**Introduction**

**Explaining models**

There are 2 approaches to explaining models

1. Use non-complex models. These algorithms are associated with traditional statistical models. The simple structure allows us to understand the intrinsic workings of the models.
2. Conduct post-hoc interpretation on models. Post-hoc interpretation can be applied on both simple and black box models. These analyses done after model training can be further broken down into model specific and model agnostic approaches.

**Direction of the post**

In this post we will explore the first approach of explaining models, using interpretable models such as logistic regression and decision trees *(decision trees will be covered in another post)* . I will be using the tidymodels approach to create these algorithms. The dataset used is the Cleveland heart dataset which is a binary classification problem if heart disease is present or absent for a patient.

**Model production pipeline**

**Glossary**

There are 3 columns in the glossary below:

1. Original: The original variable names
2. Definition: The expansion of the abbreviated original variable names and the details for each factor level for categorical variables.
3. Rename: The renamed variable names which is meant to less jargonistic than the original variable names.

#library

library(tidyverse)

library(tidymodels)

#import

heart<-read\_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data", col\_names = F)

#glossary

tribble(~Original, ~Definition, ~Rename,

"age", "" , "age",

"sex", "1= Male, 2=Female", "sex",

"cp", "Chest pain at rest, there is another variable related to chest pain during the exercise stress test. 1=typical angina, 2= atypical angina, 3= non-chestpain pain, 4=asymtomatic" ,"rest\_cp",

"tresbps", "Resting blood pressure (in mm Hg on admission to the hospital)", "rest\_bp",

"chol", "Serum cholesterol", "chol",

"fbs", "Fasting blood sugar. 1= >120 mg/dl, 0 = <120 mg/dl","fast\_bloodsugar",

"restecg", "Resting electrocardiographic results. 0=normal, 1=ST-T wave abnormality, 2=left ventricular hypertrophy", "rest\_ecg",

"thalach", "Maximum heart rate achieved. based on values, likely taken during exercise stress test", "ex\_maxHR",

"exang", "Exercise induced angina (chest pain). 1=yes, 0=no", "ex\_cp",

"oldpeak", "ST depression induced by exercise relative to rest", "ex\_STdepression\_dur",

"slope", "Slope of the peak exercise ST segment. 1=upsloping/normal, 2=flat, 3=downsloping", "ex\_STpeak",

"ca", "Number of major vessels colored by flourosopy", "coloured\_vessels",

"thal", "Thalassemia. 3=normal, 6=fixed defect, 7=reversable defect", "thalassemia",

"num", "Angiographic status of heart disease. 0= <50% diameter narrowing (no heart disease), >1= >50% diameter narrowing (heart disease present)", "heart\_disease") %>% DT::datatable(rownames = F, options = list(paging= T))

**Data wrangling**

**Variable and value names**

We will convert the numeric encoding of categorical variables to their intended meaning. Using the intended meaning will facilitate the interpretation of models. For instance, saying “typical resting chest pain is the most influential variable” is more comprehensible than “resting chest pain =1 is the most influential variable”.

# Renaming var

colnames(heart)<- c("age", "sex", "rest\_cp", "rest\_bp",

"chol", "fast\_bloodsugar","rest\_ecg","ex\_maxHR","ex\_cp",

"ex\_STdepression\_dur", "ex\_STpeak","coloured\_vessels", "thalassemia","heart\_disease")

#elaborating cat var

##simple ifelse conversion

heart<-heart %>% mutate(

sex= ifelse(sex=="1", "male", "female"),

fast\_bloodsugar= ifelse(fast\_bloodsugar=="1", ">120", "<120"), ex\_cp=ifelse(ex\_cp=="1", "yes", "no"),

heart\_disease=ifelse(heart\_disease=="0", "no", "yes"))

## complex ifelse conversion using `case\_when`

heart<-heart %>% mutate(

rest\_cp=case\_when(rest\_cp== "1" ~ "typical",

rest\_cp=="2" ~ "atypical",

rest\_cp== "3" ~ "non-CP pain",

rest\_cp== "4" ~ "asymptomatic"),

rest\_ecg=case\_when(rest\_ecg=="0" ~ "normal",

rest\_ecg=="1" ~ "ST-T abnorm",

rest\_ecg=="2" ~ "LV hyperthrophy"),

ex\_STpeak=case\_when(ex\_STpeak=="1" ~ "up/norm",

ex\_STpeak== "2" ~ "flat",

ex\_STpeak== "3" ~ "down"),

thalassemia=case\_when(thalassemia=="3.0" ~ "norm",

thalassemia== "6.0" ~ "fixed",

thalassemia== "7.0" ~ "reversable"))

**Missing data**

Missing values often reflected as NA or “?”. First, let’s identify which variables have missing values recorded as “?”

heart %>% map\_df(~sum((.x)=="?")) %>% gather(key="var", value = "value") %>% filter(value >0)

## # A tibble: 1 x 2

## var value

##

## 1 coloured\_vessels 4

We will convert the 4 observations in “coloured\_vessels” with missing values recorded as “?” into true NA.

heart<-heart%>% mutate\_if(is.character, funs(replace(., .=="?", NA)))

Now let’s tally the total number of missing values reported as NA for each variable. We will impute these missing values later in the pre-processing stage with recipe::step\_knnimpute()

heart %>% map\_df(~sum(is.na(.x)))

## # A tibble: 1 x 14

## age sex rest\_cp rest\_bp chol fast\_bloodsugar rest\_ecg ex\_maxHR ex\_cp

##

## 1 0 0 0 0 0 0 0 0 0

## # ... with 5 more variables: ex\_STdepression\_dur , ex\_STpeak ,

## # coloured\_vessels , thalassemia , heart\_disease

**Variable class**

Currently, categorical variables are treated as characters. We will need to convert them to factors before feeding them into the model.

heart %>% map\_df(~class(.x)) %>% gather(key="var", value = "class")

## # A tibble: 14 x 2

## var class

##

## 1 age numeric

## 2 sex character

## 3 rest\_cp character

## 4 rest\_bp numeric

## 5 chol numeric

## 6 fast\_bloodsugar character

## 7 rest\_ecg character

## 8 ex\_maxHR numeric

## 9 ex\_cp character

## 10 ex\_STdepression\_dur numeric

## 11 ex\_STpeak character

## 12 coloured\_vessels character

## 13 thalassemia character

## 14 heart\_disease character

I prefer to convert the categorical variables during the data wrangling stage rather than during the model pre-processing stage with recipes::step\_string2factors. Having the dataset in the right form during the data wrangling phase helps to prevent any errors further upstream during pre-processing and model production.

heart<-heart %>% mutate\_if(is.character, as.factor)

**Models**

**Training/test set**

We will use functions from Rsample to create the training and test set.

set.seed(4595)

data\_split <- initial\_split(heart, prop=0.75, strata = "heart\_disease")

heart\_train <- training(data\_split)

heart\_test <- testing(data\_split)

**Pre-processing**

The training and test sets are pre-processed using functions from recipes. We will not explicitly create one hot encoding for categorical variables as both logistic regressions and decision trees are able to accommodate them. I kept the feature engineering brief as I wanted the explanation of the models to be succinct.

# create recipe object

heart\_recipe<-recipe(heart\_disease ~., data= heart\_train) %>%

step\_knnimpute(all\_predictors())

# process the training set/ prepare recipe(non-cv)

heart\_prep <-heart\_recipe %>% prep(training = heart\_train, retain = TRUE)

**Model creation**

The model will be created using functions from parsnip

lr\_model <- logistic\_reg(penalty = 10, mixture = 0.1, mode = "classification") %>% set\_engine("glm") %>%

fit(heart\_disease ~ ., data = juice(heart\_prep))

Now that we’ve built the model, let’s interpret the white box model.

**Logistic regression**

Logistic regression is one of the classic models use in medical research to solve classification problems. Logistic regression provides us with coefficient estimates but most often we use a derivate of the coefficient estimate, odds ratio, in comprehending the model.

**Coefficient estimate**

Before elaborating about odds ratio, let me quickly define coefficient estimate. Coefficient estimate from logistic regression characterize the relationship between the predictor and the outcome on a log-odds scale. One-unit increase in a predictor (e.g. resting blood pressure rest\_bp) is associated with an increase in the log odds of the outcome (e.g. heart disease heart\_disease) by the value of the coefficient estimate (e.g. 0.0453).

broom::tidy(lr\_model$fit) %>% filter(term=="rest\_bp")

## # A tibble: 1 x 5

## term estimate std.error statistic p.value

##

## 1 rest\_bp 0.0453 0.0161 2.81 0.00492

**Odds ratio**

The odds ratio represents the odds that an outcome will occur given the presence of a specific predictor, compared to the odds of the outcome occurring in the absence of that predictor, assuming all other predictors remain constant.  
The odds ratio is calculated from the exponential function of the coefficient estimate based on a unit increase in the predictor *(on a side note, coefficient estimates are unbiased but odds ratio are biases due to transformation)*.  
An example with a continuous variable, will be for 1 mm Hg increased in resting blood pressure rest\_bp, the odds of having heart disease increases by a factor of 1.05.

Before we go onto to getting these estimates from a fitted model, here’s the R code that created that simulated data and the graphic:

library(tidyverse)

library(scales)

library(viridis)

set.seed(765)

n **<-** 100000

data **<-** data\_frame(

animal\_ **=** sample(**c**("lion", "tiger", "bear"), size **=** n, replace **=** **TRUE**, prob **=** **c**(0.2, 0.3, 0.5))

) **%>%**

mutate(diseased **=** case\_when(

animal\_ **==** "lion" **~** sample(0**:**1, size **=** n, replace **=** **TRUE**, prob **=** **c**(0.5, 0.5)),

animal\_ **==** "tiger" **~** sample(0**:**1, size **=** n, replace **=** **TRUE**, prob **=** **c**(0.75, 0.25)),

animal\_ **==** "bear" **~** sample(0**:**1, size **=** n, replace **=** **TRUE**, prob **=** **c**(0.9, 0.1))

))

true\_props **<-** data **%>%**

group\_by(animal\_) **%>%**

summarise(diseased **=** **round**(mean(diseased), 2)) **%>%**

mutate(odds **=** **c**("1:9", "1:1", "1:3"),

lab **=** paste0("Probability: ", diseased, "\nOdds: ", odds))

ggplot(data, aes(fill **=** **as.logical**(diseased), x **=** animal\_)) **+**

geom\_bar(position **=** "fill") **+**

geom\_text(data **=** true\_props, aes(label **=** lab, y **=** diseased), colour **=** "white") **+**

labs(x **=** "", y **=** "Proportion", fill **=** "Diseased or not:") **+**

guides(fill **=** guide\_legend(reverse **=** **TRUE**)) **+**

coord\_flip() **+**

ggtitle("Simulated data for demonstrating odds and risk ratios",

"Risk ratio: tiger / bear = 2.5, lion / bear = 5.0\nOdds ratio: tiger / bear = 3.0, lion / bear = 9.0")

Obtaining odds and risk ratios from a generalized linear model

Putting that aside, how do we get these estimates from a model?

Odds ratios

Frankly, I suspect the more material reason for the prevalence of odds ratios is that they fall easily out of the results of a logistic regression (generalized linear model with the canonical logit link function relating the mean of the response to the linear predictor - where the logit function is the logarithm of the odds). For a categorical explanatory variable that has been represented as dummy indicator variables, *e* to the power of the coefficient for the dummy variable will give an estimate of the odds ratio.

Here’s an example of extracting estimates of odds ratios from our data:

*# Odds ratio:*

model\_2 **<-** glm(diseased **~** animal\_, family **=** quasibinomial, data **=** data)

**exp**(coef(model\_2))[-1]

*# animal\_lion animal\_tiger*

*# 9.286664 2.997837*

#

These odds ratios are of the given animal (Lion or Tiger) relative to the disease rate of the reference level, which in this case is Bear. So these are estimates of the ratios depicted in the original diagram.

You would get the same point estimate if you used family = binomial, or family = quasi(link = 'logit', variance = "mu(1-mu)"); it’s the logit link that’s important here.

Note that these estimates are biased; despite being based on large samples of 100,000 animals they haven’t converged on the true odds ratios of 9 and 3. The estimates of the original coefficients are unbiased; but the non-linear transformation of exp(coef(model)) inevitably introduces bias. A statistic that is unbiased on one scale will be biased if you transform it to another scale; such is life.

Using binomial versus quasibinomial does make a difference to confidence intervals, but in our current case it’s not material. The confidence intervals at least contain the correct values; although the point estimate is biased (and also not in the centre of the confidence interval, again due to the non-linear transformation), the confidence interval is still valid for the transformed scale:

**exp**(confint(model\_2))[-1, ]

*# 2.5 % 97.5 %*

*# animal\_lion 8.920358 9.669207*

*# animal\_tiger 2.881889 3.118685*

#

Risk ratios

At a minimum, the *only* change that needs to be done to get risk ratios is to change the link function that relates the mean value of the response variable to the linear predictor. For estimates of odds ratios, this is logit (ie the logarithm of the odds of the mean); for estimates of relative risk ratios, this becomes logarithm. We can specify this manually, or just use a built-in family for our generalized linear model for which the logarithm is the canonical link fucntion, and hence the default. The quasipoisson is a great choice here:

*# Risk ratio:*

model\_1 **<-** glm(diseased **~** animal\_, family **=** quasipoisson, data **=** data)

**exp**(coef(model\_1))[-1]

*# animal\_lion animal\_tiger*

*# 5.112507 2.504792*

#

Again, these point estimates are biased because of the non-linear exp() transformation of the unbiased original coefficients. But the confidence intervals contain the correct values (which recall are 5.0 and 2.5):

**exp**(confint(model\_1))[-1, ]

*# 2.5 % 97.5 %*

*# animal\_lion 4.961963 5.268162*

*# animal\_tiger 2.426520 2.585758*

#

If we’d used poisson as our family instead of quasipoisson, the dispersion parameter is forced to be 1.0, instead of 0.7 which it is estimated to be by the quasipoisson model. The net effect of that “underdispersion” in the data in this case is that the confidence intervals with family = poisson are *larger* than in the quasipoisson instance. This won’t always apply, but it often will when we are modelling data such as this which is, in fact under-dispersed compared to a pure poisson distribution (because none of the counts can exceed 1).

These examples have avoided complications of other explanatory variables, but the point of using a generalized linear model for this (rather than observing proportions directly) is that we can add in other variables and report on relative risks (or odds, if we must) “while controlling for …” those other variables.

broom::tidy(lr\_model$fit) %>% filter(term=="rest\_bp") %>% mutate(odds\_ratio= exp(estimate)) %>% select(term, estimate, odds\_ratio)

## # A tibble: 1 x 3

## term estimate odds\_ratio

##

## 1 rest\_bp 0.0453 1.05

An example with a categorical variable will be chest pain during exercise stress test ex\_cpyes. If chest pain is present, the odds of having heart disease increases by a factor of 1.52. In percentage change, the odds for heart disease is 52.6% higher for individuals with chest pain during the exercise stress test than individuals with no chest pain.

broom::tidy(lr\_model$fit) %>% filter(term=="ex\_cpyes") %>% mutate(odds\_ratio= exp(estimate), percentage= (odds\_ratio-1)\*100) %>% select(term, estimate, odds\_ratio, percentage)

## # A tibble: 1 x 4

## term estimate odds\_ratio percentage

##

## 1 ex\_cpyes 0.437 1.55 54.8

Variables such as cholesterol chol where the odds ratio is 1 means that cholesterol does not affect odds of having heart disease.

broom::tidy(lr\_model$fit) %>% mutate(odds\_ratio= round(exp(estimate),2)) %>% select(term, estimate, odds\_ratio) %>% filter(odds\_ratio==1) #https://stackoverflow.com/questions/21509346/r-displays-numbers-in-scientific-notation

## # A tibble: 1 x 3

## term estimate odds\_ratio

##

## 1 chol 0.00447 1

Variables such as age, normal resting ECG rest\_ecgnormal have odds ratio less than 1 which suggests that exposure with these variables are associated with lower odds of having heart disease. We tend to ignore the intercept when talking about odds ratio.

broom::tidy(lr\_model$fit) %>% mutate(odds\_ratio= round(exp(estimate),2)) %>% select(term, estimate, odds\_ratio) %>% filter(odds\_ratio<1)

## # A tibble: 10 x 3

## term estimate odds\_ratio

##

## 1 (Intercept) -0.750 0.47

## 2 age -0.0226 0.98

## 3 rest\_cpatypical -2.13 0.12

## 4 rest\_cpnon-CP pain -1.86 0.16

## 5 rest\_cptypical -4.18 0.02

## 6 fast\_bloodsugar>120 -0.667 0.51

## 7 rest\_ecgnormal -0.779 0.46

## 8 ex\_maxHR -0.0471 0.95

## 9 ex\_STpeakup/norm -0.183 0.83

## 10 thalassemianorm -0.201 0.82

Variables such as reversable thalassemia thalassemiareversable and resting blood pressure rest\_bp have odds ratio greater than 1 which suggest exposure to these variables are associated with higher odds of heart disease.  
ST-T abnormal during resting ECG rest\_ecgST-T abnorm, 2 vessels coloured during angiogram coloured\_vessels2.0 and males sexmale have unusually large odds ratio. Our pre-processing was rather brief, only imputation of missing values was done.

broom::tidy(lr\_model$fit) %>% mutate(odds\_ratio= round(exp(estimate),2)) %>% select(term, estimate, odds\_ratio) %>% filter(odds\_ratio>1) %>% arrange(desc(odds\_ratio))

## # A tibble: 10 x 3

## term estimate odds\_ratio

##

## 1 rest\_ecgST-T abnorm 13.1 465136.

## 2 coloured\_vessels2.0 4.68 107.

## 3 sexmale 2.48 12.0

## 4 coloured\_vessels3.0 1.62 5.04

## 5 coloured\_vessels1.0 1.52 4.59

## 6 thalassemiareversable 1.26 3.53

## 7 ex\_cpyes 0.437 1.55

## 8 ex\_STdepression\_dur 0.296 1.34

## 9 rest\_bp 0.0453 1.05

## 10 ex\_STpeakflat 0.0512 1.05

**Conclusion**

In this post we looked at explaining models using white box models such as logistic regression.