Multiple Logistic Regression - Review

We are trying to predict which restaurants ended up in a Michelin guide to NYC based on their corresponding entries in Zagat's Guide.

```
mydata <- read.csv("https://awagaman.people.amherst.edu/stat495/MichelinNY.csv", header = T)
```

Steps in the Data Analysis Process

What should you do when given this analysis task?

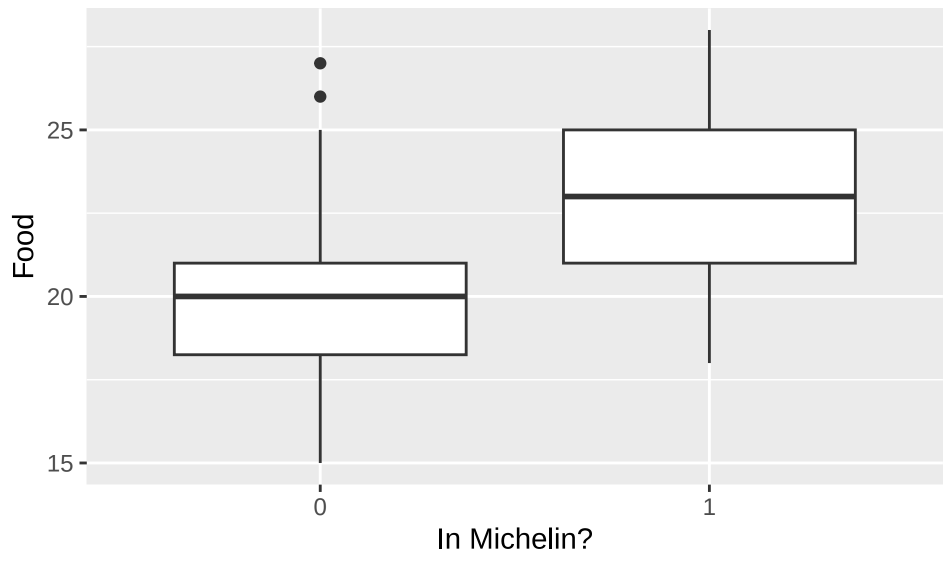
Let's Look at the Data

```
glimpse(mydata)
```

```
## Rows: 164
## Columns: 6
## $ InMichelin      <int> 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, ~
## $ Restaurant.Name <chr> "14 Wall Street", "212", "26 Seats", "44", "A", "A.O.C~
## $ Food            <int> 19, 17, 23, 19, 23, 18, 24, 23, 27, 20, 25, 23, 23, 27~
## $ Decor           <int> 20, 17, 17, 23, 12, 17, 21, 22, 27, 17, 26, 20, 27, 25~
## $ Service         <int> 19, 16, 21, 16, 19, 17, 22, 21, 27, 19, 27, 20, 23, 27~
## $ Price           <int> 50, 43, 35, 52, 24, 36, 51, 61, 179, 42, 71, 50, 82, 9~
```

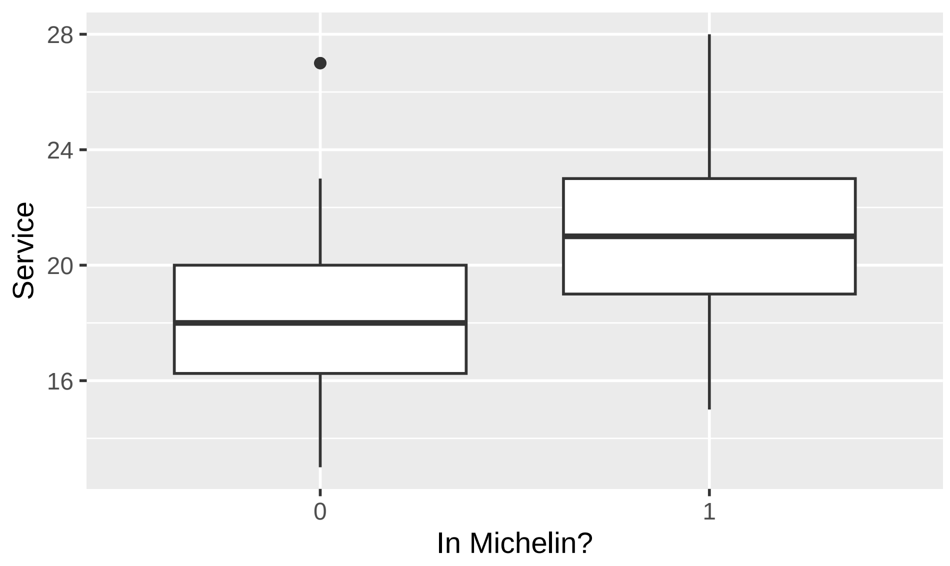
Other visualizations should be undertaken to understand the variables in the data set and their relationships. Here are examples looking at the relationships between potential predictors and our intended response variable.

```
ggplot(mydata, aes(x = as.factor(InMichelin), y = Food)) +
  geom_boxplot() +
  labs(x = "In Michelin?")
```



What does this plot suggest about whether Food should be included in the model at a first pass?

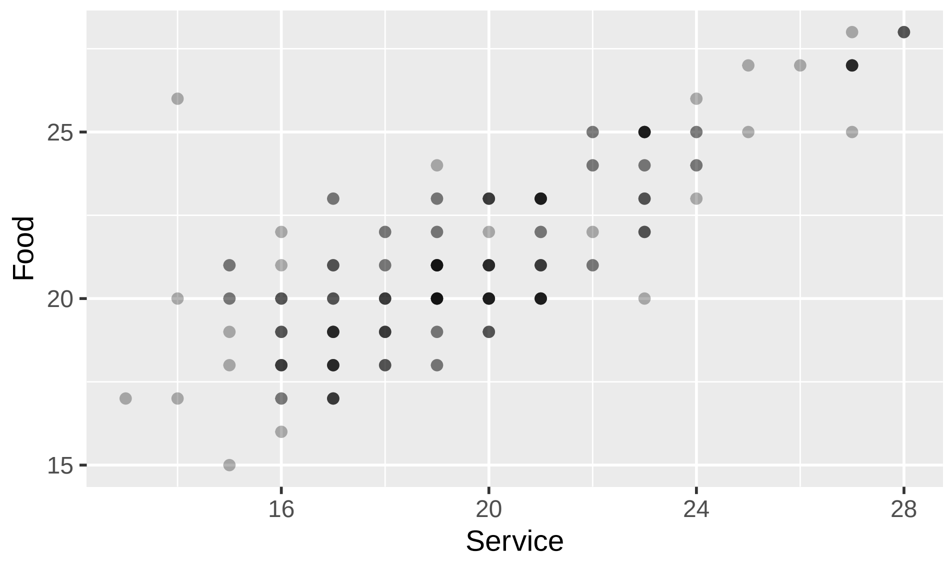
```
ggplot(mydata, aes(x = as.factor(InMichelin), y = Service)) +
  geom_boxplot() +
  labs(x = "In Michelin?")
```



What does this plot suggest about whether Service should be included in the model at a first pass?

Are there other plots we should be making?

```r
ggplot(mydata, aes(x = Service, y = Food)) +
  geom_point(alpha = 0.3)
```



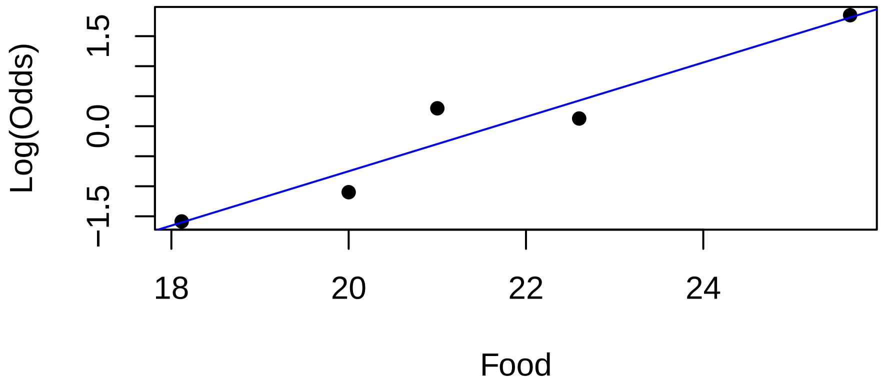
Would this plot raise any concerns for you? What concerns?

Conditions for Logistic Regression

Do you recall the conditions for logistic regression?
Write down the logistic model.
Where are the errors in your model?

You can make empirical logit plots using the function provided (requires a numeric response) in the Stat2Data package. Remember, you can change the number of breaks.

```
emplogitplot1(InMichelin ~ Food, ngroups = 5, data = mydata)
```



We can fit a full model easily. In order to demonstrate nested drop in deviance procedures, I added some interactions and transformed price based on modelling performed previously.

```
# Add log Price
mydata <- mutate(mydata, logPrice = log(Price))

#Fit model and get basic output
logm <- glm(InMichelin ~ Food + Decor + Service + Price + Food:Decor +
              Decor:Service + logPrice, data = mydata, family = binomial(logit))
msummary(logm)
```

```
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -77.05924   17.67329  -4.360  1.3e-05 ***
## Food            1.34399    0.92446   1.454 0.146002
## Decor           1.62489    0.65886   2.466 0.013655 *
## Service         0.53999    0.69811   0.774 0.439226
## Price          -0.06889    0.04627  -1.489 0.136503
## logPrice       10.80872    3.23136   3.345 0.000823 ***
## Food:Decor     -0.03562    0.04764  -0.748 0.454618
## Decor:Service  -0.04447    0.03714  -1.197 0.231220
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 225.79  on 163  degrees of freedom
## Residual deviance: 129.27  on 156  degrees of freedom
## AIC: 145.27
##
## Number of Fisher Scoring iterations: 6
```

```
# Show other tests and output
lrtest(logm)
```

```
## Likelihood ratio test
##
```

```
## Model 1: InMichelin ~ Food + Decor + Service + Price + Food:Decor + Decor:Service +
##     logPrice
## Model 2: InMichelin ~ 1
##   #Df   LogLik Df  Chisq Pr(>Chisq)
## 1   8  -64.636
## 2   1 -112.894 -7 96.517  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
exp(confint(logm))
```

```
## Waiting for profiling to be done...
```

```
##                      2.5 %       97.5 %
## (Intercept)   2.463306e-50 6.504107e-20
## Food          6.104193e-01 2.331060e+01
## Decor         1.374927e+00 1.939653e+01
## Service       4.511617e-01 7.078732e+00
## Price         8.664931e-01 1.095850e+00
## logPrice      2.101735e+01 3.250296e+07
## Food:Decor    8.803254e-01 1.062476e+00
## Decor:Service 8.858639e-01 1.025985e+00
```

```
logmaugment <- augment(logm, type.predict = "response")
```

For a nested drop in deviance procedure, we need a second reduced model. The challenge you can encounter here is that if you fit the specified reduced model on the data set, you may end up with more observations than in the full model depending on what your reduced model is because some predictors have some *missing values*, which were removed automatically by R in the model fitting process. While this does not occur here, it is important to note because you want to compare models on the *same* data. We can get around this (potential issue) by fitting our chosen reduced model on the augmented data set from the full model. That way, we will end up comparing models fit on the SAME data. Let's try to drop Food:Decor and Price from the model.

```
logm2 <- glm(InMichelin ~ Food + Decor + Service + Decor:Service + logPrice,
             data = logmaugment, family = binomial(logit))
msummary(logm2)
```

```
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)    
## (Intercept)   -63.76436   14.09848  -4.523 6.10e-06 ***
## Food            0.64274    0.17825   3.606 0.000311 ***
## Decor           1.50597    0.47883   3.145 0.001660 ** 
## Service         1.12633    0.47068   2.393 0.016711 *  
## logPrice        7.29827    1.81062   4.031 5.56e-05 ***
## Decor:Service  -0.07613    0.02448  -3.110 0.001873 ** 
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 225.79  on 163  degrees of freedom
## Residual deviance: 131.23  on 158  degrees of freedom
## AIC: 143.23
##
## Number of Fisher Scoring iterations: 6
```

```
G <- 131.23 - 129.27; G
```

```
## [1] 1.96
```

```
lrtdf <- 158 - 156 ; lrtdf
```

```
## [1] 2
```

```
pchisq(G, df = lrtdf, lower.tail = FALSE)
```

```
## [1] 0.3753111
```

You can also have the computer do these computations with:

```
anova(logm2, logm, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: InMichelin ~ Food + Decor + Service + Decor:Service + logPrice
## Model 2: InMichelin ~ Food + Decor + Service + Price + Food:Decor + Decor:Service +
##     logPrice
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       158     131.23
## 2       156     129.27  2   1.9568   0.3759
```

The reduced model should come first in this command.

This suggests (assuming we believe conditions are met) we can use the smaller reduced model. How well is that model doing? Well, we can look at something called a confusion matrix to get a sense of that, but we need to look at predictions to construct it.

To wrap up this section, we examine the predictions a bit more closely.

```
logm2augment <- logm2 %>% augment(type.predict = "response")
names(logm2augment)
```

```
##  [1] "InMichelin" "Food"       "Decor"      "Service"    "logPrice"
##  [6] ".fitted"    ".resid"     ".hat"       ".sigma"     ".cooksd"
## [11] ".std.resid"
```

```
head(logm2augment)
```

```
## # A tibble: 6 x 11
##   InMichelin  Food Decor Service logPrice .fitted .resid   .hat .sigma   .cooksd
##        <int> <int> <int>   <int>    <dbl>   <dbl>  <dbl>  <dbl>  <dbl>     <dbl>
## 1          0    19    20      19     3.91  0.398  -1.01  0.0350  0.911 0.00415
## 2          0    17    17      16     3.76  0.0778 -0.402 0.0260  0.914 0.000385
## 3          0    23    17      21     3.56  0.277  -0.806 0.0636  0.912 0.00463
## 4          1    19    23      16     3.95  0.873   0.521 0.0616  0.913 0.00170
## 5          0    23    12      19     3.18  0.0248 -0.224 0.0252  0.914 0.000112
## 6          0    18    17      17     3.58  0.0358 -0.270 0.0117  0.914 0.0000743
## # i 1 more variable: .std.resid <dbl>
```

If you examine the augmented data set, since we asked for the “response” type of prediction, our .fitted values are the probability estimated for each restaurant to have ended up in the Michelin guide.

```
favstats(~ .fitted, data = logm2augment)
```

```
##          min        Q1   median        Q3       max      mean        sd   n
##  0.001065475 0.1128282 0.410854 0.8204368 0.9994896 0.4512195 0.3463045 164
##  missing
##        0
```

If we wanted to make binary predictions for each restaurant, we could round to the nearest integer (i.e., any probability over 0.50 would imply the restaurant would be predicted to be in the Michelin guide).

```
logm2augment <- mutate(logm2augment, binprediction = round(.fitted, 0))
tally(~ binprediction, data = logm2augment)
```

```
## binprediction
##  0  1
## 96 68
```

Hmm. 41% of restaurants were implied to be in the guide, based on using the 50% cutoff.

```
68/164
```

```
## [1] 0.4146341
```

How does that reflect reality?

```
tally(~ InMichelin, data = logm2augment)
```

```
## InMichelin
##  0  1
## 90 74
```

```
74/164
```

```
## [1] 0.4512195
```

Really, about 45% ended up in the guide, so at least we aren't predicting a very different value from that.

It is possible to find the fitted values so skewed that say, none of them are over 50%, or maybe only 8% are over 50% but the data shows 20% in that group. We can use the data's percentage to alter our cutoff (this is done more appropriately with a training/test data set, but here it is for illustration).

If we made no adjustment, this is what we would get:

```
with(logm2augment, table(InMichelin, binprediction))
```

```
##           binprediction
## InMichelin  0  1
##          0 79 11
##          1 17 57
```

```
correct <- (79+57)/164; correct
```

```
## [1] 0.8292683
```

We are almost 83% correct using the 50% cutoff. Does adjusting the fraction in the guide up help? We know that roughly 45% were in the guide, and that means 55% were not.

```
with(logm2augment, quantile(.fitted, 1-0.45)) #get 55th quantile
```

```
##       55%
## 0.4655586
```

```
# Split predictions based on quantile, not just using 0.5
logm2augment <- mutate(logm2augment, binprediction2 = as.numeric(.fitted > 0.4655586))
tally(~ binprediction2, data = logm2augment)
```

```
## binprediction2
##  0  1
## 90 74
```

Now the confusion matrix looks like:

```r
with(logm2augment, table(InMichelin, binprediction2))
```

```
##            binprediction2
## InMichelin  0  1
##           0 79 11
##           1 11 63
```

```r
correct2 <- (79+63)/164; correct2
```

```
## [1] 0.8658537
```

We improved a little bit here - up to 86.5 percent accuracy. If the discrepancy between fraction predicted to be in group 1 versus the observed fraction is greater, you might improve more using a similar adjustment.

How else can we check how the model is doing? We can check concordance. Concordance looks at the observations in the data set in success-failure pairs. For each pair, it looks to see if the success observation has a higher predicted probability of being a success than the failure observation. If yes, the pair is called concordant. Otherwise, it can be discordant, or there can be a tie.

If the model is like a coin toss, concordance will be around 50%. Better fitting models have concordance values that are larger (highest is 100%).

```r
#***FUNCTION TO CALCULATE CONCORDANCE AND DISCORDANCE***#
Association <- function(model)
{
  Con_Dis_Data <- cbind(model$y, model$fitted.values)
  ones <- Con_Dis_Data[Con_Dis_Data[, 1] == 1, ]
  zeros <- Con_Dis_Data[Con_Dis_Data[, 1] == 0, ]
  conc <- matrix(0, dim(zeros)[1], dim(ones)[1])
  disc <- matrix(0, dim(zeros)[1], dim(ones)[1])
  ties <- matrix(0, dim(zeros)[1], dim(ones)[1])
    for (j in 1:dim(zeros)[1])
    {
      for (i in 1:dim(ones)[1])
      {
        if (ones[i, 2] > zeros[j, 2])
        {conc[j, i] = 1}
        else if (ones[i, 2] < zeros[j, 2])
        {disc[j, i] = 1}
        else if (ones[i, 2] == zeros[j, 2])
        {ties[j, i] = 1}
      }
    }
  Pairs <- dim(zeros)[1]*dim(ones)[1]
  PercentConcordance <- (sum(conc)/Pairs)*100
  PercentDiscordance <- (sum(disc)/Pairs)*100
  PercentTied <- (sum(ties)/Pairs)*100
  return(list("Percent Concordance" = PercentConcordance,
              "Percent Discordance" = PercentDiscordance,
              "Percent Tied" = PercentTied, "Pairs" = Pairs))
}
#***FUNCTION TO CALCULATE CONCORDANCE AND DISCORDANCE ENDS***#
```

This function will calculate concordance and discordance values. Depending on the number of observations, this computation can take a while. Here, with 164 observations, it is fairly fast.

```r
Association(logm2)
```

```
## $`Percent Concordance`
## [1] 90.15015
##
## $`Percent Discordance`
## [1] 9.84985
##
## $`Percent Tied`
## [1] 0
##
## $Pairs
## [1] 6660
```

The model concordance is 90%. We'd get 50% with a coin flip. This indicates the model fits very well.

There are also pseudo-Rsquared type statistics out there that can be used to help assess performance for this type of model. These all have different ranges, etc. so you should explore the help documentation to understand what you are seeing.

```r
PseudoR2(logm2, "all")
```

```
##        McFadden     McFaddenAdj        CoxSnell      Nagelkerke   AldrichNelson
##       0.4187987       0.3656517       0.4381867       0.5861204       0.3657180
## VeallZimmermann           Efron McKelveyZavoina            Tjur             AIC
##       0.6313544       0.4997682       0.6597938       0.4905660     143.2287550
##             BIC          logLik         logLik0              G2
##     161.8279535     -65.6143775    -112.8944069      94.5600589
```