Ryan Herrin, Luke Stodgel

DS7333

01/28/2023

**Predicting Hospital Re-admittance using Logistic Regression**

**Introduction**

The purpose of this logistic regression task is to predict if a patient would be readmitted into the hospital or not.

**Methods**

The data set is composed of different variables that have to do with a patient and their medical history. A few columns are: age, weight, race, gender, in/outpatient visits, diabetic drugs prescribed, etc. The data set has 50 features and 101,767 rows. The data contains a mix of integer features as well as categorical features. The target variable, “readmitted” contains three possible values: “No”, “<30”, “>30”. In order to use categorical data in our logistic regression models, we decided to use in-place encoding methods from the OrdinalEncoder and LabelEncoder classes. For our categorical input features we used OrdinalEncoder, and for our target variable we used LabelEncoder. The only reason we chose to encode our input features in-place as opposed to using one hot encoding was to keep from adding more columns to our data set. We do not know if having fewer columns is better or worse for model performance. Lastly, we scaled our continuous data using the StandardScaler package and we used the scaled X data combined with the encoded categorical data to build our models. We assumed that outliers in the data would be dealt with by scaling the continuous data.

**Feature Inclusion and Exclusion Reasoning**

We obtained some domain knowledge and advice from an ICU nurse to influence choices of whether to include or remove features. Talking with our domain expert we decided to keep the race feature included in the dataset. We know there are possible social implications that can come from keeping this feature, but medically speaking there are certain races that are more susceptible to certain diseases including diabetes. We removed Encounter\_id and patient\_nbr because those features are used to identify a patient and are not relevant to the population. Examide and citoglipton were also removed because all values for these features were the same and thus made no impact on our log reg model.

**Missing Values and Imputation**

The data set started with missing values in 7 columns: *race*(2273 missing), *weight*(98569 missing), *payer\_code*(40256 missing), *medical\_specialty*(49949 missing), *diag\_1*(21 missing), *diag\_2*(358 missing), *diag\_3*(1423 missing). We tried imputing predicted data for the “payer\_code” and

“medical\_specialty” columns using KNNImputer(), but that ended up making our models perform worse. In the end, this is how we handled the missing data for each of the columns with missing data:

Race - Since there were a relatively small amount of missing values, we imputed this column using its mode.

Weight - Since this column contained 97% missing values, we thought that if we tried to impute the data here using predictions, the mode, median or mean that we would cause more harm than good. So we decided to remove it from the data set completely.

Payer\_code - This column is in a similar situation to the weight column. It had so many missing values that imputing based on a mean, median, or mode seemed like it would do more harm than good. We also tried imputing data using KNNImputer() and that resulted in worse model performance. Therefore, we removed this column from the dataset completely.

Medical\_specialty - Same reason as payer\_code.

Diag\_1, Diag\_2, Diag\_3 - The diag\_1, diag\_2, and diag\_3 columns gave us issues with imputing at first, but we made it work. The problem was that these columns contained a mix of string and integer values, thus imputing any sort of numeric metric or making KNN predictions was difficult. We ended up replacing the NAs with “unknown” and then in-place encoding the values in the columns. We realized a little later that we could have in-place encoded the values first, and then imputed using a numeric metric or predictions, but we ran out of time by the time we thought to do that. Regardless, the amount of missing data in these columns was so small that imputing predicted data vs imputing the value “unknown” may not have made any difference on the model’s performance anyway.

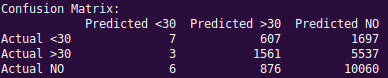
**GridsearchCV**

For our gridsearchCV algorithm, we used the parameters *solver=’liblinear’*, *cv=5* and *scoring=’roc\_auc\_ovr’*. We tested alphas from 0.001 to 0.05, incrementing by 0.005 each time. For each alpha we performed a randomly seeded 5-fold cross-validation. The best alpha we got from this was 0.41.

**Results**

We created two logistic regression models: one with features selected by lasso and gridsearchCV and another using the full feature set. The model with the full feature set performed better and resulted in a model with an accuracy of 0.57 and an F1 score of 0.491. The lasso model had an Accuracy of 0.564 and an F1 score of 0.473.

Here is a confusion matrix of the full feature model:



Confusion Matrix for Full Feature Model

As you can see from the confusion matrix, our best model guessed correctly about 50% of the time. The model looked to have had a hard time accurately predicting <30 and >30 compared to predicting “no”.

Overall, with an F1 score of 0.49, we can say that our model is not good. The model is making a significant number of false positives and false negatives.

**Conclusion**

In conclusion, our full-feature logistic regression model outperformed our lasso log reg model by a small margin. We don’t consider either models to be good predictors of patient readmittance.

A few ways we could improve our final model would be using patterns or grouping feature data to help create a stronger model by reducing the number of classifications and thus reducing complexity. We also could have used more time trying to find the optimal model parameters.

Our model picked number\_inpatient as the most important feature.

Top five most important features in the data set:

1. number\_inpatient: 0.345 coefficient
2. discharge\_dispotion\_id: 0.117 coefficient
3. diabetedMed: 0.102 coefficient
4. Number\_diagnoses: 0.077 coefficient
5. age: 0.049 coefficient

Appendix A

Link to main script: <https://github.com/Abillelatus/QTW-Case-Studies/blob/main/CaseStudy_2/MSDS-7333-CaseStudy-2_v02.py>

Link to full CaseStudy\_1 full contents:

[https://github.com/Abillelatus/QTW-Case-Studies/blob/main/CaseStudy\_2/](https://github.com/Abillelatus/QTW-Case-Studies/blob/main/CaseStudy_2/MSDS-7333-CaseStudy-2_v02.py)