Using R to predict admission into ED

NHS-R Conference
Speaker: Gary Hutson (*Principal Analyst)*

9th October 2018

A bit about me

- Using R for 5 years and I can remember back to the days of R Base and plotting without ggplot;
- Work at NUH as a Principal Analyst overseeing the Activity & Access Team;
- I have a website (hutsons-hacks.info) dedicated to R and I am a blogger on the R community:
    - https://nhsrcommunity.com/blog/a-simple-function-to-install-and-load-packages-in-r/
    - https://nhsrcommunity.com/blog/violin-and-density-plots-in-ggplot2/
    - https://nhsrcommunity.com/blog/pareto-chart-in-ggplot2/
    - https://nhsrcommunity.com/blog/histogram-with-auto-binning-in-ggplot2/
    - https://nhsrcommunity.com/blog/diverging-dot-plot-and-lollipop-charts-plotting-variance-with-ggplot2/
    - https://nhsrcommunity.com/blog/diverging-bar-charts-plotting-variance-with-ggplot2/
    - More coming soon – starting with a few machine learning algorithms and time series methods…
- If you need any help and advice post this, please do not hesitate to get in touch.

INFORMATION & INSIGHT I2

We Listen We Care

The analysis task

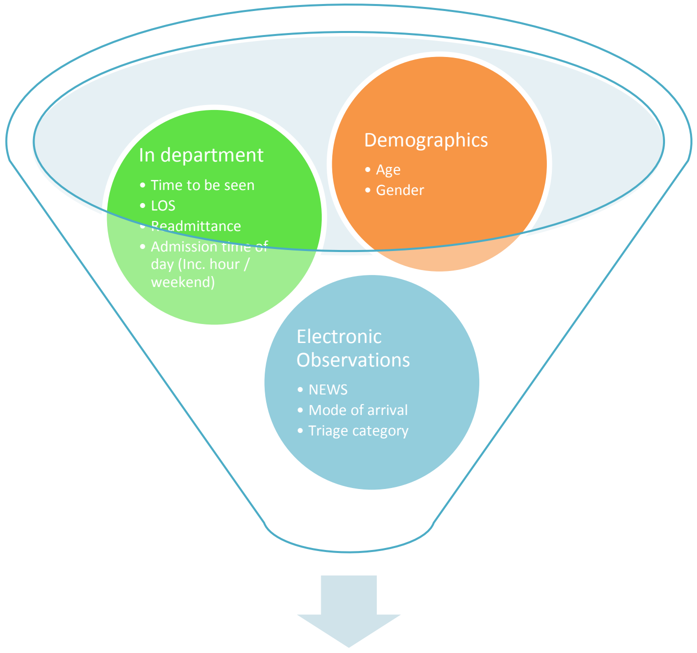
- Health Foundation funding secured to deploy the Glasgow Score into our Medway BI system to be used with predicting the probability of someone being admitted into department, at the point of triage;
- Evolution of the score to bring in additional predictor variables, building on the original analysis;
- Repeatability and deployment on to a large scale system was the main consideration;
- Utilisation of Machine Learning principles to train and test the data set in R;
- Original Logistic Regression analysis run in Excel (which took half a day to reproduce).

Choosing the right method

- The starting point for the analysis was to use the logistic regression algorithm to start to analyse how good the Glasgow score was when applied to our core BI solution (Medway);
- This score was derived from this research: https://emj.bmj.com/content/32/3/174 and when applied, was 69% accurate at predicting whether a patient would be admitted or not;
- I had a phone call from the lead consultant (Ben) to say, in his words, *"Gary you are the geekiest person I know – would you like to work on this and can we improve the score";*
- My original approach was to use the same algorithm - as used in the initial research - this was tested against other types of models – using a Machine Learning process called stacking. Logistic regression came out as the best performing predictor / classifier – however the list tested against was not extensive.

Who says repetition is not fun?

- What ensued was a journey of repeated testing of the regression model and the search for those pesky significant predictor variables



Admission into department

How and why was R important?

- The project had two phases – both needing the model to be iterated:
  - The first phase was to test the various combinations of ‘in department’, patient and NEWS (utilising E-Observations) predictor variables. This resulted in over 55 permutations of the model combination being tested;
  - The second phase was premised around building our own NUH score and assessing each of the components of the NEWS (National Early Warning Score) individually on predicting admission;
  - As stated in the previous slides, this required repeated testing of the logistic regression models, each of which needed validation and testing. The solution was R. The next set of slides outline tips I want to share from the analytical journey;
  - I am getting to the R part – enough of my preamble.

Utilising R for phase one – loading in the data

Loading data via SQL. This can be achieved by downloading and installing the package **RODBC**:

```
library(RODBC)
```

The first step is defining the connection string of the SQL database you are trying to connect to. My example uses a Microsoft SQL Server database:

```
connStr <- "Driver={SQL Server Native Client 11.0};Server=myserver;
Database=mydatabase;Trusted_Connection=yes;"
```

Next, use RODBC's *odbcDriverConnect* function to pass the variable created previously connStr to the function. This is implemented below:

```
myconn <- odbcDriverConnect(connection = connStr)
```

Now test that the connect has been established. This can be done by printing the variable to the command window **print(myconn)**.

Utilising R for phase one – loading in the data (cont.)

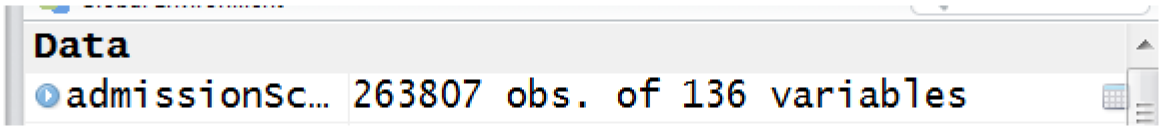
Now create a variable to pass the whole SQL string to:

```
sql_command <- “my SQL script goes here"
```

The final command you need to use to get the data read in is to use:

```
admission_score_df <- sqlQuery(myconn, sql_command)
```

This function uses the connection string previously defined and the SQL command specified in this section to add the contents to a data frame.



```
Data
admissionSc… 263807 obs. of 136 variables
```

INFORMATION & INSIGHT I2

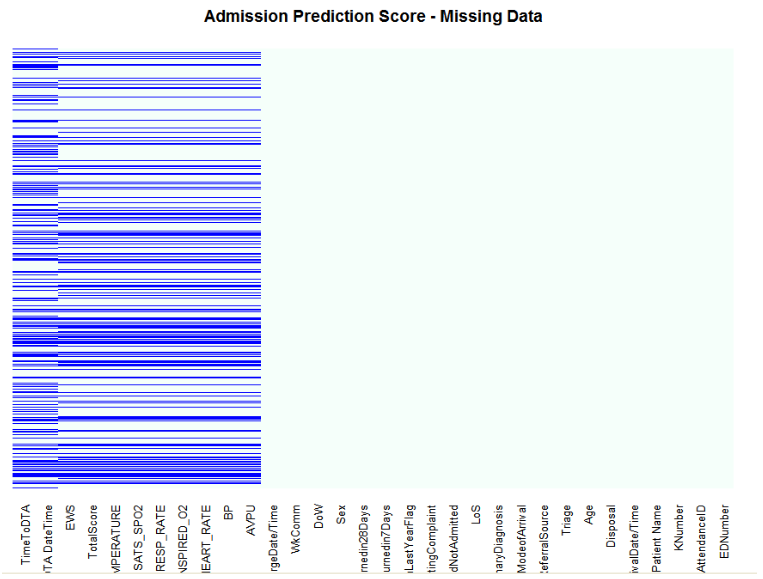
We Listen We Care

Utilising R for phase one – data cleansing

Missing values

The package to use here is **Amelia** as it allows you to produce a visualisation of missing values in your datasets. As you can see, this allows you to pick out which fields have the most NA values within your core dataset.

```
library(Amelia)
missmap(admissionScoreCopy,
        main="Missing Data Map"  col=c("blue",
"mintcream"), legend=FALSE, xaxt ='n')
```



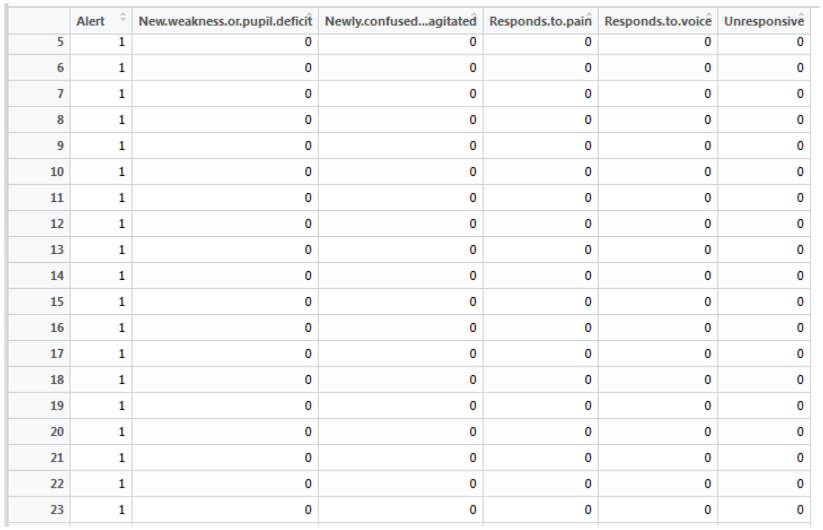
Alternatively, you could the **data.table** library to use the na.omit command to omit missing values. Amelia can also be used to impute missing values, using the **amelia_fit** function.

Utilising R for phase one – feature engineering

Categorical dummy coding

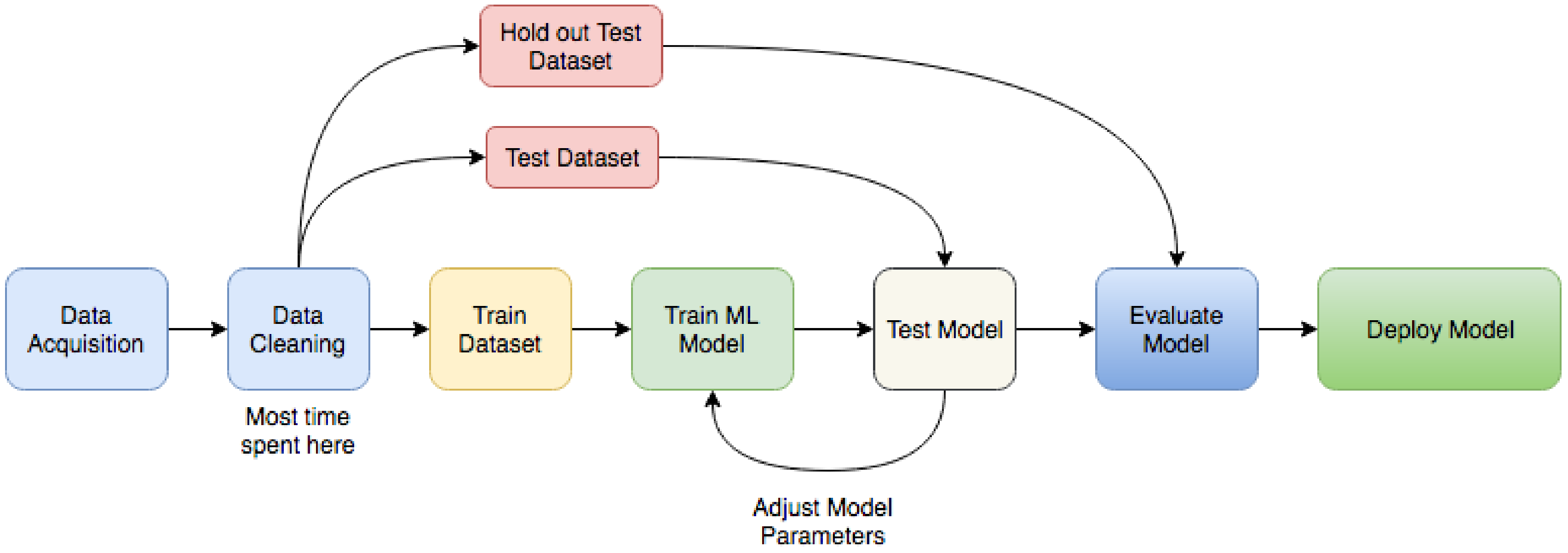
Before analysing some of the data sets in my analysis, categorical data needed to be dummy coded (https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faqwhat-is-dummy-coding/) to convert this into a numerical representation and data type. I love the package **psych** for this:

```
library(psych)
AVPUCoded <- data.frame(dummy.code(ADNMTS$AVPU))
#Refer to the AVPU column ($) in the weirdly titled data frame
AVPUCoded <- AVPUCoded[,-1]
#Drop the reference column – as dummy coding works on k-1.
AVPUCoded <- cbind(AVPUCoded, AdmitFlag=ADNMTS$AdmitFLAG)
#Bind the data frame back to the original data frame
```



	Alert	New.weakness.or.pupil.deficit	Newly.confused...agitated	Responds.to.pain	Responds.to.voice	Unresponsive
5	1	0	0	0	0	0
6	1	0	0	0	0	0
7	1	0	0	0	0	0
8	1	0	0	0	0	0
9	1	0	0	0	0	0
10	1	0	0	0	0	0
11	1	0	0	0	0	0
12	1	0	0	0	0	0
13	1	0	0	0	0	0
14	1	0	0	0	0	0
15	1	0	0	0	0	0
16	1	0	0	0	0	0
17	1	0	0	0	0	0
18	1	0	0	0	0	0
19	1	0	0	0	0	0
20	1	0	0	0	0	0
21	1	0	0	0	0	0
22	1	0	0	0	0	0
23	1	0	0	0	0	0

Hold on a minute – you mentioned the Machine Learning process – what is that?



Utilising R for phase one – testing the dataset with Logistic Regression

Selecting the fields post cleansing and dummy coding

A data frame (tibble) can now be created via utilisation of dplyr's **select** function:

```
library(dplyr)
data_frame_name %<>%
        as_data_frame %>%
                select_("field1", "field2")
```

The relevant fields for the analysis should now contain only numeric predictor (independent variables) and those coded categorical variables of interest, for the analysis.

The next slide is important, as it advocates how to split the data to deploy this as part of the Machine Learning process.

INFORMATION & INSIGHT

We Listen We Care

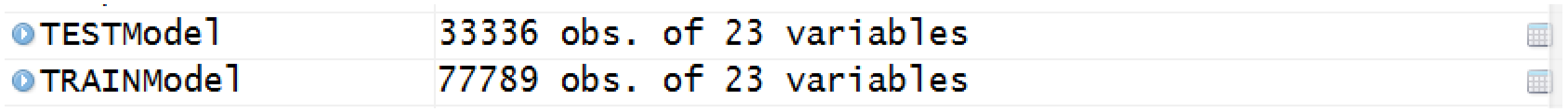
Utilising R for phase one – deciding how to validate the models

The data initially was just divided into a test and training set. Later a validation set was used. The below shows how to create a partition for a test and training set:

```
library(caret)
#The driver for all classification and regression machine learning in R – bar deep learning add ins such as H2O
dp <- createDataPartition(admission_df$AdmitFlag, p= 0.7, list=F)
 TRAINModel <- admission_df [dp,]
 TESTModel <- admission_df [-dp,]
```

This creates a data partition (candidates randomly sampled into each group) and then the TRAIN and TEST models data frames are created with 70% of observations in the training set and 30% in the test set. This partitions on the dependent variable (the data item you are trying to predict).

In the case of this analysis, that happens to be a binary dependent variable (thus logistic regression is a no brainer).

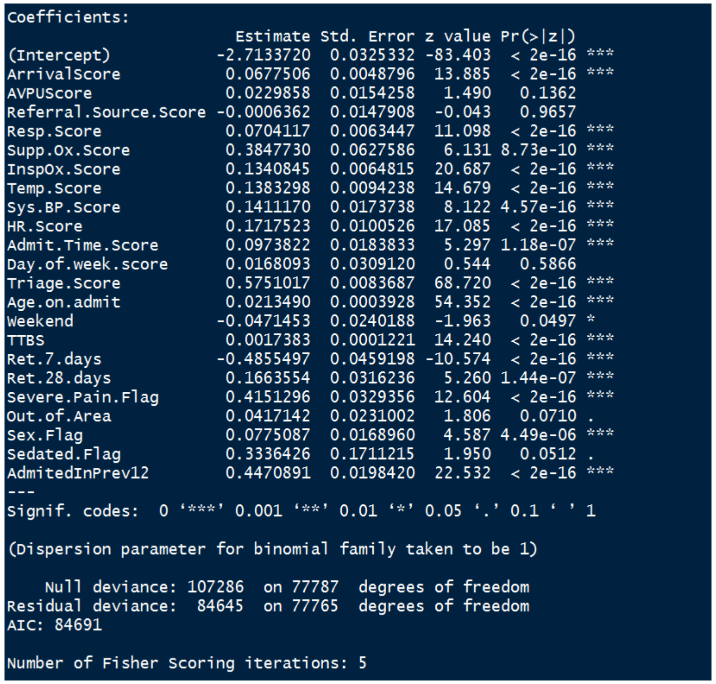


```
TESTModel       33336 obs. of 23 variables
TRAINModel      77789 obs. of 23 variables
```

Utilising R for phase one – running the algorithm on the models

To run the algorithm on the models this is relatively simple:

```
library(caret)
TRAINED_Model <- train(factor(AdmitFlag) ~ .,
                    data=TRAINModel,
                    method="glm", family="binomial")
 #Converts the AdmitFlag to a factor – based on previous admissions and discharges, uses the train model to tune the parameters and set the coefficients, specifies the family of regression and the method of generalised linear model (GLM).
```



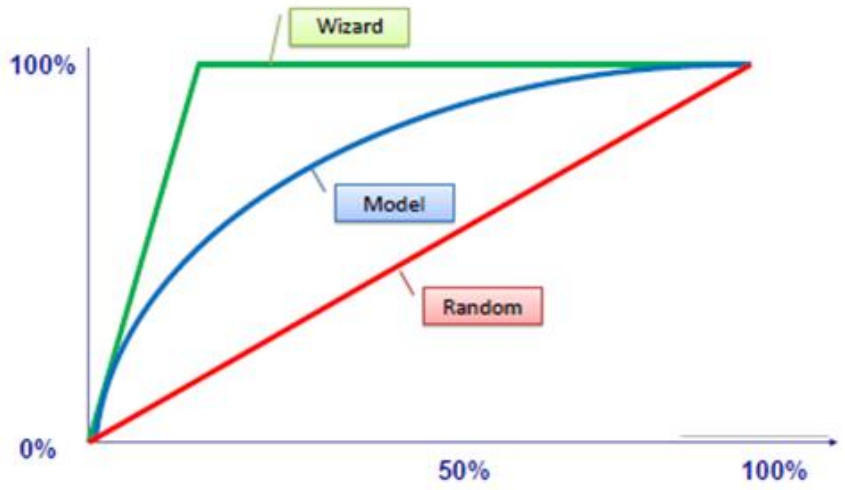
```
Coefficients:
                       Estimate Std. Error z value Pr(>|z|)    
(Intercept)          -2.7133720  0.0325332 -83.403  < 2e-16 ***
ArrivalScore          0.0677506  0.0048796  13.885  < 2e-16 ***
AVPUScore             0.0229858  0.0154258   1.490   0.1362    
Referral.Source.Score -0.0006362  0.0147908  -0.043   0.9657    
Resp.Score            0.0704117  0.0063447  11.098  < 2e-16 ***
Supp.Ox.Score         0.3847730  0.0627586   6.131 8.73e-10 ***
InspOx.Score          0.1340845  0.0064815  20.687  < 2e-16 ***
Temp.Score            0.1383298  0.0094238  14.679  < 2e-16 ***
Sys.BP.Score          0.1411170  0.0173738   8.122 4.57e-16 ***
HR.Score              0.1717523  0.0100526  17.085  < 2e-16 ***
Admit.Time.Score      0.0973822  0.0183833   5.297 1.18e-07 ***
Day.of.week.score     0.0168093  0.0309120   0.544   0.5866    
Triage.Score          0.5751017  0.0083687  68.720  < 2e-16 ***
Age.on.admit          0.0213490  0.0003928  54.352  < 2e-16 ***
Weekend              -0.0471453  0.0240188  -1.963   0.0497 *  
TTBS                  0.0017383  0.0001221  14.240  < 2e-16 ***
Ret.7.days           -0.4855497  0.0459198 -10.574  < 2e-16 ***
Ret.28.days           0.1663554  0.0316236   5.260 1.44e-07 ***
Severe.Pain.Flag      0.4151296  0.0329356  12.604  < 2e-16 ***
Out.of.Area           0.0417142  0.0231002   1.806   0.0710 .  
Sex.Flag              0.0775087  0.0168960   4.587 4.49e-06 ***
Sedated.Flag          0.3336426  0.1711215   1.950   0.0512 .  
AdmitedInPrev12       0.4470891  0.0198420  22.532  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 107286  on 77787  degrees of freedom
Residual deviance:  84645  on 77765  degrees of freedom
AIC: 84691

Number of Fisher Scoring iterations: 5
```

Utilising R for phase one – predicting against the training set results

Predicting against these results is even simpler in R:

```
library(caret)
 PREDICT_TRAIN_Model = predict(TRAINED_Model, newdata=TESTModel) #This will get the predicted classifications i.e. admitted = 1 or discharged = 0
#If you wanted the predicted probabilities, then an additional parameter would need adding of:
PREDICT_TRAIN_Model = predict(TRAINED_Model, newdata=TESTModel, type="prob")
```

The next stage is then to test whether this iteration of the model is any good:

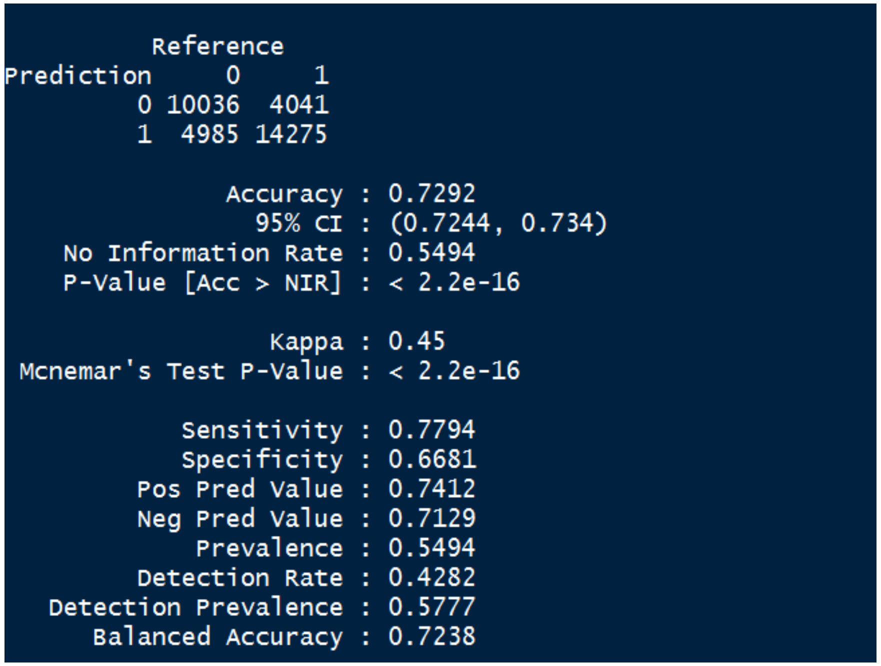


Utilising R for phase one – evaluating the model

My favoured two options here are to use confusion matrices and ROC curves to evaluate the models, as for senior stakeholder, these can be implemented as hereunder shown:

Confusion Matrixes

```
library(caret)
confusionMatrix(data=PREDICT_TRAIN_Model,TEST
Model[,"AdmitFlag"], positive = "1")
```



```
          Reference
Prediction     0     1
         0 10036  4041
         1  4985 14275

               Accuracy : 0.7292
                 95% CI : (0.7244, 0.734)
    No Information Rate : 0.5494
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.45
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.7794
            Specificity : 0.6681
         Pos Pred Value : 0.7412
         Neg Pred Value : 0.7129
             Prevalence : 0.5494
         Detection Rate : 0.4282
   Detection Prevalence : 0.5777
      Balanced Accuracy : 0.7238
```

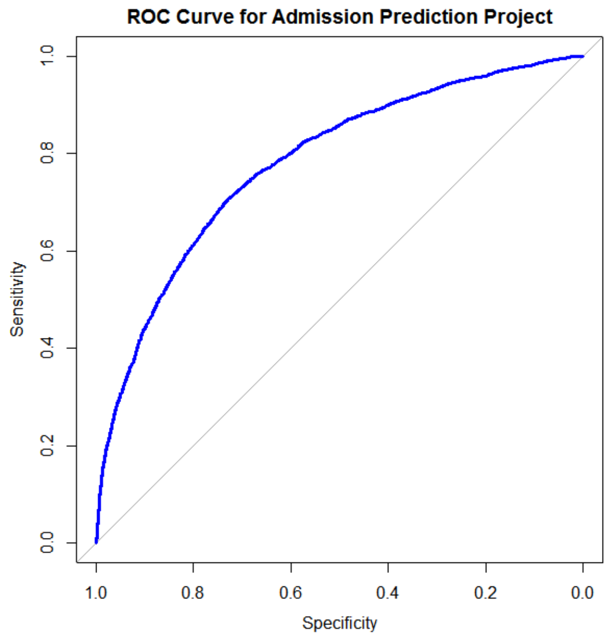
Confusion matrices: read (https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/) for interpreting every statistic reported on the R output.

Utilising R for phase one – evaluating the model

ROC Curve

The best package for this is a package called **ModelGood** as the results of a logistic regression can be directly utilised in a couple of simple instructions. Another package is **ROCR**. The script below shows how to deploy this:

```
library(ModelGood)
roc_glm <- glm(factor(AdmitFlag) ~
.,data=TRAINModel,family=binomial)
 roc.curve <- Roc(roc_glm)
 plot(roc.curve, col="blue")
 title(main="ROC Curve for NUH Score")
#To obtain the area under the curve use the below:
print(roc.curve)
```



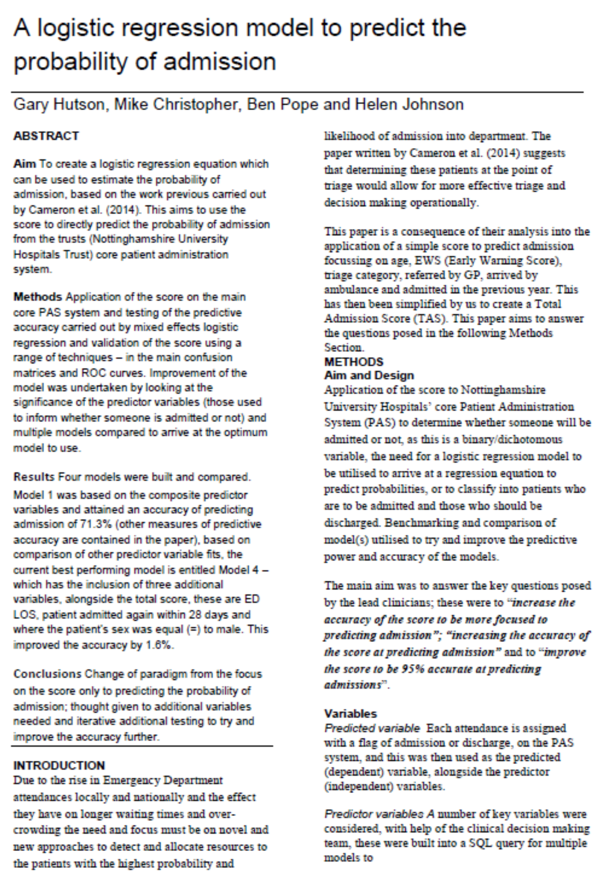
Utilising R for phase one – conclusion

Reiteration of the models, working with senior stakeholders who know the service and accuracy comparison and evaluation was key.

I recommend that if you are attempting a similar project that you engage with senior leads and staff on the group to get buy in and involve them in the idea sharing process.

The best performing model after 53 models were created (with varied permutations of the predictor variables) – over a number of months – was selected and we managed to improve the accuracy of the model from 69% to 78% on the last count.

This research was submitted to the Health Foundation and we are in the process of submitting this to the EMJ (Emergency Medical Journal).



A logistic regression model to predict the probability of admission

Gary Hutson, Mike Christopher, Ben Pope and Helen Johnson

ABSTRACT

Aim To create a logistic regression equation which can be used to estimate the probability of admission, based on the work previous carried out by Cameron et al. (2014). This aims to use the score to directly predict the probability of admission from the trusts (Nottinghamshire University Hospitals Trust) core patient administration system.

Methods Application of the score on the main core PAS system and testing of the predictive accuracy carried out by mixed effects logistic regression and validation of the score using a range of techniques – in the main confusion matrices and ROC curves. Improvement of the model was undertaken by looking at the significance of the predictor variables (those used to inform whether someone is admitted or not) and multiple models compared to arrive at the optimum model to use.

Results Four models were built and compared.

Model 1 was based on the composite predictor variables and attained an accuracy of predicting admission of 71.3% (other measures of predictive accuracy are contained in the paper), based on comparison of other predictor variable fits, the current best performing model is entitled Model 4 – which has the inclusion of three additional variables, alongside the total score, these are ED LOS, patient admitted again within 28 days and where the patient's sex was equal (=) to male. This improved the accuracy by 1.6%.

Conclusions Change of paradigm from the focus on the score only to predicting the probability of admission; thought given to additional variables needed and iterative additional testing to try and improve the accuracy further.

INTRODUCTION

Due to the rise in Emergency Department attendances locally and nationally and the effect they have on longer waiting times and over-crowding the need and focus must be on novel and new approaches to detect and allocate resources to the patients with the highest probability and likelihood of admission into department. The paper written by Cameron et al. (2014) suggests that determining these patients at the point of triage would allow for more effective triage and decision making operationally.

This paper is a consequence of their analysis into the application of a simple score to predict admission focussing on age, EWS (Early Warning Score), triage category, referred by GP, arrived by ambulance and admitted in the previous year. This has then been simplified by us to create a Total Admission Score (TAS). This paper aims to answer the questions posed in the following Methods Section.

METHODS

Aim and Design

Application of the score to Nottinghamshire University Hospitals' core Patient Administration System (PAS) to determine whether someone will be admitted or not, as this is a binary/dichotomous variable, the need for a logistic regression model to be utilised to arrive at a regression equation to predict probabilities, or to classify into patients who are to be admitted and those who should be discharged. Benchmarking and comparison of model(s) utilised to try and improve the predictive power and accuracy of the models.

The main aim was to answer the key questions posed by the lead clinicians; these were to "***increase the accuracy of the score to be more focused to predicting admission***"; "***increasing the accuracy of the score at predicting admission***" and to "***improve the score to be 95% accurate at predicting admissions***".

Variables

Predicted variable Each attendance is assigned with a flag of admission or discharge, on the PAS system, and this was then used as the predicted (dependent) variable, alongside the predictor (independent) variables.

Predictor variables A number of key variables were considered, with help of the clinical decision making team, these were built into a SQL query for multiple models to

INFORMATION & INSIGHT I²

We Listen We Care

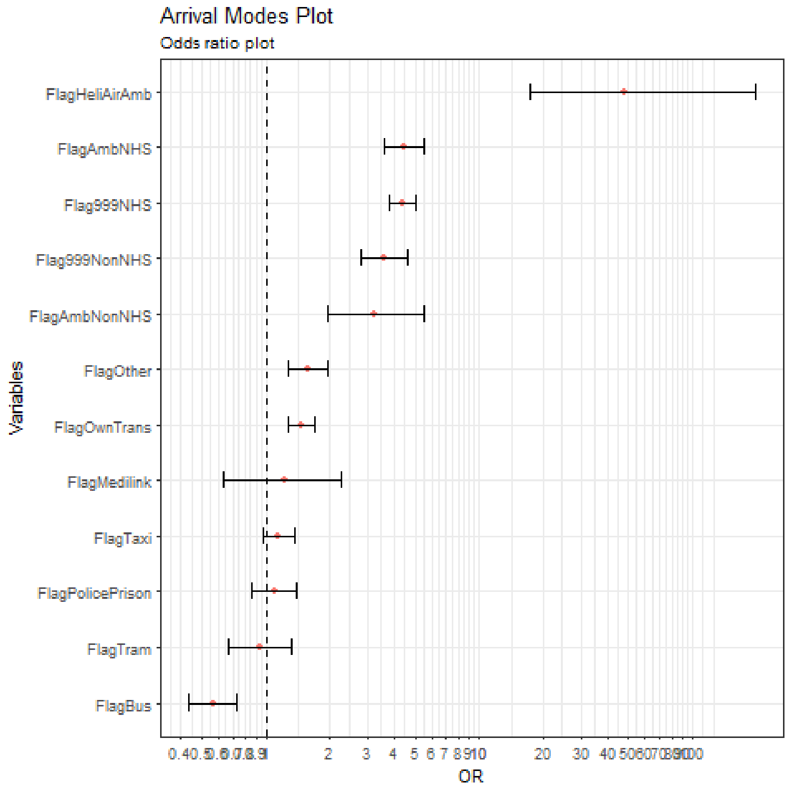
Utilising R for phase two – decomposition of the NEWS score

This involved taking each element of the NEWS scoring framework, now NEWS2 (https://www.rcplondon.ac.uk/projects/outputs/national-early-warning-score-news-2) and following this process:

1. Run a regression on each one of the components of the NEWS score (some needed dummy coding – due to being a categorical data type);
2. Create an odds plot function in R to enable me to present this back to our Emergency Team, and the lead Emergency Consultant, in a visual way. Trying to keep some of the hard-core statistical elements to a minimum.
3. Evaluate the key components of each of the NEWS types in predicting admission.

Utilising R for phase two – creating the odds plot function

Functions are the most powerful thing about R and if you want to extend the power of R, then this is the way to do it (*in my opinion*). The below set of R code shows how to create a odds plot – using ggplot2:



```
plot_odds<-function(x, title = NULL){
  tmp<-data.frame(cbind(exp(coef(x)), exp(confint(x))))
  odds<-tmp[-1,]
  names(odds)<-c('OR', 'lower', 'upper')
  odds$vars<-row.names(odds)
  ticks<-c(seq(.1, 1, by =.1), seq(0, 10, by =1), seq(10, 100, by
=10))

  ggplot(odds, aes(y= OR, x = reorder(vars, OR))) +
    geom_point(aes(color='blue')) +
    geom_errorbar(aes(ymin=lower, ymax=upper), width=.3)
+
    scale_y_log10(breaks=ticks, labels = ticks) +
    geom_hline(yintercept = 1, linetype=2) +
    coord_flip() +
    labs(title = title, subtitle = "Odds ratio plot", x =
'Variables', y = 'OR') +
    theme_bw() + theme(legend.position = "none")
 }
```

INFORMATION & INSIGHT I²

We Listen We Care

Utilising R for phase two – creating the odds plot function

Using the function – this needs the results of the trained model to work:

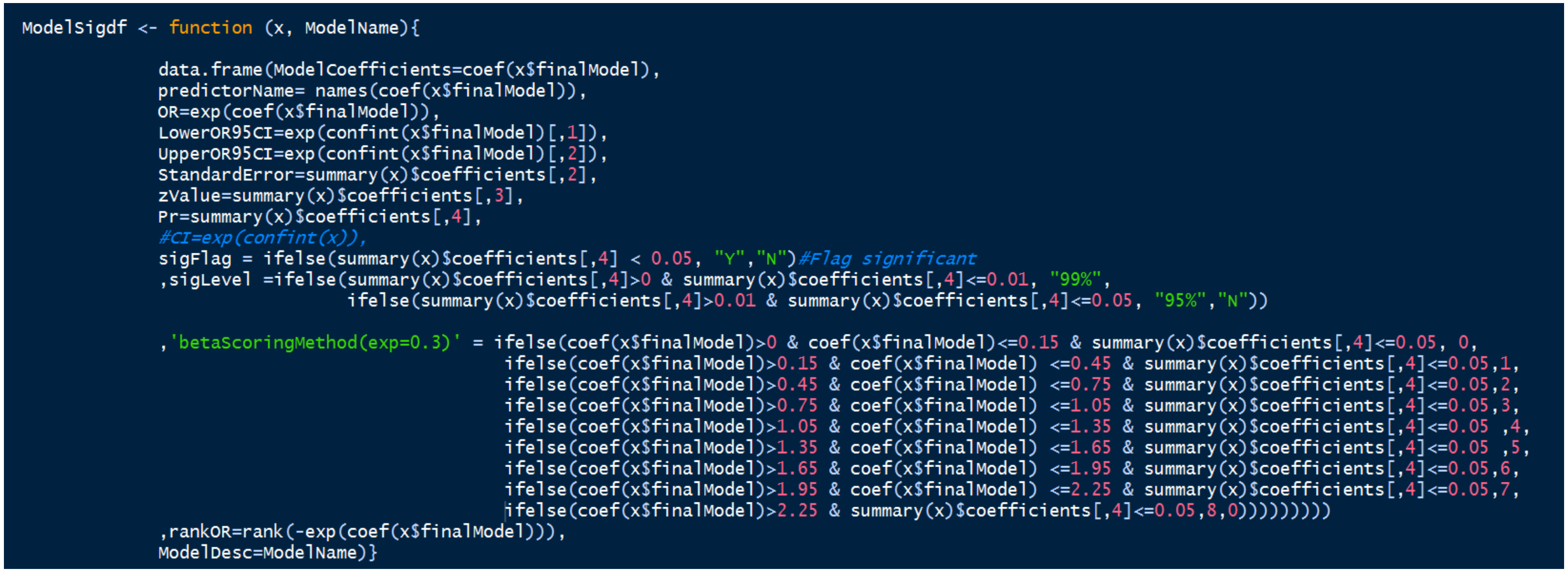
```
dev.new()
#Turn on a new graphical device
plot_odds(TRAINED_Model$finalModel, 'SATS score NEMS bandings Odds Plot')
#Pass the results of the TRAINED_Model and access the parameter finalModel to get the
coefficients and other statistical properties
dev.copy(png,'SATS score NEMS bandings Odds Plot.png')
#Copy the plot as png format and then export out with the given name and file type
dev.off()
#Turn the graphical device off
```

INFORMATION & INSIGHT

We Listen We Care

Utilising R for phase two – creating the scoring function

This below R code now scores every patient with a score and then this scoring framework is validated and retrained. This type of solution needs a full integration to our core systems.

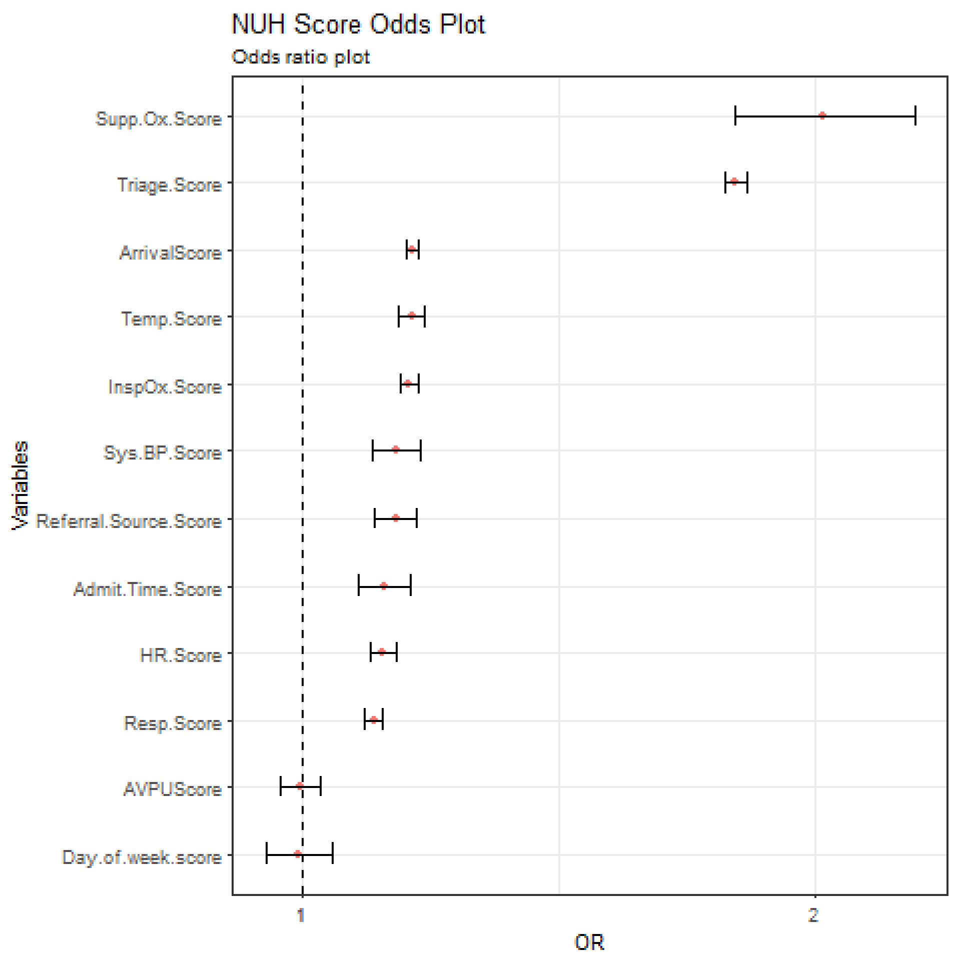


```
ModelSigdf <- function (x, ModelName){

            data.frame(ModelCoefficients=coef(x$finalModel),
            predictorName= names(coef(x$finalModel)),
            OR=exp(coef(x$finalModel)),
            LowerOR95CI=exp(confint(x$finalModel)[,1]),
            UpperOR95CI=exp(confint(x$finalModel)[,2]),
            StandardError=summary(x)$coefficients[,2],
            zValue=summary(x)$coefficients[,3],
            Pr=summary(x)$coefficients[,4],
            #CI=exp(confint(x)),
            sigFlag = ifelse(summary(x)$coefficients[,4] < 0.05, "Y","N")#Flag significant
            ,sigLevel =ifelse(summary(x)$coefficients[,4]>0 & summary(x)$coefficients[,4]<=0.01, "99%",
                             ifelse(summary(x)$coefficients[,4]>0.01 & summary(x)$coefficients[,4]<=0.05, "95%","N"))

            ,'betaScoringMethod(exp=0.3)' = ifelse(coef(x$finalModel)>0 & coef(x$finalModel)<=0.15 & summary(x)$coefficients[,4]<=0.05, 0,
                                              ifelse(coef(x$finalModel)>0.15 & coef(x$finalModel) <=0.45 & summary(x)$coefficients[,4]<=0.05,1,
                                              ifelse(coef(x$finalModel)>0.45 & coef(x$finalModel) <=0.75 & summary(x)$coefficients[,4]<=0.05,2,
                                              ifelse(coef(x$finalModel)>0.75 & coef(x$finalModel) <=1.05 & summary(x)$coefficients[,4]<=0.05,3,
                                              ifelse(coef(x$finalModel)>1.05 & coef(x$finalModel) <=1.35 & summary(x)$coefficients[,4]<=0.05 ,4,
                                              ifelse(coef(x$finalModel)>1.35 & coef(x$finalModel) <=1.65 & summary(x)$coefficients[,4]<=0.05 ,5,
                                              ifelse(coef(x$finalModel)>1.65 & coef(x$finalModel) <=1.95 & summary(x)$coefficients[,4]<=0.05,6,
                                              ifelse(coef(x$finalModel)>1.95 & coef(x$finalModel) <=2.25 & summary(x)$coefficients[,4]<=0.05,7,
                                              ifelse(coef(x$finalModel)>2.25 & summary(x)$coefficients[,4]<=0.05,8,0)))))))))
            ,rankOR=rank(-exp(coef(x$finalModel))),
            ModelDesc=ModelName)}
```

Utilising R for phase two – creating the scoring function

The scoring system was then applied and checked against the other models I had developed. Performance-wise this was on par with the other harder to understand statistical models (trying to cater for my non-statistician colleagues and reduce complexity).

So, how did we end up using this research?

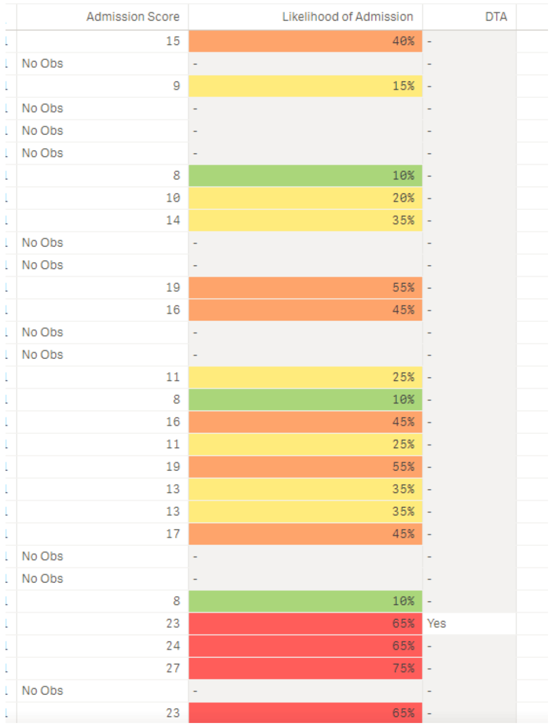


INFORMATION & INSIGHT I^2

We Listen We Care

How is the model deployed currently

The model is deployed as a regression equation, which is validated every couple of months to adjust the coefficients and make sure the predictors are still *fit for purpose,* in our BI tool Qlik Sense:



Admission Score	Likelihood of Admission	DTA
15	40%	-
No Obs	-	-
9	15%	-
No Obs	-	-
No Obs	-	-
No Obs	-	-
8	10%	-
10	20%	-
14	35%	-
No Obs	-	-
No Obs	-	-
19	55%	-
16	45%	-
No Obs	-	-
No Obs	-	-
11	25%	-
8	10%	-
16	45%	-
11	25%	-
19	55%	-
13	35%	-
13	35%	-
17	45%	-
No Obs	-	-
No Obs	-	-
8	10%	-
23	65%	Yes
24	65%	-
27	75%	-
No Obs	-	-
23	65%	-

This is currently set up to flag those with a probability of admission and indicates if there has been a decision to admit granted.

Next steps would be:

- Creation of alerts and cut offs to alert front door staff of the number of > 80% patients in department. We have a tool called Nprinting to do automated alerting;
- Continue to improve the accuracy with various different ML models (https://rdrr.io/cran/caret/man/models.html) .

Next steps

The next steps for the project were outlined as:

- Exploring different algorithms to improve accuracy, as there are better classifiers / regressors out there – such as Support Vector Machines, Neural Nets, etc.;
- Ensemble learning (boosting, bagging and stacking – see https://blog.statsbot.co/ensemble-learning-d1dcd548e936) is another avenue I am exploring at the moment – so far the 5 algorithms I have tested against my algorithm do not perform as well as my original choice;
- Agreeing what the budget is for this self learning and improving algorithm to be deployed to connect and update a live server – refreshing the probabilities real time;
- Integrating Qlik Sense with R to make the dashboard we have more accurate and updated;
- The latter two bullets need to be costed and put through a change process, which they are currently going through;
- R is just so versatile – it lets you do more or less anything you want with data – from the Tidyverse to Machine Learning. I have been in love with it for 5 years – even when it was programming it in R base.

Revolution