ASSIGNMENT 2: Machine Learning Classification Techniques on Breast Cancer Dataset

Shivani Goyal, Student Number: R00183301

Abstract

In this project we are applying different machine learning algorithms on breast cancer dataset to classify the stage of cancer as malignant or benign depending on the features available in the dataset. Malignant stage is considered as harmful while benign means the earlier stage of cancer. Before building machine learning models on the dataset will do some preprocessing of data which includes dealing with missing values, detecting outliers by plotting boxplots, feature selection, scaling the data if required and handling imbalance. Then we will further proceed by finding the best three classification algorithms and tune their corresponding hyperparameters. Furthermore, for this project exploring different techniques to handle the imbalance of data as the research area.

1 Introduction

For this project have chosen the Breast Cancer Dataset which is easily available on Kaggle. This dataset is popular and many work has been already done on this before. This is a classification Dataset. The dataset is about the Breast cancer. Breast Cancer is the most common type of cancer in women. It is the second highest in terms of mortality rates. Diagnosis of breast cancer is performed when an abnormal lump is found (from self-examination or x-ray) or a tiny speck of calcium is seen (on an x-ray). After a suspicious lump is found, the doctor will conduct a diagnosis to determine whether it is cancerous and, if so, whether it has spread to other parts of the body [2].

This dataset consists 569 observations. The dataset consists 6 variables. All these variables are numerical. This dataset contains no missing data. Hence, it is a clean dataset.

Below are the details of 5 features and the target:

1. **mean_radius:** mean of distances from center to points on the perimeter
2. **mean_texture:** standard deviation of gray-scale values
3. **mean_perimeter:** mean size of the core tumor
4. **mean_area:** mean area of the core tumor
5. **mean_smoothness:** mean of local variation in radius lengths
6. **diagnosis:** The diagnosis of breast tissues (0 = malignant, 1 = benign)

Here, ‘diagnosis’ is the target variable.

2 Research

Imbalanced Data occurs when samples from one class or from multiple classes are over-represented in a dataset. As this is the case in our dataset. The proportion of ‘benign’(1) is comparatively higher than ‘malignant’(0) classifier of target variable which leads to the imbalance of data. This dataset is imbalance as the data count of diagnosis variable as ‘benign’(1) is 63% which is more comparing to diagnosis variable as ‘malignant’(0) is 37% . The breast cancer dataset is skewed.

The problem with imbalance data is that generally the machine learning algorithms works best for the balanced data. Balanced Data means when the number of samples in each class are equal. The reason behind this is because most of the algorithms are designed to maximize accuracy and reduced error. Problem with ‘Accuracy’ as performance metric is that it is not the best metric to be used
when analyzing imbalanced datasets, because it can be very misleading [3]. Below are the metrics which are considered better to evaluate the performance metrics:

1. Confusion Matrix: Table showing correct and incorrect types of prediction.
2. Precision: It is also called Positive Predictive Value. It is calculated by dividing the number of true positives to all the positive predictions. Low precision indicates a high number of false positives.
3. Recall: It is also called Sensitivity or the True Positive Rate. It is calculated by dividing the number of true positives by the number of positive values in the test data. Low recall indicates a high number of false negatives.
4. F1- Score: It is the weighted average of precision and recall.

There are a range of strategies that attempt to address the issue of imbalance. The most common techniques focus on sampling as a means of addressing the disparity. However, there are a wide range of approaches to dealing with this issue the most common is applying common techniques of sampling that attempts to equal the distribution.

Some of the common approaches are :

1. Random Under Sampling: Removes datapoints randomly from the majority class to balance the both classes. Under Sampling can be a good choice when you have a ton of data -think millions of rows. However, this could lead to loss of the important data, underfitting and poor generalization of the test data. This could be done by using resampling module from Scikit-Learn (imblearn.over_sampling.RandomOverSampler).

2. Random Over Sampling: It gradually increases the size of minority class. The data points from the minority class is randomly duplicated to balance with majority class. It could be defined as adding more copies of the minority class. Oversampling can be a good choice when you don't have a ton of data to work with. While using this technique always be careful to split the dataset into test and train before applying this technique. Oversampling before splitting the data can allow the exact same observations to be present in both the test and train sets. This can allow our model to simply memorize specific data points.
So, this may lead to overfitting and poor generalization of the test data. This could be done by using using resampling module from Scikit-Learn (imblearn.under_sampling.RandomUnderSampler).

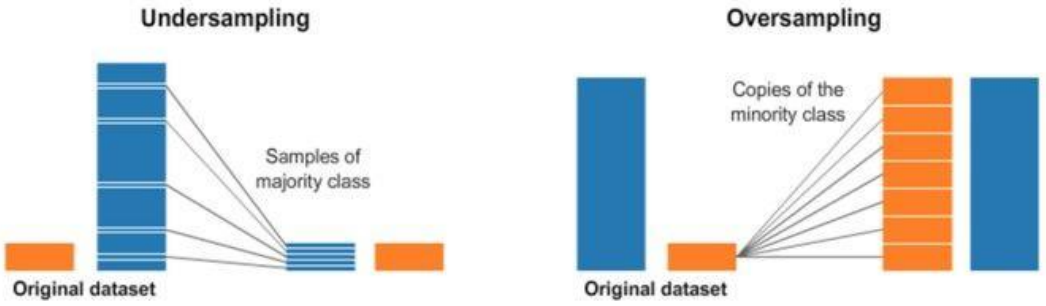

Fig.1

As we have found that our dataset is very limited and low, it is therefore not possible to do under sampling. So, here we are exploring different techniques and ways to oversample the data as the research area. One of the most popular and frequently used technique of over sampling SMOTE (Synthetic Minority Oversampling Technique).

SMOTE: It is an oversampling technique that allows to create new artificial data points in the dataset. This could be done by using resampling module from Sciit-Learn(imblearn.over_sampling.SMOTE).

SMOTE technique can be interpreted as the iterative execution of the following steps:

1. Select a data point from the minority class and identify it's k nearest neighbours in feature space.
2. Randomly select a neighbor
3. Get the difference between the original data point and the neighbour and multiply it by a random number between 0 and 1.
4. Next a new point is created in feature space between the original data point and it's neighbour by adding the random number to the original data.[Lecture Notes]

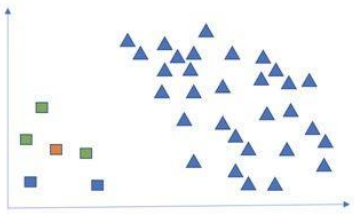

Fig.2.1

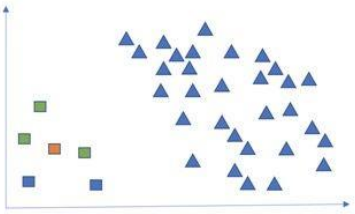

Fig. 2.2

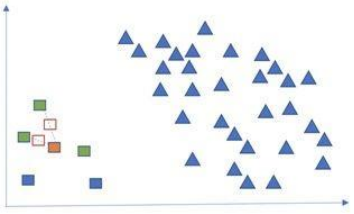

Fig.2.3

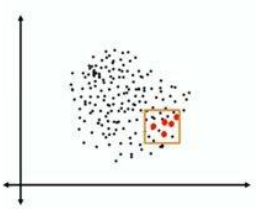

Fig.2.4

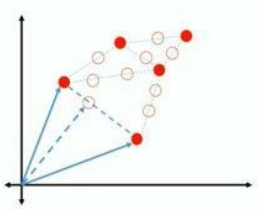

Fig.2.5

2.1 Variants Of SMOTE

SMOTE might connect inliers and outliers while ADASYN (Adaptive Synthetic Sampling) might focus solely on outliers which, in both cases, might lead to a sub-optimal decision function. In this regard, SMOTE offers three additional options to generate samples. As we are using SMOTE for over sampling in preprocessing steps, it is a good idea to use variants of SMOTE to research their effect and analyse them. These methods focus on samples near of the border of the optimal decision function and will generate samples in the opposite direction of the nearest neighbors class. Following are the three variants using for research: Borderline Smote, K Means Smote and SVM Smote [4].

2.2 Borderline SMOTE

This algorithm works by generating new synthetic data along the line between the minority borderline examples and their nearest neighbors of the same class. Thus strengthens the borderline examples. This procedure involves in finding the nearest neighbours using KNN but the only difference is that, it focuses on borderline data and over sample them. It is such, because borderline data points are more apt to be misclassified than the ones far from the borderline.
The borderline SMOTE have 2 parameters kind = 'borderline-1' and kind = 'borderline-2'. They classify each sample xi to be (i) noise which means all nearest-neighbors are from a different class than the one of xi, (ii) danger which means at least half of the nearest neighbors are from the same class than xi and (iii) safe which means all nearest are from the same class than xi. **Borderline-1** and **Borderline-2** SMOTE will use the samples *in danger* to generate new samples. In **Borderline-1** SMOTE, x_{zi} will belong to the same class than the one of the sample xi. While for **Borderline-2** SMOTE will consider x_{zi} which can be from any class.

2.2.1 SVM SMOTE

This algorithm uses an SVM classifier to find support vectors and generate samples considering them. The parameter of the SVM classifier allows to select more or less support vectors. The parameter 'm_neighbours' is used to define the neighborhood i.e. to decide if a sample is in danger, safe , or noise. This parameter is used by both borderline and SVM SMOTE techniques. It focuses on generating new minority class instances near borderlines with SVM so as to help establish boundary between classes.

2.2.2 K Means SMOTE

This algorithm uses a KMeans clustering method before to apply SMOTE. The clustering will group samples together and generate new samples depending of the cluster density. It aids classification by generating minority class samples in safe and crucial areas of the input space. The method avoids the generation of noise and effectively overcomes imbalances between and within classes.

K-means SMOTE works in three steps:

1. Cluster the entire input space using k-means
2. Distribute the number of samples to generate across clusters:
   i. Filter out clusters which have a high number of majority class samples.
   ii. Assign more synthetic samples to clusters where minority class samples are sparsely distributed.
3. Oversample each filtered cluster using SMOTE .

SMOTE need information regarding the neighborhood of each sample used for sample generation. They are using a one-vs-rest approach by selecting each targeted class and computing the necessary statistics against the rest of the data set which are grouped in a single class.

3 Methodology

Machine Learning is the science of getting computers to learn and act like humans do, and improve their learning over time in autonomous fashion, by feeding them data and information in the form of observations and real-world interactions. There are many algorithm for getting machines to learn, from using basic decision trees to clustering to layers of artificial neural networks depending on what task you're trying to accomplish and the type and amount of data that you have available.
Only implementing machine learning algorithm doesn't solves the problem. This includes many steps like pre-processing of data, finding the best algorithm depending on the quality and size of the dataset and further tuning the hyperparameters of the model. The first and foremost important step is to read and understand your dataset. Then proceed by pre-processing of data. Once, the data is cleaned then we could either split the dataset into train -test set or alternatively could use cross validation techniques. Post this, we could fit the range of models to the training data set by using pre-defined models from sklearn package. By using evaluation metrics parameters like accuracy, confusion matrix, precision, recall or f1 score could evaluate the models builded. After finding the best three models need to tune the hyperparameters.
Breast Cancer Dataset is skewed so for this we will be using f1-score parameter of metric evaluation.

3.1 Data Preprocessing

3.1.1 Dealing with Missing Values

The dataset is a cleaned Data. Means no missing values are present in our dataset. Hence, we simply checked the existence of null values in each column to verify the authenticity of the clean dataset and it turns out to be true.

3.1.2 Dealing with Outliers

An outlier is a point that falls far from the datapoint. It's a point that is extreme in some ways. If the parameter estimates change a great deal when a point is removed from the calculations, then the point is said to be influential.
Outlier could be detected by visualizing them. Outliers could be visualized by plotting the boxplot. The datapoints which lies outside the 1.5 IQR are considered to be outliers. But before dealing with these ouliers which includes removal of such datapoints from the dataset it is necessary to check weather they are potential outliers or not. Sometimes these datapoints are not sparse rather it's clustered together it becomes a really important data and can impart crucial properties too.
For our dataset, there are many datapoints lying outside the 1.5 IQR but these all points are clustered together. Hence, we didn't removed any datapoints from the dataset.

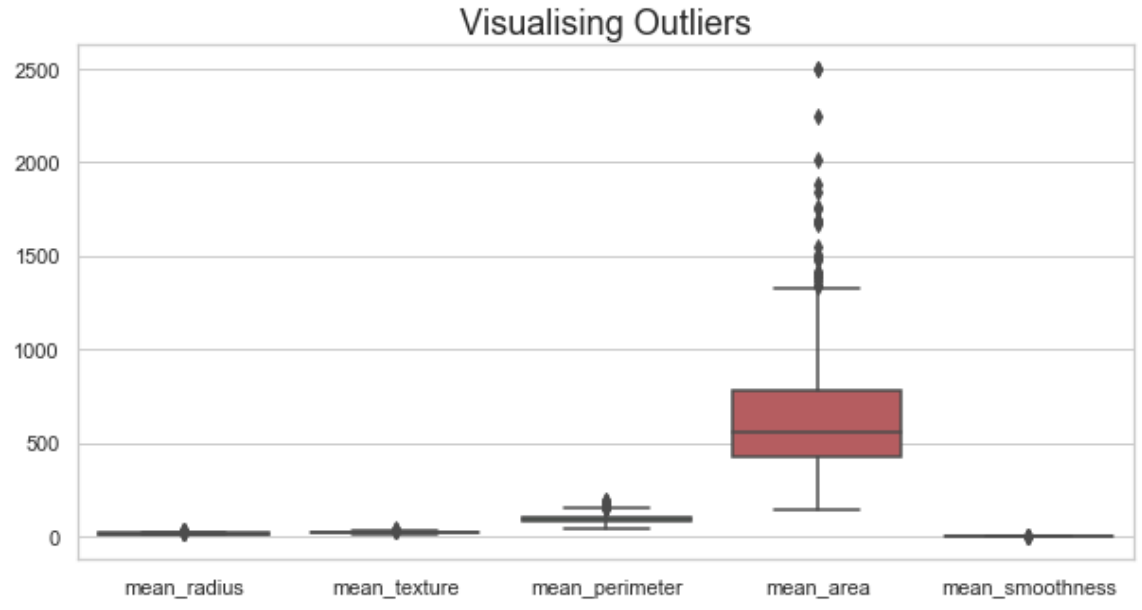

Fig.3

Here, we observe that outliers are present but doesn't seems to be potential outliers as these are clustered. Removing these outliers would lead to the loss of important data. So, here not imparting these outliers from the data.

3.1.3 Handling Categorical Data

As the dataset have only numeric values, don't have to handle categorical data.
In any case, to deal with categorical data we can convert it into numeric columns by using one-hot encoding or label encoder.

3.1.4 Scaling Data

As it's visible from box-plot above, values of feature like mean_area is at very different scale then values of other features like mean_smoothness. For machine learning algorithms to work better & efficiently we have to bring all the numeric values on the same scale and this is known as scaling. The majority of and optimization algorithms behave better if features are on the same scale [Lecture Slides].

There are 2 techniques:

1. Normalization: Data is rescaled in the range of 0 to 1. This can improve the performance for algorithms that assign a weight to features such as linear regression and in particular for algorithms that utilize geometric distance such as KNNs.

2. Standardization: It facilitates the transformation of features to a standard Gaussian(normal) distribution with a mean of 0 and a standard deviation of 1. The scaling happens independently on each individual feature by computing the relevant statistics on the samples in the training set.

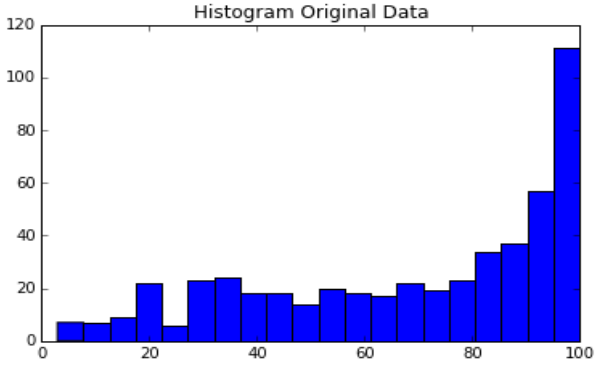

Fig.4.1

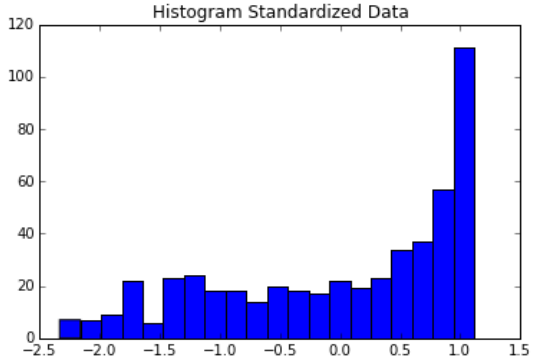

Fig.4.2

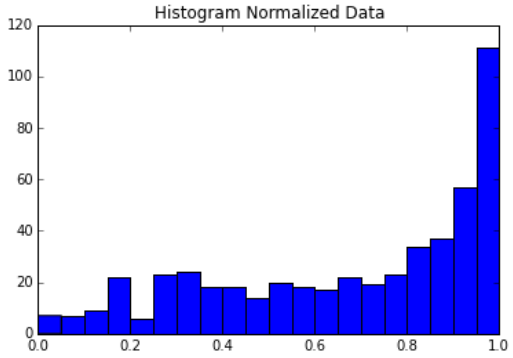

Fig.4.3

NOTE: Standardization is less sensitive to outlier than normalization.
Hence, we are using standardization technique for scaling our data. The reason being that our dataset consists outliers and as we didn't remove any outlier. Therefore, it is good to use this technique to avoid unforeseen effect due to non-removal of outliers.

3.1.5 Handling Data Imbalance

As mentioned, the Breast Cancer dataset is very small dataset and skewed towards the classifying the datapoint as benign (1), and it's about 63% of the data. While 37% of data points are classified as malignant (0). Along with this issue, it is also observed that the data is very small, i.e. just 569 rows. So rather than using random under sampling technique we are using an over sampling technique which is SMOTE(Synthetic Minority Oversampling Technique).
It is recursive flow of below mentioned steps :

1. Selection of datapoint from minority class
2. Finding nearest neighbours using knn
3. Picking up one of the neighbours at random
4. Take difference of the selected data points and multiply with random number between 0 & 1.
5. The data created is our new sample of minority class.
6. Continue until balance achieved.
7. Take difference of the selected data points and multiply with random number between 0 & 1.
8. The data created is our new sample of minority class.
9. Continue until balance achieved.

3.1.6 Feature Selection

Feature selection methods is used to identify and remove unneeded, irrelevant and redundant attributes from data that do not contribute to the accuracy of a predictive model or may in fact decrease the accuracy of the model.
Feature selection is different from dimensionality reduction. Both methods seek to reduce the number of attributes in the dataset, but a dimensionality reduction method do so by creating new combinations of attributes, where as feature selection methods include and exclude attributes present in the data without changing them[5].
We could filter out those features from the dataset which doesn't contributes much meaning in our dataset. We could find and depict the strength or relationship or could say, correlation between different features of the dataset by plotting the cor plot.

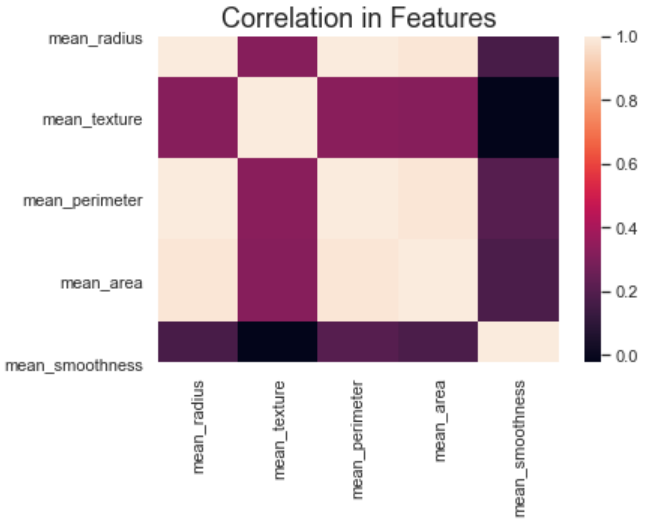

Fig.5

From the correlation heatmap visualization [Fig.5] we observes that mean_radius, mean_perimeter and mean_area are highly correlated. Yet we can't reduce our feature space to just three features. Hence, we are not extracting these features for classification.

3.1.7 Dimensionality Reduction

Dimensionality reduction is the process of converting a dataset that typically has a large number of features into a dataset with a reduced number of features [Lecture Notes]. As mentioned earlier, that our dataset is small and has less number of features. So, for our dataset dimensionality reduction is not required.

3.2 Model Selection

We are using a variety of classification algorithms to find the best model that provides the highest score for the dataset. We have applied simple machine learning algorithms to the ensembles in order to find the best model for the dataset. For all the algorithms considered we are running them on default parameters.

1. KNN
2. Logistic Regression
3. Gaussian Naïve Bayes
4. SVM
5. Ada Boosting
6. Gradient Boosting

For the evaluation of these models we have taken a common evaluation metric: F1- score. The models generated are the baseline models. We will select the top 3 best models based on scores and tune them on their hyper parameters.

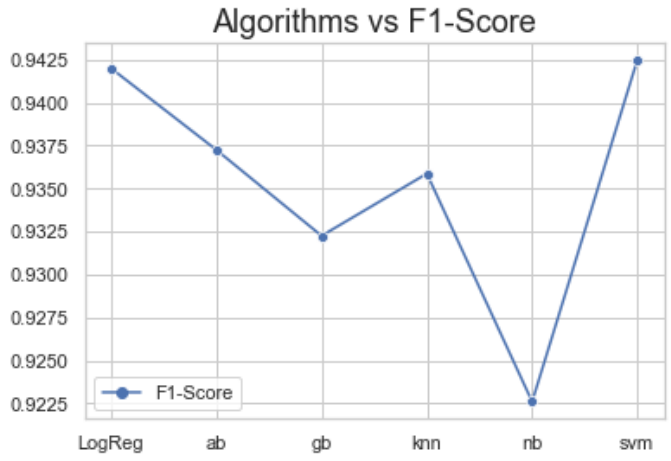

Fig. 6

In our range of models, we detected the best performing models with highest f1-scores are:

1. Logistics Regression: 0.9419975341427163
2. Ada Boosting: 0.9372528835862466
3. Gradient Boosting: 0.932252989407918

3.3 Hyper-Parameter Optimization

These are the parameters which are not learned or automatically adjusted by algorithms and needs to be experimented with a variety of values to get the best possible result. These parameters are tuned manually over range of values and then compared to get the best set of parameters having highest performance score. A model hyperparameter is a configuration that is external to the model and whose value cannot be estimated from data. When a machine learning algorithm is tuned for a specific problem then essentially you are tuning the hyperparameters of the model to discover the parameters of the model that result in the most skillful predictions[6].

The two simple strategies to optimize or tune the hyperparameters are: (i) Grid Search and (ii) Random Search

We are using Grid Search approach to tune the hyperparameter that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid. Here we pass param grid dictionary, which have parameters as keys and range of values for parameters as values.
We have taken 'no_of_splits' as 10 because the dataset is small and we need more data for training, so, with the increase in number of splits the training data increases. The another way is to use leave-one-out CV but it is computationally expensive. Hence, we are using this.
As our data is imbalance we have to use SMOTE on training data along with the top algorithms and make a pipeline which in turn is passed to the GridSearchCV. Again, for evaluation we are using F1-Score.

3.4 Parameter Tuning & Best Performing Model

3.4.1 Logistic Regression

1.C: Inverse of regularization strength; must be a positive float. Smaller values specify stronger regularization.[3].
Range: np.logspace(0, 4, 10)
2.penalty: Used to specify the norm used in the penalization. **Range:** ['l2']

3.4.2 Ada Boosting

1. n_estimators: The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early. **Range**: list(range(50, 101))
2. learning_rate: Learning rate shrinks the contribution of each classifier by learning_rate.
Range: [0.1, 0.3, 0.5, 0.7, 0.9, 1]

3.4.3 Gradient Boosting

1. n_estimators: The number of boosting stages to perform. Gradient boosting is fairly robust to overfitting so a large number usually results in better performance. **Range**: list(range(100, 200))
2. learning_rate: Learning rate shrinks the contribution of each tree by learning_rate.
Range: [0.01, 0.03, 0.05, 0.07, 0.09, .1]
3. max_depth: The maximum depth limits the number of nodes in the tree. Tune this parameter for best performance; the best value depends on the interaction of the input variables. **Range**: [1, 3, 5, 7, 9]

4 Evaluation

Machine learning algorithms are incomplete until and unless they are properly evaluated. The most crucial part of preparing end-to-end ML model is deciding the right evaluation metric.

There are a number of metrics available to judge the algorithm. The selection of evaluation metric depends on the data and how we are processing it. Some of the frequently used evaluation metrics are:

1. Accuracy: We compare the predicted target values with actual labels and find out what percentage of values are correctly predicted.

2. Confusion Matrix: It gives us deeper insight of the predicted & actual classes. It tells us about how many True Positives, True Negative, False Positives & False Negatives are found in the prediction.

3. Recall: It shows us how confident we are that all of the instances that belong to a specific class have been classified correctly using our model.

4. Precision: It shows us how confident we are that all any instance predicted as belonging to a certain class actually belongs to that class.

5. F1-Score: Rather than having Recall & Precision as 2 different metrics, they can be combined to form the F1-Score. It is harmonic mean of precision & recall. For F1 to be high both precision & recall has to be high. F1 doesn't get skewed on just a single value.

4.1 Selecting Evaluation Metric

4.1.1 Why not to use Accuracy?

Accuracy hides lot of details for basic classification. For example, an ML algorithm might be doing well are predicting one class but may be poor are predicting another class. This type of behaviour can often be masked when simply looking at classification accuracy.

Accuracy Paradox: When dealing with highly imbalanced datasets. The accuracy of the model will appear high but the model is just predicting the majority class. The situation is often referred to as the accuracy paradox. It occurs when your algorithm reports a very high level of accuracy (such as 95%), but the accuracy is only reflecting the class distribution within the dataset. [Lecture Slides]

4.1.2 What to use, Precision, Recall, F1-Score?

Precision: It will give us very low value when algorithm is not correctly predicting the values to be of actual class.

Recall: It will highlight if we do very poorly on predicting any specific class. In the imbalanced dataset it would clearly show that if our algorithms perform very poorly on minority class.

F1-Score: When dealing with imbalanced data we actually need both precision & recall for evaluation but rather than using 2 different metrics, we considered using F1-Score because we need a model having high precision & recall and F1 score works on similar lines, i.e F1 is high only when both are high.

Alternatively, we can also use classification report where we get all three i.e. precision, recall, & F1-Score.

We chose F1-Score only for ease of working through.

4.2 Baseline Models

We are evaluating our models on F1-Score over 10 iterations of Stratified K Fold Cross Validation and then we are averaging the F1 Score over no. of iterations.

Observation:
We observes that Logistic Regression, Gradient Boosting and Ada Boosting are the top performing models for breast cancer dataset. While Decision is the least performing algorithm.

4.3 Hyper Parameter Optimization on Top Models

1. Logistic Regression
Best Parameters: {'model_C': 166.81005372000593, 'model_penalty': 'l2'} Best Score: 0.936814296031363

2. Ada Boosting
Best Parameters: {'model_learning_rate': 1, 'model_n_estimators': 100}
Best Score: 0.9434503977016261

3. Gradient Boosting
Best Parameters: {'model_learning_rate': 0.7, 'model_n_estimators': 64}
Best Score: 0.9354849798923339

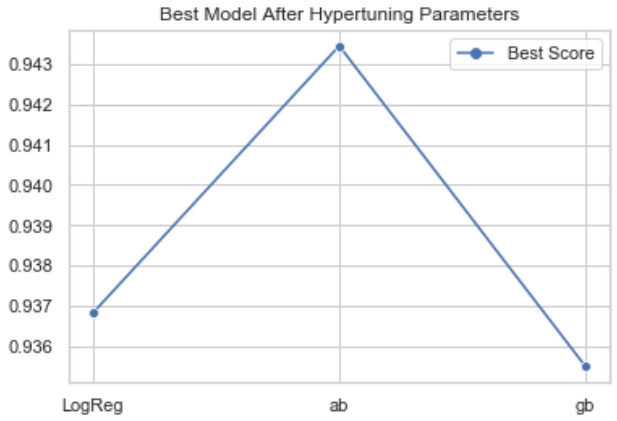

Fig. 7

Observation:
It is very evidently visible [Fig 7] that there is a great increase in F1-Score of Adaboost algorithm after hypertuning parameter.

4.4 Research Evaluation

As mentioned, we are researching various techniques of over sampling and decided on using three variants on SMOTE. The three chosen variants of smote are Borderline Smote, K-Means Smote and SVM Smote.

	Borderline Smote	K-Means Smote	SVM Smote
Logistic Regression	0.920	0.945	0.926
Ada Boosting	0.925	0.936	0.927
Gradient Boosting	0.919	0.935	0.925

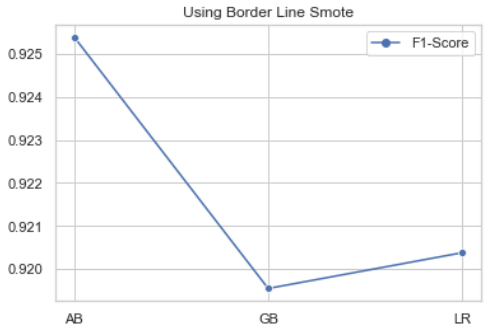

Fig.8

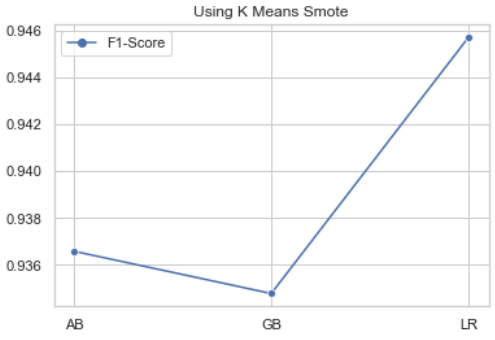

Fig. 9

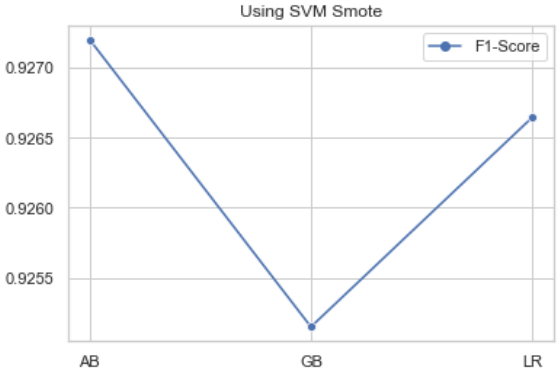

Fig. 10

Observation:
We observes from figure 8,9 and 10 that K-Means Smote is performing the best of all with highest F1-Score as 0.945. While the least performing model is Gradient Boosting with least F1-Score as 0.91.

4.5 Hyper Parameter Optimization on Research area

We tried tuning the top 3 algorithm using variants of smote to remove imbalance in the data and we are also tuning the hyper parameters in the similar way as done earlier.

	Borderline Smote	K-Means Smote	SVM Smote
Logistic Regression	0.930	0.942	0.930
Ada Boosting	0.926	0.952	0.936
Gradient Boosting	0.933	0.945	0.932

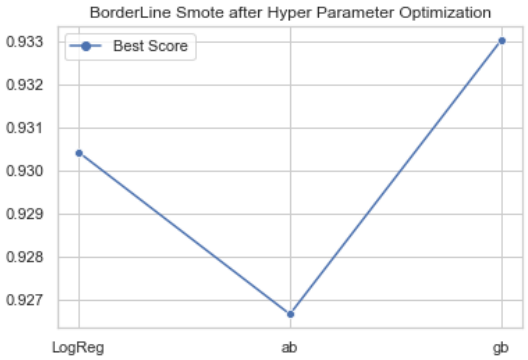

Fig.11

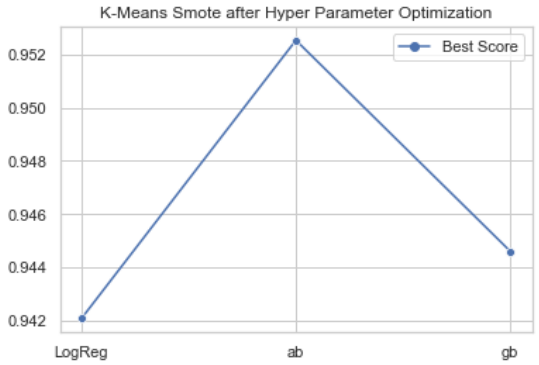

Fig. 12

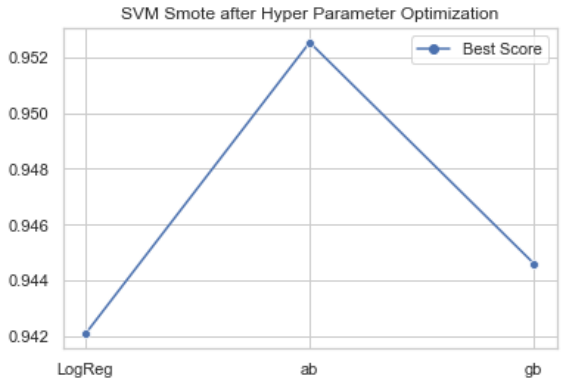

Fig. 13

Observation:
As depicted in the plots Fig 11,12 and 13, it is observed that K-Means Smote performs the best with a score of around 0.95 for all the top 3 algorithms. While Borderline Smote & SVM Smote are good but not at par with K-Means Smote.

5 Conclusion

We have successfully implemented all steps from data pre-processing till research. During the process we have come across various difficulties and resolved them. Our major observation from the evaluation are that Logistic Regression, Ada Boosting & Gradient Boosting are the best algorithms for breast cancer dataset. Only Logistic Regression has major growth on hyper-parameterization. Borderline Smote is the best performing imbalance handling technique. Also, On hyperparameter optimization, K-Means smote works the best.

References

1. https://www.datacamp.com/community/tutorials/parameter-optimization-machine-learning-models
2. https://www.slideshare.net/hecfran/borderline-smote
3. https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html
4. https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18
5. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
6. https://medium.com/vclab/tackling-class-imbalance-with-svm-smote-efa41ec3de5f
7. https://sci2s.ugr.es/keel/keel-dataset/pdfs/2005-Han-LNCS.pdf
8. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.ht
9. https://pypi.org/project/kmeans-smote/#:~:targetText=K%2DMeans%20SMOTE%20is%20an,imbalances%
10. https://towardsdatascience.com/hyperparameters-optimization-526348bb8e2d
11. https://breast-cancer.ca/diag-chnces/
12. https://www.kaggle.com/merishnasuwal/breast-cancer-prediction-dataset