Relationship between Ischemic Stroke 30-Day Mortality and 30-Day Readmission Rates, Hospital Quality Ratings, and Location.

Rachel Kersey [*1] **Helen Sparling** [*1 2]

Abstract

This paper examines the relationship between Ischemic Stroke 30-Day Mortality and 30-Day Readmission Rates, Hospital Quality Ratings, and Location. (More will be added after data description) (Rachel Kersey)

1. Data Description

The dataset name is, "Ischemic Stroke 30-Day Mortality and 30-Day Readmission Rates and Quality Ratings for CA Hospitals." We got it from Data.gov, which is a reputable source to pull data from. The hyperlink to the dataset is as follows. The purpose of this dataset is to show risk-adjusted 30 day mortality and readmission rates, as well as quality ratings and deaths/readmissions, for Ischemic Strokes treated in hospitals in California. This dataset is important as it shows how hospital ratings are related to stroke readmissions and deaths: and a possible correlation could demonstrate the need for hospitals to increase their ratings in order to lower readmission or death rates. The URL is available here:

Stroke Dataset Link

We investigated a rich dataset with 9 variables: 5 numeric, and 4 categorical. We created a subset to include only the five most populated counties in California. The variables are listed as follows:

- Year: quantitative discrete, year data was recorded).
- County: categorical, county data was recorded in.
- Hospital: categorical, hospital data was recorded in.
- OSHPDID: categorical, hospital ID, not important for research question).
- Risk Adjusted Rate: quantitative continuous (float), risk rate adjusted for age, health severity, readmissions and deaths).
- Number of Deaths/Readmissions: quantitative continuous (float), number of deaths/readmissions for hospital that year.
- Number of Cases: quantitative continuous (float), number of hospital cases
- Hospital Ratings: categorical, patient hospital rating.
- Location: quantitative (float), hospital location coordinates. (Rachel Kersey)

Data Dimensions The size and the shape of the data is 1,026 observations with 9 columns. (Helen Sparling)

Data Quality Notes: The data had missing values. They were replaced with np.Nan so no additional cleaning was needed. Data cleaning included: coercing columns with periods to numeric as a precaution, removing unnecessary wording, splitting the year column to change the format from "2011-2012" to "2011", and changing the year column to numeric. These changes will make data visualization simpler. There were no unnecessary dollar signs, commas, or other characters. The original data set was quite large, so the 5 most populated counties were subsetted to make the dataset more manageable for analysis. The location column was removed because it was not necessary to answer the research question. (Helen Sparling)

There are some outliers in the dataset. Boxplots were made for the three numeric variables: risk adjusted rate, number of cases, and number of deaths/readmissions. All of these variables were right skewed. Future steps will involve transformations to fix the data and curb the number of outliers. (Rachel Kersey)

1.1. Context and Relevance

Our project/goal is to investigate the relationships between year and county and how they relate to risk adjusted rate, deaths, readmissions, and case count. This investigation is important from a public health standpoint, and provides helpful insight into hospital functioning in the California area. Understanding which hospitals have higher deaths and cases, as well as how hospital ratings affect cases, deaths, and re admissions, would help policy makers identify hospitals in need of improvement, and provide recommendations of how to improve patient care. (Helen and Rachel)

2. Pre-Analysis

The following sections outline our pre-analysis plan for our dataset.

2.1. Refined Research Question

We first reviewed some associations in our data utilizing barplots, as well as a correlation matrix. We did not find high correlation between any of the numeric variables.

After reviewing some associations in our data, we found indicators that there is a relationship between hospital ratings and risk adjusted rate, number of cases, and number of deaths/readmissions. Thus, this led us to refine our research question: Predict hospital rating ("as expected", "worse", "better") from hospital location, year, risk adjusted rate, deaths, readmissions, and case count. This question allows us to explore with Machine Learning methods we have used in class. (Rachel)

2.2. Data Splitting Plan

For our data splitting, we plan to utilize an 80/20 train test split through a def(maxmin) function, creating target and feature variables, and using the "from sklearn.model selection import train test split." (Rachel)

2.3. Feature Selection

Before running any machine learning, certain variables were one-hot encoded. Variables that were one-hot encoded included County and measure (mortality or readmission). We did not one-hot encode by hospital due to the large number of different hospitals. County was a more reasonable location-based variable to one-hot encode. Numerical variables used will include risk adjusted rate, number of deaths/readmission, and number of cases.

Interactions between variables will be analyzed. We will predict hospital ratings based on multiple variables, both categorical and numeric. These predictor variables are hospital location (County), year, risk adjusted rate, death, re admissions, and case count. We will do the same for objective 2, using risk adjusted rate as the predicted variable instead of hospital ratings. (Helen)

2.4. Model Selection Plan

We will use a classification model for addressing objective 1 because the predicted variable is categorical. We will use a linear regression model for objective 2 because the predicted variable is numeric.

Specifically for objective 1, a logistic regression model will be used because the measure variable is a categorical variable. We will also use a K-nearest neighbor for objective 1 because these models can handle categorical targets, categorical predictors, so long as they are one-hot encoded (which was done previously).

Tree based models will be used for objective 2. This will be helpful in making a model with mixed data types (numeric and categorical) without first hot encoding. (Helen)

2.5. Evaluation Strategy

For our evaluation strategy, we plan to use accuracy and precision to evaluate how well our model does. This entails using "from sklearn import tree", making a decision classifier and fitting the classifier, making predictions on the test set, then completing a confusion matrix with an accuracy formula. (Rachel)

3. Analysis

3.1. Model implementation

Decision Tree Classifier: It is a classification model that measures the probability of a hospital being rated "as expected", "worse", or "better" based on independent variables we gave it. To set this up, we used a def(maxmin) function and set our x and y. Our x was our features, which we chose to be 'Risk Adjusted Rate', '# of Deaths/Readmissions', "# of Cases", "Year", "County", "Hospital", and "Measure". Our y, or target variable, was Hospital ratings. Next, we selected the numeric x variables for normalization, and then recombined them with the categorical x variables. We then split the data on an 80/20 train test split. Next, we one hot encoded all of the categorical variables (County, Measure, and Hospital). Then, we imported the decision tree classifier, created the classifier object, fit the classifier, and made predictions on the test set. We finally computed accuracy to evaluate how well our model did. (Rachel)

Random forest: This was a more complicated model we chose, and we used this to explore our secondary question, so we made a model where our target variable was 'Risk Adjusted Rate' and we used a multitude of features in this model. Here, our model built a lot of decision trees and the random forest averages our results. This is more accurate than just one decision tree. To build this model, we first split the data into training rows and testing rows. We then removed the rows with the target to finalize our df_train and df_test. Our next step bootstrapping, where we set T=1000, and we split the data again into an X_train and y_train, X_test and y_test, where the y's used arcsinh, and the x code dropped the risk adjusted rate. We then re-one hot encoded our categorical columns because we were having issues with our code. We then used a for loop to generate a bootstrapping sample, computed rsq, and made/saved predictions. Our next step was building an ensemble predictor, and we printed our rsq. We did the same thing but predicted

hospital rating, it just required a lot of online work and AI to help refine my code because it needed to be changed from arcsinh to a classifier random forest.

We also used random forests for classification on hospital ratings. This did not include bootstrapping. To build it, we defined our target variable y and our features x (all variables except year and OSHPDID), one hot encoded the categorical columns, and did an 80/20 train test split on the data. Then, we imported RandomForestClassifier, fit the classifier, and printed the confusion matrix and accuracy. (Rachel)

Multiple Linear Regression: This was used as a comparison model to the ensemble random forest, and it predicts risk adjusted rate utilizing the other numeric variables in our dataset as predictors. We first dropped NA's from our numerical variables as it was causing issues in the code. Next, we used def mlr(x,y) to build our multiple linear regression function. We set our intercept equal to one, and set our X to be our intercept + predictors, and our y to be Risk Adjusted Rate. Finally, we printed our MLR coefficients and r-squared. (Rachel)

K Nearest Neighbors: This model was used to determine what class (Hospital rating: "as expected", "worse", or "better") a new case will belong to based on the predictor variables. We first created the target and predictor variables. We used get_dummies to address non-numeric variables. This produced boolean values, so we adjusted the dummy variables to be 0 and 1 rather than "True" and "False", allowing us to then create the KNN model. Then, we split the data 80/20, and scaled the X_test and X_train so that values would be comparable. Next, we needed to determine the ideal k value using SEE across possible values of 1-50 for k. The ideal K was 1 to minimize classification error. This is a bit unusual, but it is possible that the ideal k can be 1 when the data is clean and lacks excessive noise. However, the model may be prone to overfitting. (Helen)

3.2. Architecture/structure

Number of trees in random forest: We set T=1000 for the ensemble random forest, which predicted risk adjusted rate.
Optimal k for KNN: k=1 (Helen and Rachel)

3.3. Pre-processing Steps

Train/Validation/Test split: We split the data into an 80/20 train test split, which allows us to train a good chunk of the data, while the other piece is used to evaluate performance. We used this method in our decision tree classifier, our random forest for hospital ratings, and our K nearest neighbors model. We followed the code from the random forest classifier class to split our data for random forest. (Rachel)

Scaling: We scaled the data for the KNN model so that their values would be comparable. Once this was completed, we were able to determine the K value that minimizes test classification error, and make the KNN model. We also used maxmin functions for our decision tree classifier. (Helen)

One hot encoding: We have one hot encoded measure (readmission/mortality), hospital rating, and county to use in our models. (Rachel)

Feature Selection: The features we used to build our model were 'Risk Adjusted Rate', '# of Deaths/Readmissions', "# of Cases", "Year", "County", "Hospital", and "Measure" if we were predicting hospital ratings. We used these measures as we saw some interesting relationships in our preliminary graphs and wanted to thus incorporate these variables into the model. For our models where we predicted risk adjusted rate, in the random forest ensemble (for risk adjusted rate), we used all of our variables in our dataset because we were curious how keeping them all in would affect ensemble rsq. The features we used in our MLR model included all of our numeric variables, except for risk adjusted rate, which was our target. (Rachel)

4. Evaluation Benchmark

4.1. Metrics Used

Accuracy: The accuracy for our decision tree classifier was 0.92.
(Rachel) The accuracy for our KNN model was 0.93. (Helen) The accuracy for our random forest (hospital ratings) model was 0.926. (Rachel)

Confusion Matrix

Table 1. KNN Confusion Matrix

Actual / Predicted	Class 0	Class 1	Class 2	Class 3
Class 0	189	2	0	0
Class 1	6	1	0	0
Class 2	1	0	0	0
Class 3	5	0	0	2

(Helen)

Table 2. MLR Confusion Matrix

Actual / Predicted	Class 0	Class 1	Class 2
Class 0	182	4	1
Class 1	4	6	0
Class 2	7	0	1

(Rachel) **R-squared**: Our ensemble rsq for the ensemble

random forest was 0.76, and our rsq for MLR was 0.0056. (Helen and Rachel) graphicx

4.2. Model comparison

The KNN model had the best accuracy. As aforementioned, however, the k value with the least test classification error was 1, which makes this model more prone to overfitting. For predicting risk adjusted rate, random forest has a far better r-squared value compared to the multiple linear regression model (0.76 and 0.0056, respectively). (Helen)

4.3. Initial Results and Comparison

Interpretation of which model performed best:For models predicting hospital ratings, k nearest neighbors performed the best in regards to accuracy, but the decision tree classifier accuracy was very close to k nearest neighbors. In addition, the decision tree classifier is less prone to overfitting vs. the k nearest neighbors because our optimal k is 1 for the KNN. Further, from our random forest model, we had an accuracy of 0.926, which was higher than the decision tree classifier, and therefore our best model.

For models predicting risk adjusted rate, MLR performed poorly with a low rsq. The random forest performed better with a significantly higher rsq.

5. Initial Results Analysis

The random forest and decision tree classifier models performed better than the KNN and MLR models. Regarding the KNN model, this one was not as ideal, though it has a high accuracy (0.93) due to the fact that the ideal k for minimizing test classification error was k=1. This model, therefore, is prone to overfitting. The model presenting these characteristics because the cleaned data set was large, clean, and lacked excessive noise. MLR performed poorly because a linear model simply failed to capture the variance in the data. The trends were more complex. The random forest model performed better because this model is non-linear and random forests account for more complex patterns. (Helen)

Random forest was likely the best when it came to predicting hospital ratings as it takes multiple trees and has them "vote" on the outcome, so there are many more trees contributing to the results. (Rachel)

6. Issues and Limitations

We had some limitations when it came to ensemble learning for random forest as our code kept throwing errors. However, we just needed to make small adjustments to how we one-hot encoded and the order of our code, and it eventually worked.

7. Figures

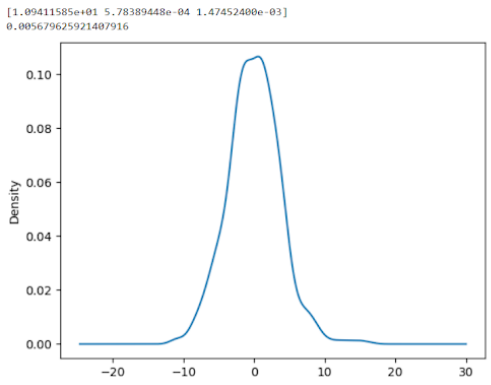

Figure 1. MLR model results

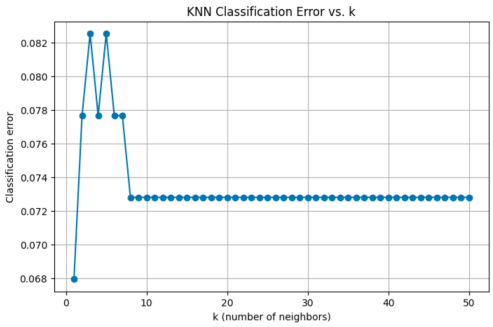

Figure 2. KNN Classification Error vs. k

References

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.