non-linear activation function $g(\cdot) : R \rightarrow R$ - like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values.

The module contains the public attributes `coefs_` and `intercepts_`. `coefs_` is a list of weight matrices, where weight matrix at index i represents the weights between layer i and layer $i + 1$. `intercepts_` is a list of bias vectors, where the vector at index i represents the bias values added to layer $i + 1$.

Quadratic Discriminant Analysis

For each input variable, we define k binary indicator variables. Furthermore, let $\mathbf{t}$ denote all our target variables, and π the prior with a subscript denoting the class. Assuming the data points are drawn independently, the likelihood function is given by

$$
\begin{aligned}
P\left(\mathbf{t} \mid \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K\right) &= \prod_{n=1}^{N} \prod_{k=1}^{K} P\left(t_n = k\right)^{t_{nk}} P\left(\mathbf{x}_n \mid t_n = k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)^{t_{nk}} \\
&= \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{t_{nk}} \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)^{t_{nk}} .
\end{aligned}
$$

To simplify notation, let $\boldsymbol{\theta}$ denote all the class priors, class-specific mean vectors, and covariance matrices.

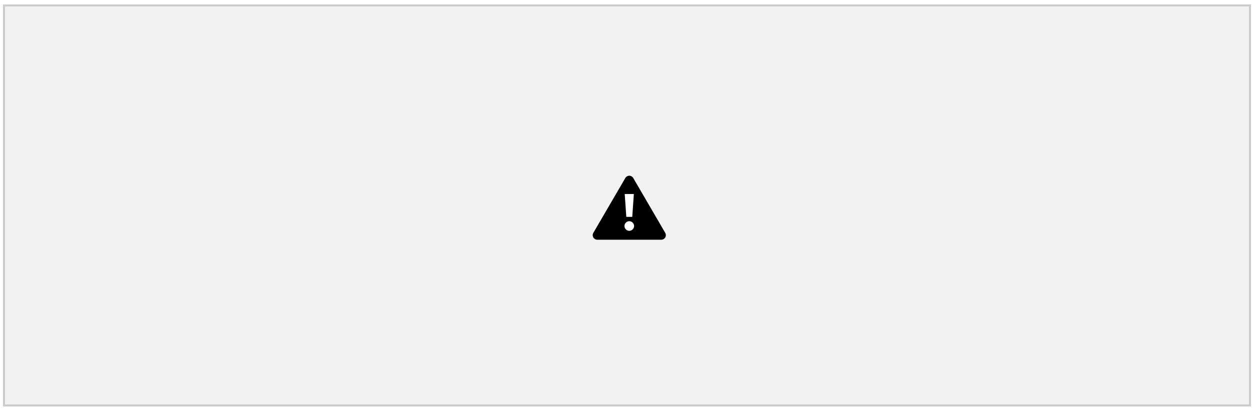
As we know, **maximizing the likelihood is equivalent to maximizing the log-likelihood**. The log-likelihood is

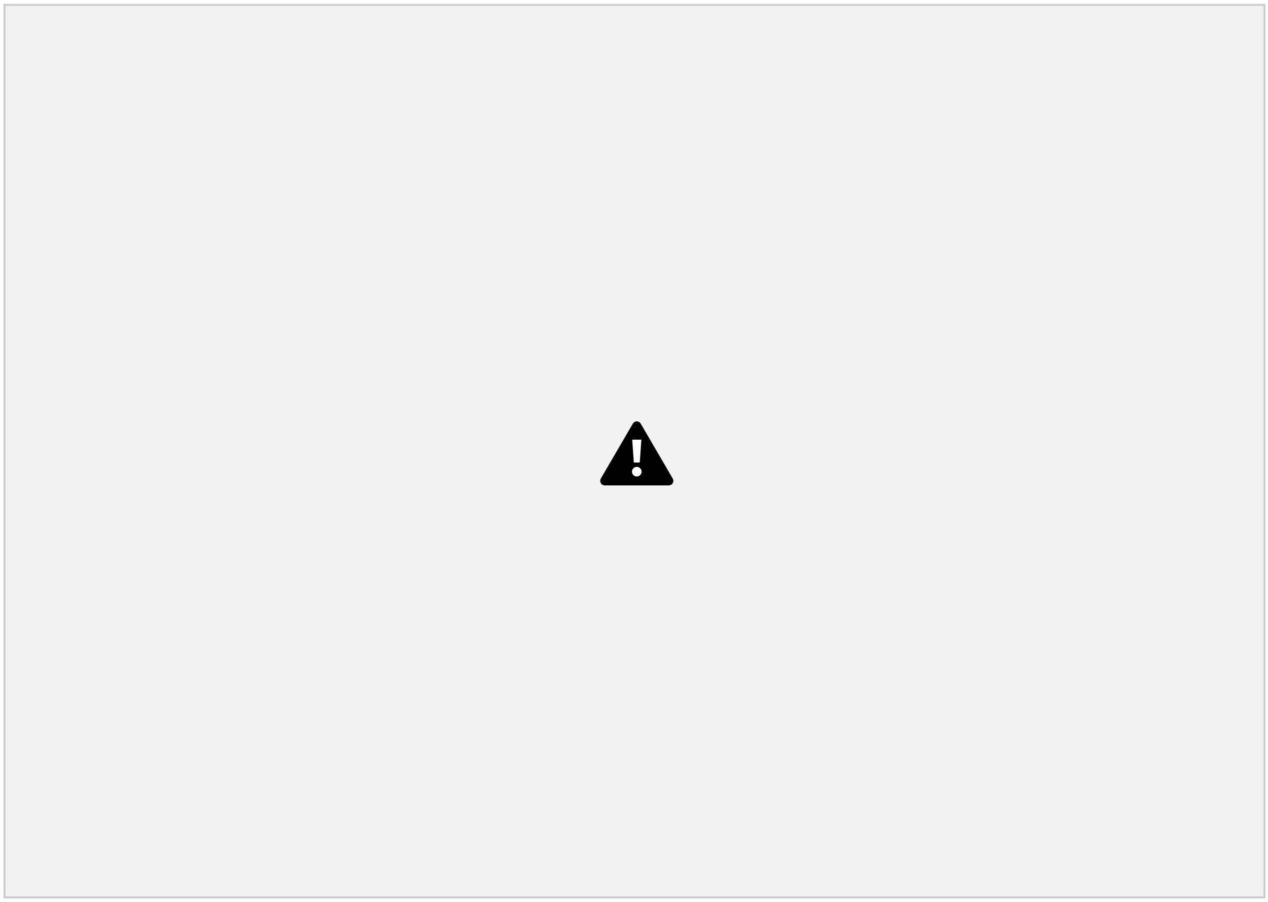
$$
\begin{aligned}
\ln P\left(\mathbf{t} \mid \boldsymbol{\theta}\right) &= \ln \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{t_{nk}} \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)^{t_{nk}} \\
&= \sum_{n=1}^{N} \ln \prod_{k=1}^{K} \pi_k^{t_{nk}} \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)^{t_{nk}} \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} \ln \pi_k^{t_{nk}} \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)^{t_{nk}} \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left(\ln \pi_k + \ln \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right) \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} \left(t_{nk} \ln \pi_k + t_{nk} \ln \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right) . \qquad (1)
\end{aligned}
$$

Expanding (1) will greatly help us in the upcoming derivations:

We have to find the maximum likelihood solution for the class-specific priors, means, and covariance matrices. Starting with the priors, we have to take the derivative of (2), set it equal to 0, and solve for the prior. However, we have to maintain the constraint

This is done by using a Lagrange multiplier λ, and instead maximizing





Just like the class-specific mean vector is just the mean of the vectors of the class, **the class-specific covariance matrix is just the covariance of the vectors of the class**, and we end up with our maximum likelihood solutions (5), (6), and (7). Thus, we can classify using the following

XGBoost:

XGBoost is an implementation of Gradient Boosted decision trees. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

The prediction scores of each individual decision tree then sum up to get If you look at the example, an important fact is that the two trees try to *complement* each other. Mathematically, we can write our model in the form

where, K is the number of trees, f is the functional space of F, F is the set of possible CARTs. The objective function for the above model is given by:

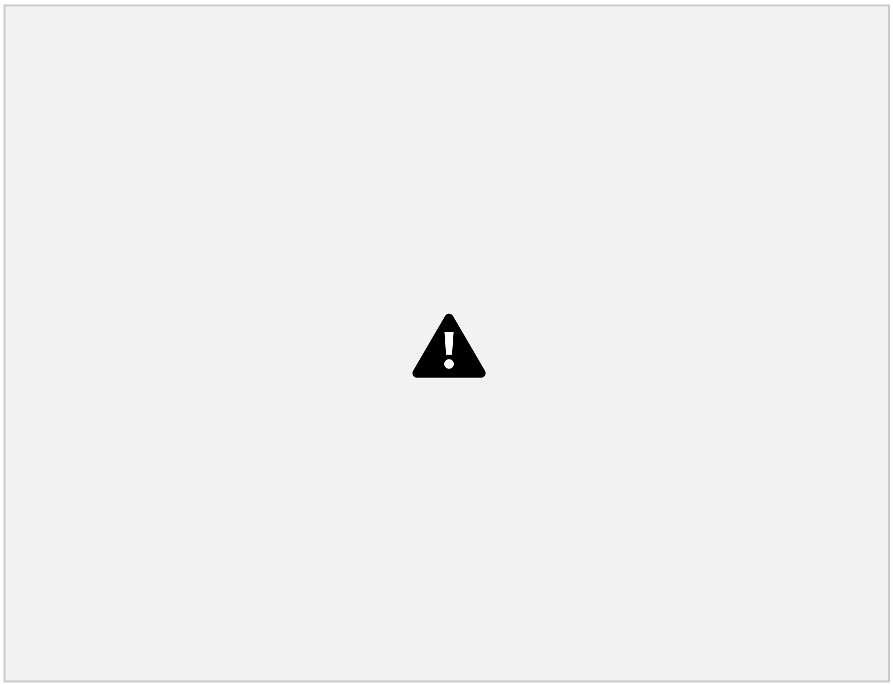
6. RESULTS

S.No.	Algorithm	Accuracy	Precision	Recall	F1 score Sensitivity Specificity AUC (ROC) Error rate	TPR	FPR	Execution time
1.	Logistic regression without regularization	0.873	0.755	0.353	0.481 0.353 0.977 0.665 0.127	0.353	0.023	0.041
2.	KNeighborsClassifier	0.854	0.763	0.180	0.291 0.180 0.989 0.584 0.146	0.180	0.011	0.412
3.	Linear SVM	0.663	0.310	0.837	0.453 0.837 0.628 0.678 0.337	0.837	0.372	0.420
4.	Naive Bayes	0.802	0.434	0.616	0.509 0.616 0.839 0.727 0.198	0.616	0.161	0.012
5.	Decision Tree(gini index)	0.735	0.363	0.783	0.496 0.783 0.725 0.754 0.265	0.783	0.275	0.0254

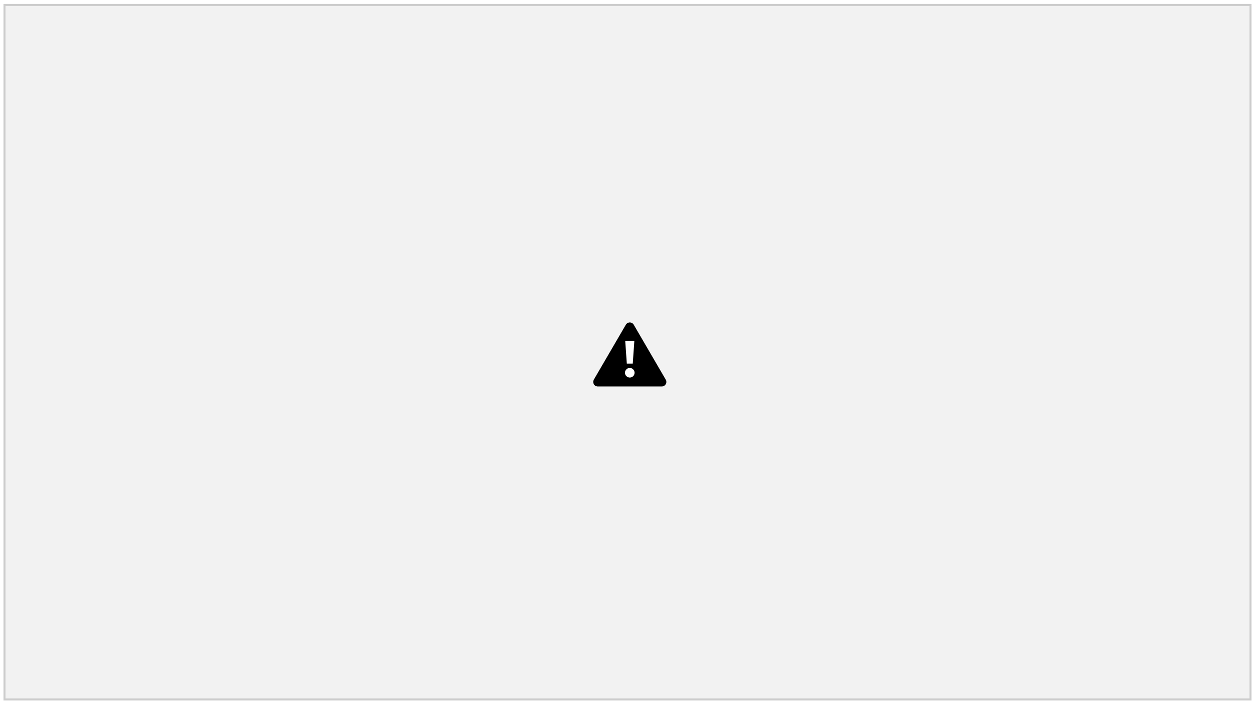
6.	Random Forest	0.894	0.748	0.547	0.632 0.547 0.963 0.743 0.106	0.547	0.037	0.243
7.	AdaBoost	0.859	0.576	0.582	0.579 0.582 0.914 0.741 0.141	0.582	0.086	0.0558
8.	MLPCl assi fier	0.863	1.00	0.017	0.033 0.017 1.00 0.509 0.164 0.496 0.783 0.725 0.754	0.017	0.00	0.536
9.	Quadratic Discrim ina nt Analysis	0.735	0.363	0.783	0.265	0.783	0.275	0.0254
10.	Gradient Boosting Classifier	0.882	0.643	0.495	0.559 0.495 0.951 0.723 0.118	0.495	0.049	0.243
11.	Light Gradient Boosting Machine	0.893	0.728	0.574	0.642 0.574 0.957 0.766 0.107	0.574	0.043	0.0104
12.	XGBoost	0.897	0.743	0.584	0.654 0.584 0.960 0.772 0.103	0.584	0.040	0.5633
13.	Extra tree classifier	0.887	0.783	0.448	0.570 0.448 0.975 0.714 0.113	0.448	0.025	0.714

Confusion Matrices:

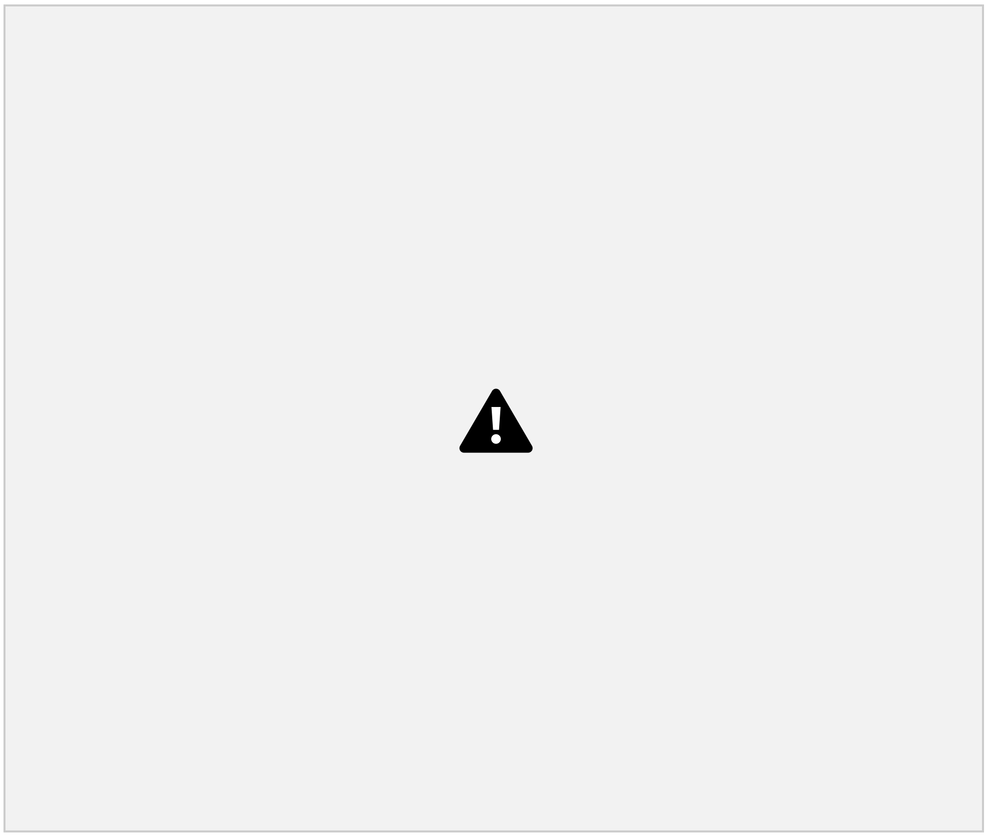
KNN:



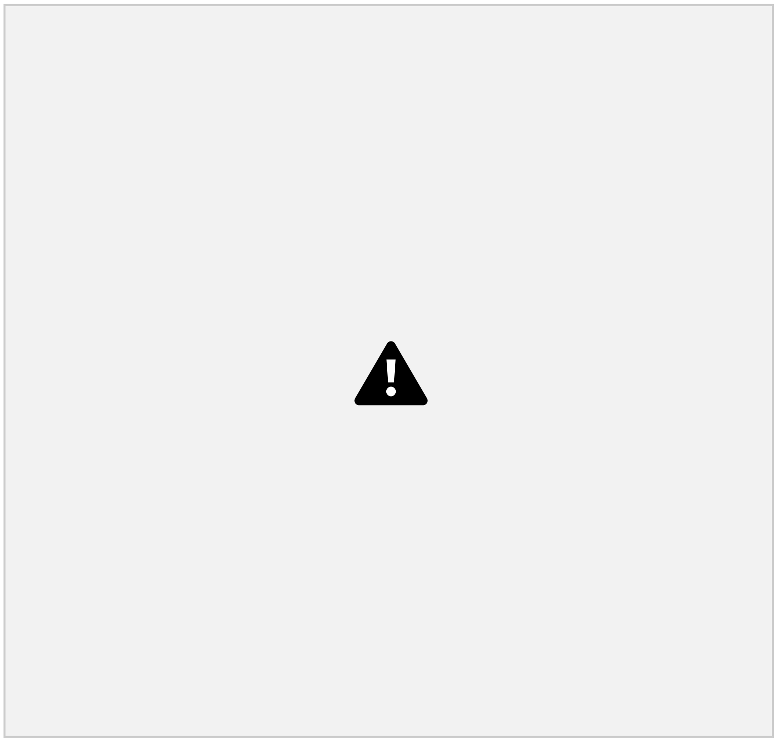
Logistic Regression :
Naive Baye:



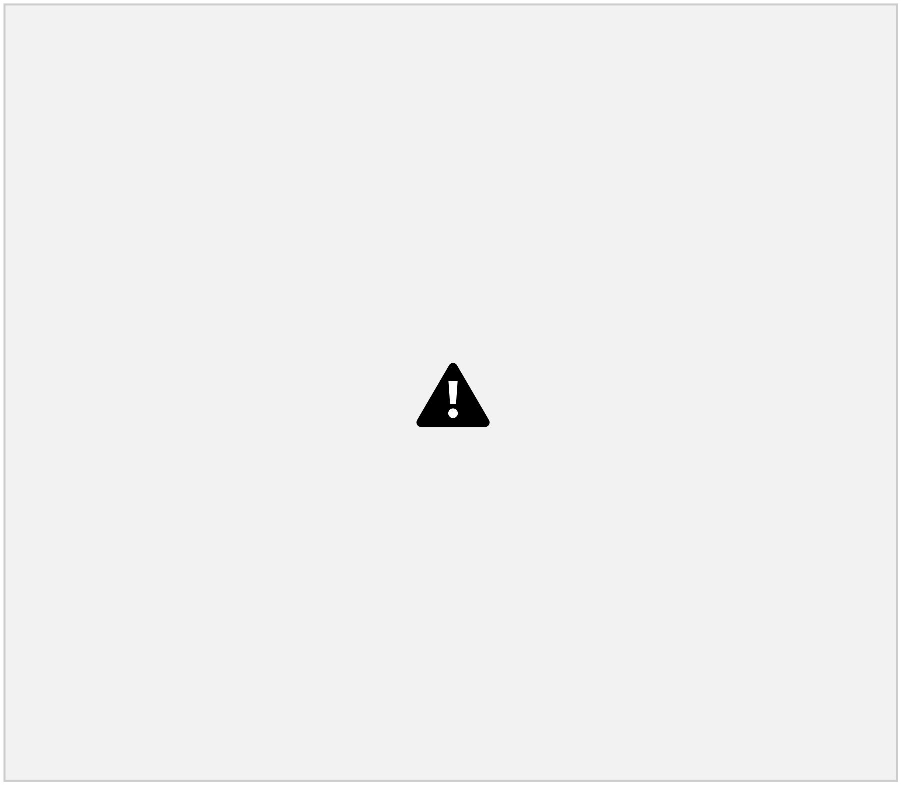
Random Forest:



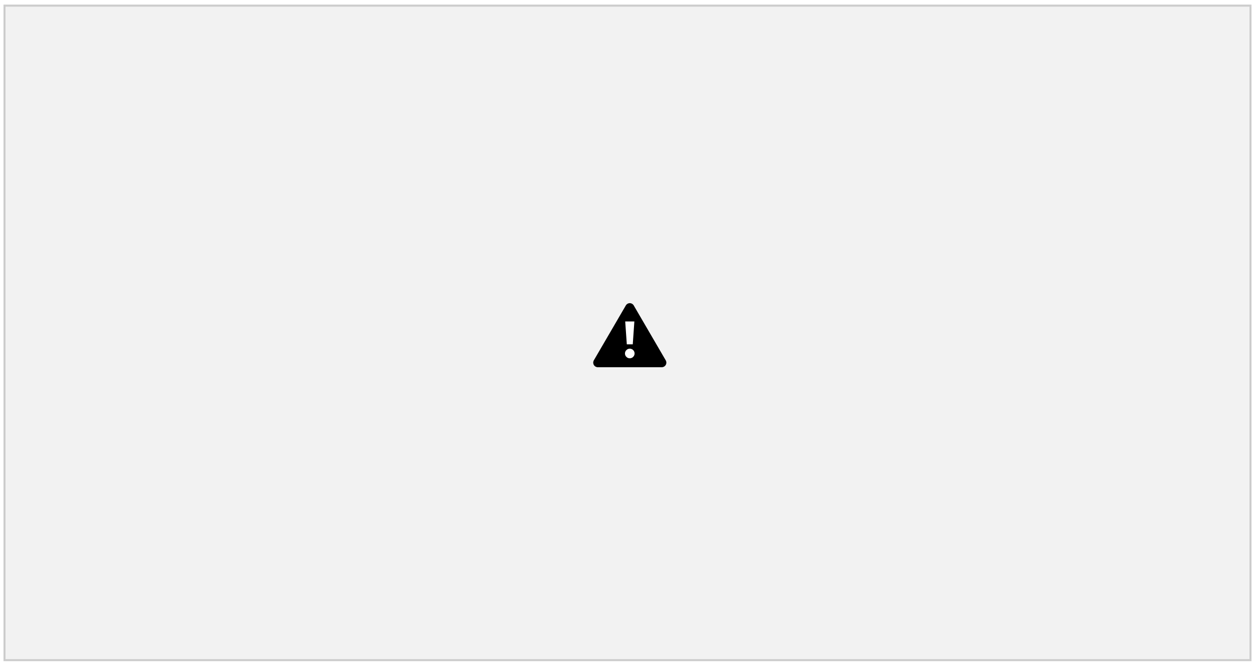
Gradient Boosting Classifier:



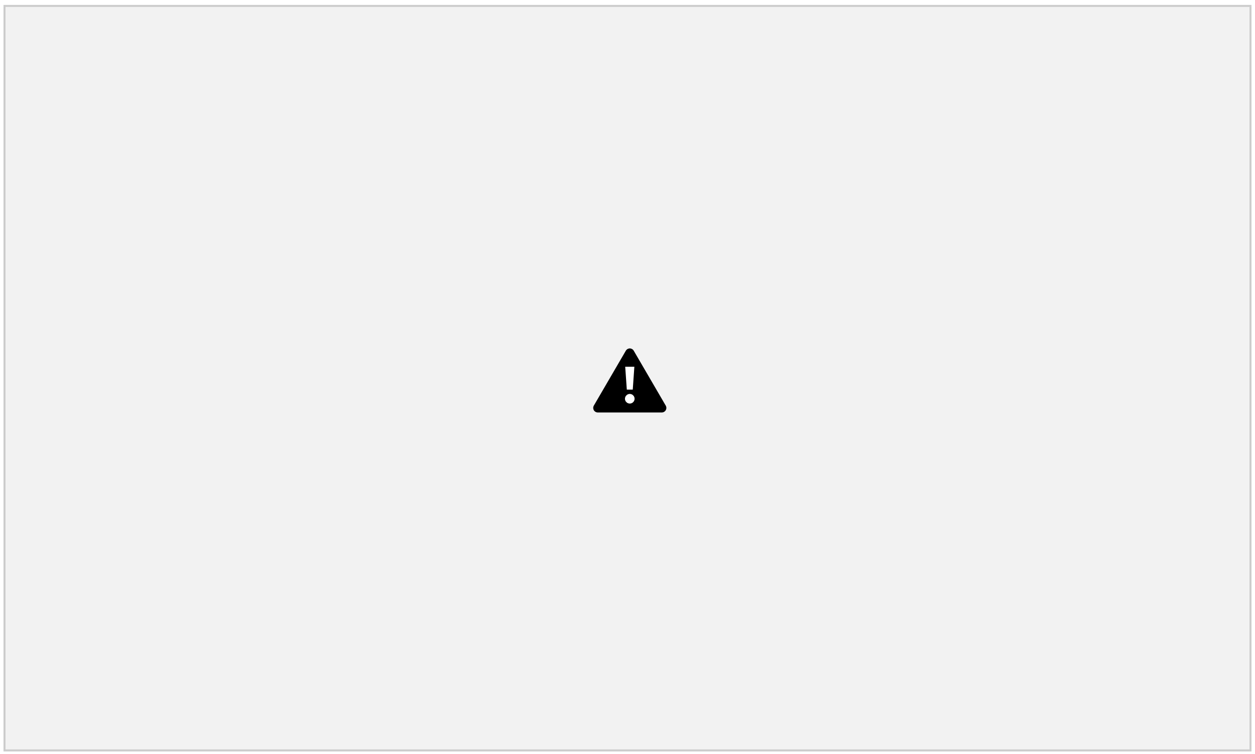
Light Gradient Boosting Classifier:



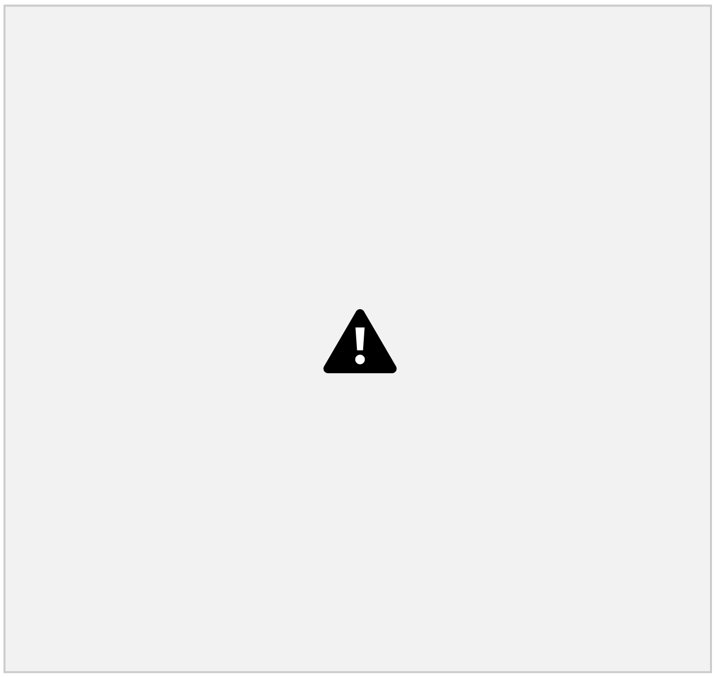
XGBOOST



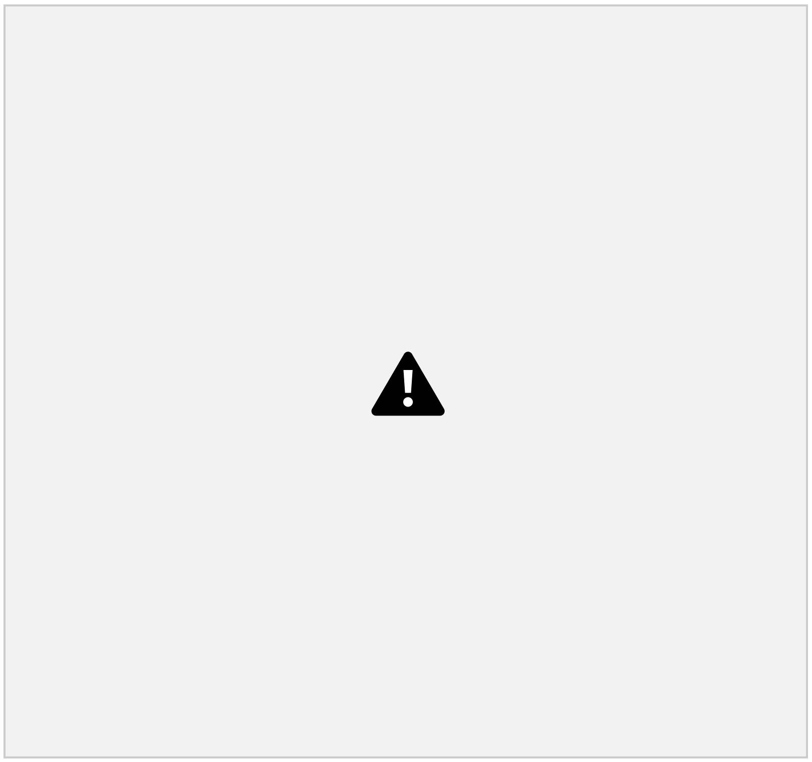
MLP:



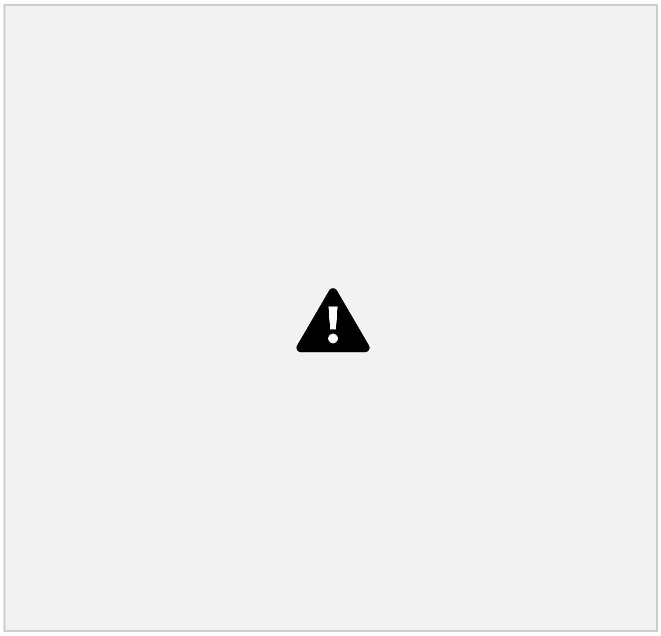
Ada Boost:



Extra Tree:



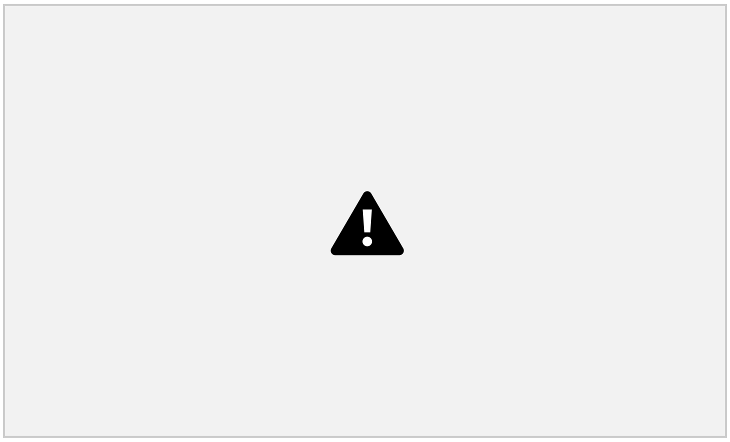
Quadratic Discriminant Analysis

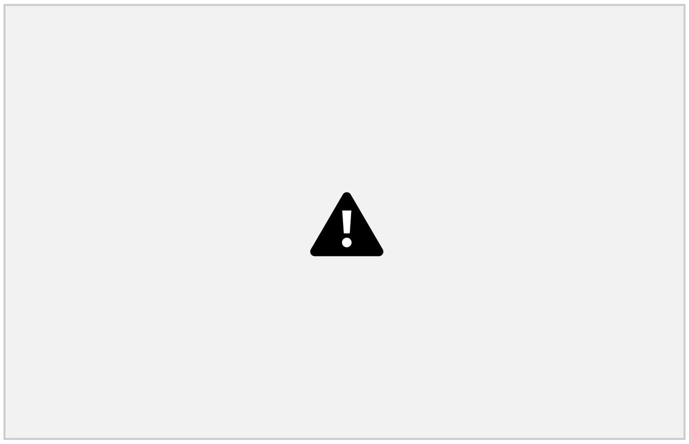


Linear SVM

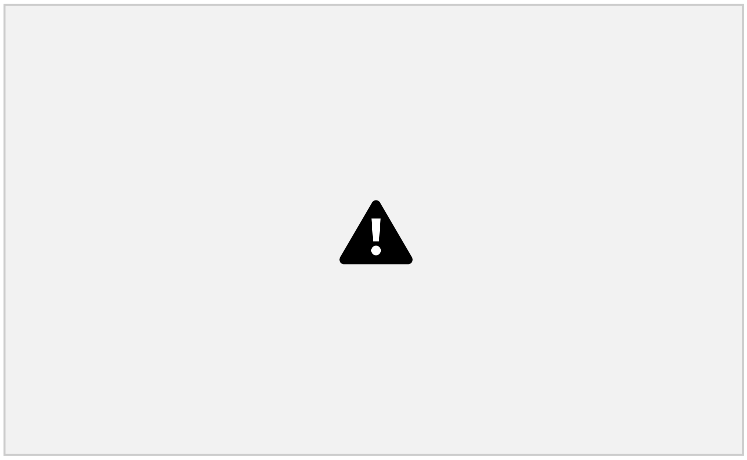
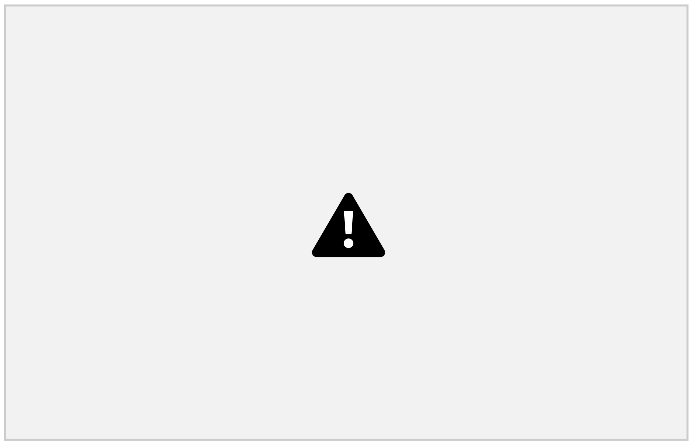
ROC Curves:

Logistic Regression KNN

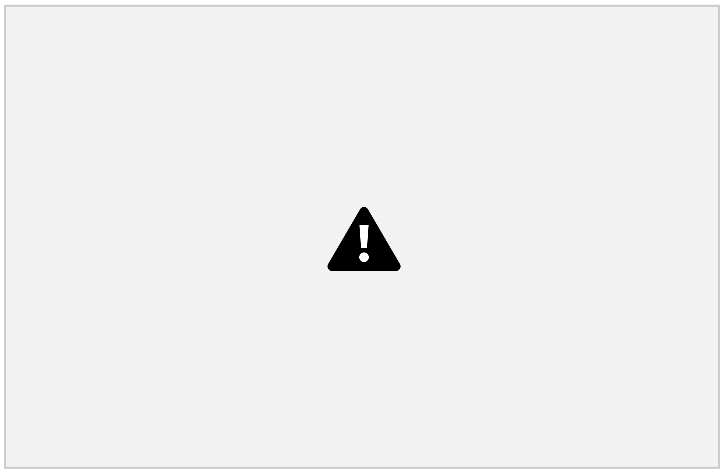
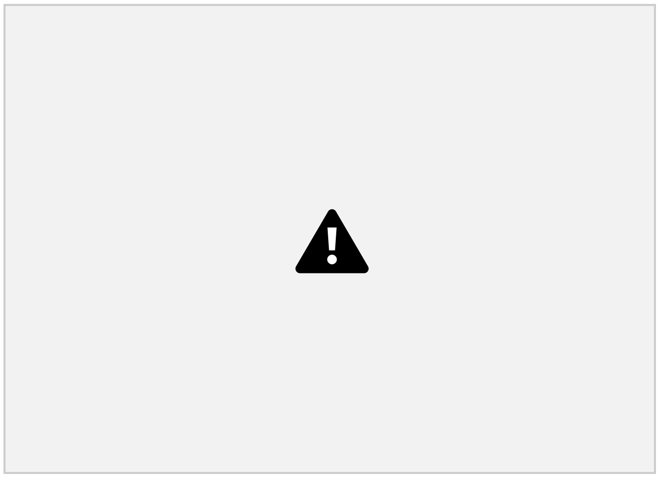




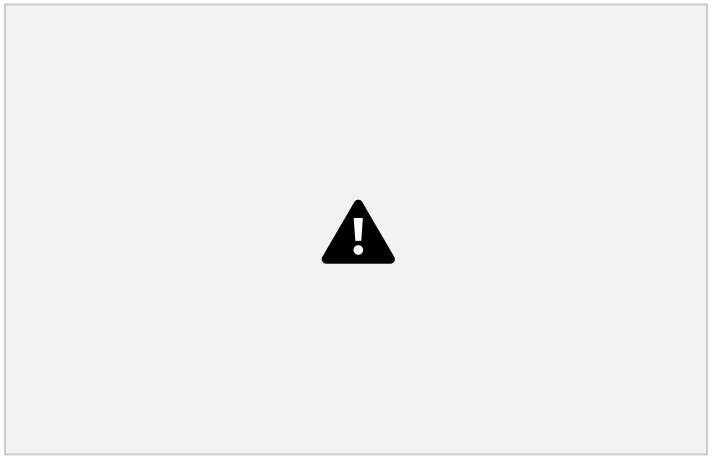
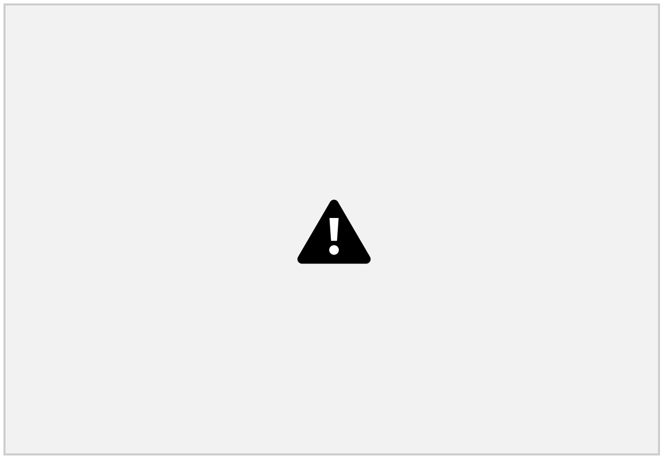
Naive Baeye: Random Forest:

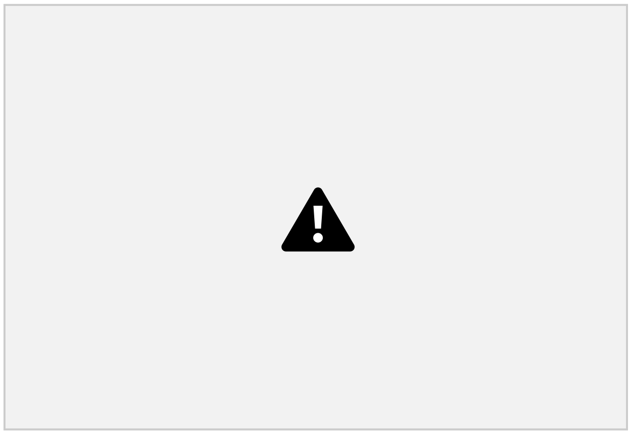
Gradient Boosting: Light Gradient Boosting:

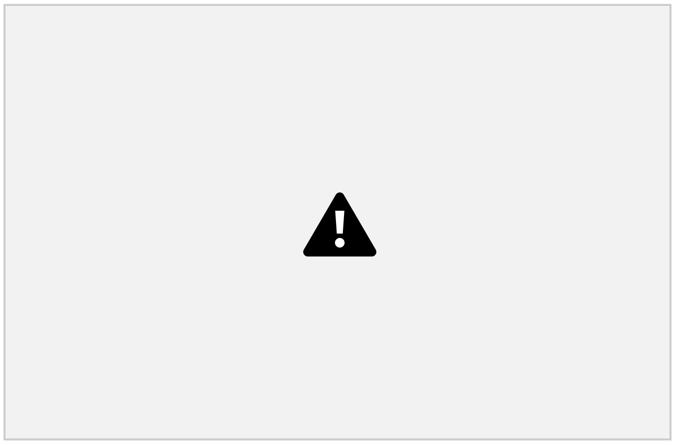



Decision Tree: Extra tree:

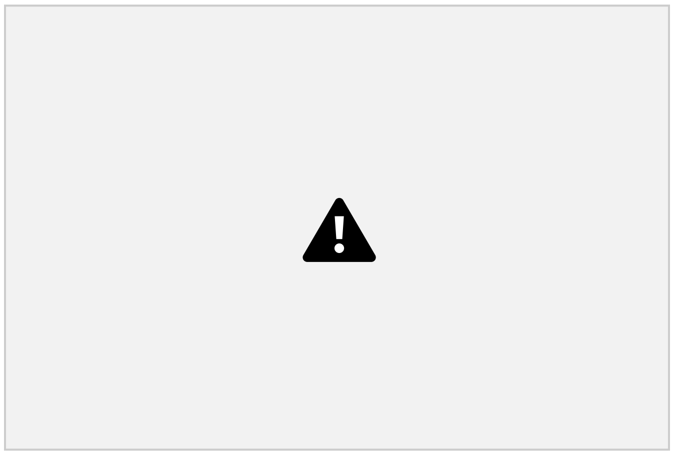



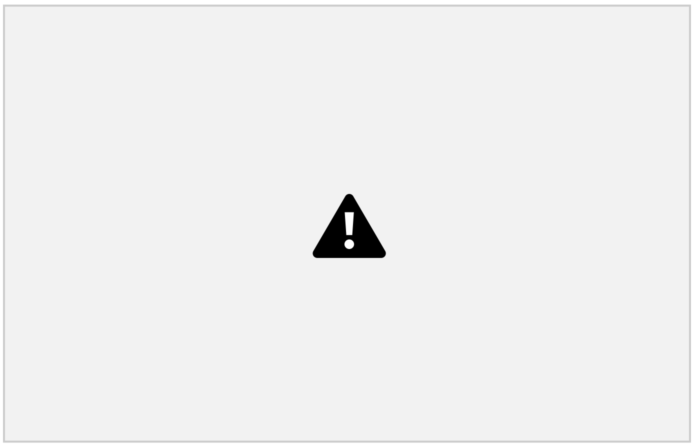
Quadratic Discriminant XGB:





MLP ADA Boost:





6. CONCLUSION

There are so many machine learning algorithms available, so, it is necessary to choose among those algorithms depending on the dataset being processed. Our data was based on the online shoppers purchasing intention so we have to compare the following results of different algorithms and observed that the relative accuracy of these algorithms were highest among the other algorithms and their values are quite relative to each other. Those algorithms are Random forest, Light gradient boost and XG boost.

Lowest execution time is required by light gradient boosting with accuracy of 0.893 and error of 0.118 whereas MCP classifier gives precision of one which means that every result retrieved by a search was relevant (but says nothing about whether all relevant documents were retrieved).
Random forest classifier gives accuracy of 0.894 with error of 0.106. XG Boost gives an accuracy of 0.897 which is the highest from all other algorithms and an error of 0.103 which is also the lowest from all other algorithms.

7. REFERENCES

1. Carmona CJ, Ramírez-Gallego S, Torres F, Bernal E, del Jesús MJ, García S (2012) Web usage mining to improve the design of an e-commerce website: OrOliveSur. com. Expert Syst Appl 39(12):11243–11249
2. Albert TC, Goes PB, Gupta A (2004) A model for design and management of content and interactivity of customer-centric web sites. MIS Q 28(2):161–182
3. Moe WW (2003) Buying, searching, or browsing: differentiating between online shoppers using in-store navigational clickstream. J Consum Psychol 13(1–2):29–39
4. Ding AW, Li S, Chatterjee P (2015) Learning user real-time intent for optimal dynamic web page transformation. Inf Syst Res 26(2):339–359
5. Keng Kau A, Tang YE, Ghose S (2003) Typology of online shoppers. J Consum Mark 20(2):139–156
6. Fernandes RF, Teixeira CM (2015) Using clickstream data to analyze online purchase intentions. Master's thesis, University of Porto