Identifying the best model for Class Imbalance Data in a Multiclass Problem

Submitted By
Nirmal Raj Eganathan
Deekshith Basvoju
Batch 56 – INSOFE, Bengaluru.

Guide: Dr. Soumen Manna, Senior Data Scientist.

International School of Engineering, Bengaluru.
31st July 2019

Table of Contents

Abstract

The Purpose of this internship is to do a research on an Infinite possibility of imbalanced class of data and we would like to investigate what are the best models through all possible imbalanced situation of a data set.

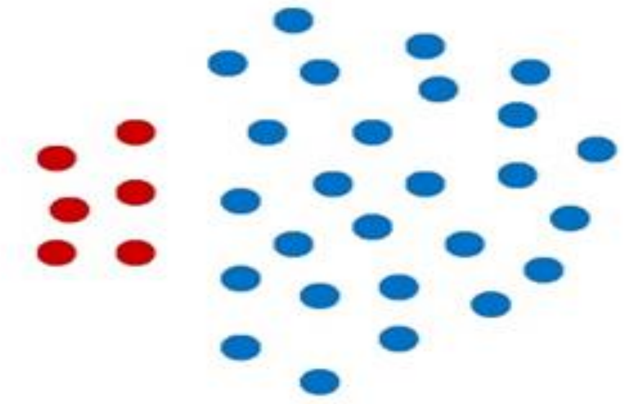
1.Introduction

1.1 Problem Statement:

Generating Data points on our own and make the balance dataset in to imbalance data in different proportions and check the classification report on all possible situation of the data and suggesting which are the models are giving best results.

2.Generating Data points:

2.1 What is Class – Imbalance?



Imbalance Class - Distributions

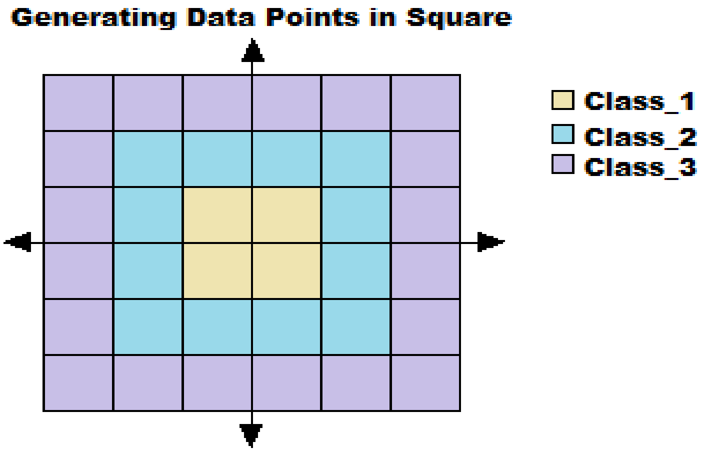
There is no particular definition for imbalanced class of data. In general, data that is not balanced is called imbalanced. Referring to the above that at least one of the class having significantly less number of training examples or the examples in the training data belonging to one class heavily outnumber the examples in the other class.

Currently, most of the Machine learning algorithms assume the training data to be balanced like SVM, Logistic-Regression, Naïve-Bayes etc.,

Last few decades, some effective methods have been proposed to attack this problem like up-sampling, down-sampling, Smote etc…

2.2 Getting Dataset:

Generating Data Points in Square Pattern Keeping a boundary classifying the data points as belongs to multiple class and name them as class_1, class_2 and class_3.

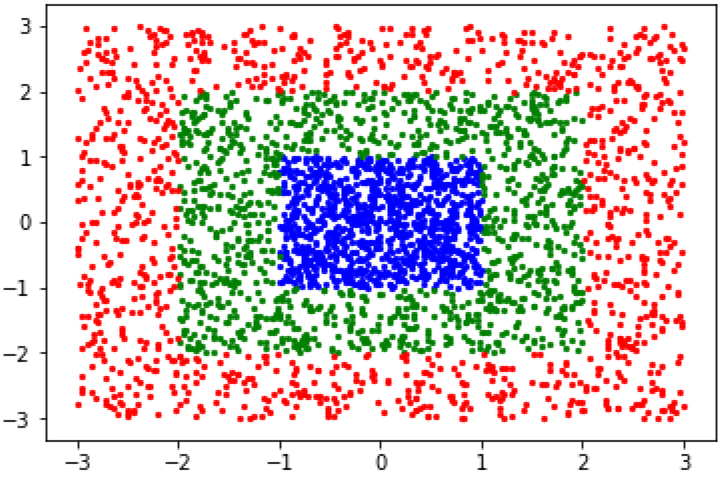


The data-points created here with the help of a graph generating a random data points quadrant wise and once every data points generated we are concatenating every data points and making a target column and assigning those data points to one class.

Similarly, for other two classes we follow the same steps to generate data points and assign them to their respective class.

Now Concatenate All the three classes and make it a Data frame.

Output After creating Data Points



For our Analysis we want our data set to be misclassified so adding impurity will make the data points in the dataset fall under different class and making it misclassified though it will be a good representation for our analysis in the future.

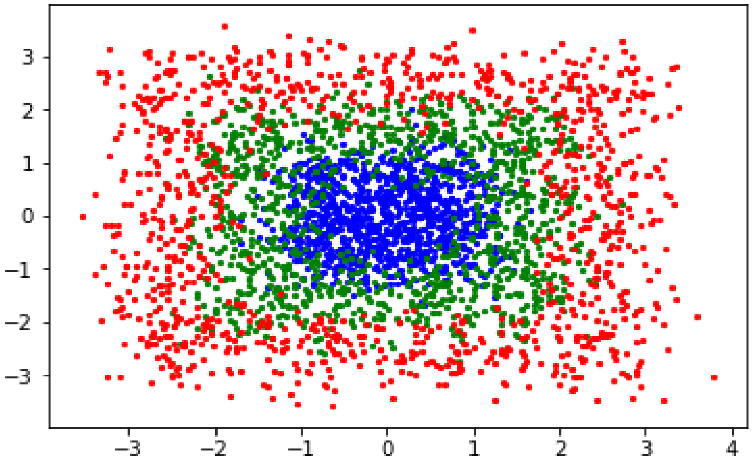
2.3 Adding Noise to Data:

For doing Classification Experiments in Machine Learning Algorithms using our created dataset which is clearly classified we are adding some jitter points to every data points to make every data points fall under different class and make them misclassify in itself. Following Code will explain how we are adding the noise to the data.

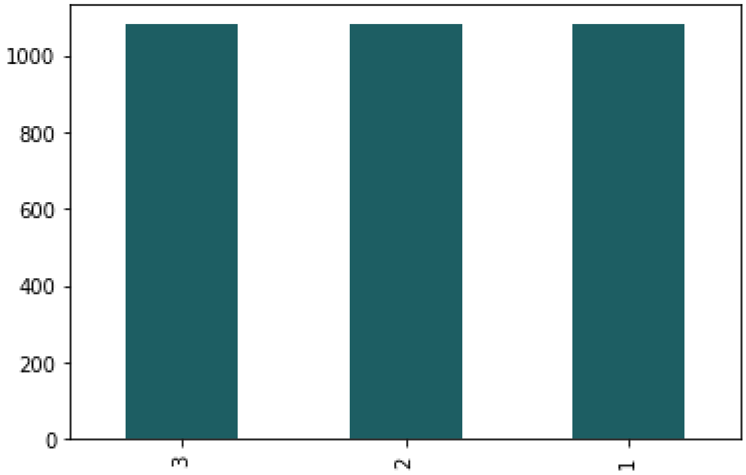
Code For Adding Noise :

```
x_noise = [ ]                               ------------Make a list for X column
y_noise = [ ]                               ------------similarly for Y column
for index,row in data.iterrows():           -------using loop for each & every points
    mu,sigma = row['X'],0.3                 ------Generating data point in interval of value and 0.3
    noise_x = np.random.normal(mu,sigma)-----Random Data point between two values
    x_noise.append(noise_x)                  ------Append those values in the list
for index,row in data.iterrows():            -----Similarly for Y column
    mu,sigma = row['Y'],0.3
    noise_y = np.random.normal(mu,sigma)
    y_noise.append(noise_y)
x_noise = pd.DataFrame(x_noise
x_noise.rename(columns = {0:'X',},inplace = True)
y_noise = pd.DataFrame(y_noise)
y_noise.rename(columns = {0:'Y',},inplace = True)
```

Now convert that appended List of X and Y values in to a Data Frame and make a dataset with a noise and assign the targets respect to them. After adding Noise to the data,



Is the Dataset Imbalanced?...............No



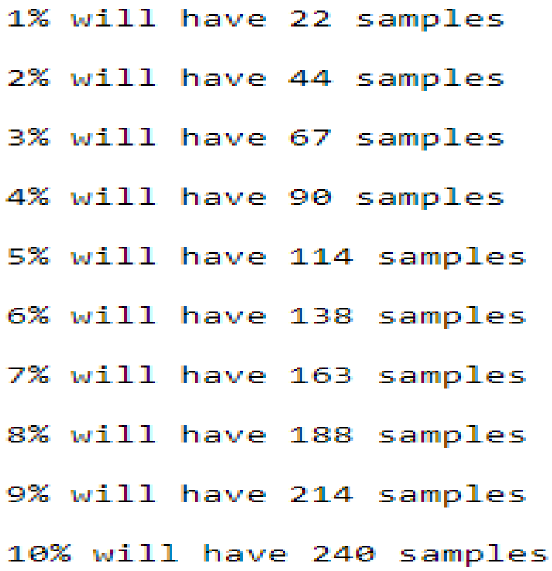
Every Class got Balanced though our analysis is for Imbalanced data we make this dataset an imbalance for our further process.

2.4 Proportions Considered for Each Class

Now let’s make the balance dataset to imbalance by making one class with the proportion of samples like 1%,2%,3%.......10% keeping other classes same.

1%? Means how many samples should we take to make that class as minority?

$$Samples\ to\ take\ for\ making\ one\ class\ as\ minority = \frac{data\ points\ in\ the\ other\ classes *percentage}{1-percentage}$$



```
1% will have 22 samples
2% will have 44 samples
3% will have 67 samples
4% will have 90 samples
5% will have 114 samples
6% will have 138 samples
7% will have 163 samples
8% will have 188 samples
9% will have 214 samples
10% will have 240 samples
```

Example:

If the three classes have shape (1080,3) (1080,3) (1080,3)

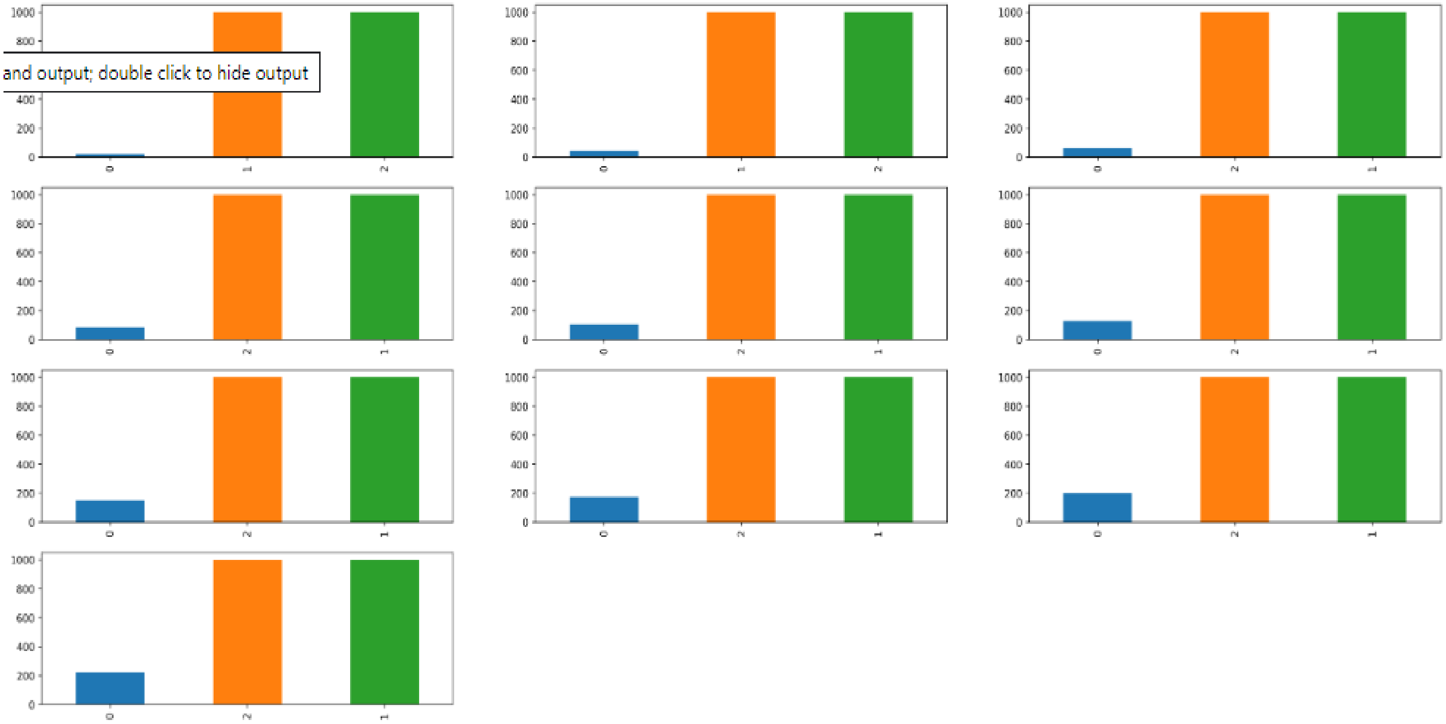
$$if\ its\ 1\%, y = \frac{1}{100} = 0.01$$

$Similarly\ for\ every\ percentages$

$$samples = \frac{(1080+1080)*0.01}{1-0.01} = 21.81 = 22$$

Total 22 samples for making that class as minority, similarly for other percentages we calculated as shown.

After Creating the Dataset Class_1 as minority and created 10 data set as per percentage proportions. The Count of the records in each class is shown in the figure.



Similarly, for other 2 classes create a data set. With the 30 data sets created try to implement every Machine Learning Algorithms and collect the Classification report on only Minority class and see which algorithms performing well on the given sample of minority class.

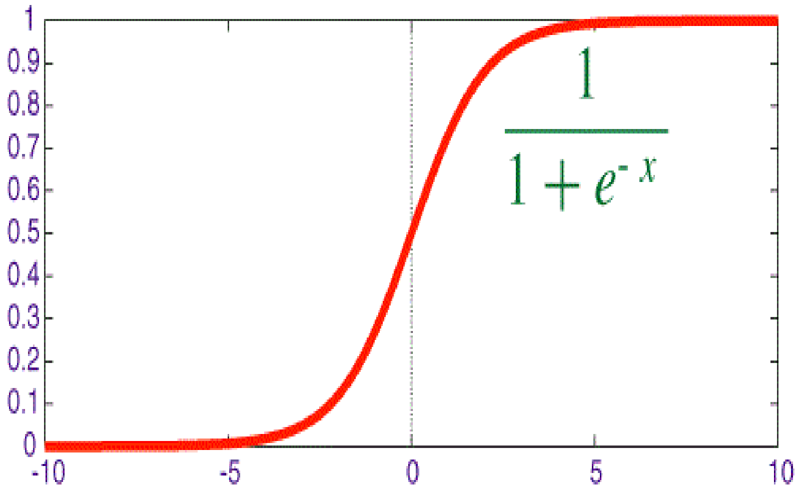
3.Models Built

3.1 Logistic Regression:

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

Logistic Regression is one of the most popular ways to fit models for categorical data, especially for binary response data in Data Modelling. It is the most important (and probably most used) member of a class of models called generalized linear models. Unlike linear regression, logistic regression can directly predict probabilities (values that are restricted to the (0,1) interval); furthermore, those probabilities are well-calibrated when compared to the probabilities predicted by some other classifiers, such as Naive Bayes. Logistic regression preserves the marginal probabilities of the training data. The coefficients of the model also provide some hint of the relative importance of each input variable.

Logistic Regression is used when the dependent variable (target) is categorical.



For example,

To predict whether an email is a spam (1) or (0)

Whether the tumour is malignant (1) or not (0)

Logistic Regression Assumptions:

1. Binary logistic regression requires the dependent variable to be binary.
2. For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.
3. Only meaningful variables should be included.
4. The independent variables should be independent of each other.i.e., the model should have little or no multi-collinearity.
5. The independent variables are linearly related to the log odds.
6. Logistic regression requires quite large sample sizes.

Even though logistic (**logit**) regression is frequently used for binary variables (2 classes), it can be used for categorical dependent variables with more than 2 classes. In this case, it’s called Multinomial Logistic Regression.

Types of Logistic Regression:

1. **Binary Logistic Regression**: The categorical response has only two 2 possible outcomes. E.g.: Spam or Not
2. **Multinomial Logistic Regression:** Three or more categories without ordering. E.g.: Predicting which food is preferred more (Veg, Non-Veg, Vegan)
3. **Ordinal Logistic Regression:** Three or more categories with ordering. E.g.: Movie rating from 1 to 5.

Sigmoid or Logistic Function

The Sigmoid Function curve looks like a S-shape. The main reason why we use sigmoid function is because it exists between (0 to 1).

The function is **differentiable**.That means, we can find the slope of the sigmoid curve at any two points.

$$E(Y|X=x)=f(x)=P(Y=1|X=x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{1+e^{x}}$$

3.2 Naïve-Bayes:

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:

$$\mathrm{P(c|x)} = \frac{\mathrm{P(x|c)} * \mathrm{P(c)}}{\mathrm{P(x)}}$$

$$P(c|X) = P(x_1|c) * P(x_2|c) * \ldots\ldots..* P(x_n|c) * P(c)$$

Above,

P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes)
P(c) is the prior probability of class
P(x|c) is the likelihood which is the probability of predictor given class
P(x) is the prior probability of the predictor

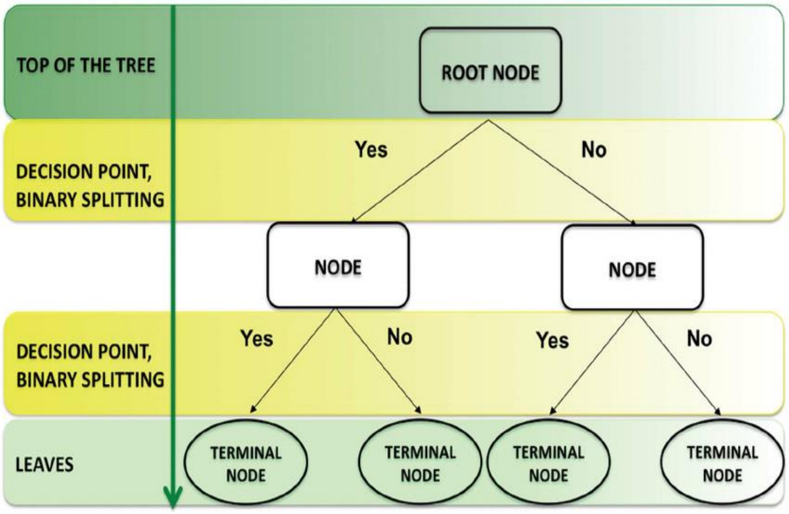
3.3 Decision Tree Classifier:

A decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter/differentiator in input variables.

Let's look at the basic terminology used with Decision trees:

1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called a decision node.
4. **Leaf/ Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
6. **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.

7. **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the children of the parent node.



Advantages: -

1. Simple to understand and to interpret. Trees can be visualized.
2. Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed.
3. Able to handle both numerical and categorical data. Other techniques are usually specialized in analyzing datasets that have only one type of variable.
4. They work well for both regression and classification tasks.
5. They require relatively less effort for training the algorithm.
6. They can be used to classify non-linearly separable data.
7. They're very fast and efficient compared to KNN and other classification algorithms.
8. Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by Boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.

Dis - Advantages: -

1. Decision-tree learners can create over-complex trees that do not generalize the data well. This is called overfitting. Mechanisms such as pruning - setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
2. Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
3. The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
4. Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

3.4 Random Forest Classifier:

Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object. Suppose training set is given as: [X1, X2, X3, X4] with corresponding labels as [L1, L2, L3, L4], random forest may create three decision trees taking input of subset for example,

[X1, X2, X3]
[X1, X2, X4]
[X2, X3, X4]

So finally, it predicts based on the majority of votes from each of the decision trees made.

Why use Random Forest Machine Learning Algorithm?

1. It maintains accuracy when there is missing data and is also resistant to outliers.ie we could have run this model on our original data without removing customers with NA values.
2. Capable of handling numerical, binary and categorical features, without scaling, transformation or modification. Our data has variables which have largely different values
3. Implicit feature selection as it gives estimates on what variables are important in the classification. We have few features and the models select what it presumes well.

Pros and cons: -

1. Both training and prediction are very fast, because of the simplicity of the underlying decision trees. In addition, both tasks can be straightforwardly parallelized, because the individual trees are entirely independent entities.
2. The multiple trees allow for a probabilistic classification: a majority vote among estimators gives an estimate of the probability.
3. The nonparametric model is extremely flexible, and can thus perform well on tasks that are under-fit by other estimators.

A primary disadvantage of random forests is that the results are not easily interpretable: that is, if you would like to draw conclusions about the *meaning* of the classification model, random forests may not be the best choice.

3.5 K-Nearest Neighbors (Knn):

The algorithm uses **‘feature similarity’** to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set.

The first step is to calculate the distance between the new point and each training point. There are various methods for calculating this distance, of which the most commonly known methods are – Euclidean, Manhattan (for continuous) and Hamming distance (for categorical).

1. Euclidean Distance:

Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (y).

$$Euclidean = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

2. Manhattan Distance:

This is the distance between real vectors using the sum of their absolute difference.

$$Manhattan = \sum_{i=1}^{k}|x_i - y_i|$$

3. Hamming Distance:

It is used for categorical variables. If the predicted value (x) and the real value (y) are the same, the distance D will be equal to 0 Otherwise D=1.

$$D_H = \sum_{i=1}^{k}|x_i - y_i|$$

$$x = y \ \rightarrow D = 0$$

$$x \neq y \ \rightarrow D = 1$$

Once the distance of a new observation from the points in our training set has been measured, the next step is to pick the closest points. The number of points to be considered is defined by the value of k.

3.6 Xg-Boost:

Xg-Boost (extreme Gradient Boosting) is an advanced implementation of the gradient boosting algorithm. Xg-Boost has proved to be a highly effective ML algorithm, extensively used in machine learning competitions and hackathons. Xg-Boost has high predictive power and is almost 10 times faster than the other gradient boosting techniques. It also includes a variety of regularization which reduces overfitting and improves overall performance. Hence it is also known as **regularized boosting** technique.

Let us see how Xg-Boost is comparatively better than other techniques:

1. Regularization:
   Standard GBM implementation has no regularization like Xg-Boost. Thus Xg-Boost also helps to reduce overfitting.
2. Parallel Processing:
   Xg-Boost implements parallel processing and is faster than GBM. Xg-Boost also supports implementation on Hadoop.
3. High Flexibility:
   Xg-Boost allows users to define custom optimization objectives and evaluation criteria adding a whole new dimension to the model.
4. Handling Missing Values:
   Xg-Boost has an in-built routine to handle missing values.
5. Tree Pruning:
   Xg-Boost makes splits up to the max_depth specified and then starts pruning the tree backward and removes splits beyond which there is no positive gain.
6. Built-in Cross-Validation:
   Xg-Boost allows a user to run cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run.

3.7 Ada-Boost:

Adaptive boosting or AdaBoost is one of the simplest boosting algorithms. Usually, decision trees are used for modeling. Multiple sequential models are created, each correcting the errors from the last model. AdaBoost assigns weights to the observations which are incorrectly predicted and the subsequent model works to predict these values correctly.

Below are the steps for performing the AdaBoost algorithm:

1. Initially, all observations in the dataset are given equal weights.
2. A model is built on a subset of data.
3. Using this model, predictions are made on the whole dataset.
4. Errors are calculated by comparing the predictions and actual values.
5. While creating the next model, higher weights are given to the data points which were predicted incorrectly.
6. Weights can be determined using the error value. For instance, higher the error more is the weight assigned to the observation.
7. This process is repeated until the error function does not change, or the maximum limit of the number of estimators is reached.

3.8 Gradient Boosting Classification:

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion as other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

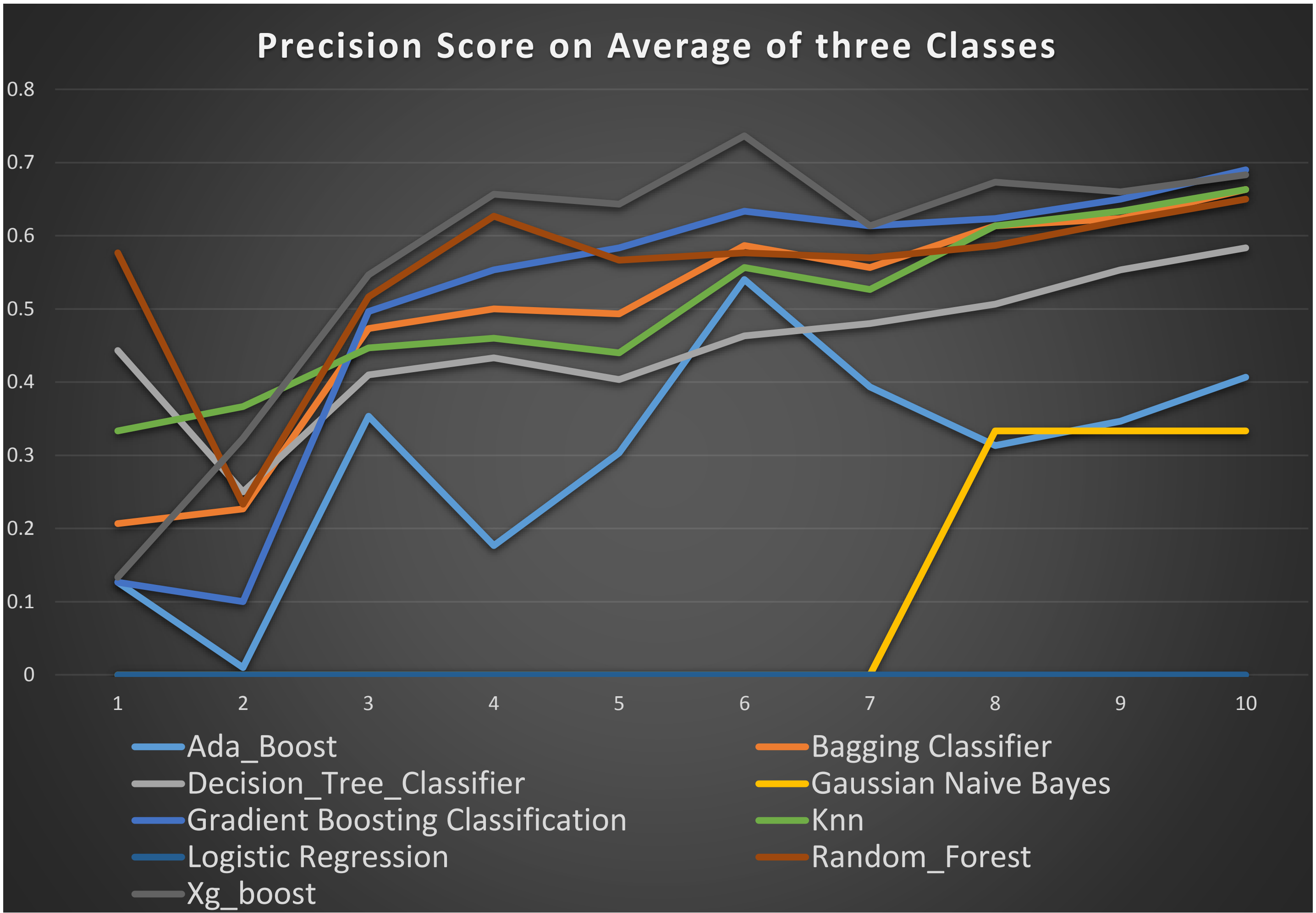
The idea of gradient boosting originated in the observation by Leo Breiman that boosting can be interpreted as an optimization algorithm on a suitable cost function. Explicit regression gradient boosting algorithms were subsequently developed by Jerome H. Friedman, simultaneously with the more general functional gradient boosting perspective of Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Then latter two papers introduced the view of boosting algorithms as iterative *functional gradient descent* algorithms. That is, algorithms that optimize a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in many areas of machine learning and statistics beyond regression and classification.

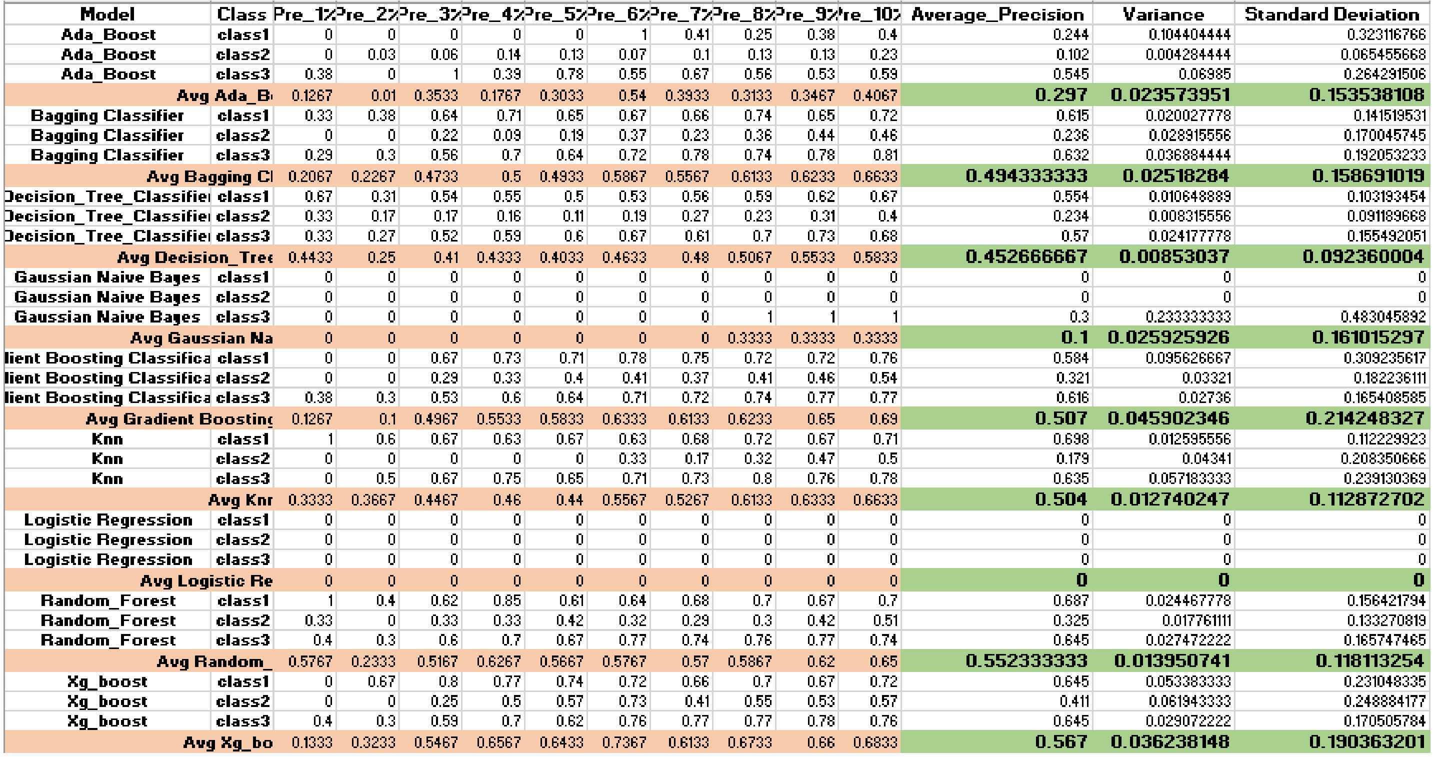
3.9 Bagging Classifier:

Bagging (Bootstrap Aggregating) is a widely used an ensemble learning algorithm in machine learning. The algorithm builds multiple models from randomly taken subsets of train dataset and aggregates learners to build overall stronger learner. In this post, we’ll learn how to classify data with Bagging Classifier class of a sklearn library in Python.

4. Reports:

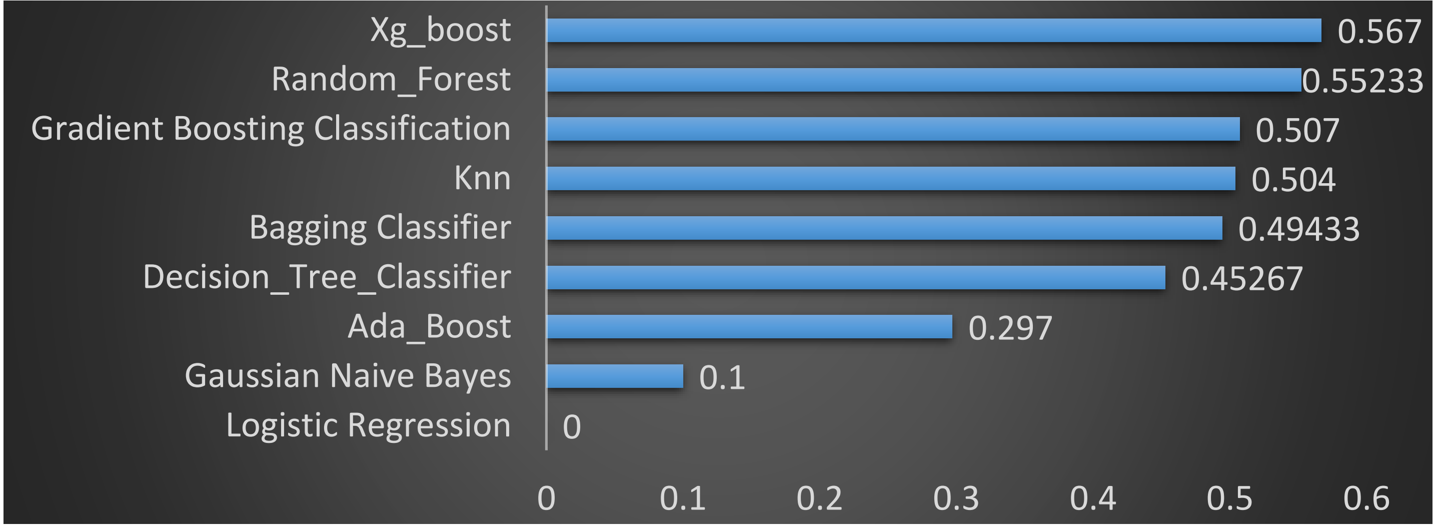
4.1 Model Building and their reports



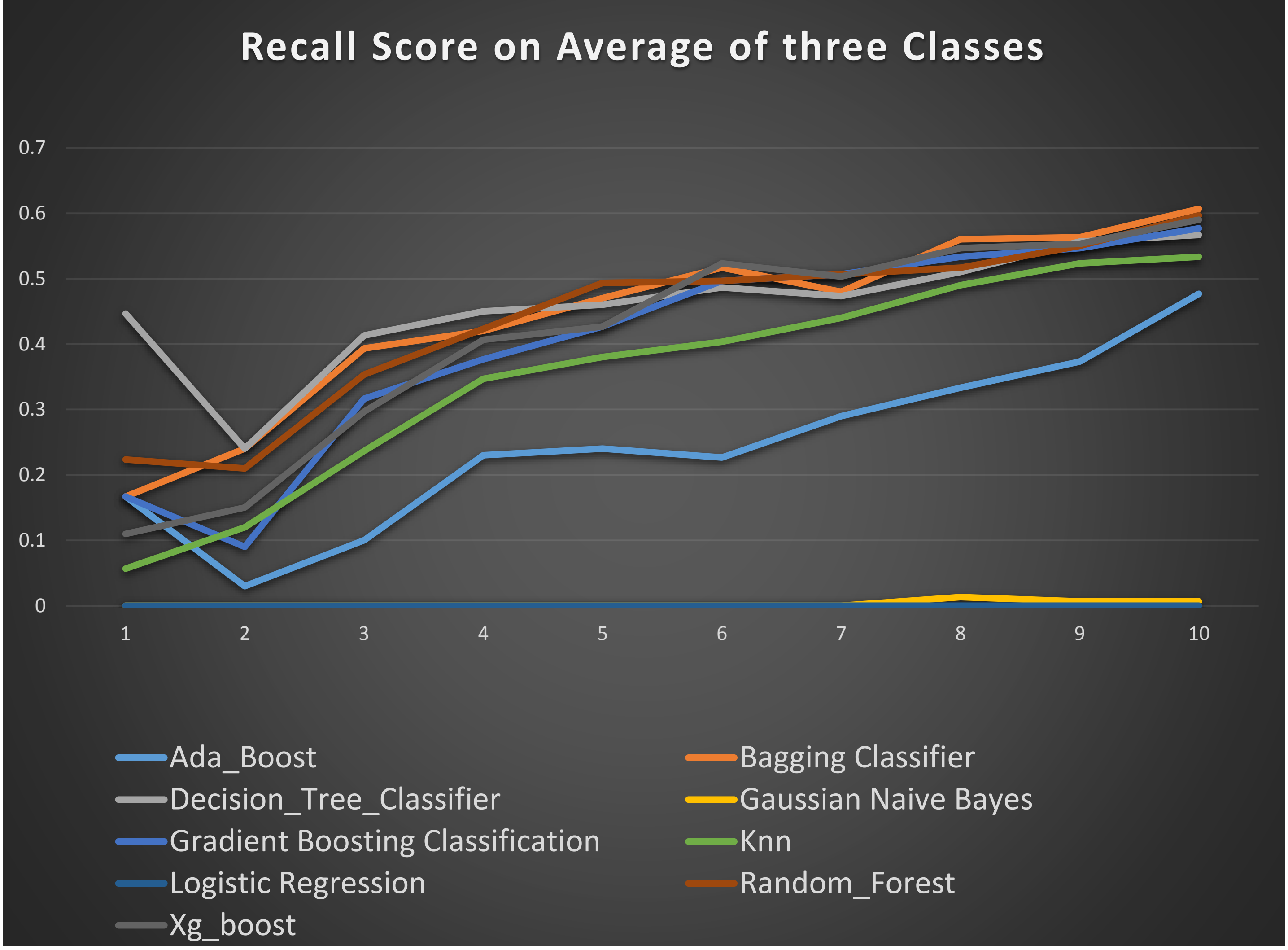


Model	Class	Pre_1%	Pre_2%	Pre_3%	Pre_4%	Pre_5%	Pre_6%	Pre_7%	Pre_8%	Pre_9%	Pre_10%	Average_Precision	Variance	Standard Deviation
Ada_Boost	class1	0	0	0	0	0	1	0.41	0.25	0.38	0.4	0.244	0.104404444	0.323116766
Ada_Boost	class2	0	0.03	0.06	0.14	0.13	0.07	0.1	0.13	0.13	0.23	0.102	0.004284444	0.065455668
Ada_Boost	class3	0.38	0	1	0.39	0.78	0.55	0.67	0.56	0.53	0.59	0.545	0.06985	0.264291506
Avg Ada_B		0.1267	0.01	0.3533	0.1767	0.3033	0.54	0.3933	0.3133	0.3467	0.4067	**0.297**	**0.023573951**	**0.153538108**
Bagging Classifier	class1	0.33	0.38	0.64	0.71	0.65	0.67	0.66	0.74	0.65	0.72	0.615	0.020027778	0.141519531
Bagging Classifier	class2	0	0	0.22	0.09	0.19	0.37	0.23	0.36	0.44	0.46	0.236	0.028915556	0.170045745
Bagging Classifier	class3	0.29	0.3	0.56	0.7	0.64	0.72	0.78	0.74	0.78	0.81	0.632	0.036884444	0.192053233
Avg Bagging Cl		0.2067	0.2267	0.4733	0.5	0.4933	0.5867	0.5567	0.6133	0.6233	0.6633	**0.494333333**	**0.02518284**	**0.158691019**
Decision_Tree_Classifie	class1	0.67	0.31	0.54	0.55	0.5	0.53	0.56	0.59	0.62	0.67	0.554	0.010648889	0.103193454
Decision_Tree_Classifie	class2	0.33	0.17	0.17	0.16	0.11	0.19	0.27	0.23	0.31	0.4	0.234	0.008315556	0.091189668
Decision_Tree_Classifie	class3	0.33	0.27	0.52	0.59	0.6	0.67	0.61	0.7	0.73	0.68	0.57	0.024177778	0.155492051
Avg Decision_Tree		0.4433	0.25	0.41	0.4333	0.4033	0.4633	0.48	0.5067	0.5533	0.5833	**0.452666667**	**0.00853037**	**0.092360004**
Gaussian Naive Bayes	class1	0	0	0	0	0	0	0	0	0	0	0	0	0
Gaussian Naive Bayes	class2	0	0	0	0	0	0	0	0	0	0	0	0	0
Gaussian Naive Bayes	class3	0	0	0	0	0	0	0	1	1	1	0.3	0.233333333	0.483045892
Avg Gaussian Na		0	0	0	0	0	0	0	0.3333	0.3333	0.3333	**0.1**	**0.025925926**	**0.161015297**
lient Boosting Classifica	class1	0	0	0.67	0.73	0.71	0.78	0.75	0.72	0.72	0.76	0.584	0.095626667	0.309235617
lient Boosting Classifica	class2	0	0	0.29	0.33	0.4	0.41	0.37	0.41	0.46	0.54	0.321	0.03321	0.182236111
lient Boosting Classifica	class3	0.38	0.3	0.53	0.6	0.64	0.71	0.72	0.74	0.77	0.77	0.616	0.02736	0.165408585
Avg Gradient Boosting		0.1267	0.1	0.4967	0.5533	0.5833	0.6333	0.6133	0.6233	0.65	0.69	**0.507**	**0.045902346**	**0.214248327**
Knn	class1	1	0.6	0.67	0.63	0.67	0.63	0.68	0.72	0.67	0.71	0.698	0.012595556	0.112229923
Knn	class2	0	0	0	0	0	0.33	0.17	0.32	0.47	0.5	0.179	0.04341	0.208350666
Knn	class3	0	0.5	0.67	0.75	0.65	0.71	0.73	0.8	0.76	0.78	0.635	0.057183333	0.239130369
Avg Knn		0.3333	0.3667	0.4467	0.46	0.44	0.5567	0.5267	0.6133	0.6333	0.6633	**0.504**	**0.012740247**	**0.112872702**
Logistic Regression	class1	0	0	0	0	0	0	0	0	0	0	0	0	0
Logistic Regression	class2	0	0	0	0	0	0	0	0	0	0	0	0	0
Logistic Regression	class3	0	0	0	0	0	0	0	0	0	0	0	0	0
Avg Logistic Re		0	0	0	0	0	0	0	0	0	0	**0**	**0**	**0**
Random_Forest	class1	1	0.4	0.62	0.85	0.61	0.64	0.68	0.7	0.67	0.7	0.687	0.024467778	0.156421794
Random_Forest	class2	0.33	0	0.33	0.33	0.42	0.32	0.29	0.3	0.42	0.51	0.325	0.017761111	0.133270819
Random_Forest	class3	0.4	0.3	0.6	0.7	0.67	0.77	0.74	0.76	0.77	0.74	0.645	0.027472222	0.165747465
Avg Random_		0.5767	0.2333	0.5167	0.6267	0.5667	0.5767	0.57	0.5867	0.62	0.65	**0.552333333**	**0.013950741**	**0.118113254**
Xg_boost	class1	0	0.67	0.8	0.77	0.74	0.72	0.66	0.7	0.67	0.72	0.645	0.053383333	0.231048335
Xg_boost	class2	0	0	0.25	0.5	0.57	0.73	0.41	0.55	0.53	0.57	0.411	0.061943333	0.248884177
Xg_boost	class3	0.4	0.3	0.59	0.7	0.62	0.76	0.77	0.77	0.78	0.76	0.645	0.029072222	0.170505784
Avg Xg_bo		0.1333	0.3233	0.5467	0.6567	0.6433	0.7367	0.6133	0.6733	0.66	0.6833	**0.567**	**0.036238148**	**0.190363201**

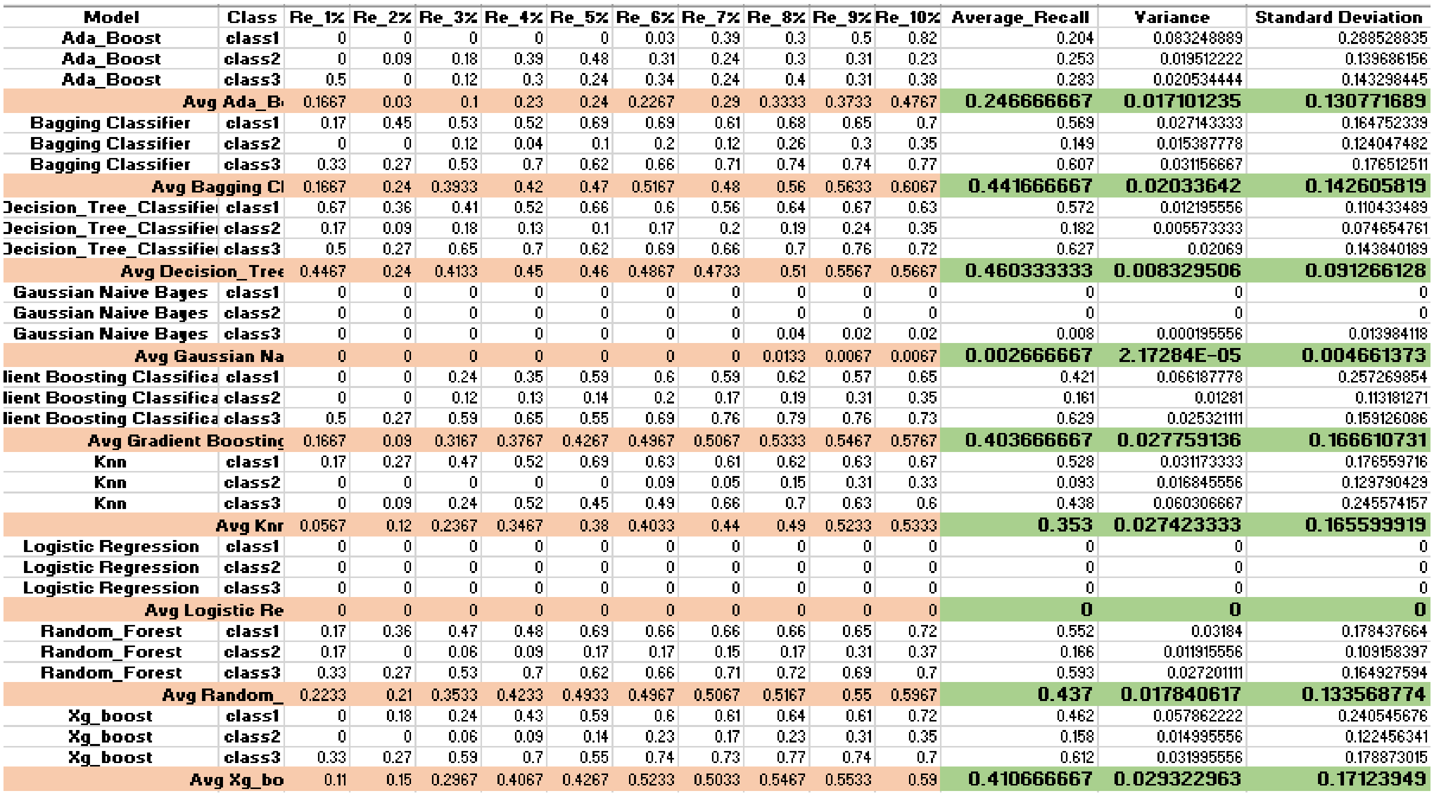
From the above it is seen that Naïve Bayes Algorithm and Logistic Regression is not performing well.



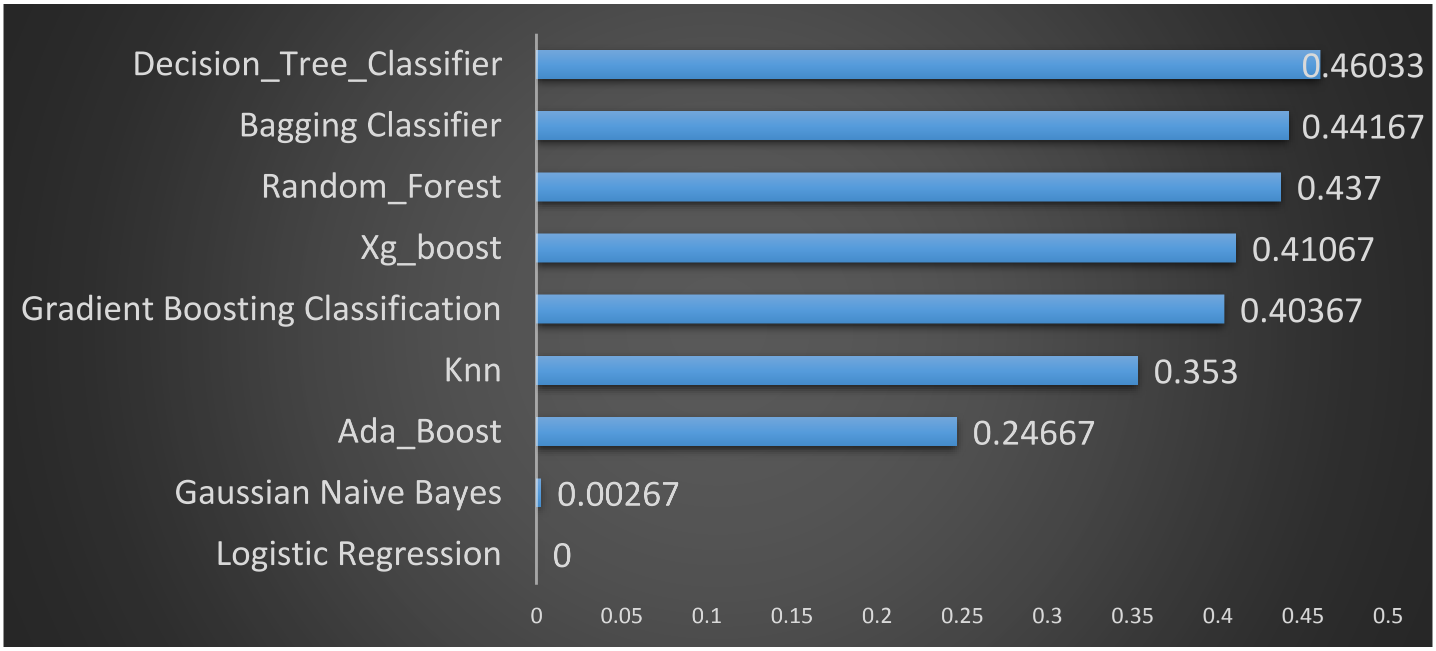
Overall when considering Precision score for imbalance data from our analysis Xg_Boost is performing well comparing other Machine Learning Algorithms



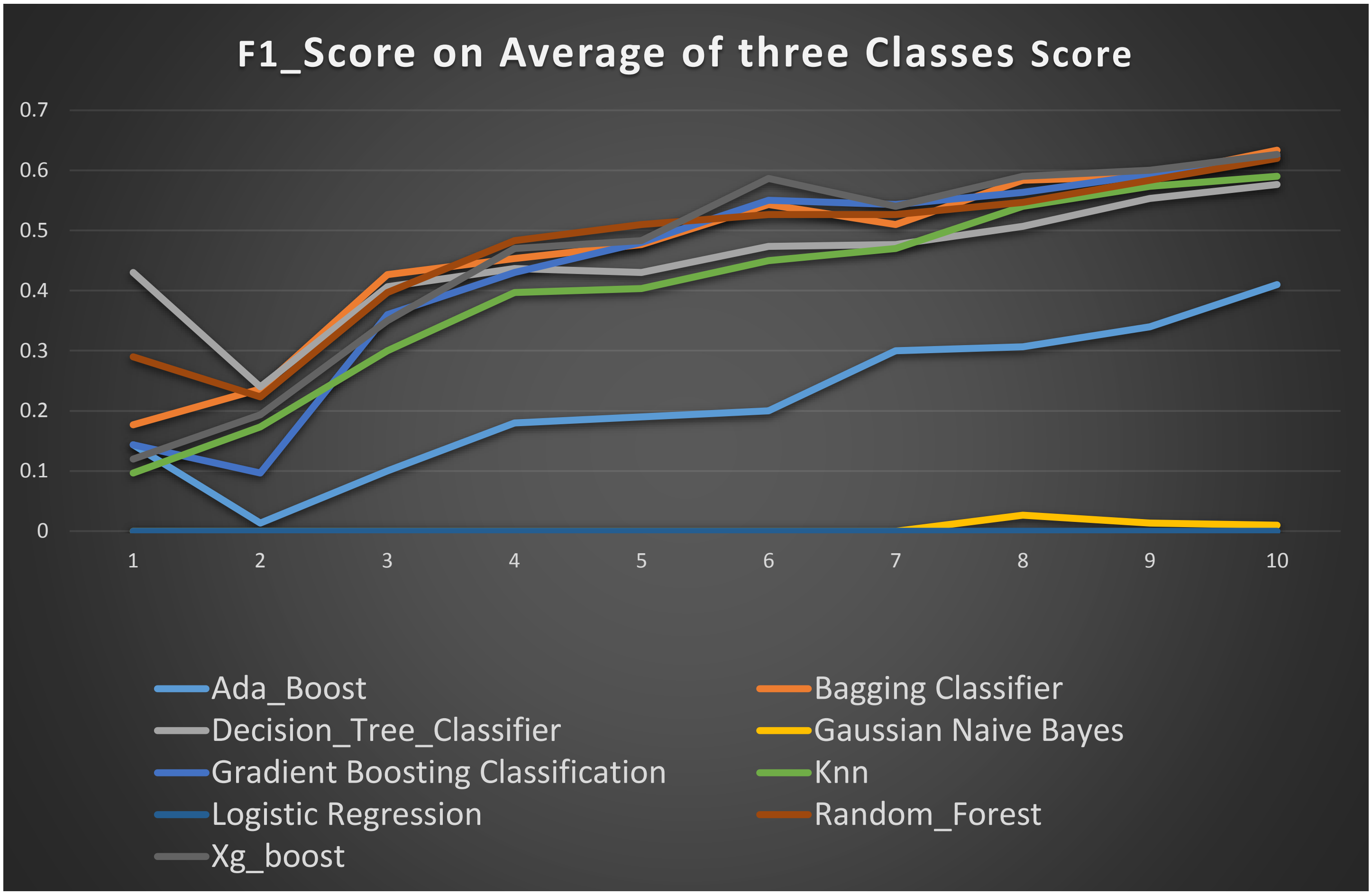
From the above it is seen that Naïve Bayes and Logistic Regression is not Performing well.



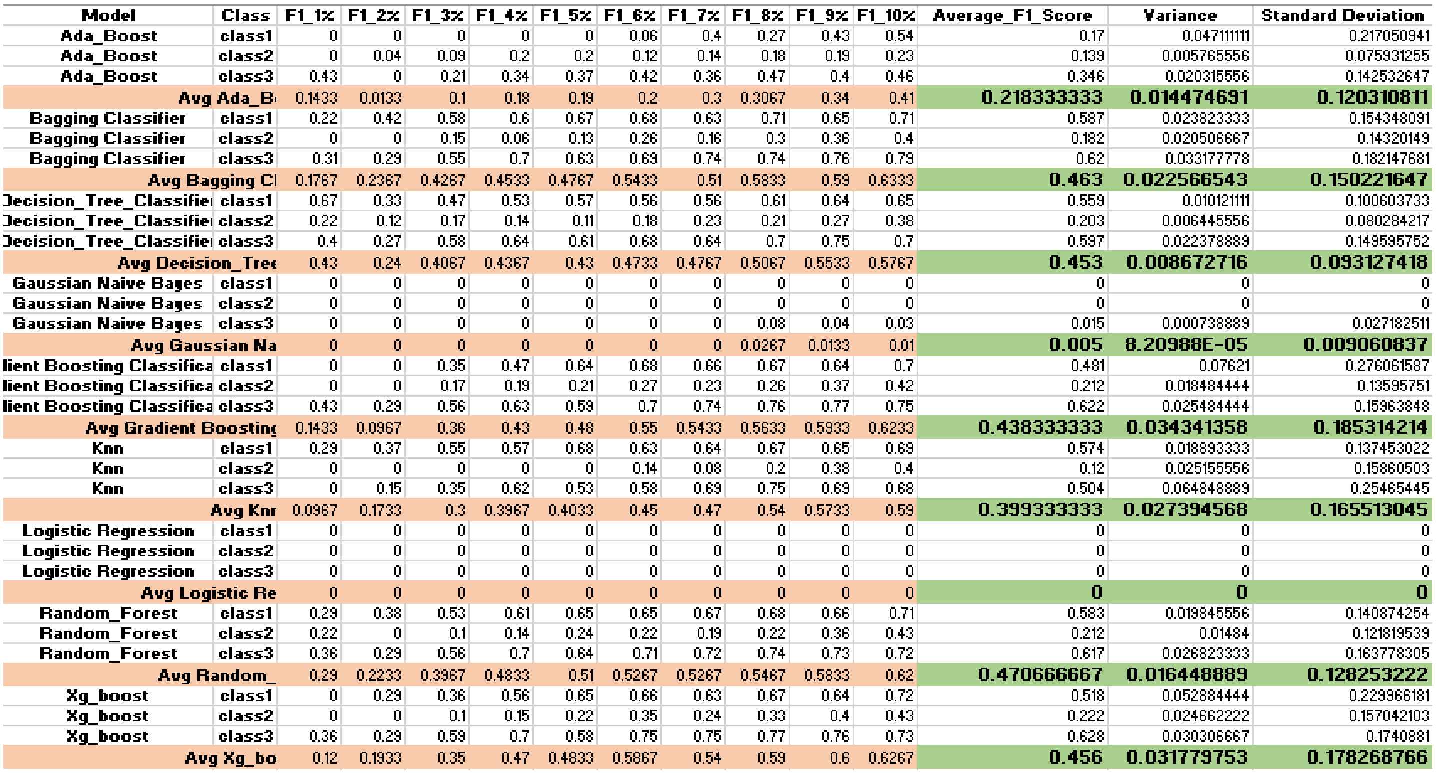
Model	Class	Re_1%	Re_2%	Re_3%	Re_4%	Re_5%	Re_6%	Re_7%	Re_8%	Re_9%	Re_10%	Average_Recall	Variance	Standard Deviation
Ada_Boost	class1	0	0	0	0	0	0.03	0.39	0.3	0.5	0.82	0.204	0.083248889	0.288528835
Ada_Boost	class2	0	0.09	0.18	0.39	0.48	0.31	0.24	0.3	0.31	0.23	0.253	0.019512222	0.139686156
Ada_Boost	class3	0.5	0	0.12	0.3	0.24	0.34	0.24	0.4	0.31	0.38	0.283	0.020534444	0.143298445
Avg Ada_B		0.1667	0.03	0.1	0.23	0.24	0.2267	0.29	0.3333	0.3733	0.4767	**0.246666667**	**0.017101235**	**0.130771689**
Bagging Classifier	class1	0.17	0.45	0.53	0.52	0.69	0.69	0.61	0.68	0.65	0.7	0.569	0.027143333	0.164752339
Bagging Classifier	class2	0	0	0.12	0.04	0.1	0.2	0.12	0.26	0.3	0.35	0.149	0.015387778	0.124047482
Bagging Classifier	class3	0.33	0.27	0.53	0.7	0.62	0.66	0.71	0.74	0.74	0.77	0.607	0.031156667	0.176512511
Avg Bagging Cl		0.1667	0.24	0.3933	0.42	0.47	0.5167	0.48	0.56	0.5633	0.6067	**0.441666667**	**0.02033642**	**0.142605819**
Decision_Tree_Classifie	class1	0.67	0.36	0.41	0.52	0.66	0.6	0.56	0.64	0.67	0.63	0.572	0.012195556	0.110433489
Decision_Tree_Classifie	class2	0.17	0.09	0.18	0.13	0.1	0.17	0.2	0.19	0.24	0.35	0.182	0.005573333	0.074654761
Decision_Tree_Classifie	class3	0.5	0.27	0.65	0.7	0.62	0.69	0.66	0.7	0.76	0.72	0.627	0.02069	0.143840189
Avg Decision_Tree		0.4467	0.24	0.4133	0.45	0.46	0.4867	0.4733	0.51	0.5567	0.5667	**0.460333333**	**0.008329506**	**0.091266128**
Gaussian Naive Bayes	class1	0	0	0	0	0	0	0	0	0	0	0	0	0
Gaussian Naive Bayes	class2	0	0	0	0	0	0	0	0	0	0	0	0	0
Gaussian Naive Bayes	class3	0	0	0	0	0	0	0	0.04	0.02	0.02	0.008	0.000195556	0.013984118
Avg Gaussian Na		0	0	0	0	0	0	0	0.0133	0.0067	0.0067	**0.002666667**	**2.17284E-05**	**0.004661373**
lient Boosting Classifica	class1	0	0	0.24	0.35	0.59	0.6	0.59	0.62	0.57	0.65	0.421	0.066187778	0.257269854
lient Boosting Classifica	class2	0	0	0.12	0.13	0.14	0.2	0.17	0.19	0.31	0.35	0.161	0.01281	0.113181271
lient Boosting Classifica	class3	0.5	0.27	0.59	0.65	0.55	0.69	0.76	0.79	0.76	0.73	0.629	0.025321111	0.159126086
Avg Gradient Boosting		0.1667	0.09	0.3167	0.3767	0.4267	0.4967	0.5067	0.5333	0.5467	0.5767	**0.403666667**	**0.027759136**	**0.166610731**
Knn	class1	0.17	0.27	0.47	0.52	0.69	0.63	0.61	0.62	0.63	0.67	0.528	0.031173333	0.176559716
Knn	class2	0	0	0	0	0	0.09	0.05	0.15	0.31	0.33	0.093	0.016845556	0.129790429
Knn	class3	0	0.09	0.24	0.52	0.45	0.49	0.66	0.7	0.63	0.6	0.438	0.060306667	0.245574157
Avg Knr		0.0567	0.12	0.2367	0.3467	0.38	0.4033	0.44	0.49	0.5233	0.5333	**0.353**	**0.027423333**	**0.165599919**
Logistic Regression	class1	0	0	0	0	0	0	0	0	0	0	0	0	0
Logistic Regression	class2	0	0	0	0	0	0	0	0	0	0	0	0	0
Logistic Regression	class3	0	0	0	0	0	0	0	0	0	0	0	0	0
Avg Logistic Re		0	0	0	0	0	0	0	0	0	0	**0**	**0**	**0**
Random_Forest	class1	0.17	0.36	0.47	0.48	0.69	0.66	0.66	0.66	0.65	0.72	0.552	0.03184	0.178437664
Random_Forest	class2	0.17	0	0.06	0.09	0.17	0.17	0.15	0.17	0.31	0.37	0.166	0.011915556	0.109158397
Random_Forest	class3	0.33	0.27	0.53	0.7	0.62	0.66	0.71	0.72	0.69	0.7	0.593	0.027201111	0.164927594
Avg Random_		0.2233	0.21	0.3533	0.4233	0.4933	0.4967	0.5067	0.5167	0.55	0.5967	**0.437**	**0.017840617**	**0.133568774**
Xg_boost	class1	0	0.18	0.24	0.43	0.59	0.6	0.61	0.64	0.61	0.72	0.462	0.057862222	0.240545676
Xg_boost	class2	0	0	0.06	0.09	0.14	0.23	0.17	0.23	0.31	0.35	0.158	0.014995556	0.122456341
Xg_boost	class3	0.33	0.27	0.59	0.7	0.55	0.74	0.73	0.77	0.74	0.7	0.612	0.031995556	0.178873015
Avg Xg_bo		0.11	0.15	0.2967	0.4067	0.4267	0.5233	0.5033	0.5467	0.5533	0.59	**0.410666667**	**0.029322963**	**0.17123949**



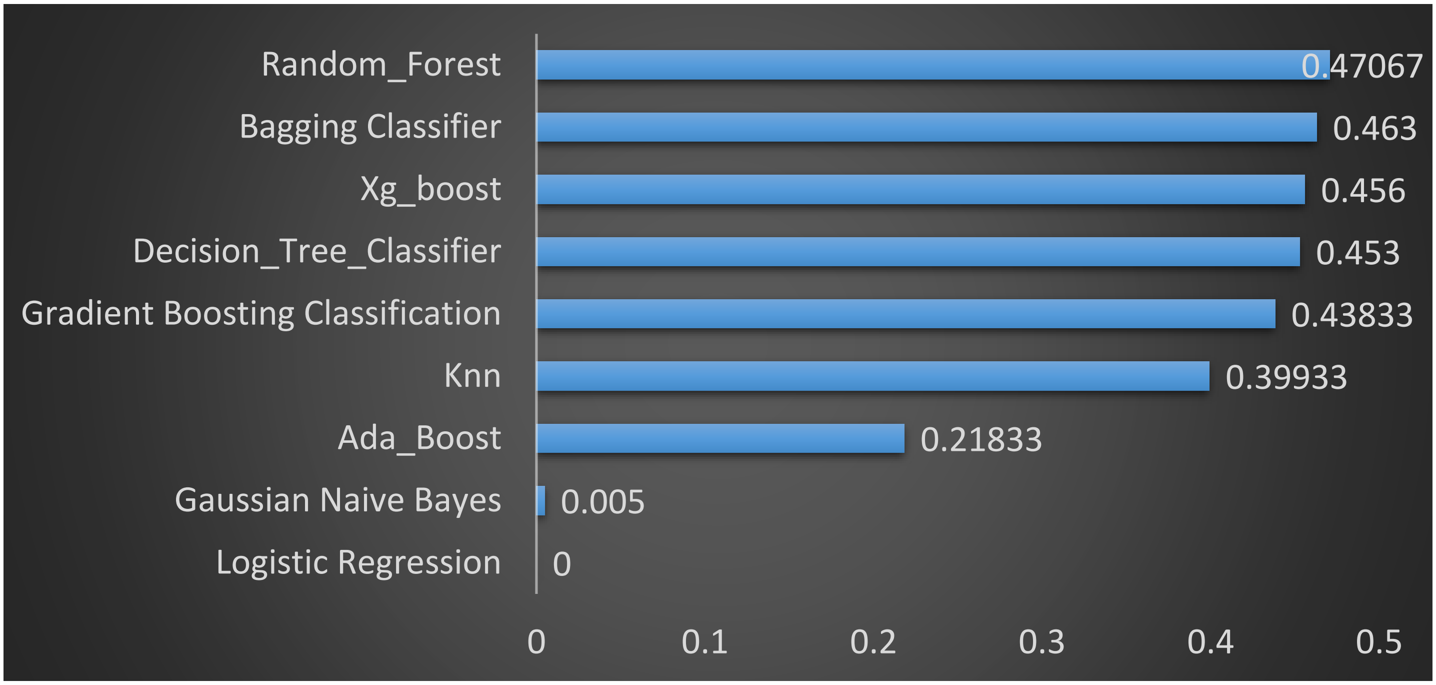
Overall when we considering Recall Score for Imbalance Data It is clearly Seen that from our analysis Decision Tree Algorithm is Performing Well Comparing Other Algorithms.



From the above it is seen that Naïve Bayes and Logistic Regression is not Performing well.



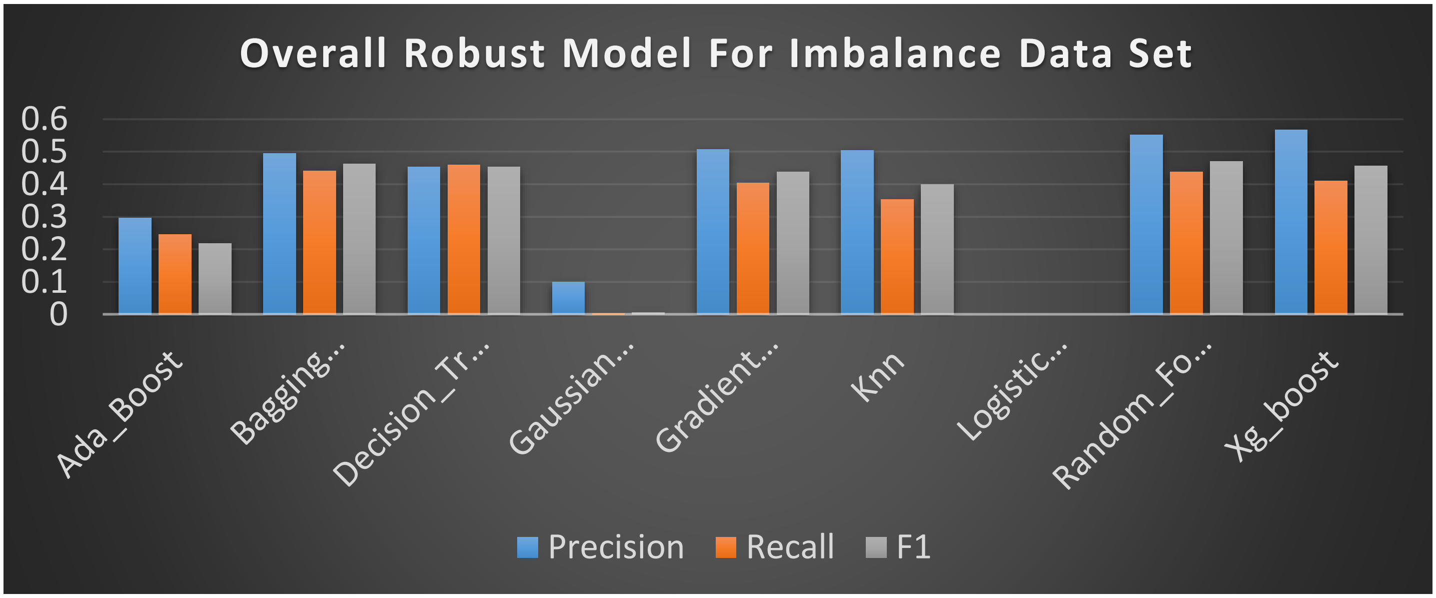
Model	Class	F1_1%	F1_2%	F1_3%	F1_4%	F1_5%	F1_6%	F1_7%	F1_8%	F1_9%	F1_10%	Average_F1_Score	Variance	Standard Deviation
Ada_Boost	class1	0	0	0	0	0	0.06	0.4	0.27	0.43	0.54	0.17	0.047111111	0.217050941
Ada_Boost	class2	0	0.04	0.09	0.2	0.2	0.12	0.14	0.18	0.19	0.23	0.139	0.005765556	0.075931255
Ada_Boost	class3	0.43	0	0.21	0.34	0.37	0.42	0.36	0.47	0.4	0.46	0.346	0.020315556	0.142532647
Avg Ada_B		0.1433	0.0133	0.1	0.18	0.19	0.2	0.3	0.3067	0.34	0.41	0.218333333	0.014474691	0.120310811
Bagging Classifier	class1	0.22	0.42	0.58	0.6	0.67	0.68	0.63	0.71	0.65	0.71	0.587	0.023823333	0.154348091
Bagging Classifier	class2	0	0	0.15	0.06	0.13	0.26	0.16	0.3	0.36	0.4	0.182	0.020506667	0.14320149
Bagging Classifier	class3	0.31	0.29	0.55	0.7	0.63	0.69	0.74	0.74	0.76	0.79	0.62	0.033177778	0.182147681
Avg Bagging Cl		0.1767	0.2367	0.4267	0.4533	0.4767	0.5433	0.51	0.5833	0.59	0.6333	0.463	0.022566543	0.150221647
Decision_Tree_Classifie	class1	0.67	0.33	0.47	0.53	0.57	0.56	0.56	0.61	0.64	0.65	0.559	0.010121111	0.100603733
Decision_Tree_Classifie	class2	0.22	0.12	0.17	0.14	0.11	0.18	0.23	0.21	0.27	0.38	0.203	0.006445556	0.080284217
Decision_Tree_Classifie	class3	0.4	0.27	0.58	0.64	0.61	0.68	0.64	0.7	0.75	0.7	0.597	0.022378889	0.149595752
Avg Decision_Tree		0.43	0.24	0.4067	0.4367	0.43	0.4733	0.4767	0.5067	0.5533	0.5767	0.453	0.008672716	0.093127418
Gaussian Naive Bayes	class1	0	0	0	0	0	0	0	0	0	0	0	0	0
Gaussian Naive Bayes	class2	0	0	0	0	0	0	0	0	0	0	0	0	0
Gaussian Naive Bayes	class3	0	0	0	0	0	0	0	0.08	0.04	0.03	0.015	0.000738889	0.027182511
Avg Gaussian Na		0	0	0	0	0	0	0	0.0267	0.0133	0.01	0.005	8.20988E-05	0.009060837
lient Boosting Classifica	class1	0	0	0.35	0.47	0.64	0.68	0.66	0.67	0.64	0.7	0.481	0.07621	0.276061587
lient Boosting Classifica	class2	0	0	0.17	0.19	0.21	0.27	0.23	0.26	0.37	0.42	0.212	0.018484444	0.13595751
lient Boosting Classifica	class3	0.43	0.29	0.56	0.63	0.59	0.7	0.74	0.76	0.77	0.75	0.622	0.025484444	0.15963848
Avg Gradient Boosting		0.1433	0.0967	0.36	0.43	0.48	0.55	0.5433	0.5633	0.5933	0.6233	0.438333333	0.034341358	0.185314214
Knn	class1	0.29	0.37	0.55	0.57	0.68	0.63	0.64	0.67	0.65	0.69	0.574	0.018893333	0.137453022
Knn	class2	0	0	0	0	0	0.14	0.08	0.2	0.38	0.4	0.12	0.025155556	0.15860503
Knn	class3	0	0.15	0.35	0.62	0.53	0.58	0.69	0.75	0.69	0.68	0.504	0.064848889	0.25465445
Avg Knn		0.0967	0.1733	0.3	0.3967	0.4033	0.45	0.47	0.54	0.5733	0.59	0.399333333	0.027394568	0.165513045
Logistic Regression	class1	0	0	0	0	0	0	0	0	0	0	0	0	0
Logistic Regression	class2	0	0	0	0	0	0	0	0	0	0	0	0	0
Logistic Regression	class3	0	0	0	0	0	0	0	0	0	0	0	0	0
Avg Logistic Re		0	0	0	0	0	0	0	0	0	0	0	0	0
Random_Forest	class1	0.29	0.38	0.53	0.61	0.65	0.65	0.67	0.68	0.66	0.71	0.583	0.019845556	0.140874254
Random_Forest	class2	0.22	0	0.1	0.14	0.24	0.22	0.19	0.22	0.36	0.43	0.212	0.01484	0.121819539
Random_Forest	class3	0.36	0.29	0.56	0.7	0.64	0.71	0.72	0.74	0.73	0.72	0.617	0.026823333	0.163778305
Avg Random_		0.29	0.2233	0.3967	0.4833	0.51	0.5267	0.5267	0.5467	0.5833	0.62	0.470666667	0.016448889	0.128253222
Xg_boost	class1	0	0.29	0.36	0.56	0.65	0.66	0.63	0.67	0.64	0.72	0.518	0.052884444	0.229966181
Xg_boost	class2	0	0	0.1	0.15	0.22	0.35	0.24	0.33	0.4	0.43	0.222	0.024662222	0.157042103
Xg_boost	class3	0.36	0.29	0.59	0.7	0.58	0.75	0.75	0.77	0.76	0.73	0.628	0.030306667	0.1740881
Avg Xg_bo		0.12	0.1933	0.35	0.47	0.4833	0.5867	0.54	0.59	0.6	0.6267	0.456	0.031779753	0.178268766



Overall when we considering F1- Score for Imbalance Data It is clearly Seen that from our analysis Random-Forest Algorithm is Performing Well Comparing Other Algorithms.

4.2 Robust Model for Imbalance Data

From the above reports now will see from Overall Reports Which model is performing well in either of the proportions.



Based on the results shown above when each class is proportioned the following is observed:

Gradient Boosting, Xg-Boost, Random Forest, Knn, Bagging Classifier and Decision Tree are performing better across the imbalanced classes.

Boosting Models are more consistent across the imbalanced classes.

Logistic Regression, NB are inconsistent with imbalanced data.

For high Precision and Recall, Random Forest and Boosting models are the best.

Conclusion

Whenever using a machine learning algorithm, evaluation metrics for the model have to be chosen cautiously: we must use the metrics that give us the best overview of how well our model is doing with regards to our goals.

When dealing with an imbalanced dataset, if classes are not well separable with the given variables and if our goal is to get the best possible accuracy, the best classifier can be a “naive” one that always answers the majority class.

Reworking the problem itself is often the best way to tackle an imbalanced classes problem: the classifier and the decision rule have to be set with respect to a well-chosen goal that can be, for example, minimizing a cost.

Reference

- https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/
- https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28
- https://sci2s.ugr.es/keel/pdf/algorithm/articulo/2011-IEEE%20TSMC%20partC-%20GalarFdezBarrenecheaBustinceHerrera.pdf
- https://towardsdatascience.com/xgboost-mathematics-explained-58262530904a
- https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf
- https://link.springer.com/article/10.1007/s10462-011-9272-4
- https://www.cs.cmu.edu/~kdeng/thesis/logistic.pdf
- https://www.geeksforgeeks.org/naive-bayes-classifiers/
- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.815&rep=rep1&type=pdf