IITM-CS5691 : Pattern Recognition and Machine Learning — Release Date: 23 Oct, 2021
New Method Writeup — Due Date: 10 Nov, 2021

Teammate 1 Roll No: CS21M057 — **Teammate 1 Name:** Sanjanaa G V
Teammate 2 Roll No: CS21M021 — **Teammate 2 Name:** Gudivada Harsha Vardhan

- Dear Student, You may have tried different methods for predicting each of the clinical descriptors in the data contest. Submit a write-up of the methods chosen for the data contest in the template provided below. **You will have to add the details in your own words and submit it as a team in gradescope**.
- We will run plagiarism checks on codes/write-up, and any detected plagiarism in writing/code will be strictly penalized.

1. (points) [Name of the Method chosen for each descriptor, and its Paradigm: paradigm could be linear/non-linear models, etc.)]:

Solution: Various machine learning classifiers were explored, such as Logistic regression, Naive Bayes classifier, support vector machines with radial basis kernel, linear kernels, ensemble bagging techniques like random forests, ensemble boosting techniques like gradient boosting classifier, adaBoost classifier. We were given with six clinical descriptors and for every descriptor, the problem is a single binary classification problem, to predict if the given gene has a neurodegenerative disease. The above mentioned models were applied using sklearn library.

1. Logistic Regression: This is a probabilistic model for binary classification. It learns the probability mass function of the output label given the input, $p(y|\mathbf{x})$.

$$p(y|\mathbf{w},\mathbf{x}) = \sigma(\mathbf{w}^\top\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{1} + \mathbf{e}^{-\mathbf{w}^\top\mathbf{x}}}$$

Here, the $\mathbf{w}^\top\mathbf{x}$ is the score for the input $\mathbf{x}$. Sigmoid function, σ turns it into a probability and the $\mathbf{w}^\top\mathbf{x}$ indicates a linear model.

2. Ensemble methods: These methods use a combination of models to increase the accuracy and combine a series of k learned models with an aim of creating an improved model.
There are 2 paradigms of an ensemble method, they could either be a parallel ensemble or a sequential ensemble.

(i) Parallel Ensemble - Bagging - Random Forest Classifier - These models combines several models, each of which were built separately to fit the data. These allow Naive Parallelism. Bagging is a type of parallel ensemble. In bagging, to increase the variance, each individual node of each tree is learned on only a subset of features. This helps in case there is a highly predictive feature as all learners will use the feature and hence will be correlated. This ensemble method is the random forest classifier.

(ii) Sequential Ensemble - Boosting : In Sequential models, each model built explicitly uses other models and tries to make up for the weakness of the rest of the models. In other words, these models use the weakness of each submodel to boost the performance. Boosting combines multiple base classifiers to produce a strong classifier.

- AdaBoost Classifier - Adaptive boosting is a boosting algorithm, in which, at each iteration, it changes the sample distribution by modifying the weights of each instance. It increases the weights of the wrongly predicted instances and reduces the weights of the correct ones. After being trained, the weak learner is added to the strong learner according to its performance on the new distribution.

- Gradient Boosting classifier - This classifier does not modify the sample distribution, but instead, trains on a new sample distribution. The weak learner trains on the remaining error of the strong learner. At each iteration, the remaining error is calculated and a weak learner is fit to these errors and then it is added to the strong classifier using gradient descent.

Out of these classifiers, the combination of AdaBoost and Logistic regression gave us the best score.
CD01: AdaBoost Classifier, a sequential ensemble with maximum of 400 estimators at which boosting is terminated and a learning rate of 0.1. This classifier was implemented with the help of sklearn library.
CD02: AdaBoost Classifier, a sequential ensemble with maximum of 400 estimators at which boosting is terminated and a learning rate of 1. This classifier was implemented with the help of sklearn library.
CD03: Logistic Regression, a linear model with default parameters in sklearn.
CD04: Logistic Regression, a linear model with default parameters in sklearn.
CD05: AdaBoost Classifier, a sequential ensemble with maximum of 20 estimators at which boosting is terminated and a learning rate of 0.3. This classifier was implemented with the help of sklearn library.
CD06: AdaBoost Classifier, a sequential ensemble with maximum of 20 estimators at which boosting is terminated and a learning rate of 0.3. This classifier was implemented with the help of sklearn library.

This combination gave the best score of 0.48229.

2. (points) [Brief description on the dataset: could show graphs illustrating data distribution or brief any additional analysis/data augmentation/data exploration performed]

Solution:
Data processing is the conversion of the original data into modified datasets which can help the models predict better. We are given with 2 datasets. Here are a few processing procedures to take into consideration.

(i) Handling missing values: In this step, in case the dataset has a missing value, we fill it in with the average value available in that feature column or we can delete all the rows containing missing features or remove the features which have the missing values. However, the given 2 datasets had no missing values and no action was required for this purpose.

(ii) Feature Scaling: It is a method used to normalise the range of independent variables or features of the data. The sklearn.preprocessing package provides several methods that can fit and transform classes to change the raw feature vectors into a representation that is more suitable for modelling our classifier.

- Standardization using StandardScaler: Standardization is performed to make sure the individual features is more or less like a standard normally distributed data, that is, Gaussian with zero mean and unit variance (else it might behave badly). StandardScalar is used to transform the data to center by removing the mean value and scale it by dividing non-constant features by their standard deviation. This is helpful when the objective function of a learning algorithm assumes that all features are zero-centered and have unit variance.

- Normalization using Normalizer: It is the process of scaling individual samples to have unit norm.

We have explored the above two methods of feature scaling. However, it did not generate a better score in our model.

(iii) Class Imbalance problem: This refers to the problem with classification where the classes are not represented equally. For example in our dataset 1, the clinical descriptor 1 with over 130 instances has only 33 instances labelled with Class-1 and the rest 97 instances labelled with Class-0. This is an imbalanced dataset and the ratio of Class-1 and Class-2 is 33:97 which is almost 1:3. This imbalance might lead to prediction biased towards the majority class. To solve this problem, we have explored the following methods:

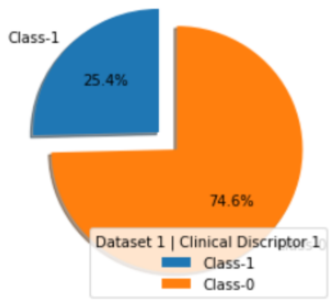
- Resampling the dataset: This involves creating a new transformed version of the training dataset in which the selected examples have a different class distribution. Random Oversampling is a method which involves randomly creating duplicate examples in the minority class and Random Undersampling involves randomly deleting examples in the majority class.

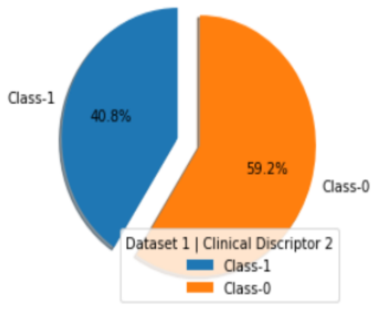
- Synthetic minority Oversampling Technique SMOTE: SMOTE selects samples that are close in the feature space, draws a line between the samples in the feature space and draws a new sample at a point along that line, i.e. a random sample from the minority class is first chosen. Then k of the nearest neighbors for that sample are found. A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two samples in the feature space. The method works as it causes the classifier to build greater decision regions that contain nearby minority class points. A con of this approach is that the samples are created without considering the majority class which could result in contradictory samples if there is a strong overlap for the classes.

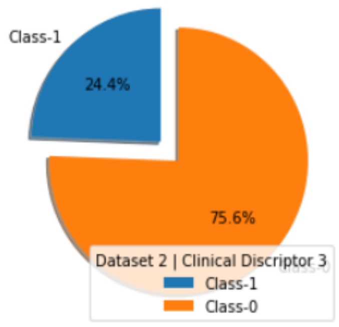
- Adaptive Synthetic Sampling Approach ADASYN: It is a generalization of SMOTE and aims to oversample the minority class by generating synthetic instances of it. However, it considers the density distribution which decides the number of synthetic instances generated for samples which are difficult to learn. Thus, it helps in adaptively changing the decision boundaries based on the samples difficult to learn.

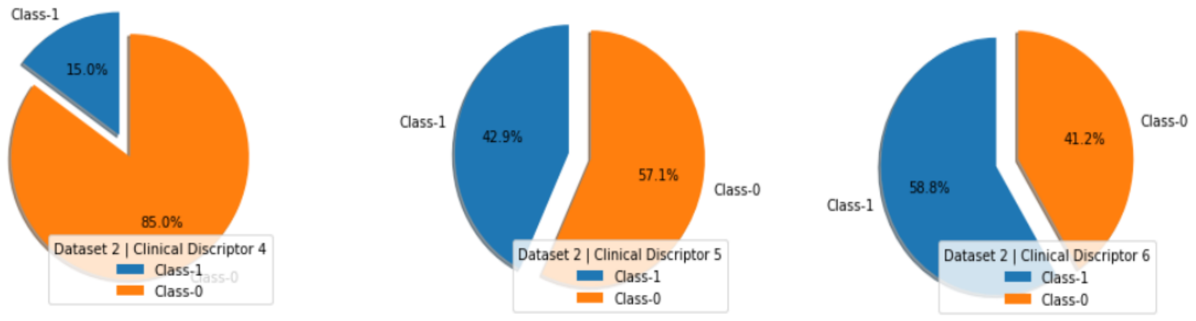
- Using the inbuilt parameter 'class-weight': Some of the classifiers have an inbuilt parameter to balance the classes of an imbalanced dataset. We have tried balancing the classes using inbuilt parameters of such classifiers. These parameters balance classes by sampling data points using the frequencies of the classes. This method showed good results on random forest classifier (on all 6 descriptors) by generating a better score.

ADASYN worked better on our model. However, implementing it was quite challenging and we have implemented it using the imblearn library.









The above figures show the class distribution in the given six clinical descriptors.

(iv) Dimensionality reduction: The given 2 datasets have the following dimensions.
-Training dataset of the first study contains 130 training examples and 22282 feature columns.
-Training dataset of the second study contains 340 training examples and 54674 feature columns.
-Test dataset of the first study contains 100 training examples and 22282 feature columns.
-Test dataset of the second study contains 214 training examples and 54674 feature columns.

This is a very high dimensional data of gene expressions. This 'Curse of dimensionality' leads to data sparsity and high variance. So, we tried methods to reduce dimensionality of the dataset. This is is done under the assumption that not all the thousands of genes will contribute significantly to the clinical descriptor. We have tried Principal Component Analysis (PCA) for this purpose using the package, sklearn.decomposition. PCA is a process of reducing the dimension of the features with minimum to nil loss of data. However, this procedure did not improve the score generated by our model.

3. (points) [Brief introduction/motivation: Describe briefly the reason behind choosing a specific method for predicting each of the descriptors (could show plots or tables summarizing the scores of training different model)]

Solution:

Classifier Used	**Data Preprocessing techniques**	**Score**
Support Vector Machine with linear Kernel	-	0.35628
Support Vector Machine with Gaussian Kernel	Feature Scaling using Standard Scalar and Dimensionality reduction using PCA	0.21060
Naive Baye's Classifier	Feature Scaling with Standard Scalar Dimensionality Reduction using PCA	0.01077
Gradient Boosting classifier - sequential ensemble	Feature Scaling using Standard Scalar and Dimensionality reduction using PCA	0.20071
Gradient Boosting classifier - sequential ensemble	Feature Scaling using Standard Scalar	0.39075
Gradient Boosting classifier - sequential ensemble	-	0.40014
Gradient Boosting classifier - sequential ensemble	SMOTE oversampling method from imblearn	0.35460
Gradient Boosting classifier - sequential ensemble	ADASYN oversampling method from imblearn	0.45129
Random Forest classifier - parallel ensemble	-	0.30790
Random Forest classifier - parallel ensemble	ADASYN oversampling method	0.34730
Random Forest classifier - parallel ensemble	ADASYN oversampling method Feature selection	0.32672
Random Forest classifier - parallel ensemble	Inbuilt parameter class_weight = "balanced"	0.41799
Logistic Regression	ADASYN oversampling method	0.32162

AdaBoost sequential Ensemble	-	0.40144
AdaBoost sequential Ensemble	Inbuilt Parameter base_estimator = Logistic Regression	0.34995
Logistic Regression on CD01, CD02 AdaBoost sequential Ensemble on CD03, CD04, CD05, CD06	- -	0.45129
Gradient Boosting sequential Ensemble on CD01, CD02 AdaBoost sequential Ensemble on CD03, CD04, CD05, CD06	- -	0.38930
Random Forest Classifier on CD01, CD02 AdaBoost sequential Ensemble on CD03, CD04, CD05, CD06	Inbuilt parameter class_weight = "balanced" -	0.40761
Random Forest Classifier on CD01, CD02 AdaBoost sequential Ensemble on CD03, CD04, CD05, CD06	Inbuilt parameter class_weight = "balanced" Inbuilt parameter base_estimator = Decision Trees	0.33754
Logistic Regression on CD01, CD02, CD03, CD04 AdaBoost sequential Ensemble on CD05, CD06	- -	0.45755
Logistic Regression on CD01, CD02, CD03, CD04 AdaBoost sequential Ensemble on CD05, CD06	Feature Scaling with Normalizer	0.36665
Logistic Regression on CD01, CD02, CD03, CD04 AdaBoost sequential Ensemble on CD05, CD06	ADASYN oversampling method to handle imbalanced dataset	0.47600

Logistic Regression on CD03, CD04 AdaBoost sequential Ensemble on CD01, CD02, CD05, CD06	**ADASYN oversampling method to handle imbalanced dataset Crossvalidation to tune hyperparameters**	**0.48299**

Initially we started by using the same classifier on all six descriptors to figure out which classifier gives an overall better performance. As you can see from the table above, Naive Baye's classifier, support vector machine with linear/ non-linear kernels gave a poor performance. After this we went on to explore Ensemble methods. Ensemble methods performed better than general classifiers. We can see from the table that bagging ensemblers like Random classifiers performed better than decision trees. And boosting ensemblers like AdaBoost and Gradient Boosting classifier performed better than general classifiers. Further, a combination of logistic regression(CD03, CD04) and adaboost classifiers (CD01, CD02, CD05, CD06) outperformed (0.48299) the results obtained by simple ensemblers (all descriptors modelled using a particular ensembler). Keeping these results in mind, we have chosen the combination of logistic regression(CD03, CD04) and adaboost classifiers (CD01, CD02, CD05, CD06) as the classifiers for predicting the given clinical descriptors.

4. (points) [As the data comes from a clinical setting, model interpretation is also an important task along with prediction. Describe briefly your preferred choice of methods if you are asked to determine the significant genes behind the regulation of the endpoints or restrict the number of features/genes.]:

Solution: Preferred choice of methods to choose significant genes:
1. Observe the coefficients: This could be the easiest way to examine the importance of a feature, observe the model's coefficients. For example, logistic regression boils down to an equation in which coefficients are assigned to each input feature. Larger the coefficients, higher the influence on the model. Finally, we can proceed by choosing the features which have higher coefficients and train our model with the newly transformed dataset.

2. Feature importance from a tree-based models: Tree based models like Random Forest classifiers have a built in parameter called *'feature_importances'* which can be used to discard unimportant features and extract the features with high *'feature_importances'*. However, the results from this approach might end up a little biased as the tendency of this approach is to inflate the importance of the continuous features or high cardinality categorical variables.

3. Principal Component Analysis (PCA): PCA will not give us the most important features directly. However, it will transform the given higher dimensionality data into N dimensions and it will return the N principal components, where N is the number of original features.

4. Univariate Selection: Univariate statistics tests like Chi Square Test can be used to select those features that have a stronger relation with the output variable. Chi-square test measures dependence between stochastic variables. We can define our Null hypothesis as "The feature is independent or the feature does not provide sufficient information for the output variable". Chi-square value can be calculated using the formula

$$X_c^2 = \frac{\Sigma(O_i - E_i)^2}{E_i}$$

where, c = degree of freedom, O = observed values, E = expected values, X = random variable. We can use this to remove the features/genes that are the most likely to be independent of class and therefore irrelevant for classification.

5. Variance: Genes with low variance across the samples can be removed. It means we can remove the genes which have similar values across the samples.

6. Pre-modelling: Before the final classifier, we can fit the data through another classifier/estimator and check genes/features with high coefficients. Those significant genes/features with high coefficients can then be selected for training in another classifier.

7. Greedy selection: We can use greedy selection of genes/features iteratively. We can choose the best new gene/feature and add it to our selected genes/features based a cross-validation score. That is, we start with 0 features and choose the best single feature with the highest score. The procedure is repeated until we reach the desired number of selected features.

8. Column Modified: In case of missing values, we can remove the features which have missing values from the dataset as the presence of large number of missing values outweighs the information provided by these features.

5. (points) [Share your thoughts on each of the endpoints: whether easy/difficult to predict]:

Solution: We have tried various classifiers on each of the endpoints. For each classifier on every descriptor, we have tried to train each model with a list of learning rates, regularization parameters and different number of estimators and chose the learning rate, regularization parameters, n_estimators that performed the best on that particular data model.

We have also used cross validation for tuning the hyper parameters. In the basic approach, called k-fold CV, the training set is split into k smaller sets. The following procedure is followed for each of the k "folds":
1. A model is trained using $k - 1$ of the folds as training data
2. the resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).

Performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. Further, we repeated K-Fold n times. It is used to run K-Fold n times, producing different splits in each repetition. And then, we used StratifiedKFold, a variation of k-fold which returns stratified folds: each set contains approximately the same percentage of samples of each target class as the complete set. Together with this StratifiedKFold cross validation, we used Grid Search to find the best hyperparameters.

Clinical Descriptor 1: Prediction on this descriptor had the best result when we used ensemble models. AdaBoost with learning_rate = 0.1 and n_estimators, the maximum number of estimators at which the boosting is terminated as 400, and the default base_estimator SAMME.R, gave us a good result.

Clinical Descriptor 2: Prediction on this descriptor had the best result when we used ensemble models. AdaBoost with learning_rate = 1 and n_estimators, the maximum number of estimators at which the boosting is terminated as 400, and the default base_estimator SAMME.R, gave us a good result.

Clinical Descriptor 3: Prediction on this descriptor had the best result when we used linear models. Logistic regression with default regularization parameter C = 1 gave us a good result.

Clinical Descriptor 4: Prediction on this descriptor had the best result when we used linear models. Logistic regression with default regularization parameter C = 1 gave us a good result.

Clinical Descriptor 5: Prediction on this descriptor had the best result when we used ensemble models. AdaBoost with learning_rate = 0.3 and n_estimators, the maximum number of estimators at which the boosting is terminated as 20, and the default base_estimator SAMME.R, gave us a good result.

Clinical Descriptor 6: Prediction on this descriptor had the best result when we used ensemble models. AdaBoost with learning_rate = 0.3 and n_estimators, the maximum number of estimators at which the boosting is terminated as 20, and the default base_estimator SAMME.R, gave us a good result.

6. (points) [Add any additional information: like challenges faced or some details that would help us to better understand the strategies that you have utilized for model development]:

Solution: First, the datasets given are highly unbalanced. Handling the unbalanced data was quite challenging. Some methods did not improve our scores. However, on using the imblearn package in python to implement the Adaptive Synthetic Sampling oversampling method, improved our score a little.

Next, we are given with a very high dimensional data and working on this was quite challenging. Since the number of features are exponentially higher than the number of training examples, most of the classifiers we used would overfit resulting in poor scores. Using methods like PCA and the other feature selection methods were not very effective.

Since the model development procedure is highly based on data assessment and trial and error using multiple classifiers, choosing different combination of classifiers for each descriptor and analysing the score was highly challenging as well. Even if one of the six models worked perfectly well and the other five didn't, we would still end up getting a bad score and not realise if any of the models performed well at all. So, trying the different combination of classifiers without knowing the individual descriptor's performance was quite difficult.

Further, any method used to improve the efficiency, like feature scaling using StandardScalar, Normalizer, Dimensionality reduction using PCA or different feature selection methods, did not improve the performance of our models.

For cross validation purposes, some of the methods used from Scikit are 'RepeatedStratifiedKFold' for 'Repeated Stratified K-Fold cross validation' and 'GridSearchCV' for 'Grid Search cross validation'. However, these cross validation methods are highly time consuming tasks. Google Colab has an idle time out period which stops the code execution abruptly and hence resulted in loss of training information.

Out of all the classifiers learnt in class, ensemble methods had the best performance without any data processing steps. Ensemble methods involve combining different weak models to produce a strong model. Hence, it is less prone to overfitting than the other classifiers. Even among the ensemble methods, using the built in parameters to handle data processing would reduce the score. AdaBoost sequential ensembler along with Logistic regression for the third and the fourth clinical descriptors gave the best score.