Lorenzo Scaturchio

Michael Ngo

Bharadwaj Satyanarayana

9 October 2019

Math 032: Probability and Statistics

Parkinson's Analysis

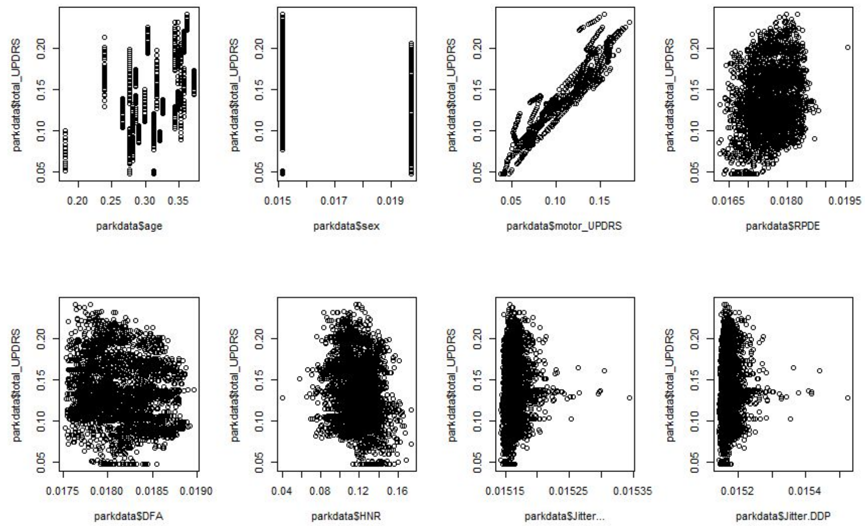
Abstract

We seek to define if there is a relationship between auditory function and the severity of Parkinson's Disease using linear regression models. In the event that a relationship is discovered, the auditory function of a given person with Parkinson's Disease can be used to dictate how severe their affliction with Parkinson's Disease is. That is, using the predictive variables, we hope to be able to create a model that can predict a patient's total UPDRS score using their auditory functions. We hope that by finding a relationship between a patient's auditory functions and the severity of their Parkinson's, we will be able to mathematically predict the rate at which a patient's Parkinson's disease will deteriorate.

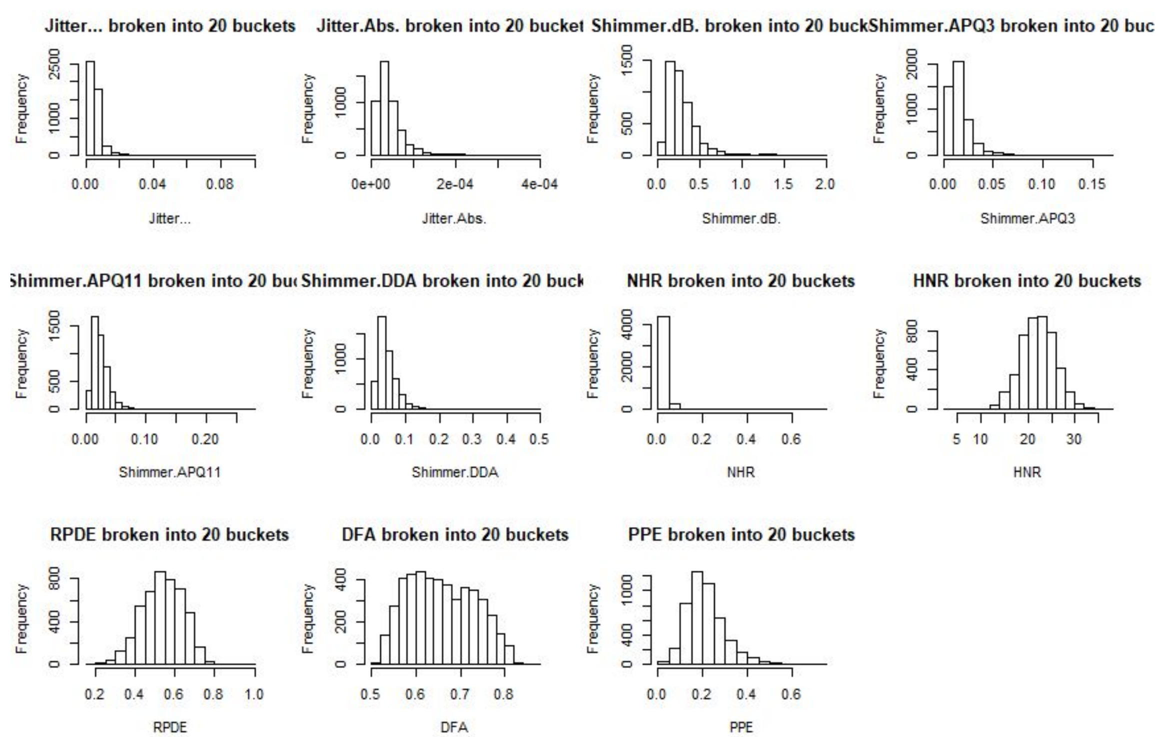
Exploring Data

When analyzing the dataset that we were given, there were approximately 4700 rows that consist of each patient, and there are approximately 21 columns in which there are several auditory factors that have been measured with respect to each person's Parkinson's severity.

When exploring the data, we typically used the total UPDRS score as the variable we are trying to predict. We did this so that we could minimize our work by restricting prediction to a single variable, reducing run and test times on machines for each of the many models we create. We chose total because it seemed to be more encompassing of the severity of a patient's Parkinson's, as UPDRS is a measure of the severity of the disease.



One of the first steps that we took in exploring our data was to take the plots of some of the predictors that we researched to potentially be good for determining the total UPDRS score so we plotted each of the predictors on the x axis of each plot, and total_UPDRS on the y axis. Our results were



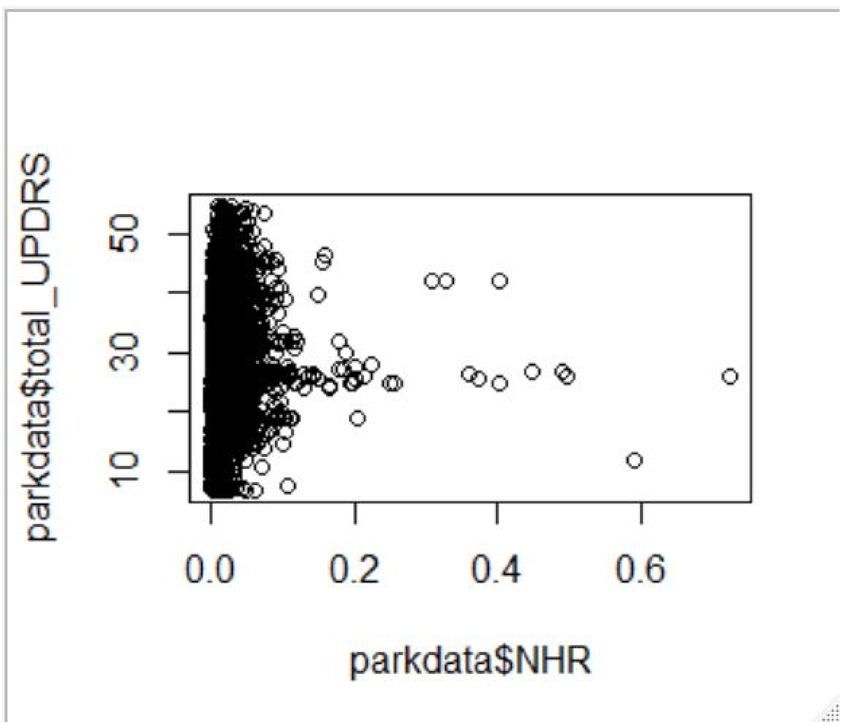
drastically diverse looking at the first few graphs, however, when looking at the last three predictors that we chose, you can begin to see a common trend among them.

The next step we took in exploring our data was to create histograms of each variable, minus the identifying variables and the variables we wish to predict, in order to observe how frequent certain values appeared across the board. We used 20 buckets in each case in order to standardize our observations between the data. From this, we found that almost all of the predictive variables had heavy skews towards certain ranges of values. For example, the majority of NHR was extremely small, and that seemed abnormal.

```
> cor(parkdata$total_UPDRS, parkdata$NHR)
[1] 0.07486251
```

```
> summary(parkdata$NHR)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.000286 0.010352 0.017526 0.023771 0.029251 0.725370
```

In the frame of further exploring why there is such a heavy skew towards lower values in NHR, we took a summary of that specific variable and found that the histogram is true to the data. NHR is a ratio of noise to harmonics of the voice, which may suggest a higher Parkinson's score in the case that the noise is much larger than the harmonics. In taking the correlation between the two variables, we found



that they were not particularly correlated to each other. In terms of plotting NHR against UPDRS, we did not see any obvious pattern at first glance.

Our next idea was to iteratively look for correlations between our prediction variable and the predictor variables. In doing so, we found that RPDE had the highest absolute correlation with a correlation value of 0.239 and a Jitter.DDP had a minimum absolute correlation with a value of 0.053. No particularly correlative variables here.

Following this, we decided to investigate the correlation amongst variables themselves. We created a matrix and populated it by finding the correlation between the variables in the row and column of each individual cell. From there, we took the maximum correlation values and recorded them for future use and reference in creating models. In order, the five largest correlated pairs are: 16,13 | 10,8 | 11,12 | 11,14 | 10,6

We did notice that a large portion of the data contained very small values, and when compared to other points of data, these portions would be vastly overshadowed. In order to combat this, we would need to normalize the data so that each set has an equal weight on the variable we wish to predict: total_UPDRS. We first decide to explore models without a normalized dataset in order to ascertain whether or not it is even necessary in the first place.

Modeling & Methods

Model 1:

When creating and testing our models, we aim to use a simple 80/20 train/test split of the data so there is a minimum time wasted when exploring models. Later on, when we felt more confident in our models, we cross-validated using different ratios, using ten-fold validation, and LOOCV. These are expressed as they are used in the process.

The very first thing that we did when beginning to model the data was to foolishly create a linear model containing all variables excluding the identifying first four variables, and the two target prediction variables, such that an lm was created excluding subject number, subject age, subject gender, time is taken, and both UPDRS scores. From there, we trained the model on a randomly (set seed of 42) selected 80% of the given park data, and then tested it on the remaining 20%, taking the mean squared error, and using this value as a measure of the strength of the model itself is predicting a total UPDRS score. This foolish model had an MSE (mean squared error) of 49.11508, which is much larger than our “goal” score of a zero MSE. We acknowledge that a zero MSE is unrealistic and often problematic, but we wish to reduce the MSE of our model as much as possible within reason. For

```
Call:
lm(formula = ytrain ~ . - subject. - age - sex - test_time -
    motor_UPDRS - total_UPDRS, data = xtrain)

Residuals:
           Min             1Q         Median             3Q            Max
-20.705482865642  -5.351866421349  -0.627041127693   5.593916060961  19.200511196968

Coefficients:
                   Estimate        Std. Error   t value   Pr(>|t|)
(Intercept)    4.47236365365e+01  2.81549602650e+00  15.88482 < 2.22e-16 ***
Jitter...      3.52561703466e+02  2.16049342148e+02   1.63186 0.10279930
Jitter.Abs.   -3.19422319906e+04  9.51418550786e+03  -3.35733 0.00079538 ***
Jitter.RAP    -2.23086080886e+04  4.44787436763e+04  -0.50156 0.61601092
Jitter.PPQ5   -4.74031468867e+02  2.51918266620e+02  -1.88169 0.05996118 .
Jitter.DDP     7.50656985394e+03  1.48269311005e+04   0.50628 0.61269233
Shimmer        1.94191819227e+00  8.78620572155e+01   0.02210 0.98236793
Shimmer.dB.    2.10819508834e+00  5.02207296431e+00   0.41979 0.67466762
Shimmer.APQ3  -6.24614534755e+04  4.51491509006e+04  -1.38345 0.16661590
Shimmer.APQ5   2.67053682504e+01  7.62194321547e+01   0.35037 0.72607847
Shimmer.APQ11  1.29798625162e+02  3.32139031469e+01   3.90796 9.4835e-05 ***
Shimmer.DDA    2.07468077543e+04  1.50501170964e+04   1.37851 0.16813233
NHR           -4.75682518352e+01  8.84223692788e+00  -5.37966 7.9511e-08 ***
HNR           -3.97679492812e-01  6.55348235385e-02  -6.06822 1.4310e-09 ***
RPDE           8.43371207600e+00  1.75276843471e+00   4.81165 1.5599e-06 ***
DFA           -3.59434951976e+01  2.14389808224e+00 -16.76549 < 2.22e-16 ***
PPE            1.58623858029e+01  3.12153172645e+00   5.08160 3.9381e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.19860904919 on 3508 degrees of freedom
Multiple R-squared:  0.14289189606,	Adjusted R-squared:  0.138982623066
F-statistic: 36.5520382634 on 16 and 3508 DF,  p-value: < 2.220446049e-16
```

reference, the variance of the y-test, also known as a randomly selected portion of total UPDRS scores equal to 80% of the observations is 100.5168, meaning this foolish model is already 20% stronger than just randomly choosing a UPDRS score!

Model 2:

Following this, our next idea was to blindly use a threshold to remove variables that have a large p-value, as a large p-value suggests the variable's end coefficient would be close to zero, and would, therefore, cause more harm than help in constructing an efficient model. We chose a threshold of .70. That is, we removed any variables that had a p-value that was higher than .7, and ended up removing Shimmer.APQ5 and Shimmer. This caused our MSE to reduce to 49.069366. A small improvement, but an improvement nonetheless.

```
Coefficients:
  (Intercept)          RPDE           DFA    Jitter.DDP  Shimmer.APQ11
       36.729        26.363       -38.948         6.894         61.725
```

Model 3:

Moving on from that model, we tried a different approach. This time, we only include a select number of variables. From before, we saw that RPDE had the highest correlation with the target variable, so we will use that variable. From there, we also chose variables that had a high correlation with each other, but only chose ones that appeared relatively less in order to avoid variables masking each other, so we chose DFA and Jitter.DDP. From there we iteratively added variables until we had 4, with the lowest MSE. Our end model for this method was one that used

RPDE, DFA, Shimmer. APQ11, and Jitter.DDP, for an average MSE of 49.60784, which ends up being slightly worse than our old model.

Using what we found by iterating through the different variables, we thought about using mathematical transformations in the lm formula for model 3. Our idea was to take the best predictive variables and divide them by the least predictive variables. While this may not make the most sense in terms of how the variables themselves may influence a patient’s UPDRS score, it may help us predict the values better. Our first model divided DFA by HNR and RPDE by NHR and had a better MSE than swapping HNR and NHR by 0.44352 and beats our previous record by 0.289916. From there, we decided to shuffle values around and found that DFA/RPDE + HNR/NHR gave slightly larger reliability of 48.66958 MSE, better than model 1 by 0.399786.

```
Call:
lm(formula = ytrain ~ (DFA/RPDE) + (HNR/NHR), data = xtrain)

Residuals:
     Min       1Q   Median       3Q      Max 
-19.4401  -5.7757  -0.3232   5.9714  19.4939 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  49.71658    1.95824  25.388  < 2e-16 ***
DFA         -40.64795    2.24982 -18.067  < 2e-16 ***
HNR          -0.40976    0.04847  -8.455  < 2e-16 ***
DFA:RPDE     19.22390    2.40824   7.983 1.89e-15 ***
HNR:NHR      -1.74920    0.43884  -3.986 6.85e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.362 on 3755 degrees of freedom
Multiple R-squared:  0.1026,	Adjusted R-squared:  0.1016 
F-statistic: 107.3 on 4 and 3755 DF,  p-value: < 2.2e-16
```

Model 4:

We wanted to have two more models under our belts, so we decided to try a different method of choosing predictor variables. Judging by how small most of our predictor variables are, we tried using a normalization function on the data in order to remove bias from comparatively large numbers. The formula we used was $z_i = \frac{x_i - min(x)}{max(x) - min(x)}$. Then, we iteratively chose the five variables that, when ran individually, gave the lowest MSE, and found DFA, Jitter.RAP, Jitter.DDP, Jitter…, and NHR. Interestingly, we see DFA and NHR again. From there, we tried random mathematical permutations until we arrived at:

$(\frac{DFA}{HNR}) * (\frac{Jitter.DDP}{Jitter...}) * Jitter.RAP$.

```
Coefficients:
                 (Intercept)                          DFA                          Jitter.DDP
                      45.616                       45.423                            -841.633
                         HNR                     DFA:RPDE                 Jitter.DDP:Jitter...
                      -0.180                     -110.802                            9709.325
              DFA:Jitter.DDP                      DFA:HNR                      Jitter.DDP:HNR
                   -2561.825                       -3.526                             176.038
     DFA:Jitter.DDP:Jitter...          DFA:RPDE:Jitter.DDP                        DFA:RPDE:HNR
                    1284.437                     5927.016                               5.695
     Jitter.DDP:Jitter...:HNR           DFA:Jitter.DDP:HNR         DFA:RPDE:Jitter.DDP:Jitter...
                   -6590.472                     -135.926                          -27282.024
 DFA:Jitter.DDP:Jitter...:HNR      DFA:RPDE:Jitter.DDP:HNR     DFA:RPDE:Jitter.DDP:Jitter...:HNR
                   11503.948                     -140.962                           -4666.707
```

<u>Using All 4 Models:</u>

Our next idea to improve the accuracy of each model was to run LOOCV with all of our models, then remove all of the data observations that had an MSE percentage difference of over 100% when compared to the variance of the total UPDRS. That is, we removed as observations all occurrences when any prediction from any of our models had an MSE greater than 300. Finally, we created a major model that uses normalized data, bad data removed, and the best formula we had so far.

Results

```
> dm
         [,1]     [,2]     [,3]         [,4]
[1,] 52.52285 52.16364 52.90452 0.0008209416
[2,] 62.83111 64.04490 62.96459 0.0013769800
[3,] 62.99414 66.28218 64.13853 0.0013689776
[4,] 63.47855 66.67862 64.46791 0.0013499905
[5,] 63.56291 63.56291 64.68819 1.1559284855
> mean(dm[, 1])
[1] 61.07791
> mean(dm[, 2])
[1] 62.54645
> mean(dm[, 3])
[1] 61.83275
> mean(dm[, 4])
[1] 0.2321691
> |
```

```
> PercentError(model1_3Variance, mean(dm[, 1]))
[1] 25.78268
> PercentError(model1_3Variance, mean(dm[, 2]))
[1] 23.44277
> PercentError(model1_3Variance, mean(dm[, 3]))
[1] 24.57387
> PercentError(model4Variance, mean(dm[, 4]))
[1] 124.6308
> var(parkdata[-terr, ]$total_UPDRS)
[1] 79.15594
> var(normalize(parkdata[-terr, ])$total_UPDRS)
[1] 1
> |
```

In order to analyze our results, we cross-validated every single model we created, all four of them, using ten-fold, fifty-fold, LOOCV, 80% train, 75%train. In doing this, we attempt to test the accuracy of each model against each other. Furthermore, by doing 5 different CV methods, we help to protect ourselves from overfitting. For reference, models 1-3 were ran using the main data set, with certain observations withheld, and model 4 was run using the normalized data set, with certain observations withheld. Our Model 4 in LOOCV was almost 200% better than randomly guessing. However, in order to get a better idea of how each model did with each cross-validation, we used the mean for each calculation of accuracy. The models performed roughly 25% better than randomly guessing, and model 4 predicted over 125% better than randomly guessing.

Conclusion

To conclude, we have established 4 models that reign different results based on the form of cross-validation that had been implemented. The way in which we chose to implement our models, although not the most efficient way of writing so, at least allowed for us to identify what could potentially be a good predictor for each given model. The model that ended up being the most effective is Model 4. As when we ran predictions, Model 4 performed 125% better than randomly guessing, while the other three models only performed roughly 25% better. As this dataset was primarily predicting continuous data, binding the variable that we were trying to predict into a classification problem proved ultimately pointless, as even if we had a high performing logistic regression model, we would lose a large amount of nuance in terms of the predictor variable and the actual values we need to predict. For the most part, the modeling was rather straightforward. We did not have any particular NA data, and since we removed the 4 indicator variables, we never had to run into a situation where we needed to ever consider doing logistic regression or classification issues.

In the end, we decided that model 4 was the most applicable model for predicting Parkinson's severity in terms of UPDRS scores, given the variable predictors in the data set. In summary, we found that normalizing all columns of data had a great impact on the predictive strength of our model, as it removes bias in the dataset. When considering LOOCV, which typically defends the best against overfitting, model 4 performs 200% better than randomly guessing, while model 1 performs roughly 17% better than randomly guessing.

References

1. “Unified Parkinson Disease Rating Scale (UPDRS).” *Theracycle Physical Therapy Exercise Bike and Rehabilitation Equipment*, www.theracycle.com/resources/links-and-additional-resources/updrs-scale/.
2. Editor, Minitab Blog. “How to Identify the Most Important Predictor Variables in Regression Models.” *Minitab Blog*, blog.minitab.com/blog/adventures-in-statistics-2/how-to-identify-the-most-important-predictor-variables-in-regression-models.