Plot of Odds of Ration and Checking for the accuracy of logistic models though creation of confusion matrix and ROC curves and Multiple Cross Validation

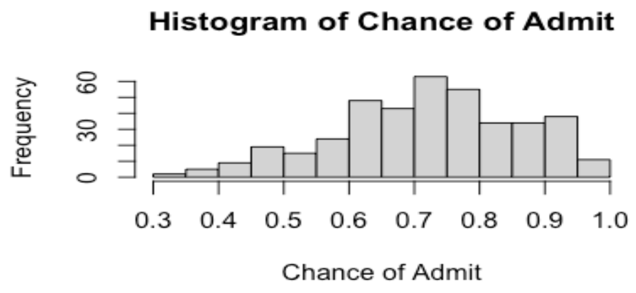
Professor Esfandiari

I Plot of odds Ratios – Best summary for clients

Logistic regression is a methodology that is used quite often in different disciplines. The major thing that we have to think about is what would be the best way to present the findings to the clients; who are most non-statisticians. Based on my experience in the world of consulting, the best way to do this is the table of odds, as well as the plot of odds ratios.

Using the admit data set from Kaggle, I will give you an example.

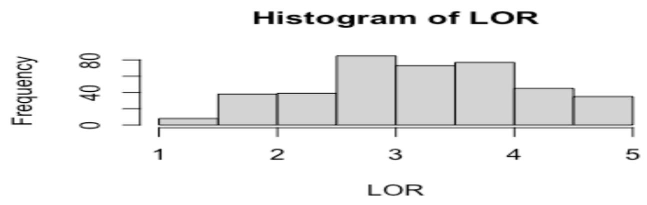
Research question: Can chance of admission to graduate school be predicted from GRE scores, research, and letter of recommendation?



We need to change chance of admit to binary

```
> admit<-cut(`Chance of Admit`,br=c(0.34,0.73,0.98),labels=c("low","high"))
> table(admit)

admit
 low         high
 207         191
> table(admit,Research)
        Research
admit    0      1
  low   147     60
  high  32      159
```



```
> summary(LOR)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   3.000   3.500   3.453   4.000   5.000
> m1<-glm(admit~`GRE Score`+Research+LOR+family="binomial")
> summary(m1)

> m1<-glm(admit~`GRE Score`+Research+LOR,family="binomial")
> summary(m1)

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)          -64.85210    8.10704  -7.999 1.25e-15 ***
`GRE Score`            0.18914    0.02543   7.438 1.02e-13 ***
Research               0.97144    0.33908   2.865  0.00417 **
LOR                    1.21330    0.22940   5.289 1.23e-07 ***
---

    Null deviance: 551.10  on 397  degrees of freedom
Residual deviance: 251.12  on 394  degrees of freedom
  (2 observations deleted due to missingness)
AIC: 259.12

Number of Fisher Scoring iterations: 6
```

```
> exp(coef(m1))
 (Intercept)       `GRE Score`           Research                LOR
6.840548e-29      1.208205e+00        2.641759e+00       3.364578e+00

> exp(confint(m1))
                2.5 %              97.5 %
(Intercept)     2.830538e-36       1.986608e-22
`GRE Score`     1.152859e+00       1.274098e+00
Research        1.357179e+00        5.150069e+00
LOR              2.181752e+00       5.383018e+00
```

The best way to summarize the results for the client would be to make a summary table of the odds ratios and the 95% confidence interval for the odds ratio of each predictor.

Table of odds and 95% odds ratio for the model predicting admission to graduate school as a function of GRE scores, research, and letter of recommendation (LOR).

Predictor	Odds ratio	95% confidence interval	p-value
GRE score	1.21	1.15 - 1.27	0.000
Research	2.64	1.36 - 5.15	0.000
LOR	3.36	2.18 - 5.38	0.000

Interpretation
Keeping all else constant…

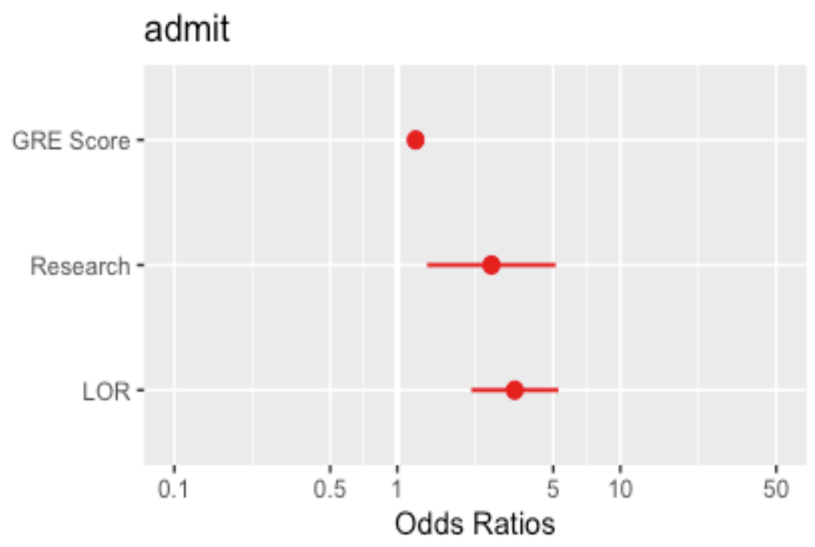
- For one point increase in GRE scores, the odds admission increases by 21%.
- For candidates who do research, the odds of admission is 2.64 times higher.
- For one point increase in letter of recommendation the odds of admission increases 3.36 times.

As you notice all the p-values are less than 0.05 and none of the confidence intervals include one and they are all positive meaning that higher GRE scores, doing research, and stronger letters of recommendation increase the chance of admission to graduate school.

The best way to present this visually is to create plot of the odds. As you see from the following plot, all the confidence intervals are sitting to the right of one. The reason for the lack of a clear confidence interval for GRE is the wide range for GRE scores and the fact that the mode tells us what happens to the odds of admission as GRE increases by one unit.

```
> summary(`GRE Score`)
  Min.    1st Qu.    Median    Mean    3rd Qu.    Max.
 290.0    308.0      317.0     316.8    325.0     340.0
```

R codes
>library(sjPlot)
>Plot_model(m1)



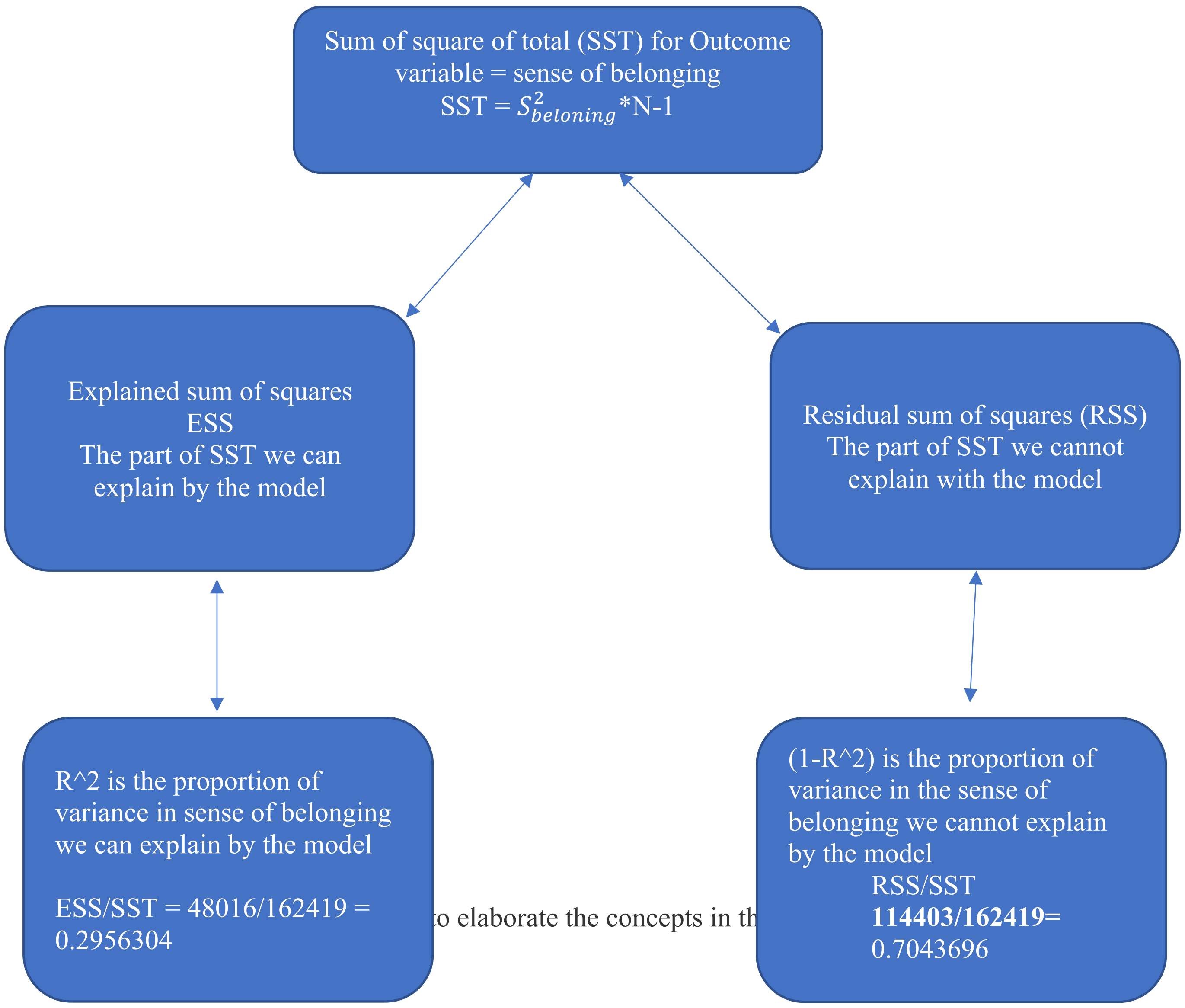
If we exponentiate the log of odds associated with GRE by ten times, then we see that for five points increase in GRE scores the odds of admission increases 2.57 times (almost as high as doing research).

```
> exp(0.18914 *5)
[1] 2.574615
```

II Checking for the accuracy of logistic models though creation of confusion matrix and ROC curve and Multiple Cross Validation

As it was elaborated in the prior lecture on logistic regression, we cannot check the accuracy of logistic models by calculation of R-squared. R-squared is used for estimating the accuracy of linear regression models and it is a measure of the proportion of the sum of squares explained by the model divided by the sum of the square of the total. Conceptually, we can explain this by the following schematic.

Let us assume that we want to predict students' perception of sense of belonging to our campus from academic self-confidence, socio-economic struggle, and whether student is transfer from community college. R-squared would be the proportion of variance that we can explain by the predictors in the model including, academic self-confidence, socio-economic, struggle, and GPA. This can be depicted in the following schematic.


Sum of square of total (SST) for Outcome variable = sense of belonging
SST = $S^2_{beloning}$*N-1
Explained sum of squares ESS The part of SST we can explain by the model
Residual sum of squares (RSS) The part of SST we cannot explain with the model
R^2 is the proportion of variance in sense of belonging we can explain by the model
ESS/SST = 48016/162419 = 0.2956304
(1-R^2) is the proportion of variance in the sense of belonging we cannot explain by the model
RSS/SST
114403/162419=
0.7043696

o elaborate the concepts in th

We can use ANOVA to calculate sum of square of each factor and then using these sums of squares to calculate R^2 and (1-R^2)..

```
> m1=aov(Belonging~Academic+SES+Transfer)
> summary(m1)
              Df    Sum Sq      Mean Sq     F value     Pr(>F)
Academic      1     45726       45726       327.348     < 2e-16 ***
SES           1     1405        1405        10.058       0.00157 **
Transfer       1     885        885         6.332       0.01205 *
Residuals     819   114403      140
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
43 observations deleted due to missingness
```

TSS = SS(academic) + SS (transfer) + SS(SES) + SS (Residual) = 45726+1405+885+114403 = **162419**

ESS = SS(academic) + SS (SES) + SS(Transfer) = 45726+1405+885 = **48016**
RSS = 114403

If we run regression, we get R^2 for the whole model.

```
> m1<-lm(Belonging~Academic+SES+Transfer)
> summary(m1)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 36.80199    2.16245  17.019  < 2e-16 ***
Academic     0.47985    0.03064  15.659  < 2e-16 ***
SES         -0.05149    0.01720  -2.994  0.00284 **
TransferYes -2.62572    1.04345  -2.516  0.01205 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 11.82 on 819 degrees of freedom
  (43 observations deleted due to missingness)
Multiple R-squared:  0.2956,          Adjusted R-squared:  0.293
F-statistic: 114.6 on 3 and 819 DF,  p-value: < 2.2e-16
```

In multiple linear regression, mathematically
TSS = RSS + ESS = 162419
Total sum of squares = Residual sum of squares + ESS (or sum of square of regression)

Mathematically, the above does not hold true for logistic regression because the outcome is binary and not numerical, and it is measured in terms of log of odds.

In multiple linear regression the equation for the outcome variable is as follows:

$$\boldsymbol{Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_j X_j}$$

In logistic regression, the equation for the outcome variable is displayed as follows:

$$\log\left(\frac{sucess}{failure}\right) = \text{Log}\left(\frac{\theta}{1-\theta}\right) = \boldsymbol{\beta_0 + \beta_1 X_1 + \cdots + \beta_j X_j}$$

Now let us use the same variables and run logistic regression. In order to do that, we need to transform the outcome variable of sense of belonging to a binary factor.

```
> summary(Belonging)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  11.10   50.00   61.10   60.11   69.40   97.20      28
> belongingcat<-cut(Belonging,br=c(11.1,61.1,97.2),right=FALSE)
> table(belongingcat)
belongingcat
[11.1,61.1) [61.1,97.2)
        411         426

> library(car)
> belongingcat<-recode(belongingcat,"'[11.1,61.1)'='0';'[61.1,97.2)'='1'")
> table(belongingcat)
belongingcat
  0                1
411              426
```

Logistic model
> m2<-glm(belongingcat~Academic+SES+Transfer,family="binomial")
> summary(m2)

```
Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)            -3.182182   0.445273  -7.147  8.9e-13 ***
Academic                0.068787   0.006857  10.031  < 2e-16 ***
SES                    -0.008877   0.003247  -2.734  0.00626 **
TransferYes            -0.680335   0.200011  -3.401  0.00067 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
    Null deviance: 1139.36  on 821  degrees of freedom
Residual deviance:  949.95  on 818  degrees of freedom
  (44 observations deleted due to missingness)
AIC: 957.95
```

As we discussed before…
Null Deviance = 1139.36 (residual for the intercept only model)
Residual deviance = 949.95 (residual for the model with three predictors)
Thus, deviance or residual decreases (1139.36 - 949.95 = 189.41)

Pseudo R-squared can be calculated as an estimate of the strength of the model in logistic regression. However, since the mathematical underpinning of multiple linear regression and logistic regression are different; pseudo-R-squared is not interpretable the same way like multiple regression.

pseudo-R-squared = $1 - \frac{residual\ deviance}{null\ deviance}$ = 1 – (949.95/1139.36) = 0.16624

We cannot say that based on the logistic model, 16.62% of the variance in the odds for perception of belonging to UCLA is explained by academic confidence, socio-economic struggle and being transfer from community college. This is mathematically incorrect.

The strength of the logistic models need to be calculated by estimation of accuracy. This is explained below.

III Estimating the strength of the logistic models through confusion matrix

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

We will start with an **example confusion matrix for a binary classifier**. In the following you are given the data for 200 patients that have diabetic. Based on a logistic regression model, we have made predictions. The following table presents the actual values and the predicted values.

Actual	Predicted		Colum totals
	No	Yes	
No	60 TRUE NEGATIVE	12 FALSE POSITIVE TYPE I ERROR	72
Yes	8 FALSE NEGATIVE TYPE II ERROR	120 TRUE POSITIVE	128
Row totals	68	132	200

Based on the prediction model,
Of the two hundred patients, 132 were classified as having the disease and 68 were classified as not having the disease.

Based on the actual results,
Of the two hundred patients, 128 actually had diabetic II and 72 did not.

We will now define, the different cells of the confusion matrix
TRUE POSITIVE
True positive are the cases in which the patient actually has the disease and the model also predicts that the patient has the disease.

FALSE POSITIVE
False positive are the cases in which the patient actually has the disease, but, the model predicts that the patient does has the disease. **This is known as TYPE I ERROR.**

FALSE NEGATIVE
False negative are the cases in which the patient actually does not have the disease, but, the model predicts that the patient has the disease. **THIS IS KNOWN AS TYPE II ERROR.**

TRUE POSITIVE
False positive are the cases in which the patient actually has the disease and the model predicts that the patient has the disease.
Other Concepts Underlying the Model

ACCURACY

Overall, how well did the prediction based on the logistic model work?

Accuracy $= \frac{\boldsymbol{True\ positive + Ture\ negative}}{\boldsymbol{Total\ sample}} = \frac{\mathbf{120+60}}{\mathbf{200}}$ **= 180/200 = 0.90**

This means that in 90% of the cases the prediction based on the logistic model makes the correct classification.

MISCLASSIFICATION RATE – ERROR RATE
Overall, how often is the classification done by the prediction model wrong?

Misclassification Rate $= \frac{\boldsymbol{false\ positive + false\ negative}}{\boldsymbol{Total\ sample}} = \frac{\mathbf{12+8}}{\mathbf{200}}$ **= 20/200 = 0.10**

Misclassification Rate = 1- accuracy rate = 1 – 0.90 = 0.10

This means that in 10% of the cases the prediction based on the logistic model makes the incorrect classification.

TRUE POSITIVE RATE – SENSITIVITY RATE
True positive rate: When the patient "**actually has diabetic II**", and the **model also predicts "yes".**

=TP/actual yes
= 120/128 = 0.9375

FALSE POSITIVE RATE – SENSITIVITY RATE
False positive rate: When the patient "**does not have diabetic II**", and **model predicts** "**yes**".
=FP/Actual No
= 12/72 = 0.167

- **Specificity:** When it's actually no, how often does it predict no?
  Specificity = $\frac{\boldsymbol{True\ No}}{\boldsymbol{Actual\ No}}$ **= 60/72 = 0.833**

  equivalent to 1 minus False Positive Rate = 1- 0.167 = 0.833

- **Precision:** When it predicts yes, how often is it correct?
  Precision = $\frac{\boldsymbol{True\ positive}}{\boldsymbol{Predicted\ yes}}$ **= 120/132 = 0.9090909**

- **Prevalence:** How often does the yes condition actually occur in our sample?
  actual yes/total = 128/200 = 0.64

Calculation of the Confusion Matrix using R

```
>library(nnet)
> m3<-multinom(belongingcat~Academic+SES+Transfer,stemfeb20)
# weights:  5 (4 variable)
initial  value 569.766982
final  value 474.976703
converged
> p<-predict(m3,belongingcat)
> tab<-table(p,belongingcat)
> tab
   belongingcat
p          0            1
  0        271          125
  1        134          292
```

Predicted values	Actual Value		Row totals
	Low sense of belonging	High sense of belonging	
Low sense of belonging	271 True negative	125 False Positive	396
High sense of beloning	134 False Negative	293 True positive	427
Column Total	405	418	823

Accuracy of our model = (true negative + true positive)/Total number of observations
Accuracy of our model = (271+293)/823 = 564/823 = **0.6852977**
Accuracy of our model is about 69%. This implies that in 69% of the cases we predict low and high sense of belonging correctly.

An example of using logistic regression and calculation of confusion matrix in medicine

Using the diabetic data, we will create a logistic regression model for the prediction of having diabetic type II from age, smoking, and hypertension.

```
> m1<-glm(Diabetes.new~Age+SmokingStatus_NISTCode+HypertensionDX,
family="binomial")
> summary(m1)


Coefficients:
                                    Estimate          Std. Error z value  Pr(>|z|)
(Intercept)                          -3.380674        0.174617 -19.361   < 2e-16 ***
Age                                  0.025801         0.002736   9.430   < 2e-16 ***
SmokingStatus_NISTCodeFORMER -0.355067                0.085528  -4.151   3.3e-05 ***
SmokingStatus_NISTCodeTRUE       -0.334447             0.130085  -2.571    0.0101 *
HypertensionDXyes                   1.129735          0.090237  12.520   < 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: 4404.9  on 4402  degrees of freedom
Residual deviance: 3953.5  on 4398  degrees of freedom
  (5545 observations deleted due to missingness)
AIC: 3963.5

Number of Fisher Scoring iterations: 5
```

We will now use R to create the confusion matrix

R Codes

```
> library(nnet)
> m1<-
multinom(Diabetes.new~Age+HypertensionDX+SmokingStatus_NISTCode,diabetic)

# weights:  6 (5 variable)
initial  value 3051.927036
iter  10 value 1976.747412
final  value 1976.731777
converged
```

R codes for the creation of the confusion matrix continued

```
> p<-predict(m1,diabetic)
> tab<-table(p,diabetic$Diabetes.new)
> tab
```

```
  p      0             1
 0     3517          876
 1     6             4
```

Predicted values	Actual Value		Row totals
	Does not have diabetic	Has diabetic	
Does not have diabetic	3517 True negative	876 False Positive	4393
Has Diabetic	6 False Negative	4 True positive	10
Column Total	3523	880	4403

- **Accuracy:** Overall, how often is the classifier (predicted model) correct?
  (TP+TN)/total = (4+3517)/4403 = **0.799682**
- **Misclassification Rate or error rate:** Overall, how often is the classifier wrong?
  (FP+FN)/total = (876+6)/4403 = **0.200318**
  equivalent to 1 minus Accuracy
- **True Positive Rate:** When it's actually yes, how often does it predict yes?
  TP/actual yes = 4/880 = **0.004545455**
  also known as "Sensitivity"
- **False Positive Rate:** When it's actually no, how often does it predict yes?
  FP/actual no = 876/3523 = **0.2486517**
- **Specificity:** When it's actually no, how often does it predict no?
  TN/actual no = 3517/3523 **= 0.9982969**
  equivalent to 1 minus False Positive Rate
- **Precision:** When it predicts yes, how often is it correct?
  TP/predicted yes = 4/10 = **0.4**
- **Prevalence:** How often does the yes condition actually occur in our sample?
  actual yes/total = 880/4403 = **0.1998637**

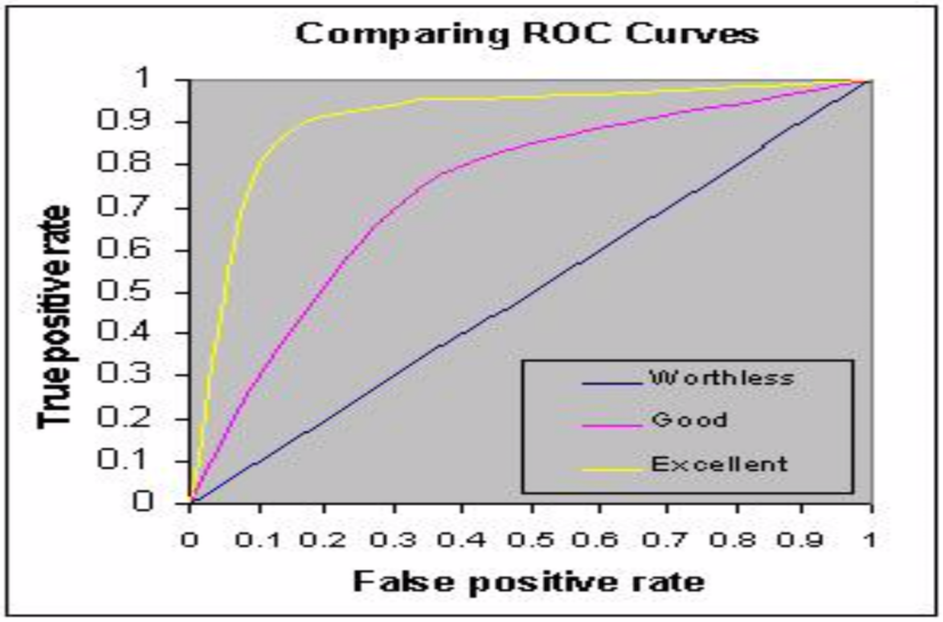
CONCLUSION: **Based on the confusion matrix given above, the accuracy rate is 80%. So, the predictive power of the model is good.** You want most of the points to be on the diagonal; that is true positive and true negative. The model seems to be doing well with respect to true negative but not so well with respect to true positive.

IV ROC curves (Receiver Operating Characteristic Curves) in Logistic Regression

A **ROC curve** (**receiver operating characteristic curve**) **is** a graph showing the performance of a classification model at all classification thresholds. **ROC curves** in **logistic regression** are used for determining **the** best cutoff value for predicting whether a new observation **is** a "failure" (0) or a "success" (1). The observed outcome in **logistic regression can** ONLY be 0 or 1. **The** predicted probabilities from **the model can** take on all possible values between 0 and 1.

The schematic below shows three ROC curves representing excellent, good, and worthless tests plotted on the same graph. The accuracy of the test depends on how well the test separates the group being tested into those with and without the disease in question. Accuracy is measured by the area under the ROC curve; referred to as AUC (Area Under Curve) in the statistics literature. An area of 1 represents a perfect test; an area of .5 represents a worthless test. AUC of 0.50 represents a prediction as good as guessing 50% success and 50% failure. This is represented by the area above and below the straight line. A rough guide for classifying the accuracy of a diagnostic test is the traditional academic point system:

- .90-1 = excellent (A)
- .80-.90 = good (B)
- .70-.80 = fair (C)
- .60-.70 = poor (D)
- .50-.60 = fail (F)



In the following we will present examples of …

- Using campus climate data to construct different confusion matrices based on multiple cutoffs. These cutoffs are determined based on the prediction scores resulting from cross validation.
- Using hsb2.csv to construct and interpret ROC curves resulting from cross validation.

Suppose that we want to predict the odds of planning to leave UCLA as a function of being first generation and our students' perception of satisfaction with academics and friendliness on our campus.

You need to take the following steps to construct the ROC curve and confusion matrix at the same time.

# Install library (caTools)

# Split the sample and call it split

```
>split<-sample.split(campusclimate$leaveUCLA,SplitRatio=0.65)
>train<-subset(campusclimate,split=T)
>test<-subset(campusclimate,split=F)
```

# Make a model and view the predicted results based on the model constructed

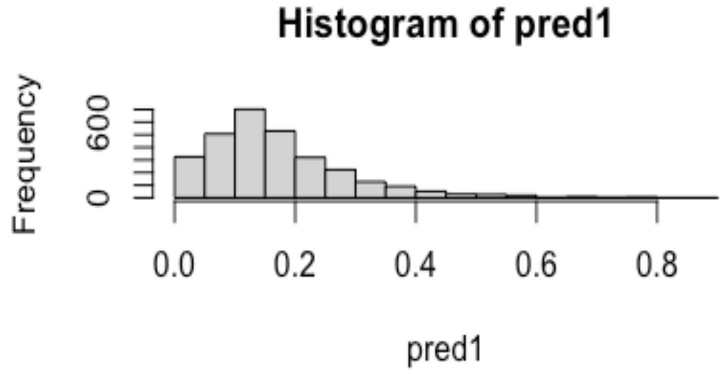
```
> m1<-glm(leaveUCLA~FIRSTGEN+academicenvp+friendlyenvp,family="binomial")
> pred1<-predict(m1,newdata=test,type="response")
> View(pred1)
```

Below is the sample of the predicted values we get as a result of the function (pred1)

2	0.06626226
3	NA
4	NA
5	0.07040798
6	0.01907223

# Make a histogram of the predicted scores

hist(pred1)



V Using cutoff from ROC curves to calculate accuracy via the construction of multiple confusion matrices

# Pick a threshold based on the histogram of predicted scores and create the confusion matrix. In the following, we have created a confusion matrix based on placing the threshold greater than 0.10

```
> table(test$leaveUCLA,pred1>0.1)

 predicted      FALSE (0)    TRUE (1- actual values)
 no   (0)        781          1678
 yes  (1)       50            464
```

calculation of accuracy

```
> (781+ 464)/(781+ 1678 + 50+ 464)
```

[1] 0.4187689

Based on the standards given above, with a threshold of 0.1, the accuracy falls in the fail range. It is less than 0.50. So, it is even lower than random guessing.

If we change the threshold to 0.15, which seems closer to the median of the predicted scores, accuracy will increase from 0.42 to 0.60. Accuracy of 0.60 is not good based on the range given. However, given that we are dealing with an outcome that is potentially affected by many confounding factors, accuracy of 0.60 may not be that bad. **We need to judge the magnitude of accuracy based on the problem we are dealing with.** For instance, if we are dealing with a medical diagnosis compared to a decision in social science, the two stories and acceptable thresholds for accuracy, false positive, and false negative will be totally different.

```
> table(Test$leaveUCLA,pred1>0.15)
        FALSE(No)    TRUE(yes)
  no    1398         1061
  yes   135          379

> 1398+1061
[1] 2459
> 1398+379
[1] 1777
> 1398+379+1061+135
[1] 2973
> 1777/2973
```
[1] 0.5977127

We will now use hsb2.csv data to construct the ROC curve

Using hsb2

First Step
Gender will be our outcome variable and it needs to be coded as “0” and “1”.

```
> table(gender)
gender
female          male
  109           91

> gender<-recode(gender,"'female'='0';'male'='1'")
> table(gender)
gender
  0   1
109  91
```

Second step: We will split the data to testing and training with 65% of the data assigned to training and the rest assigned to testing.

```
>library(caTools)
> split<-sample.split(hsb2$gender,SplitRatio=0.65)
> train<-subset(hsb2,split=T)
> test<-subset(hsb2,split=F)
```

Step three: We will create the likelihood of being male or female from science and reading scores.

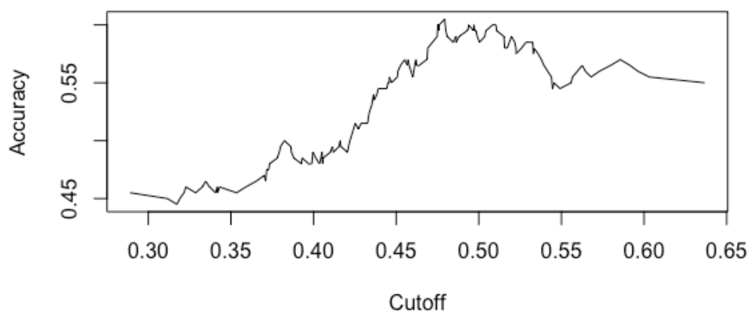
```
> m1<-glm(gender~reading+science,family="binomial")
> result_m1<-predict(m1,newdata=test,type="response")
> View(result_m1)
```

Sample predicted score by R

1	0.4051043
2	0.5084334
3	0.5224497

Step four – We will install the ROCR library and create the ROC curve

```
> library(ROCR)
> pred_m1<-prediction(result_m1,test$gender)
> acc<-performance(pred_m1,"acc")
> plot(acc)
```



In the following you will see that we can create the confusion matrices using different cutoff values provided by the ROC curve

```
> table(test$gender,result_m1>0.47)
```

predicted	FALSE(0)	TRUE(1)
female(0)	71	38
male (1)	45	46

Accuracy = (71+46)/(71+46+38+45) = 0.585

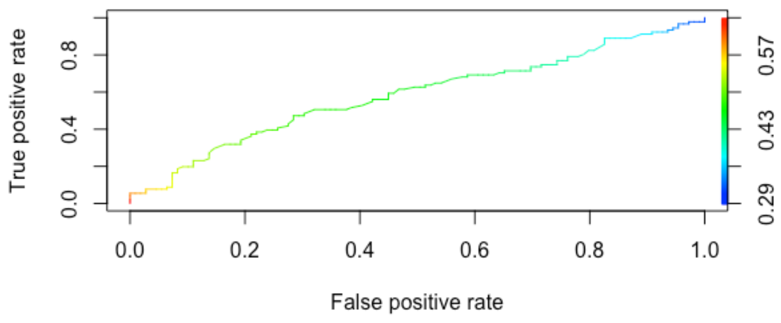
This is on a threshold of 0.47cut off where accuracy seems maximum

In the following you see the graph of true positive (sensitivity) vs. false positive (specificity) values.

```
> roc_curve<-performance(pred_log,"tpr","fpr")
> plot(roc_curve,colorize=T)
```

Sensitivity: the ability of a test to correctly identify patients with a disease. **Specificity**: the ability of a test to correctly identify people without the disease. True positive: the person has the disease and the test is positive.

- Sensitivity is identification of true positive
- Specificity is identification of true negative
- Depending on the question to be answered, sensitivity could be more important than accuracy.



```
> table(test$gender,result_m> 0.6)

            FALSE       TRUE
 female     109          0
 male        89         2
```

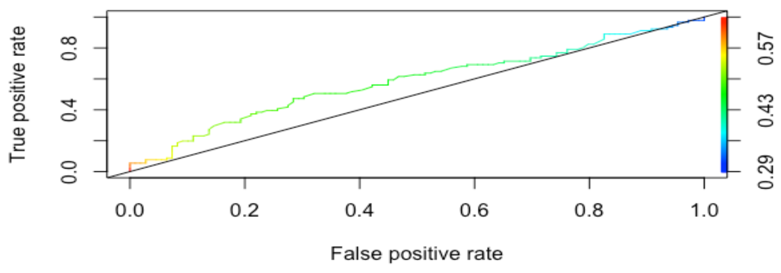
Accuracy changes to

109+2/89 = 0.555

```
> print(auc_ROCR)
```
[1] 0.5836778

One can add a straight line to the above plot to further clarify the AUC or the area under the curve which shows the accuracy of the model. It is clear from the area below the ROC curve that our accuracy is not that high and male vs. female cannot be classified based on their math and reading scores.

```
>abline(0,1)
```



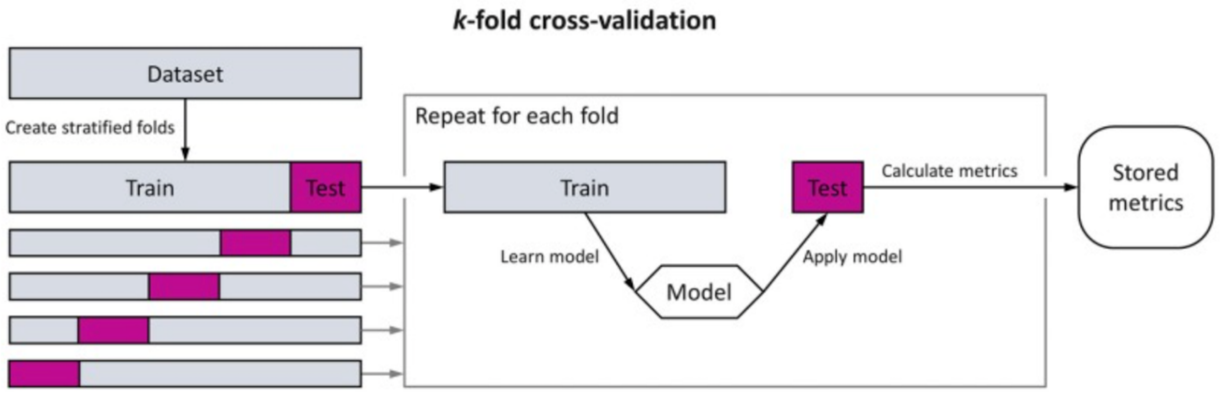
VI K-fold cross validation

Useful references

https://www.youtube.com/watch?v=Zd9GRoQjKvo
https://www.youtube.com/watch?v=c-kqw0Yf6BE

K-fold Cross Validation in Logistic Regression

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. **That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.** K-fold cross validation is a procedure used to estimate the skill of the model on a new data.



Schematic overview of cross-validation

The data set is randomly split into k-stratified folds. Each fold is used on a test set once, while the other folds are temporarily combined to form a training set for model generation. Performance metrics on the test data are calculated and stored, and the process is repeated for the number of folds that have been generated.

This is what happens in the K-fold cross validation..

- The data is divided into k subsamples.
- One out of the K-samples is used as testing and K-1 as the training data.
- A model is fit to the training data and it is evaluated on the test data
- There are common tactics that you can use to select the value of k for your dataset.
- There are commonly used variations on cross-validation such as stratified and repeated cv.

This is what happens in repeated K-fold cross validation
Repeated k-fold cross validation involves simply repeating the **cross-validation** procedure multiple times and reporting the mean result across all **folds** from all runs. This is where the k-fold cross validation is repeated n times, where the data sample is shuffled prior to each repetition, which results in a different split of the sample.

In the following I will demonstrate how you can conduct repeated K-fold cross validation in logistic regression. I will be using the pima diabetic data set that consists of the following variables.

Codebook for pima diabetic data set

Variable	Variable label	scale
Diabetic type II	outcome	0= No 1 = Yes
pregnancies	pregnancies	Number of pregnancies
Glucose	Glucose	Numerical
Blood Pressure	BloodPressure	Numerical
Skin thickness	skinthickness	Numerical
Insulin	Insulin	Numerical
Body Mass Index	bodymassindex	Numerical
**Diabetic pedigree function	diabeticpedigreefuntion	Numerical
Age	Age	Numerical

DiabetesPedigreeFunction: **Diabetes pedigree function (a **function** which scores likelihood of **diabetes** based on family history)

This data set can be found in the data folder on week eight of CCLE. We are going to run two types of regression on the binary outcome given (person has (1) and does not have (0) diabetic) as a function of all the given predictors.

The first type of regression is called gbm (gradient boosting machine). Gradient boosting machines (GBMs) are an extremely popular machine learning algorithm that have proven successful across many domains. The main idea of boosting is creating a series of sequential models with each model boosting the performance of the prior models. The main idea of boosting is to add new models to the ensemble ***sequentially***. In essence, one can start with a *weak*, and sequentially *boost* its performance by continuing to build new trees, where each new model in the sequence tries to fix up where the previous one made the biggest mistakes (i.e., each new tree in the sequence will focus on the training rows where the previous tree had the largest prediction errors).

The second type of regression is "**glm**" and in this particular example it is "logistic regression" because our outcome variable only has two levels.

Check out the following reference to learn about gbm models

https://bradleyboehmke.github.io/HOML/gbm.html

Step one: You need to tell R that the outcome variable is a factor and measured as (0,1). R should not assume that the outcome is a numerical variable.

```
> diabeticpima$Outcome<-as.factor(diabeticpima$Outcome)
> levels(diabeticpima$Outcome)<-c('No','Yes')
```

Step two. You need to download library (caret – classification and regression training). To install R, you also need to install “ggplot2”,

```
> library(caret)
```

Step three. You need to divide the data into testing and training set

```
> set.seed(777)
> partitionRule<-createDataPartition(diabeticpima$Outcome,p=0.7,list=F)
> trainingSet<-diabeticpima[partitionRule,]
> testingSet<-diabeticpima[-partitionRule,]
> splitRule<-trainControl(method="repeatedcv",number=10,repeats=3,
+ classProbs=T,summaryFunction=twoClassSummary)
```

In the above we have assigned 70% of the data to the training and the rest to the testing

Step four: Run the gbm model (Outcome~., means use all the predictors)
You need to download the “e1071” library

```
>gbmModel<-train(Outcome~.,data=trainingSet,trControl=splitRule,
+method="gbm",preProc=c("center","scale"),metric="ROC")
```

The following is a sample of what happens during the cross validation

```
Iter   TrainDeviance   ValidDeviance   StepSize        Improve
     1        1.2466             nan             0.1000     0.0207
     2        1.2090             nan             0.1000     0.0116
     3        1.1833             nan             0.1000     0.0115
     4        1.1562             nan             0.1000     0.0140
     5        1.1294             nan             0.1000     0.0089
     6        1.1058             nan             0.1000     0.0071
     7        1.0883             nan             0.1000     0.0056
     8        1.0791             nan             0.1000     0.0005
     9        1.0636             nan             0.1000     0.0073
```

Install the gbm package

Step five. Predict the outcome using the gbm Model

```
> gbmTest<-predict(gbmModel,newdata=testingSet)
```

Step. Six. Use the predicted data found in step five and the actual data in the testing data set to build a confusion matrix and check the accuracy of your model

```
> confusionMatrix(data=gbmTest,testingSet$Outcome)
Confusion Matrix and Statistics

                Reference(actual)
Prediction      No          Yes
      No        133         35
      Yes       17          45
```

- 133 means that 133 people did not have diabetic and the model predicted the same.
- 45 means 45 people had diabetic and the model predicted the same
- 35 means that 35 people had diabetic and model predicted no (false negative)
- 17 means that 17 people did not have diabetic and the model predicted they do, (false positive)

Accuracy = (133+45)/(133+45+35+17) = 0.773913 – the same value given by R

Accuracy : 0.7739
 95% CI : (0.7143, 0.8263)

We are 95% confident that the accuracy of our model is between 71.4% to 82.6%
You will also get the values for false positive, false negative, etc.

```
   No Information Rate : 0.6522
   P-Value [Acc > NIR] : 4.208e-05
                 Kappa : 0.4741
```

You will also get the values for

```
Mcnemar's Test P-Value : 0.0184
           Sensitivity : 0.8867
           Specificity : 0.5625
        Pos Pred Value : 0.7917
        Neg Pred Value : 0.7258
            Prevalence : 0.6522
        Detection Rate : 0.5783
  Detection Prevalence : 0.7304
     Balanced Accuracy : 0.7246

      'Positive' Class : No
```

Now we run the glm model

We now build the glm Model and compare the results with "gbm" model

```
> glmModel<-train(Outcome~.,data=trainingSet,trControl=splitRule,
+ method="glm",preProc=c("center","scale"),metric="ROC")
> glmTest<-predict(glmModel,newdata=testingSet)
> confusionMatrix(data=glmTest,testingSet$Outcome)
Confusion Matrix and Statistics

          Reference
Prediction   No    Yes
       No    128   37
       Yes   22    43

               Accuracy : 0.7435
                 95% CI : (0.6819, 0.7986)
```

Accuracy is higher with the gbm Model.

```
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 0.001872

                  Kappa : 0.4087

 Mcnemar's Test P-Value : 0.068357

            Sensitivity : 0.8533
            Specificity : 0.5375
         Pos Pred Value : 0.7758
         Neg Pred Value : 0.6615
             Prevalence : 0.6522
         Detection Rate : 0.5565
   Detection Prevalence : 0.7174
      Balanced Accuracy : 0.6954

       'Positive' Class : No
```

Now we resample

```
> resamps<-resamples(list(GBM=gbmModel,GLM=glmModel))
> summary(resamps)

Call:
summary.resamples(object = resamps)

Models: GBM, GLM
Number of resamples: 30
```

- We have ten folds and we repeat each ten times and that makes 30 resamples

ROC

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
GBM	0.7507937	0.7947368	0.8342105	**0.834127**	0.8792607	0.9079365
GLM	0.7398496	0.8030075	0.8385965	**0.828279**	0.8571429	0.8932331

Sens (sensitivity) = ability of the model to diagnose true positive

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
GBM	0.7714286	0.8285714	0.8571429	**0.8695238**	0.9142857	0.9714286
GLM	0.8000000	0.8571429	0.9142857	0.8952381	0.9428571	0.9714286

Spec (specificity) = ability of the model to diagnose true negative

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
GBM	0.3684211	0.4868421	0.5672515	**0.5760234**	0.6315789	0.7894737
GLM	0.3684211	0.4736842	0.5526316	0.5628655	0.6578947	0.7894737

The above information is useful as the mean ROC is higher for GBM (classification method) compared to GLM model (0.834 compared to 0.828)

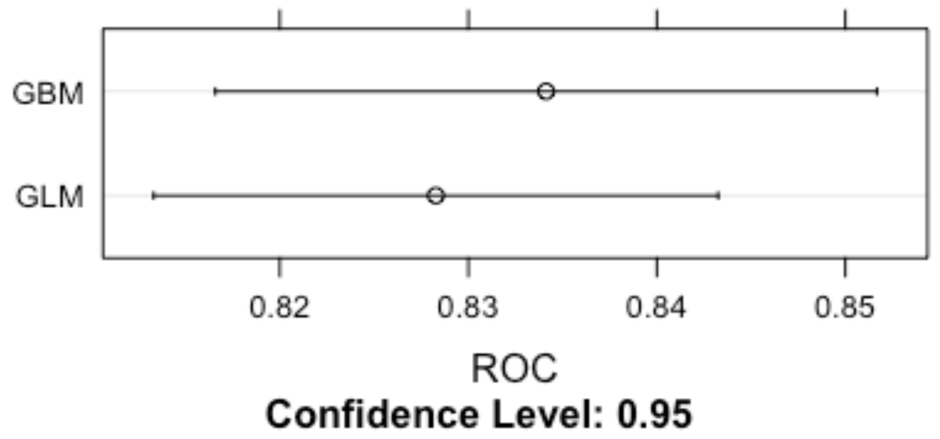
Both models seem to perform better with estimating sensitivity or diagnosis of cases in which the individual is diabetic. (correct identification of 1) and weaker in diagnosis of cases in which the individual is not diabetic (correct identification of zero)

We get similar results from the confusion matrix
Sensitivity : 0.8533
Specificity : 0.5375

To compare the GBM and GLM models with respect to accuracy it is best to visualize the comparison. We could do this by using the caret library.

We now draw the confidence interval for the "gbm" and "glm" methods

```
> trellis.par.set(caretTheme())
>dotplot(resamps,metric="ROC")
```



Based on the above plot, we can see that mean accuracy is higher for the GBM compared to the GLM model. This plot is more understandable and easier to explain for both statistical and non-statistical clients compared to the summaries given on the prior page.

Guidelines for interpretation of Cohen's Kappa in classification methods

Value of Kappa	Level of agreement
<=0	None
0.01 to 0.20	None to slight
0.20 to 0.40	Fair
0.40 to 0.60	Moderate
0.61 to 0.80	Substantial
0.81 to 1.0	Almost perfect

Kappa is an index of how closely the instances classified by the machine learning classifier matches the real data. As you noticed Kappa is generally lower than accuracy.