Applying Supervised and Unsupervised Methods to Predict Stroke Patient

Hong Beom Hur

Introduction

As a novice and trainee in the data analysis method, this hands-on experience of curating the dataset from beginning to end was very meaningful. The process of data exploration and thinking through the properties of the data set gave some insights into how to approach the classification problem. Although I could not produce any meaningful results, with a deeper understanding of the data set accumulated from the exploration, the practice of scaling, modeling, and visualization afterward provided me with a greater understanding of each practice and strategy in a practical sense. Running the models with different parameter settings and assessing the model to find optimal performance is important, but when input data features are not relevant to target variable and or not properly scaled or curated, the model will not be able to provide new insight. Most likely be the latter in this case. This report summarizes the process I undertook and the insights I gained throughout.

About the dataset

Among the myriads of the dataset available on Kaggle, I found a stroke prediction dataset. It is a simple binary classification problem set with properties that had socio-economic and behavioral measurements like employment status, residence type, smoking, and marital status along with some biological markers like Body Mass Index, age, and average glucose level. If a model can accurately predict stroke patients using these properties, it can provide some meaningful insights in making connection between the health and social surroundings which can be valuable knowledge.

Data Exploration

Though it has plenty of samples to build a model, the data set is heavily imbalanced and there is not so much contrast between the two classes in the features. Among 5110 total data points, about 95% of the data points are marked ‘0’ for no-stroke, and only 5% of the data points are marked ‘1’ for stroke. Since the goal is to predict patients with stroke, the imbalance in the set is problematic.

After examining the count values of the two classes, I began searching for missing data. Among 12 columns, only the BMI column had missing data. I deleted 201 rows without BMI values associated with them. I also plotted distributions of features separately for stroke and no stroke samples to see if they are distinct. If a disparity is shown among two groups, the feature can contribute to distinguishing the group, so this process can work as feature selection. The most notable and only feature showing the distinguishable disparity between the two classes is age. When plotted, stroke sample distribution is heavily skewed and observed highly in the older age group, but the age for no-stroke samples shows a relatively uniform distribution.

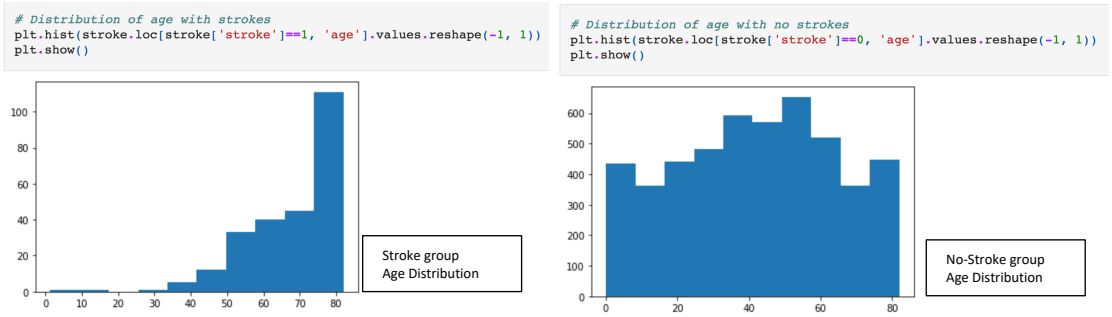

Figure 1. The distribution of age is distinguishable between the two groups.

Since there are only 2 samples under age 30 who are marked stroke, 2 out of 1570 samples, I decided to get rid of this age group to bring some balance between the two sets. By eliminating 1568 samples from the no-stroke group, the ratio became 92.5:7.5. As shown in Figure.2, no other features showed recognizable disparities in distributions which also raised a flag as a potential constraint in predicting the stroke samples. I looked at all the features, but only the distributions of numerical features are included here to conserve space. Given the observations, I picked two categorical features: hypertension, heart disease and two numerical features age and bmi as input features of supervised methods. No other transformation was done on the data for supervised method.

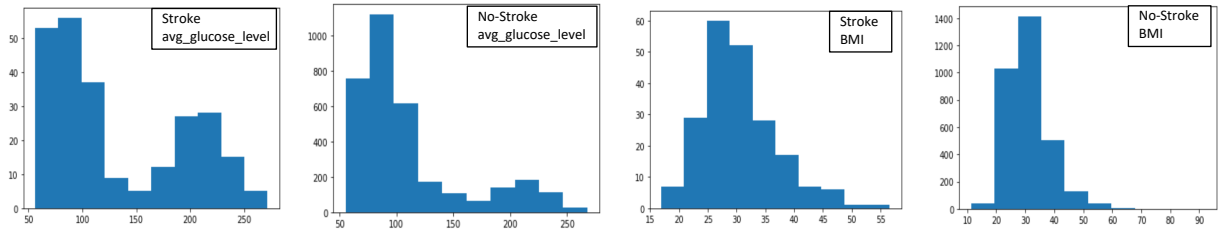

Figure 2. The distribution of average glucose level and BMI are not clearly distinguishable between the two groups. Note that X-axis scale is different in the two BMI plots.

Supervised Methods

Two supervised methods used in this project are logistic regression and random forest. Logistic regression can be used in classification because it fits a line using a logistic function. Instead of fitting a straight line to predict a continuous value like most linear regression models, it can predict a categorical value by fitting an S-shaped curve through the data points. Proper parameter settings are an important factor to consider to find an optimal performance of the model. Among many parameters inside the scikit-learn library logistic regression method, the 'C' parameter adjusts the amount of penalty that should be given when fitting the line to prevent the model from overfitting to the training data. Sklearn's method takes the inverse of the 'C' value given, so a lower 'C' parameter value means higher penatly and will make a model more generalizable and simple producing less variance between data sets by adjusting to the majority of the training set data points. In contrast, a higher 'C' parameter value will compose a model that focuses more on individual training data points with higher complexity and more variance between data sets but less bias toward the training data. In addition, 'penalty' and 'solver' parameters set the type of regularization method and algorithm used in regularization respectively. The 'random state' sets a

random seed number to make the result reproducible by making the same random combination accessible again later. Lastly, using 'GridSearchCV', a set of 'C' values was assessed and cross-validated.

All parameters produced the same accuracy metrics. R^2 is .96 and both RMSE and MSE are 0.04 suggesting the efficiency of model. However, the logistic regression method could not predict any stroke sample when the model was tested with test data. In fact, all prediction were 0 with not a single sample predicted to be labeled '1'. This is faulty and misleading because it will produce both high false negatives and high false positives. After all, there are 42 stroke samples included in the testing set though they are marginally represented. So, after short research on the web, I found the parameter 'class_weight' which will force the two classes to be in a balanced state by giving them inverse proportional weights based on their frequencies. This parameter set to 'balanced' in conjunction with a higher value of the 'C' parameter, set to 10, the logistic model was able to make more predictions of stroke samples from the test set.

```
# C=10
log_L1_gs = LogisticRegression(random_state=1, penalty="l1", solver='liblinear', C=10, class_weight='balanced')
log_L1_gs.fit(X_train, y_train)
y_pred_gs = log_L1_gs.predict(X_test)

print(y_pred_gs)
print("Number of predicted 0: ", list(y_pred_gs).count(0))
print("Number of predicted 1: ", list(y_pred_gs).count(1))
print("Number of true 0: ", list(y_test).count(0))
print("Number of true 1: ", list(y_test).count(1))

print("Test set R^2: {:.2f}".format(log_L1_gs.score(X_test, y_test)))
print("Test set RMSE: {:.2f}".format(mean_squared_error(y_test, y_pred_gs)))
print("Test set MSE: {:.2f}".format(mean_absolute_error(y_test, y_pred_gs)))

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_gs))
```

```
Number of predicted 0:  704
Number of predicted 1:  278
Number of true 0:  940
Number of true 1:  42
Test set R^2: 0.74
Test set RMSE: 0.26
Test set MSE: 0.26
              precision    recall  f1-score   support

           0       0.99      0.74      0.85       940
           1       0.12      0.79      0.21        42

    accuracy                           0.74       982
   macro avg       0.55      0.76      0.53       982
weighted avg       0.95      0.74      0.82       982
```

Figure 3. Using the 'class_weight' parameter produced better result for the imbalanced data set.

The other supervised method used is the random forest. The random forest method complements and extends the capabilities of the decision tree method by combining various decision trees. A major limitation of the decision tree is that it tends to overfit the training data and does not generalize well to test data sets. The random forests model overcomes this by building multiple decision trees with randomly selected training data points and features. Each of these attempts captures different elements of the data set by asking different sets of if/else questions resulting in a flexible model with improved accuracy with less variance between data sets compared to a single decision tree model. The parameter 'n_estimators' decides the number of trees that goes into the random forests model. The 'random_state' can also be specified so that same randomness can be accessible to make it reproducible. In the case of the random forests model, this parameter is particularly more important because the model's performance intrinsically rely on introducing randomness to it. GridSearchCV was also used here to cross validate with different set of number of estimators.

Like the logistic regression model, the random forest model did well in predicting no-stroke samples but does poorly in predicting stroke samples. Surprisingly though, the best result was observed when the number of estimators was set to 5. Comparing the accuracy metrics between two methods, notably, in logistic regression recall rate or an ability to correctly predict stroke samples out of all true stroke samples is 0.79, but in random forests recall rate is 0.10. However, they both have a low precision rate in predicting 1s. In other words, only the small fractions are truly stroke samples among all predicted stroke samples. This suggests the higher recall rate in the weight corrected logistic regression model is achieved by 'carelessly' labeling more samples as 'stroke' while the random forests is being more 'conservative' in

labeling samples ‘stroke’ and at least correctly label no strokes. Also, given that the precision rate of calling 1s is just a bit higher in the random forest method, I concur the RF model is the lesser evil of the two. But they are equally weak and cannot be trusted with much confidence.

Logistic regression with balanced weight

```
Number of predicted 0:  704
Number of predicted 1:  278
Number of true 0:  940
Number of true 1:  42
Test set R^2: 0.74
Test set RMSE: 0.26
Test set MSE: 0.26
              precision    recall  f1-score   support

           0       0.99      0.74      0.85       940
           1       0.12      0.79      0.21        42

    accuracy                           0.74       982
   macro avg       0.55      0.76      0.53       982
weighted avg       0.95      0.74      0.82       982
```

Random Forest n_estimator = 5

```
Number of predicted 0:  960
Number of predicted 1:  22
Number of predicted 0:  940
Number of predicted 1:  42
Test set R^2: 0.94
Test set RMSE: 0.06
Test set MSE: 0.06
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       940
           1       0.18      0.10      0.12        42

    accuracy                           0.94       982
   macro avg       0.57      0.54      0.55       982
weighted avg       0.93      0.94      0.93       982
```

Figure 4. The RF model seems to be the lesser evil.

Principal Component Analysis

As the next step, Principal Component Analysis is used to examine features as a feature selection step and to transform data as a preprocessing before employing clustering methods. PCA is used to maximize the variance among the data features and thereby comes in handy to make an intuitive visualization. Also, PCA was tried with both scaled and unscaled data using a standard scaler to see which makes a better distinction between the classes. Because PCA assumes linearity in the data and the training data does not have clear linearity, data scaling is a recommended process. However, both versions of the data could not produce any meaningful classification. But, both highlighted ‘Age’ and ‘BMI’ to be important features in explaining variance between groups using the first and second principal components. Because the separation between the points made better sense when unscaled data’s principal components were plotted, I decided to continue with unscaled data. The PCA transformed data was not able to yield a better result for the logistic regression model and the random forest model examined in the previous section.

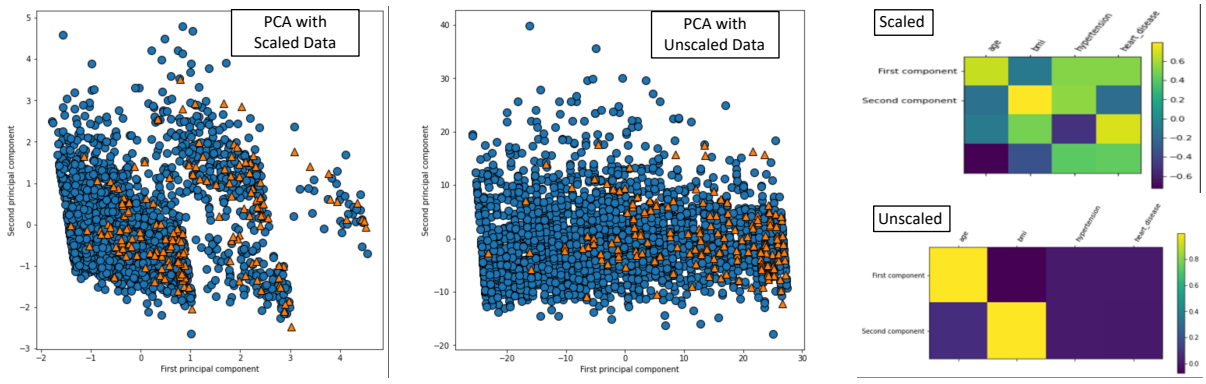

Figure 5. PCA with scaled and unscaled data do not clearly separate between classes

Clustering Methods

Then as instructed, K-means, agglomerate, hierarchical, and DBSCAN methods were used with and without the PCA transformed data. But none of the methods predicted as expected with the stroke data. The k-means clustering method is a mean-based clustering heuristic using a specified k number to partition data points by moving them closer to the k number of centroids that are placed randomly. The

algorithm concurs when convergence boundaries are met after the iterative process of updating the data points and the centroids. And as the name suggests, the agglomerate algorithm starts with an assumption of each data point being a cluster and then accumulates data points to a lesser number of clusters using some linkage methods to bring similar data points to more generalized clusters. A hierarchical cluster is what is produced after the agglomerate algorithm. It summarizes the steps of merging similar data points to the finalized number of clusters. DBSCAN algorithm does not require any apriori knowledge to bring data points to clusters and can be used for data distribution that has complex shapes. Using important parameters 'eps' and 'min_pts', the algorithm finds densely populated areas and separates points from low-density areas.

All of the above clustering methods also did not produce a meaningful separation between the classes. Given that the dataset is heavily unbalanced and only a small percentage of data points belong to the group 'stroke', clustering methods results should also only label a relatively small number of data points. The result was far off from the expectation for both PCA transformed and not transformed data. Because PCA failed to identify features that maximizes the variance among the data, it didn't really help in clustering method neither. Even at a glance, there are about an equal number of points for the two clusters. So, I used the breast cancer data from the sklearn library and was able to produce a much cleaner result.

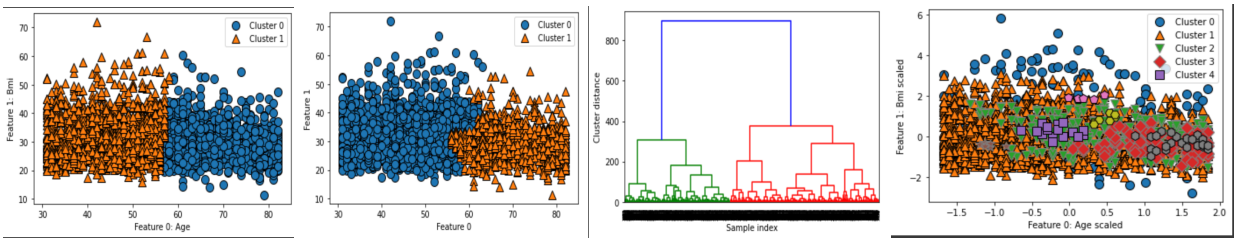

Figure 6. Clustering methods classify too many data points as 'stroke' samples. From left to right: K-means, agglomerate, hierarchical and DBSCAN.

Conclusion

In closing, I learned to appreciate the complexity of the endeavor. Drawing generalizable knowledge from any data or, in this case, developing a workable model to classify is more than applying the methods and using proper data hygiene techniques. Understanding the algorithm and the workflow is obviously important but merely a beginning. It requires an iterative process of forming a hypothesis and relevant questions, then checking and reaffirming the knowns and unknowns like in any scientific endeavor. Above all, the success of analysis is also heavily dependent on starting with relevant and comprehensive data. Understanding the variance in the data and the appropriate domain knowledge to understand why such variance exists or lacks goes long way in assessing the model's performance. No matter how accurately the algorithms can develop a model, it doesn't matter when data is not relevant to questions and hypotheses. Undoubtedly, the experience and understanding garnered from getting my hands dirty with the codes that are combined with the theoretical basis of supervised and unsupervised learning is very exciting for my personal and professional development.