Introduction

There is a significant interest in studying the classification localization sites of protein into known classes from their amino acid. An expert system developed by Nakai and Kanehisa was the first study for predicting the localization sites of protein. Later, there was number of investigations to improve the prediction accuracy using different classifier systems.[9]

In this report we implemented random forest classifier to investigate the performance of predict the Localization site of protein. Random forest is an ensemble-based schemes which make it suitable for learning from imbalanced data.

We used three approaches to deal with imbalanced data. The first one, implement the random forest algorithm and predict the 30% of dataset. The second trial was using class weights or cost sensitive learning. That what make random forest more suitable for this problem. [666].

The last experiment was resampling the training dataset with oversampling algorithm SMOTE and then training the random forest algorithm from balanced data. Class weighted random forest has shown an improvement better than the oversampling algorithm.

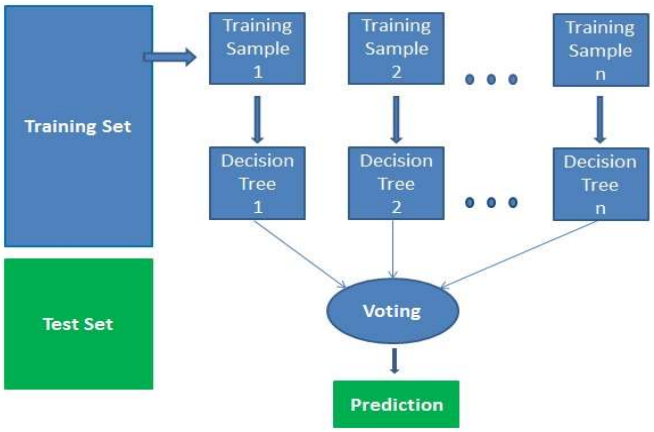
1- Methodology

Most classification problems in real world would be imbalanced dataset. There are two approaches to deal with these cases. First method, weighting the patterns and the other one is a sampling either undersampling the majority or oversampling the minority.

In our research we used the weighted approach and SMOTE oversampling to balance the data and avoid using under sampling way because it discards the data from the majority and that can lead to lose information from training instances.[666]

1.1 Random Forest Algorithm (RF)

It is one kind of ensemble techniques that based on generate multiple decision trees from random subsets of dataset and collect the prediction from these trees and select the best result based on majority voting.[1]



This graph is taken from

The random forest algorithm has two parts [4]:

- Generate number of decision tress
    1- In each decision tree model, select randomly “k” of features from the total “m” (k << m)
    2- With the k features, use the best split to split the node and generate children nodes.
    3- Iterate the steps 1 to 3 until reach the required number of nodes.

    These steps done for “n” times to create “n” number of decision trees.

- Random forest prediction

1- Using test features and the rules of each tree to predict the targets for test set
2- Select the best predict from the tree’s predictions based on the majority voting.

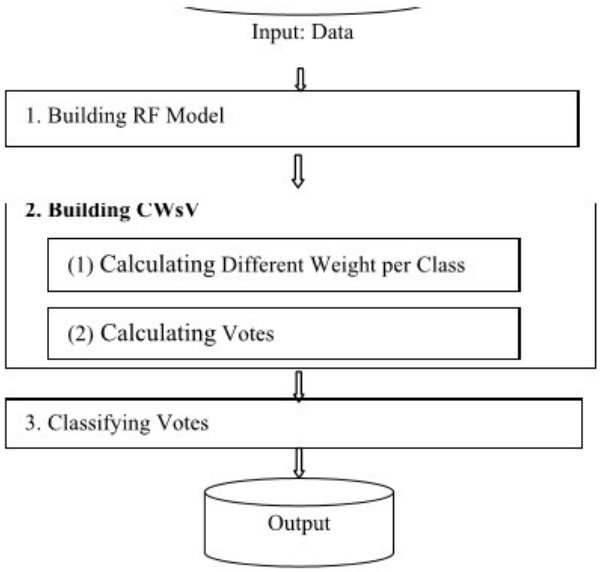
The majority voting is simply when we have three classifiers results prediction

classifier 1 -> class 0 classifier 2 -> class 0 classifier 3 -> class 1

with majority vote, we would classify the sample as "class 0."

1.2 Class weights random forest (CWsRF)

In random forest different classes have the same weight and tends to be biased to the majority class that lead to reduce the performance to predict the minority classes. [5] Therefore, we used class weights method which assign different weights to the patterns with high cost for the minority classes and low cost for majority classes. [666]



The graph shows the steps of building CWsRF[5]

The algorithm of this method has three main procedures. [5]

1- Build the RF model to get the vote of classifiers.

2- Build the class weights vote (CWsV) which is done in two steps: -

The first step is calculating different weights to each class by find out the scores for each classifier for each sample. The second step is to find the accuracy for each class per a classifier. Calculating the weight for a class in a classifier which is equal to the accuracy for the class in that classifier. obtain the voting based on the weights.

3- compute a weighted majority vote by associating a weight with classifier of unique class.
For the example assign the weights {0.2, 0.2, 0.6} of the classes from three classifier with the voting as below [6]:

classifier 1 -> class 0 classifier 2 -> class 0 classifier 3 -> class 1

this would vote to class 1 has high weight so that

$$\arg\max_i [0.2 \times i_0 + 0.2 \times i_0 + 0.6 \times i_1] = 1$$

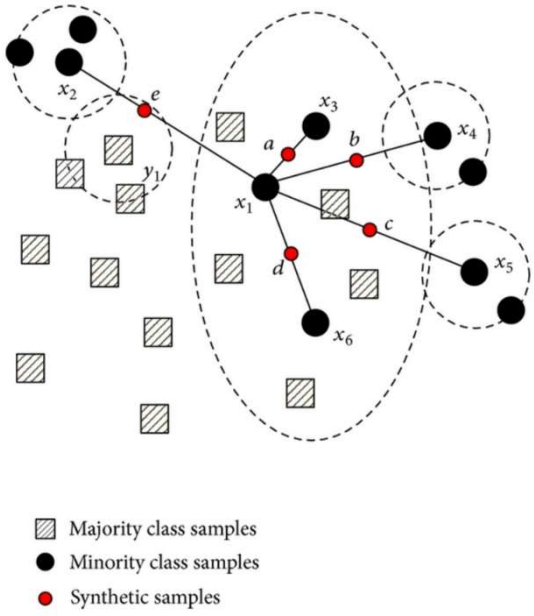
1.3 Oversampling random forest

Another way to tackle with imbalanced data for classification is resampling the dataset approach. We focused on SMOTE (Synthetic Majority Oversampling Technique) as one of techniques used to generate synthetic examples of existing minority points that sit along the line segments of the different classes.
This is to improve performance as having these additional points helps in creating large and less specific decision boundaries between the classes which helps the classifiers in improving their generalisations ability. [santos]

the algorithm of smote is basically one of the minority data points is chosen randomly for example x1 as show in the figure and based on k of nearest neighbours which specified by the user in the figure four points x2, x3, x4 and x5 are taken and new four points are generated (a, b,c,d) and interpolating every two samples on the line between them.[8]

For instance, the new sample a is created according to the formula $a = x1 + \omega (x1 - x3)$

where ω is a random weight between 0 and 1.



The figure is taken from
https://www.researchgate.net/publication/287601878/figure/fig1/AS:316826589384744@1452548753581/The-schematic-of-NRSBoundary-SMOTE-algorithm.png

2. Tuning the classifier's parameters

In our experiments we used grid search technique to hyperparameter tuning which build and fit and evaluate the model with all possible combination of the parameter's values specified in the grid function.

We used four main parameters with range of values that could tune to improve the performance of the model recommended [7]

n_estimators: [100,300,500,700,900]

max_features: ["sqrt", "log2", None]

max_depth': [10, 20, None]

min_samples_split: [2,5,10]

the method takes long computation time as it builds the model many times. In our experiments it generates 135 possible combinations with 5-fold validation.

3. Evaluation methods

To get better understanding for classification errors, we considered confusion matrix and other evaluation the performance of classification model for instance,

Precision represents a measure the exactness of a classifier. The large number of false positive the low precision.[12]

Precision = TP / TP+FP

Recall indicates as a measure of a classifier's completeness. The large number of false negative the low recall.[12]

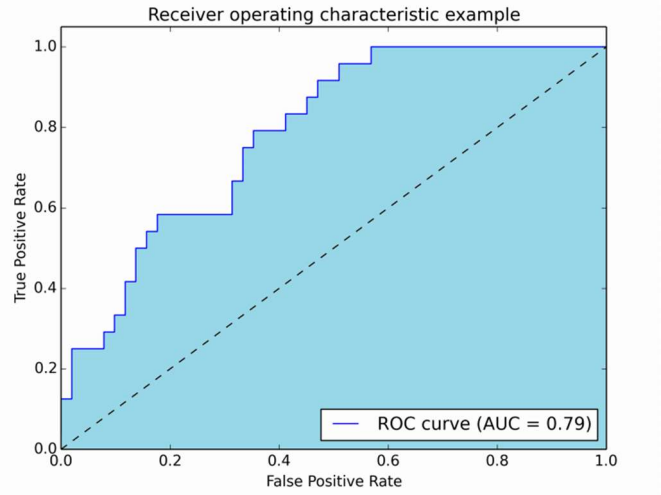
Recall = TP / TP+FN

AUC_ROC

AUC stands for area under curve. The curve is the receiver operating characteristic curve (ROC). The curve is plotting the true positive rate = TP / TP+FN with the false positive rate =FP / FP+TN at different probability between [0 -1]. [black b]

The area under curve represents the degree of separability and measure the ability of the model to distinguish between the classes.

This figure is taken from



https://stats.stackexchange.com/questions/132777/what-does-auc-stand-for-and-what-is-it

When the AUC near to 0 that indicate the worse measure of separability for the model. Whereas, the degree of separability is good when the AUC near to 1.[11]

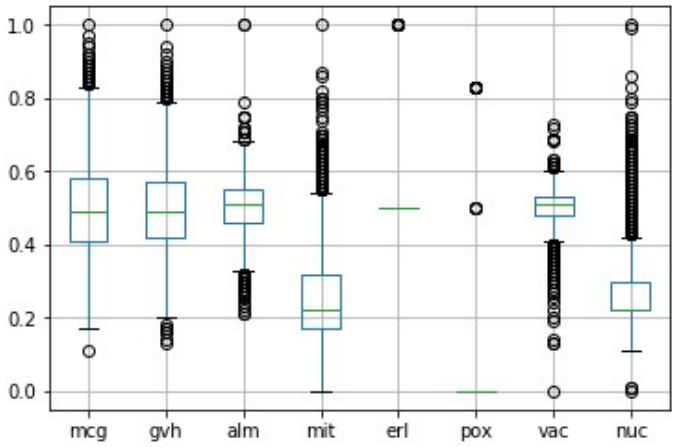
The dash line in the figure represent random guessing and the AUC =0.5 [black].

For multi classes, there are AUC measure for each class each time take one class with all other classes to find the AUC and take the average to measure the separability of the model.[11]

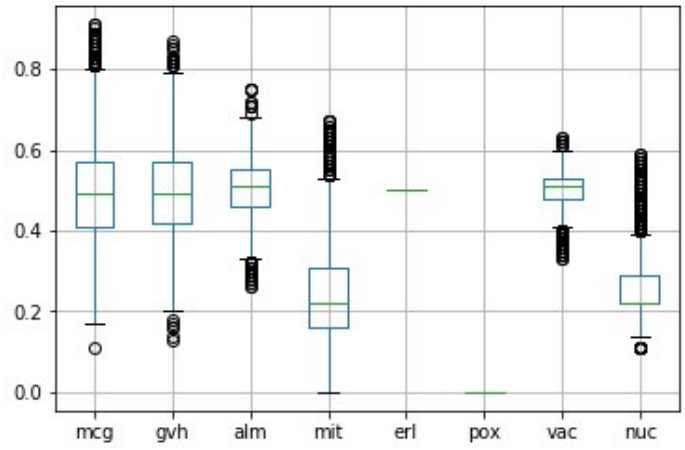
4. Dataset and pre-processing

The dataset used was Yeast proteins have 1484 Instances and 9 (8 predictive, 1 target) with no missing values.
There are 10 classes

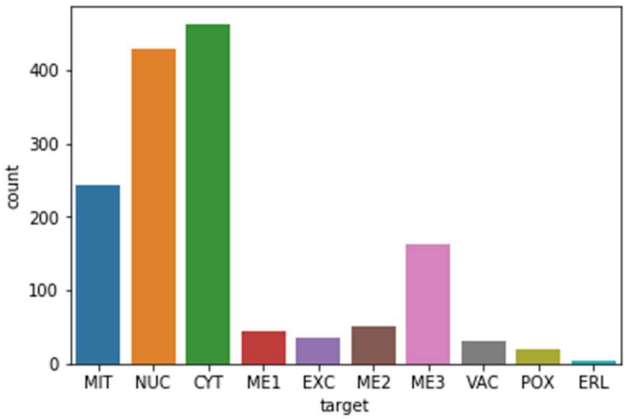
CYT (cytosolic or cytoskeletal)	463
NUC (nuclear)	429
MIT (mitochondrial)	244
ME3 (membrane protein, no N-terminal signal)	163
ME2 (membrane protein, uncleaved signal)	51
ME1 (membrane protein, cleaved signal)	44
EXC (extracellular)	37
VAC (vacuolar)	30
POX (peroxisomal)	20
ERL (endoplasmic reticulum lumen)	5



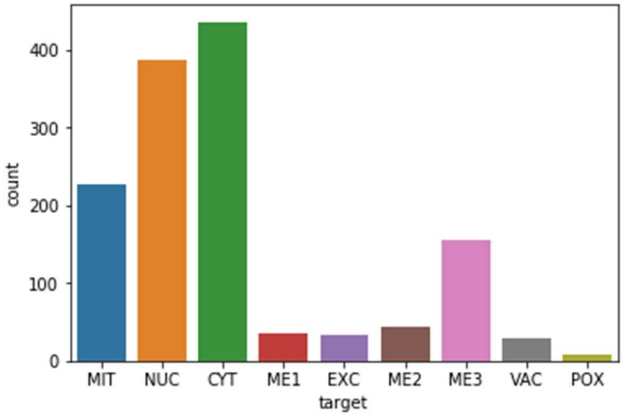
Yeast data box plot with outliers



yeast data box plot after reduce outliers



with outliers



after reduced the outliers

Bar graph data point numbers per class

It is clear from box plot above that there are many numbers of outliers. I used z-score method to find out the outlier and reduce it. This method assumes the outlier's data are the points far from the mean of Gaussian distribution and normalized any data point (xi) that it's zi larger than a threshold zth.
Where (zi = xi – mean / sd) and zth = +3 , -3.[10]

However, when 126 outlier's data point removed, ERL class has been removed all its data points completely which cause another problem so I did the experiments with the original dataset.

5. Experiments and results

We implemented three random forest experiments using different techniques to tackle with the imbalanced data problem and in all of them the parameters of the random forest classifier function are assign as baseline as below:
n_estimators = 200 , oob_score = True , random_state = 10

5.1 Experiment 1

Random Forest (RF)

Classify yeast Protein Localization by implement random forest using 70% training set. The random forest was used to train 200 trees as baseline experiment. To tuning the parameters, we used grid search hyper parameter using range of values for the parameters which were

n_estimators: [100,300,500,700,900], max_features: ["sqrt", "log2", None],

'max_depth' : [10, 20, None], min_samples_split: [2,5,10]

find the best training accuracy result 61.17 % from parameter's values after different combination with cross validation =5. We specify the cv = 5 as the minority class has 5 data points and to ensure that each class can be in each partition for the training.
The values of tuning parameters are:

max_depth = 10, max_features = sqrt, min_samples_split = 2 , n_estimators = 300

then fit the model and predict the classification using 30% (446 dataset) test set and get test accuracy 63.90 %. The table below shows the performance and the misclassification in confusion matrix result.

Class label	precision	recall	f1-score	Support (no. of patterns)	CYT	ERL	EXC	ME1	ME2	ME3	MIT	NUC	POX	VAC
CYT	0.57	0.72	0.64	140	101	0	0	0	2	3	6	26	1	1
ERL	1.00	1.00	1.00	1	0	1	0	0	0	0	0	0	0	0
EXC	0.64	0.64	0.64	11	3	0	7	0	1	0	0	0	0	0
ME1	0.88	0.74	0.80	19	0	0	2	14	1	1	0	0	1	0
ME2	0.33	0.33	0.33	12	3	0	1	2	4	0	2	0	0	0
ME3	0.71	0.89	0.79	45	2	0	0	0	0	40	0	3	0	0
MIT	0.73	0.60	0.66	77	19	0	0	0	2	4	46	6	0	0
NUC	0.66	0.54	0.59	128	42	0	0	0	2	6	9	69	0	0
POX	0.60	0.60	0.60	5	2	0	0	0	0	0	0	0	3	0
VAC	0.00	0.00	0.00	8	5	0	1	0	0	2	0	0	0	0
micro avg.	0.64	0.64	0.64	446										
macro avg.	0.61	0.61	0.61	446										
weighted avg.	0.64	0.64	0.63	446										

5.2 Experiment 2

Class weights Random Forest (CWsRF)

Random forest with class weights was conducted on training 70% of the data set. The minority class got high weight while the majority had the low weight using the function class_weight.compute_class_weight.

'CYT': 0.32136223, 'ERL':25.95, 'EXC': 4.325 , 'ME1': 4.152 , 'ME2': 2.66153846, 'ME3': 0.87966102, 'MIT':0.62155689, 'NUC':0.3448505, 'POX': 6.92, 'VAC':4.71818182

We fit the random forest classifier on grid search function with same range of the parameter's values in the previous experiment to tuning the parameters with 5-fold validation. The best training accuracy was 61.07% by assign the parameters values as below:

max_features = 'sqrt', max_depth = 20, min_samples_split = 10, n_estimators = 500

Using the model to predict the 30% testing data set. The test accuracy was 62.33% and the Table below illustrates the performance and confusion matrix.

Class label	precision	recall	f1-score	Support (no. of patterns)	CYT	ERL	EXC	ME1	ME2	ME3	MIT	NUC	POX	VAC
CYT	0.60	0.62	0.61	140	87	0	0	0	2	4	13	30	1	3
ERL	0.50	1.00	0.67	1	0	1	0	0	0	0	0	0	0	0
EXC	0.64	0.64	0.64	11	1	0	7	0	1	0	1	0	0	1
ME1	0.81	0.89	0.85	19	0	0	1	17	0	1	0	0	0	0
ME2	0.33	0.33	0.33	12	0	1	1	4	4	0	0	1	0	1
ME3	0.68	0.91	0.78	45	2	0	0	0	0	41	0	2	0	0
MIT	0.65	0.58	0.62	77	14	0	1	0	2	5	45	9	1	0
NUC	0.63	0.57	0.60	128	35	0	0	0	3	7	10	73	0	0
POX	0.60	0.60	0.60	5	2	0	0	0	0	0	0	0	3	0
VAC	0.00	0.00	0.00	8	5	0	1	0	0	2	0	0	0	0
micro avg.	0.62	0.62	0.62	446										
macro avg.	0.54	0.62	0.57	446										
weighted avg.	0.62	0.62	0.62	446										

5.3 Experiment 3

Oversampling Random Forest (OSRF)

We implemented the same previous experiment but, instead of weighted the classes we used one of the resampling algorithms to solve the imbalanced dataset.

We split the data to 70% training set and applied the SMOTE algorithm to get (323) data points for each class. Following that, we train the random forest algorithm using grid search with the same parameter's values range as previous experiments and 5-fold validation. We got the best combination of parameters
max_features= 'sqrt', max_depth = 20, min_samples_split= 10, n_estimators= 900

and training accuracy 87.08% which was higher than the accuracy of previous experiments. When predicted the 30% testing set on the model, the testing accuracy was 60.76%.

Below the result of performance and confusion matrix of the testing data set

Class label	precision	recall	f1-score	Support (no. of patterns)	CYT C	ERL	EXC	ME1	ME2	ME3	MIT	NUC	POX	VA
CYT	0.60	0.58	0.59	140	81	0	0	0	3	3	13	33	2	5
ERL	1.00	1.00	1.00	1	0	1	0	0	0	0	0	0	0	0
EXC	0.64	0.64	0.64	11	1	0	7	0	1	0	1	0	0	1
ME1	0.83	0.79	0.81	19	0	0	0	15	1	1	0	0	1	1
ME2	0.31	0.42	0.36	12	2	0	0	3	5	1	0	0	0	1
ME3	0.67	0.87	0.76	45	0	0	1	0	1	39	0	3	1	0
MIT	0.67	0.61	0.64	77	11	0	1	0	2	5	47	9	1	1
NUC	0.61	0.57	0.59	128	35	0	1	0	3	7	9	73	0	0
POX	0.38	0.60	0.46	5	1	0	0	0	0	0	0	1	3	0
VAC	0.00	0.00	0.00	8	5	0	1	0	0	2	0	0	0	0
micro avg.	0.61	0.61	0.61	446										
macro avg.	0.57	0.61	0.58	446										
weighted avg.	0.61	0.61	0.61	446										

The table 1 below shows the training and test accuracy and auc_roc for the three algorithms. Overall, what stands out from the table is that the best test accuracy result was RF with 63.9% and 61.1% training accuracy.

In details, it can be seen that the CWsRF method had no clear improvement from the RF with 62.3% and 61% test and train accuracies respectively. Another interesting point is that the oversampling method achieved improvement in the training accuracy by 87.%. However, the test result was quite lower than the training. That can be explained by the model was overfit as a result of generate samples for balance the data.

We can notice that the areas under curve were similar for the three algorithms. The weighted random forest experiment seems to be slightly superior than the others to 78.2%.

Algorithms	Training accuracy%	Test accuracy%	Auc_roc%
RF	61.175	63.901	77.84
CWsRF	61.078	62.331	78.29
OSRF	87.089	60.762	77.80

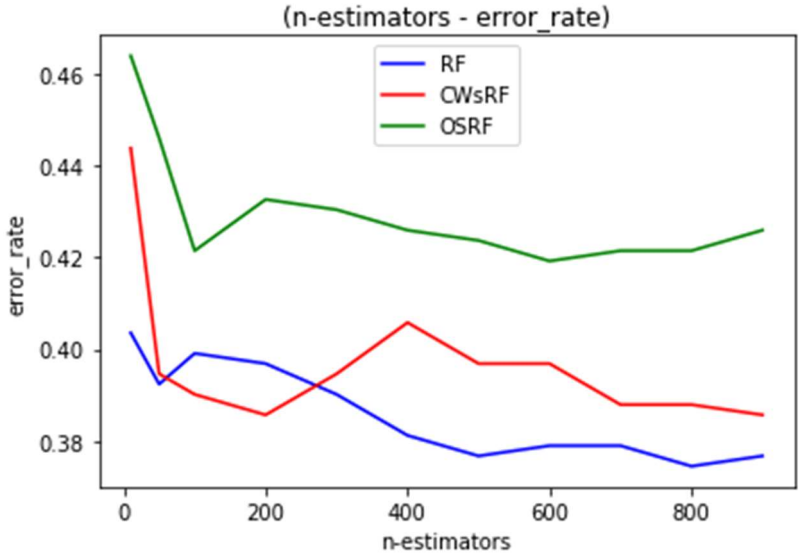
Table 1

It is important when the data was imbalanced to focus on how correct the model predict the minority classes [666]. The table 2 below presents the accuracy for each class result from predict 30% from the data set (446) data points. It can be seen that the VAC class with 8 data points for testing was misclassified in all three methods. By contrast, the ERL has 1 data point which was predicted correctly in all algorithms.

patterns	Class	RF %	CWsRF %	OSRF %
140	CYT	72.14	62.14	57.85
1	ERL	100	100	100
11	EXC	63.63	63.63	63.63
19	ME1	73.68	89.47	78.94
12	ME2	33.33	33.33	41.66
45	ME3	88.88	91.11	86.66
77	MIT	59.74	58.44	61.03
128	NUC	53.90	57.03	57.03
5	POX	60	60	60
8	VAC	0	0	0
Mean		60.53	61.51	60.68

Table 2

With respect to POX class, it was the same accuracy in all algorithms with 60%. In general, the class weights random forest algorithms had been made significant improvement in model performance compare with oversampling random forest classifier as methods to deal with imbalanced data set.



The line graph above compares the test error rates for the three experiments RF, CWsRF and OSRF in range of number of trees between 10 to 900 and the other parameters were default.

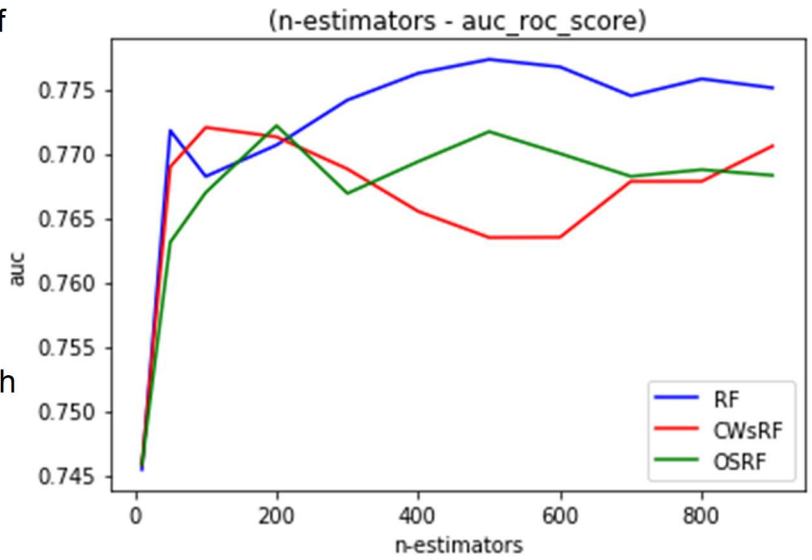
Overall, we can see that the greater number of trees generate to fit the model the lower error rate that model perform.

What stands out from the graph is that the RF algorithm had the best performance when the error tends to be minimized to nearly 33% at no. of trees around 500. Whereas, CWsRF predicted the test data with 39% error rate when there were 200 trees. Regard the OSRF when no. of trees around 600, the error rate decreased to approximately 42% which was lowest performance comparing with the other algorithms.

By comparing the algorithms with range of number of trees by the values of area under curve ROC.

It is clear to notice that the more trees contribute to build the model the higher auc_roc get.

Another interesting point is that at number of trees 500, the RF and OSRF experiments predicted the testing data with auc_roc to around 77.6% and 77% respectively. However, CWsRF reached to highest auc_roc value at 77.0% when the number of trees were 900.



These lines graph results of predicted 30% of the data set after fitted three algorithms with range of n-estimators 10 to 900 with default values of other parameters.

Conclusion

In this research we explored the use of random forest classifier to predict the yeast data set.

We first used random forest algorithm and then implement the class weights to reduce the affect of imbalanced data and the third trial, SMOTE oversampling method implemented to generate samples of minority classes data points. Generally, the best result we got with the random forest .

In the future we will train and build the model using k-fold cross validation when using oversampling algorithm. We would expect avoiding the model overfitting as we would generate the new samples for each train partition.

1- DataCamp, Understanding Random Forests Classifiers in Python, available online at https://www.datacamp.com/community/tutorials/random-forests-classifier-python. Last accessed 20/04/2019

2- Dimensionless, Introduction to Random forest, available online at https://dimensionless.in/introduction-to-random-forest/. Last accessed 22/04/2019

3- Silicon valley data science, Learning from Imbalanced Classes, available online at https://www.svds.com/learning-imbalanced-classes/. Last accessed 20/04/2019

4- Dataaspirant, How the random forest algorithm works in machine learning, available online at http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/. Last accessed 19/04/2019

5- MIN ZHU, JING XIA1, 2018. Class Weights Random Forest Algorithm for Processing Class Imbalanced Medical Data. IEEE access

6- Mlxtend, EnsembleVoteClassifier, available online at http://rasbt.github.io/mlxtend/user_guide/classifier/EnsembleVoteClassifier/. Last accessed 20/04/2019

7- Liu, Y(Hayden)., 2017. Python machine learning by example. Birmingham-Mumbai: Packt.

8- Last F., Douzas G., Bacao F.,2017, Oversampling for Imbalanced Learning Based on K-Means and SMOTE. NOVA Information Management School.

9- Aristoklis D. Anastasiadis, George D. Magoulas, and Xiaohui Liu, 2003. Classification of Protein Localisation Patterns via Supervised Neural Network Learning, Department of Information Systems and Computing, Brunel University.

10- Kdnuggets, Four Techniques for Outlier Detection, available online https://www.kdnuggets.com/2018/12/four-techniques-outlier-detection.html. Last accessed 15/04/2019.

11- towardsdatascience, Understanding AUC - ROC Curve, available online https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5. Last accessed 17/04/2019.

12- machinelearningmastery, Classification Accuracy is Not Enough: More Performance Measures You Can Use, available online https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/. Last accessed 17/04/2019