MA 679 Final Project

Models for predicting readmission in 30 days

Yuhan Pu

05.08.2024

Abstract

The main goal of this project is to analyze big data, train and test the data using different models to find the most suitable model to predict whether a patient will be readmitted to the hospital within 30 days. I evaluated the performance of the model through a number of different aspects, and by comparing the performance of different models, I finally selected the model that I was most satisfied with. The dataset I used is NRD_2020_core.csv, which was from Nationwide Readmission Database. It includes data on hospitalizations for almost all patients in 2020. When I consider the performance of many models, I think Decision Tree and XGBoost are the two best models.

Introduction

Because of the need to analyze big data, and I am not allowed to download it, I chose to analyze it on SCC, and the core idea was to clean the data and then filter out the factors that I think could influence whether a patient is readmitted in 30 days or not. After that, I will train different models by comparing their various characteristics to get the most satisfying one.

Procedure and Methodology

1. Import data and data cleaning:

   Import the first 100000 lines from the dataset as my initial dataset, and assign the column names to the dataset. Delete those columns which have a large amount of NA values. Then check the datatype to make sure they can be used in further analysis. Finally, based on NRD_Visitlink, select those who have at least two rows of data as my main focus(which means they have the history of readmission).

2. Create variables that I want to predict:

   I created a continuous variable ‘days_between’ to represent how many days there are between two admissions for each patient, and a categorical variable ‘days_between_new’ to indicate whether their readmissions are less than or equal to 30 days or more than 30 days.

3. Select predictors:

   Do some EDA and also feature selection to get the 12 most related variables as my predictors.

4. Try training the model:

   I first tried linear regression for predicting days_between and logistic regression for predicting days_between_new. The performance of the model is pretty bad, then I checked the distribution of the predicted value, I found that it is an imbalance(amount of 0 and 1 so different).

5. Resample the dataset:

   I used the random sampler to resample the data and append the new data to the original dataset to make the amount of 0 and 1 equal. This new dataset will be used in the further model training part.

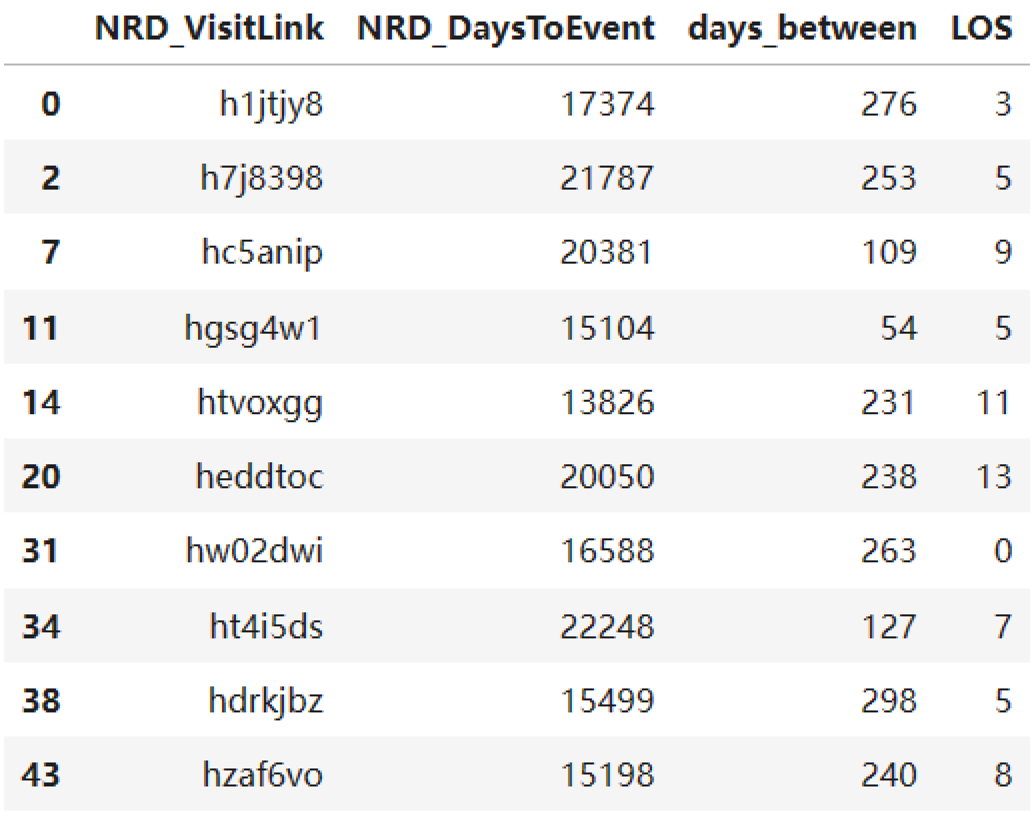
6. Retrain the model on the new dataset:

   This time, I trained and tested different models: logistic regression, Neural Network(MLP), decision tree, XGBoost, and SVM. By evaluating the performance of the model in many aspects, I concluded that decision tree and XGBoost are two models that perform better.

Specific Results In Each Steps

1. Data Cleaning and Variables Creating:

   I deleted columns which have a large amount of NA values, and calculating days_between by: first, use the max number of Days_to_Event to minus the min number of Days_to_Event for each patient, and then use this result to minus the LOS(length of stay) in the same rows with the min number of Days_to_Event. I screenshotted these columns as below:

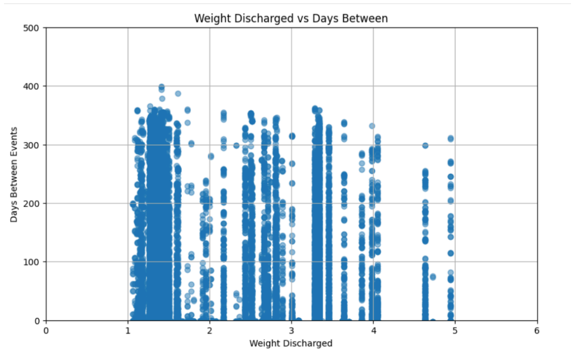


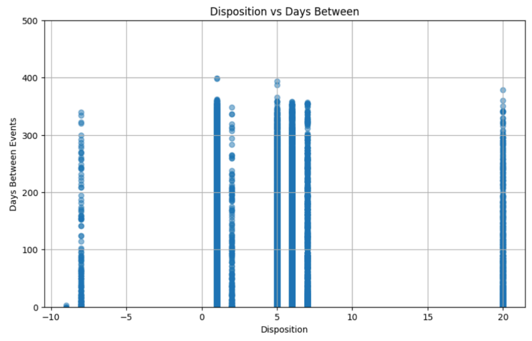
	NRD_VisitLink	NRD_DaysToEvent	days_between	LOS
0	h1jtjy8	17374	276	3
2	h7j8398	21787	253	5
7	hc5anip	20381	109	9
11	hgsg4w1	15104	54	5
14	htvoxgg	13826	231	11
20	heddtoc	20050	238	13
31	hw02dwi	16588	263	0
34	ht4i5ds	22248	127	7
38	hdrkjbz	15499	298	5
43	hzaf6vo	15198	240	8

2. Predictors Selecting:

   I did some EDA with some variables that I think might have a relationship with days_between and I listed some of them:

Weight Discharged vs days_between Disposition vs days_between





By only doing the EDA, it is so hard to determine the proper predictors since there are too many columns in the dataset, so I decided to use feature selection by importance to determine the potential predictors.

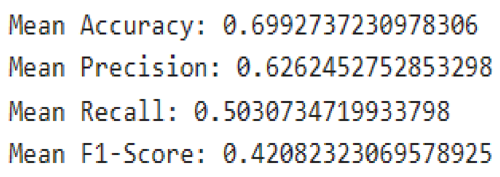
After I have done this, I choose the 12 most relevant predictors as below:

ZIPINC_QRTL, DISCWT, DRG,PRDAY1, AGE,DRG_NoPOA, TOTCHG, I10_NDX, KEY_NRD, HOSP_NRD, DMONTH, DISPUNIFORM

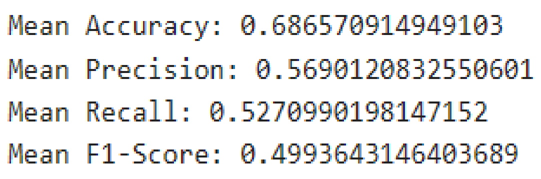
3. Model training (before resampling)

I fit the logistic regression model and MLP Neural Network both with K-fold Cross Validation(to prevent overfitting), but their performance are both bad：

Logistic Regression: MLP:

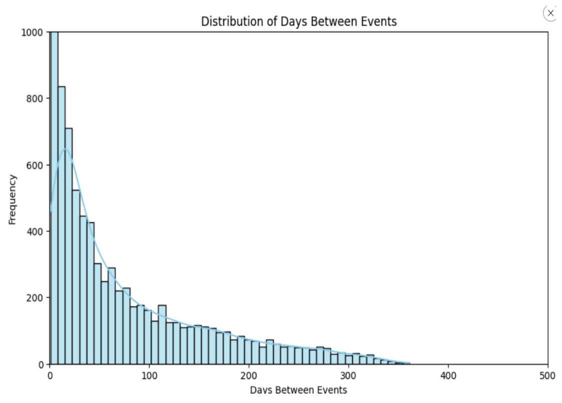


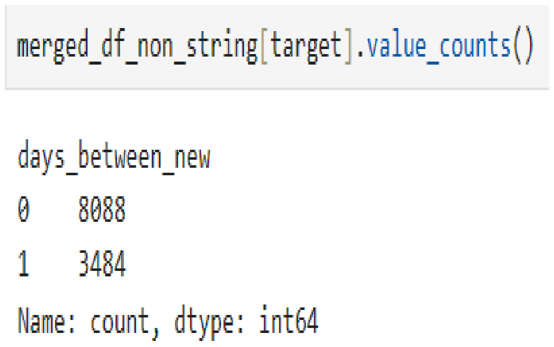
```
Mean Accuracy: 0.6992737230978306
Mean Precision: 0.6262452752853298
Mean Recall: 0.5030734719933798
Mean F1-Score: 0.42082323069578925
```



```
Mean Accuracy: 0.686570914949103
Mean Precision: 0.5690120832550601
Mean Recall: 0.5270990198147152
Mean F1-Score: 0.4993643146403689
```

You can notice that both of their accuracies are around 69%, this seems not bad. HOWEVER, the distribution of days_between shows us the problem:





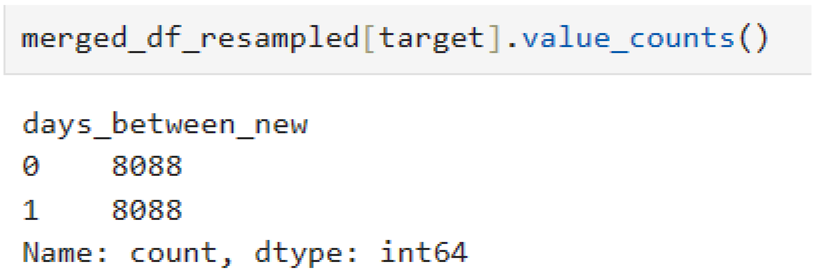
```
merged_df_non_string[target].value_counts()
```

```
days_between_new
0    8088
1    3484
Name: count, dtype: int64
```

I notice that those whose readmission is more than 30 days are much higher than those within 30 days, and the proportion is exactly around 69%. This means that these two models are always guessing the same number(always guessing 0 can still reach the accuracy of 0.69), they are not actually trained well! Because of this, I plan to make the dataset more balance by randomly resampling

4. Data resampling

I used the random resampler to make the dataset more balance (50% for 0 and 1 in predicted value)



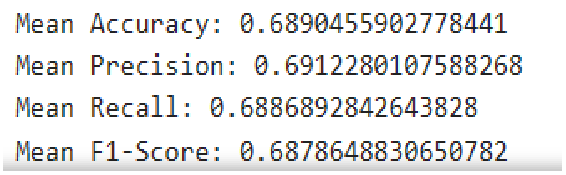
```
merged_df_resampled[target].value_counts()
```

```
days_between_new
0    8088
1    8088
Name: count, dtype: int64
```

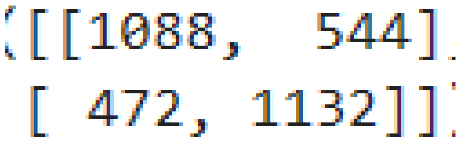
5. Model Retraining

I tried four different models after resampling the data. (all of them were trained with k-fold cross validation)

Neural Network (MLP): Key Characteristics and Confusion Matrix:



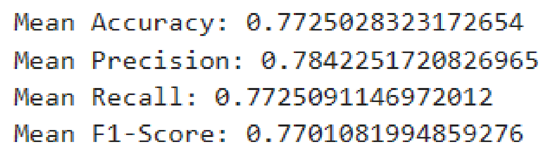
```
Mean Accuracy: 0.6890455902778441
Mean Precision: 0.6912280107588268
Mean Recall: 0.6886892842643828
Mean F1-Score: 0.6878648830650782
```



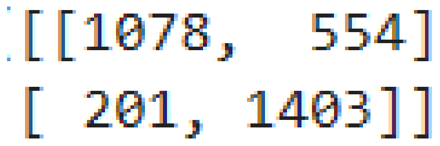
```
[[1088,  544]
 [ 472, 1132]]
```

MLP Neural Network performs more reasonably now (if it is still keeping guessing the same number, the accuracy should be around 0.50).

Decision Tree:



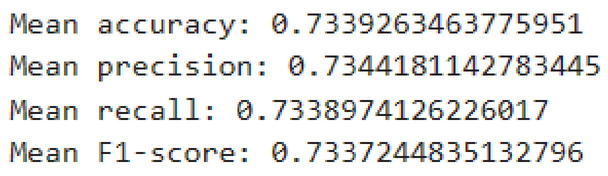
```
Mean Accuracy: 0.7725028323172654
Mean Precision: 0.7842251720826965
Mean Recall: 0.7725091146972012
Mean F1-Score: 0.7701081994859276
```



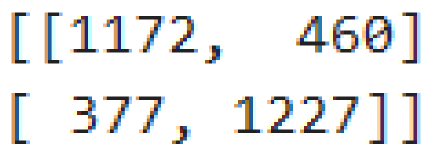
```
[[1078,  554]
 [ 201, 1403]]
```

The Decision Tree model performs better than Neural Network in almost every aspect.

XGBoost:



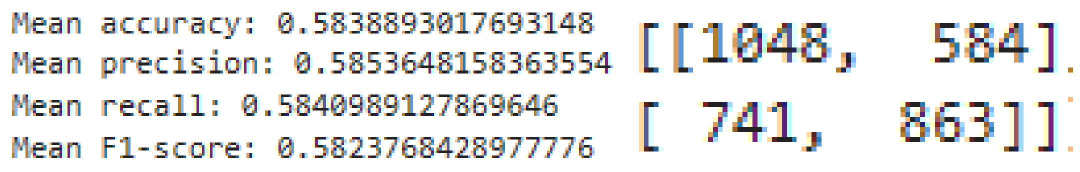
```
Mean accuracy: 0.7339263463775951
Mean precision: 0.7344181142783445
Mean recall: 0.7338974126226017
Mean F1-score: 0.7337244835132796
```



```
[[1172,  460]
 [ 377, 1227]]
```

XGBoost seems to perform a little bit worse than the Decision Tree.

SVM:



```
Mean accuracy: 0.5838893017693148
Mean precision: 0.5853648158363554
Mean recall: 0.5840989127869646
Mean F1-score: 0.5823768428977776
```

```
[[1048,  584],
 [ 741,  863]]
```

SVM Model performs the worst among these models.

Based on this, I want to select the Decision Tree and XGBoost to specifically explain their performance.

<u>Accuracy</u>: Both model’s accuracy are higher than 73%, which means they correctly predict the outcome more than 73% of the time, suggesting that the model is fairly good at handling the given dataset.

<u>Precision</u>: The models’ precisions indicates the accuracy of positive predictions. Their precisions are around 78% and 73%, suggesting the time that they correctly predict the positive class.

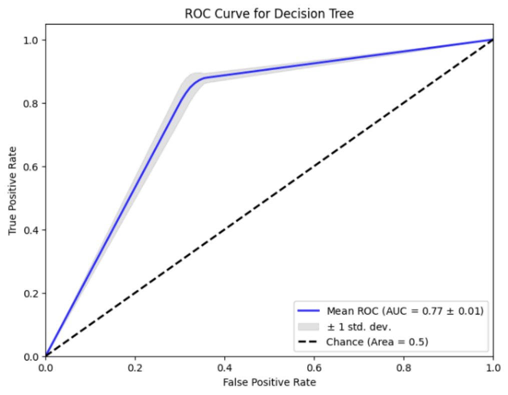
<u>Recall</u>: The model's recall, which measures their ability to find all relevant instances in a dataset, are about 77% and 73%. This shows that they successfully identify 77% and 73% of all actual positives.

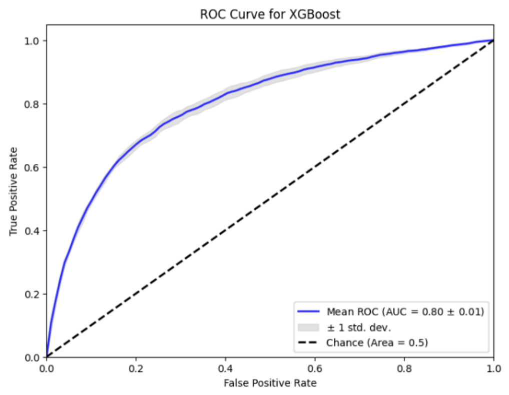
<u>F1-score</u>: It is a balance between precision and recall. Their F1-scores are around 0.77 and 0.73. This is fairly high and suggests that the model is reasonably balanced between precision and recall.

The precision, recall, and F1-score are all higher than 73%, indicating a balanced performance by these models between identifying positive classes and minimizing false positives.Besides this These models have a higher number of false positives compared to false negatives(which can be found in confusion matrices). This suggests that these two models are more prone to incorrectly predicting the positive class

Then I want to show the ROC curve of these two models.

Decision Tree: XGBoost:





Based on these two ROC curve, we could get more conclusions for these two models:

AUC: Area under the curve, both of them indicate good abilities to distinguish positive and negative classes, XGBoost seems to be a little bit better on classification performance. This might be due to XGBoost's ability to model complex nonlinear relationships and interactions between features more effectively than a single Decision Tree.

Curve Shape: The ROC curve for XGBoost shows a smoother and more gradual increase towards the upper left corner, which indicates a better balance between sensitivity and specificity at varying thresholds compared to the Decision Tree.

Conclusion

While both models demonstrate good classification capabilities, XGBoost's higher AUC and smoother ROC curve indicate it is likely a better choice for tasks requiring high discrimination ability between classes, particularly in complex datasets. However, the Decision Tree model has higher accuracy, f1-score, recall than XGBoost, which also indicates a better performance. Under this condition, it is hard to determine which model should be better.

From my analysis: Metrics such as accuracy, precision, recall, and f-1 score give a snapshot at a specific threshold, which usually defaults to 0.5 for binary classification. They indicate how well the model performs at this cutoff while the ROC curve reflects the model’s performance across all classification thresholds. Given the complexity of the decision, it's crucial to align model selection with business objectives and operational constraints. If threshold flexibility and error-cost considerations are pivotal, XGBoost could still be the right choice despite currently lower conventional metric scores at the default threshold. Conversely, if immediate performance on these metrics is crucial and the operational environment values simplicity, a well-tuned Decision Tree could be preferable.[1]

References

[1] The last part explaining the different situation of model selection was combined from my idea and a more academic tone by GPT

Acknowledgement

I would like to express my sincere gratitude to Dean and Anand for granting me access to the dataset, which has been invaluable for the success of this project. I am also profoundly thankful to Professor Masanao for providing me with this incredible opportunity to undertake this project and present my findings to our esteemed clients. Their support and guidance have been instrumental in the completion of this work.

Contribution

Yuhan Pu (myself) did everything.