# **IEE 520 Final Project**

## Mayur Waghela

## ASU ID – 1213523166

# Objective:

Given the training data, we were to build a model that would classify the given test data well which will be evaluated on the balanced error rate.

# Overview of steps:

# Data Preprocessing:

Before we go on to any steps, we need to understand out training data well. I looked for any missing data and found that there wasn’t any. Total number of rows and columns in the training set is summarized below. There are two target classes: -1 and 1.

## Training data:

Number of Instances: 2500

Number of columns: 68 (excluding target variable)

Missing data: None

## Checking data types:

The first row was row number. It was dropped, as it will not be contributing to our model building. All the other columns are continuous variables except for ﻿the columns 'x5', 'x13', 'x64' and 'x65'. These columns have nominal variables. For these columns, I wrote a function for one hot encoding them. This increased the count of columns to 79.

## Train Test Split:

Now to make sure we do not overfit or underfit the model, we split the training data into training and testing sets with 80% being used for training and 20% used for testing.

## Class balance check: ﻿

After splitting the training data, summary of the count of instances for the two classes is as follows:

|  |  |
| --- | --- |
| Class | Count (training data as is) |
| -1 | 1516 |
| 1 | 484 |

Thus, it is very clear that there is a significant class imbalance in the training set. Since, it is important to get good accuracy for both the classes and not just one of them, I decided to do up-sampling with replacement for minority class and down-sampling with replacement for majority class and bring the total count of each class instance to 1000. To avoid overfitting of minority class, I decided up-sample only up to 1000 instances and bring down the majority class to match the minority class samples instead of only up-sampling minority class to 1500 to match the majority class count. After resampling, the class count was as follows:

|  |  |
| --- | --- |
| Class | Count (after resampling) |
| -1 | 1000 |
| 1 | 1000 |

Now, the total training data size is 2000.

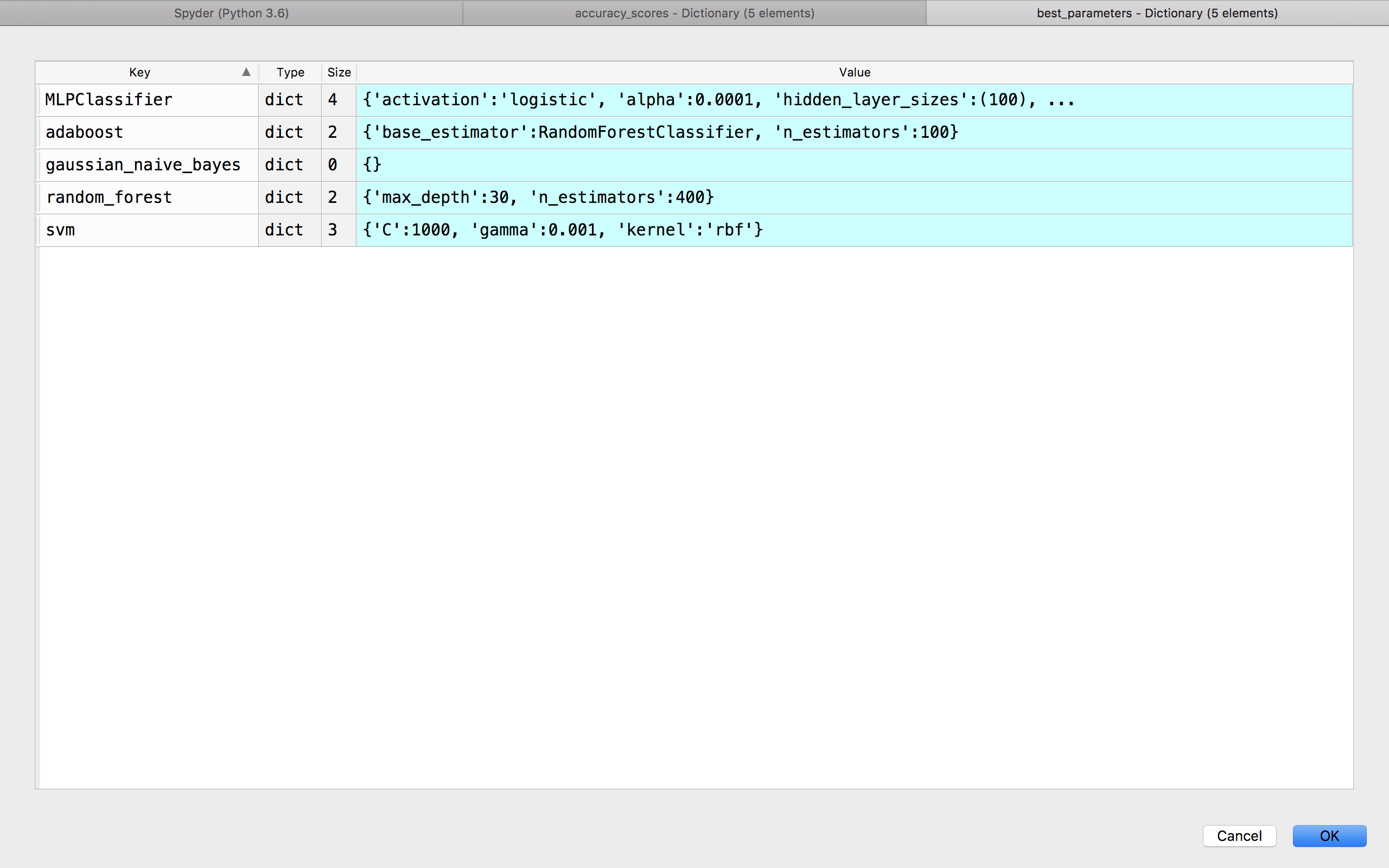
# Data Scaling:

Generally, scaling data helps to improve the time required for converging to a solution and also helps improve the accuracy. However. For this dataset, I found that unscaled data for random forest and gaussian Naïve Bayes gives better accuracy. Even though it might require more time and iteration to converge to a solution, considering the size of dataset, it is negligible. Hence, I decided to not scale the data for these classifiers. However, it was obvious that we will need to scale the data for MLP classifier, SVM and adaboost. MinMaxScaler was used to scale the data for these classifiers.

# Model Building:

To check the predictive capability of the model and tune the parameters a GridSearchCV with 5-fold cross validation was implemented. A class called ‘﻿CLASSIFIER\_SELECTION’ was defined which returned the classifier initialization and parameter dictionary of the selected classifier for GridSearchCV. Then a 5-fold cross validation was done using GridSearchCV and hyper=parameters for each classifier were tuned. The best classifiers were used to fit the training data and its performance was evaluated on the test data using total error rate, balanced error rate and mean cross validation error.

Hyper-parameters selected by each classifier:



The dictionary with various error scores is shown below:

{'gaussian\_naive\_bayes':

{'Total Error Rate': 27.400000000000006, 'Balanced Error Rate': 24.133333333333333, 'Mean CV error': 26.8},

'random\_forest':

{'Total Error Rate': 17.400000000000006, 'Balanced Error Rate': 30.533333333333328, 'Mean CV error': 15.000000000000002},

'svm':

{'Total Error Rate': 23.0, 'Balanced Error Rate': 21.2, 'Mean CV error': 23.050000000000004},

'MLPClassifier':

{'Total Error Rate': 17.60000000000001, 'Balanced Error Rate': 26.666666666666668, 'Mean CV error': 18.05},

'adaboost':

{'Total Error Rate': 18.0, 'Balanced Error Rate': 33.6, 'Mean CV error': 17.049999999999997}}

﻿

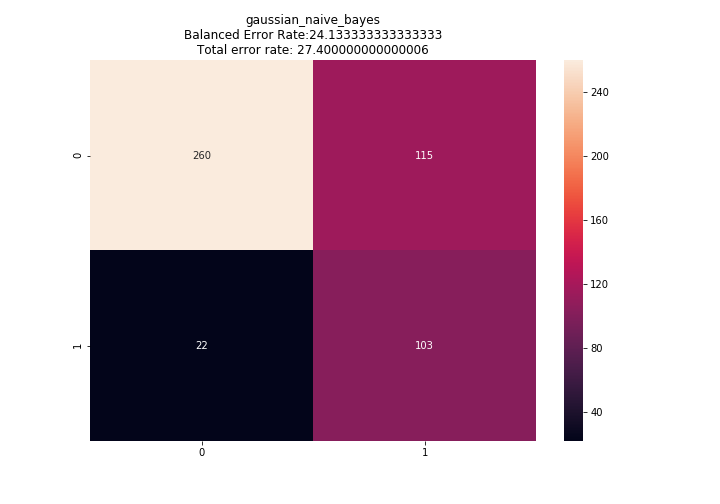
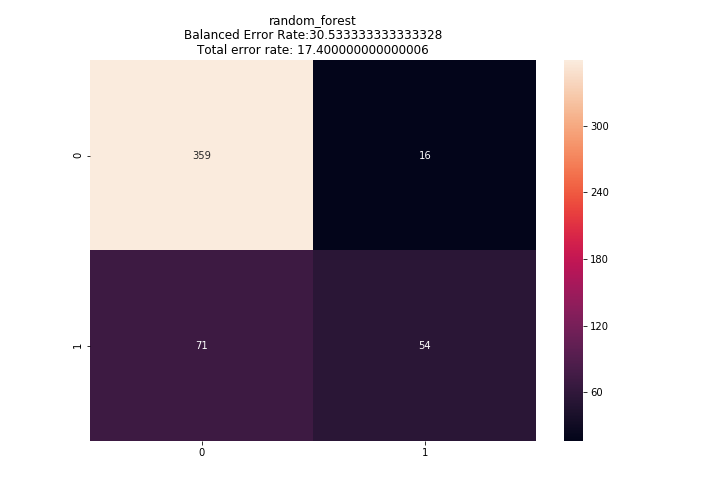
# Results and Model Selection:

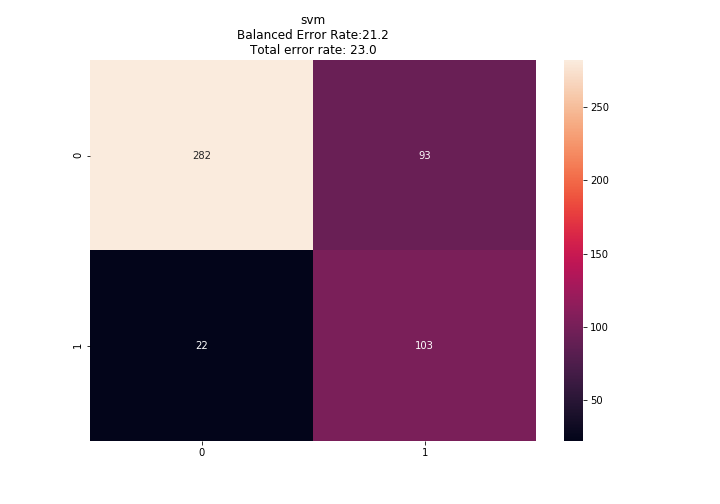
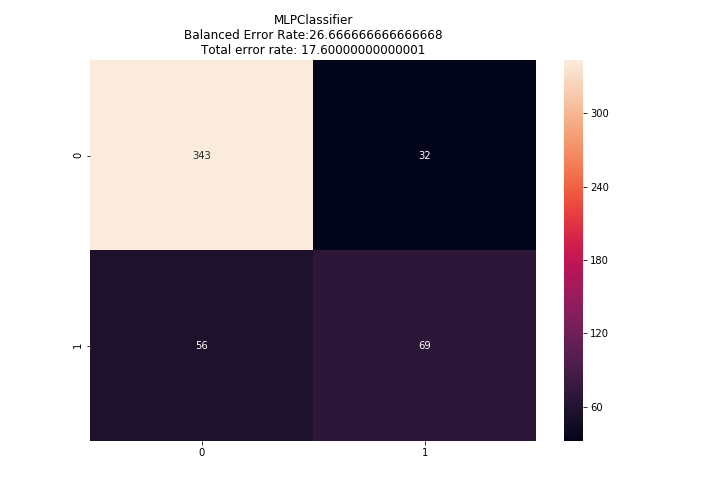
The classifiers used, and their respective Balanced error rates and Total error rates are summarized in the below table:

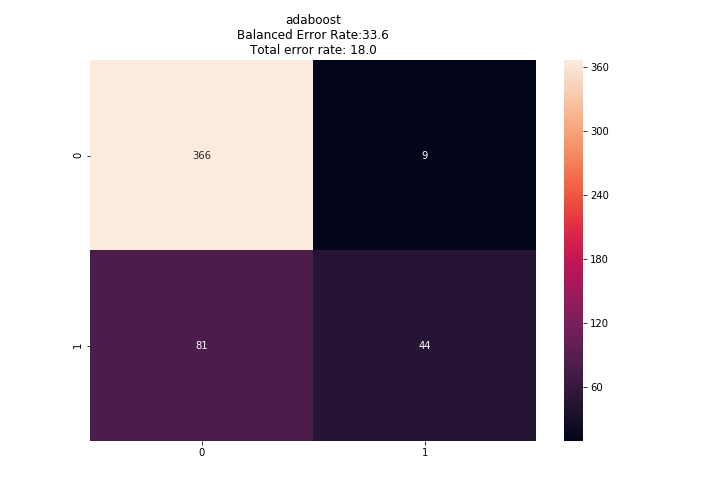
|  |  |  |
| --- | --- | --- |
| Classifier | Total Error Rate | Balanced Error Rate |
| gaussian\_naive\_bayes | 27.4% | 24.13% |
| random\_forest | 17.4% | 30.53% |
| svm | 23.0% | 21.2% |
| MLPClassifier | 17.6% | 26.67% |
| Adaboost | 18.0% | 33.6% |

From the above-mentioned dictionary, we see that there is no overfitting or underfitting in any of the models (the cross-validation error rate and test error rates are close to each other.) However, the balanced error rates are higher, except for SVM which is also the lowest.

The confusion matrices for the classifiers are as follows:



We are interested in balanced error rate, i.e. we want the accuracy in both the classed to be equally good. A high accuracy in one class and poor accuracy in another will lead to very high balanced error rate as can be seen from the confusion matrices of the classes and their balanced accuracy scores. Since SVM performs the best in terms of balanced error rate, with its total error rate also being close to the balanced error rate, I chose that classifier for predicting the test data outcome.

Selected classifier: Support Vector Machines (SVM): sklearn.svm.svc()

Classifier parameters: (

﻿C = 1000,

gamma = 0.001,

kernel = 'rbf',

random\_state = 235,

probability = True,

class\_weight='balanced'

)

# Test data prediction:

Before prediction, the test data was subject to OneHotEncoding and scaling using the same instance of LabelEncoder(), OneHotEncoder() and MinMaxClassifier that was used to fit the training data. This process was done simultaneously while performing those operations on the training data. Then using the above SVM classifier, we predicted the y for the test data given in a separate csv worksheet. Then it was written to a separate csv without heading and containing two columns – row numbers and predicted y.