**CIS 6930 - Applied Machine Learning Using Python**

**Project 2 Report**

1. Discovery of attributes that can be removed and added

When looking at the dataset, the first attribute that can be immediately removed is the ‘Name’ attribute as it won’t have any impact in determining whether or not a player lasts five years. As such the ‘Name’ column was removed and put into a new data frame which will be tested.

Further inspection of the data shows that there are some redundant attributes. For example, there is a ‘FGM’ (field goals made) attribute, a ‘FGA’ (field goals attempted) attribute, a ‘FG%’ (field goal percentage) attribute. ‘FG%’ is simply the percentage value of ‘FGM’ divided by ’FGA’. As such we can remove the attributes ‘FGM’ and ‘FGA’. By this logic, we can also remove the attributes ‘3PM’, ‘3PA’, ‘FTM’, and ‘FTA’. Similarly, the attribute ‘REB’ (rebounds) is a summation of the attributes ‘OREB’ (offensive rebounds) and ‘DREB’ (defensive rebounds). Thus, we can remove the attributes ‘OREB’ and ‘DREB’.

The ‘PPT’ (points per game) attribute can be misleading as a player can play for varying amounts of time during a particular game which could skew this attribute in one direction. A more reliable statistic would be the number of points made per minute as it better shows the consistency of the player. We can calculate this by multiplying the values in the ‘PPT’ attribute by the values in the ‘GP’ (games played) attribute and then dividing by the values in the ‘MIN’ (minutes played) attribute. By doing this, we get a new attribute ‘P/MIN’ (points per minute) and delete the attributes ‘GP’ and ‘PPT’. In fact we could do this for the remaining attributes as well. We create the attributes ‘REB/MIN’, ‘AST/MIN’, ‘STL/MIN’, ‘BLK/MIN’ and ‘TOV/MIN’ by dividing the respective attribute by the ‘MIN’ attribute. We remove the attributes ‘REB’, ‘AST’, ‘STL’, ‘BLK’ and ‘TOV’. We do all these new changes and create a new data frame which will be tested.

We have two data frames, one with only the ‘Name’ attribute removed and another with additional attributes removed/added as discussed above. We must now test to see if our changes actually improve the F1 scores of the models. The dataset has been split using the train\_test\_split() function with 80% data for training and 20% for testing. The random\_state parameter has been given a value to ensure that the training and test data splits are always the same. The models will go through 10-fold cross validation which will be executed with both a custom cross validation method as well as with GridSearchCV where default parameter values will be given.

1. K-Nearest Neighbors normalization and improvements

Below are the F1 scores produced by the K-Nearest Neighbors and Logistic Regression models on the first data frame (**without the removed/added attributes**):





Below are the F1 scores produced by the K-Nearest Neighbors and Logistic Regression models on the first data frame (**with the removed/added attributes**):





As we can see, the results produced by the custom cross validation method and GridSearchCV are the same indicating that they cross-validate in a similar fashion. We can see that F1 scores for the K-Nearest Neighbors model improves significantly while the F1 scores for the Logistic Regression model decrease marginally.

Due to the nature of the Random Forests and Artificial Neural Networks models, we will get a range of F1 scores. In order to gauge whether or not the removed/added attributes improve these models, we will run each model 5 times essentially giving us ten F1 scores (one from custom cross validation and the other from GridSearchCV).

**Note**: The early\_stopping parameter for the MLPClassifier (Artificial Neural Network) was set to ‘True’ because it was found that later when testing the best hyperparameters for GridSearch, the model ended up taking a long time when it was ‘False’.

Here are the tabulated results (F1 scores have been rounded to three places after the decimal point):







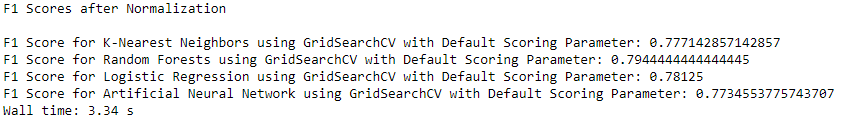


We can see that there is an improvement in F1 scores for both models albeit not a lot. Variance was also included to give an indication of the range of F1 scores produced by both models. The variance of the Random Forests model was less than that of the Artificial Neural Networks model and in both cases the variance improved when the data frame with the removed/added attributes was used.

From these results, we can see that that data frame with the removed/added attributes produce overall better F1 scores even if it is not by a lot. As such, we will be using this data frame for the rest of our testing. Also, since the custom cross validation method and GridSearchCV with default scoring parameters produced identical results for the K-Nearest Neighbors and Logistic Regression models, we will be using GridSearchCV for cross validation from now on.

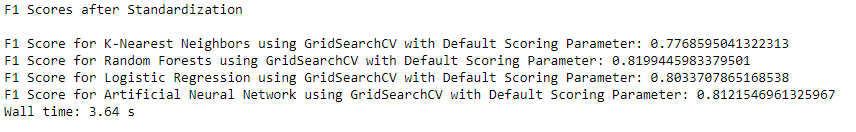
Two scaling methods were used: Normalization and Standardization. The data was normalized using the in-built normalize() function of sklearn while the data was standardized using the in-built scale() function of sklearn.

Here are the results once that data was normalized (note that the F1 scores for Random Forests and Artificial Neural Networks can vary and that this is simply one run of each model):



We notice that there is not any significant changes in the F1 scores, if anything each model’s F1 score slightly decreased. Normally, normalization would help a distance based model such as K-Nearest Neighbors but here we see a slight decrease in performance.

Below are the results once that data was standardized:

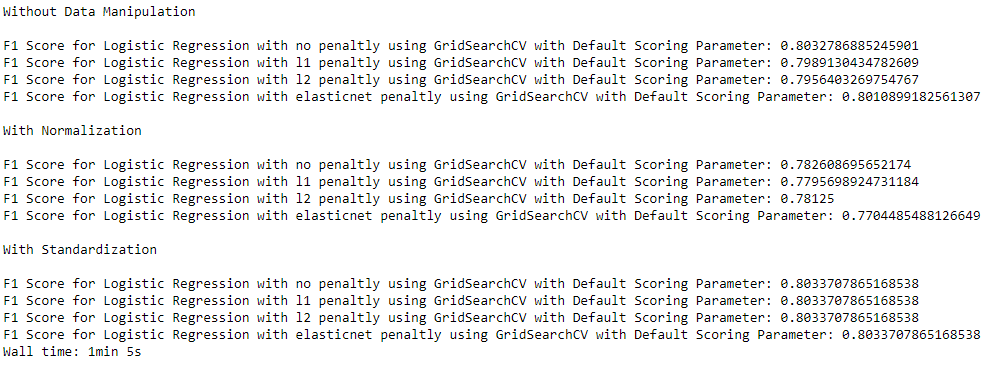


Again not any big improvements, but there is a slight increase in F1 scores for each model with the exception of the K-Nearest Neighbors model.

1. Regularization and Logistic Regression

Now we will try regularization on the Logistic Regression model to help combat any overfitting.

Below are the results (note that ‘saga’ solver was used as it is the only solver that supports all the penalty parameters, in addition an l1\_ratio of 0.5 was given when the elasticnet penalty was used):



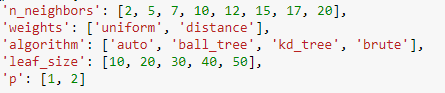
There aren’t many differences in the F1 scores, though it appears that the F1 score for “no penalty” is the highest both for when the data is not manipulated and when it has been normalized. An interesting thing to note here is that the F1 scores when the data has been standardized are identical regardless of the penalty parameter used.

1. Hyperparameters selection using GridSearch

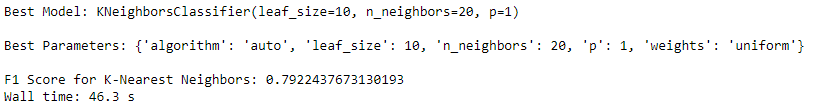
Now we try using GridSearch to select the best hyperparameters from a test set and see if that improves the F1 score. We will try this for when the data has not been scaled, when it has been normalized and for when it has been standardized.

For the K-Nearest Neighbors Model:

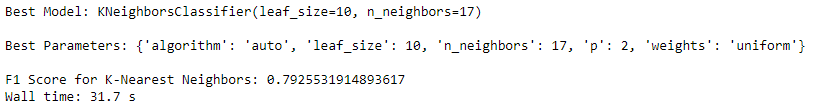
Parameters tested:



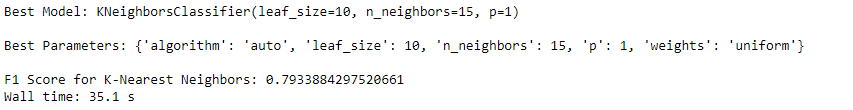
Results for no Data Manipulation:



Results for Normalized Data:



Results for Standardized Data:



We see that there is almost no change in the F1 scores while the best parameters are almost the same with only n\_neighbors and the power parameter varying.

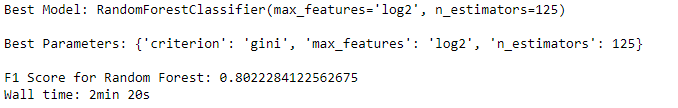
For the Random Forests Model:

(Note the F1 scores can vary between runs)

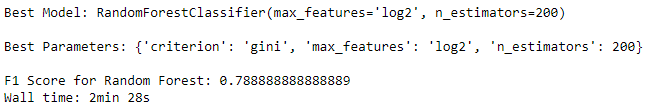
Parameters tested:



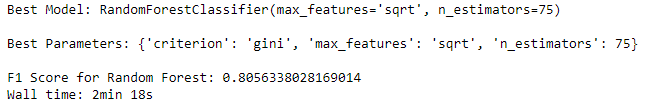
Results for no Data Manipulation:



Results for Normalized Data:



Results for Standardized Data:



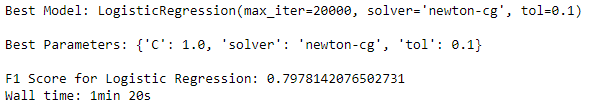
Again, there are no major differences in F1 scores. All the parameters tested vary with the gini impurity criterion being the only identical parameter.

For the Logistic Regression Model:

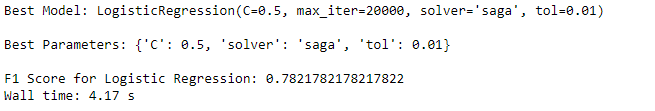
Parameters tested:



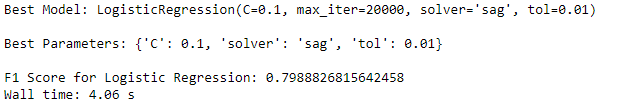
Results for no Data Manipulation:



Results for Normalized Data:



Results for Standardized Data:

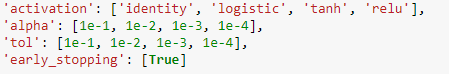


Once again, no big differences in F1 scores and the best parameters are different for each case. An interesting thing to note is that the run time for the model reduced a great deal once the data was scaled.

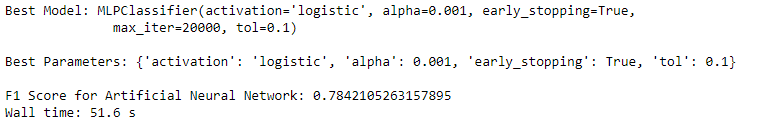
For the Artificial Neural Networks Model:

(Note the F1 scores can vary between runs)

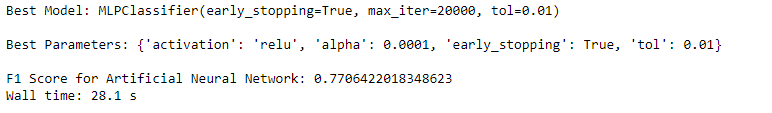
Parameters tested:



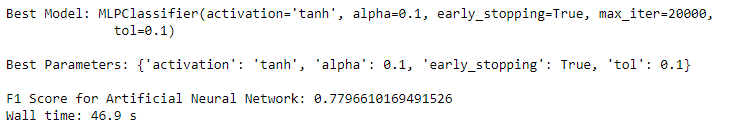
Results for no Data Manipulation:



Results for Normalized Data:



Results for Standardized Data:

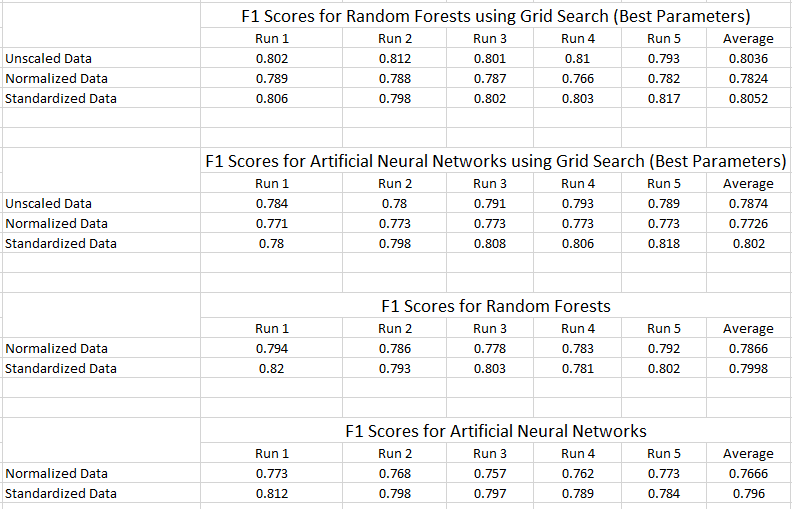


We don’t see any significant differences in F1 scores and the best parameters are different for each case.

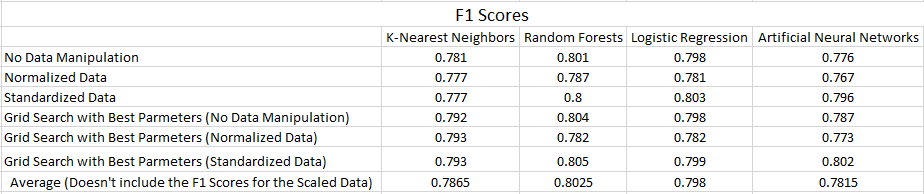
1. Best F1 score model

Finally, we will see which model produced the best F1 score on testing. In order to get a more accurate F1 scores for the Random Forests and Artificial Neural Networks models, we will run each model 5 times on both of the normalized and standardized datasets. In the case of GridSearch using the best parameters, we will run each model 5 times on the unscaled dataset as well as 5 times each on the normalized and standardized datasets.

Below are those results:

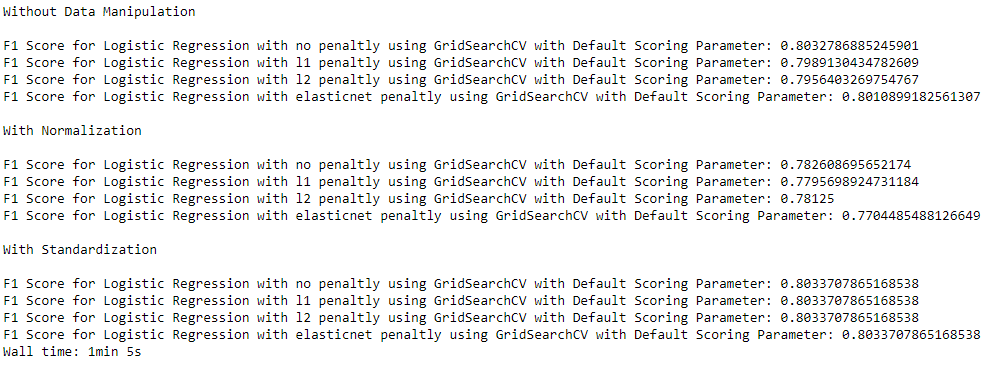


Now, I will compile all the F1 scores for each model and scenario into a table. Note that the F1 scores for the Random Forests and Artificial Neural Networks models are averages.



Now we can answer which model gave the best F1 score in two ways. One way is that we can take the average of F1 scores in each scenario and pick the model that gives the best F1 score. In this case, let us not consider the scenarios in which the data was scaled as that may unfairly benefit one model more than others. From the results we’ve obtained, we can say that the Random Forests model produced the best overall F1 score.

Another way we can choose the best model is to pick the highest F1 score in all the scenarios we’ve tested. This would include the above table and the F1 scores from when we used regularization on the Logistic Regression model. They have been re-posted them below.



From all the F1 scores, we see that the Random Forests model with GridSearch picking the best parameters and with the data standardized has produced the highest F1 score of approximately 0.805.

These are the best parameters chosen by GridSearch that produced this F1 Score:

