HAMLINE UNIVERSITY

Guadalupe Torres[1]

02/23/2025

–> > The purpose of this document is to simulataneously analyze admission data. I am interested in how variables such graduate record exam scores, GPA, and prestige of the undergraduate institution, affect admission into graduate school.

Exploratory Data Analysis

```
##      admit             gre             gpa             rank      
##  Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000  
##  Median :0.0000   Median :580.0   Median :3.395   Median :2.000  
##  Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :2.485  
##  3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000  
##  Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :4.000  
```

```
view(admit)
print(str(admit))
```

```
## spc_tbl_ [400 × 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ admit: num [1:400] 0 1 1 1 0 1 1 0 1 0 ...
##  $ gre  : num [1:400] 380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num [1:400] 3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : num [1:400] 3 3 1 4 4 2 1 2 3 2 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   admit = col_double(),
##   ..   gre = col_double(),
##   ..   gpa = col_double(),
##   ..   rank = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr> 
## NULL
```

```
sum(is.na(admit))
```

```
## [1] 0
```

```
x <- c(1, 1, 4, 5, 4, 6)
duplicated(x)
```

```
## [1] FALSE  TRUE FALSE FALSE  TRUE FALSE
```

```
x[duplicated(x)]
```

```
## [1] 1 4
```

–> The data set has 400 observations with 4 columns, the column categories consist of `admit`, `gre`, `gpa`, and `rank`. We can see that all the variables are numeric data types, r represents these variables as col_double. The dataset contains 0 missing values. To check for duplicates the code will return true if there are no duplicates and false if there is. We can see that in this case there are duplicate values in the dataset.

1. Checking for balanced classes in the dataset

```
table(admit$admit)
```

```
## 
##   0   1 
## 273 127
```

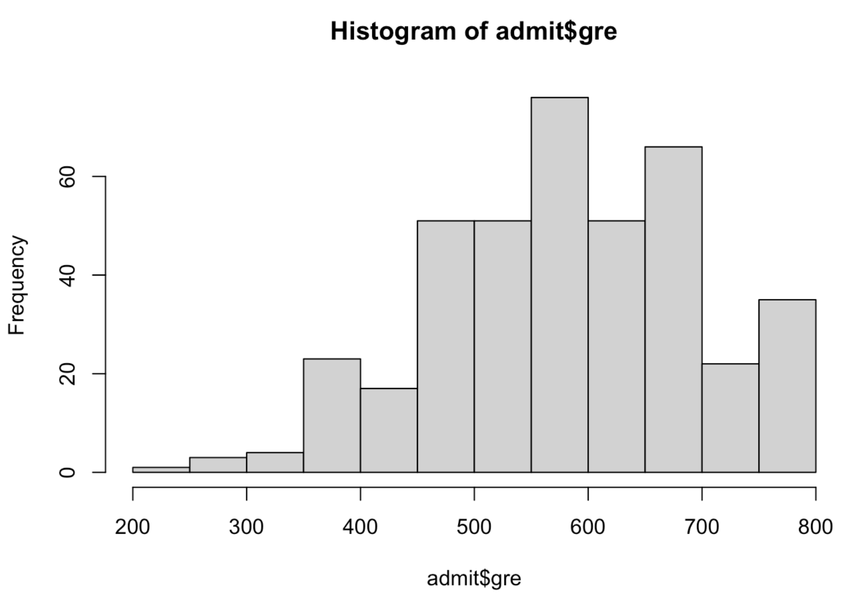
```
class_proportions <- prop.table(table(admit$admit))
class_proportions
```

```
## 
##      0      1 
## 0.6825 0.3175
```

–> To determine if the admit and don't admit classes are balanced in the dataset, we need to check if the number of data points belonging to each class is roughly equal. In this case, one class has significalty more data points then the other making the dataset imbalanced.

2. Describing the distribution of GRE scores.

```
hist(admit$gre)
```



–> Based on the Histogram the distribution of GRE scores is left skewed.

Seperating data into training and testing sets.

```
set.seed(1)
```

```
split <- sample.split(admit$admit, SplitRatio = 0.7)
```

```
train_data <- subset(admit, split == TRUE)
test_data <- subset(admit, split == FALSE)
```

```
dim(train_data)
```

```
## [1] 280   4
```

```
dim(test_data)
```

```
## [1] 120   4
```

Fitting the Model

```
log_model <- glm(admit ~ gre + gpa + rank, data = train_data, family = binomial)
```

```
summary(log_model)
```

```
## 
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = binomial, data = train_data)
## 
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -3.091615   1.296665  -2.384 0.017112 *  
## gre          0.000809   0.001253   0.646 0.518578    
## gpa          0.903378   0.384148   2.352 0.018691 *  
## rank        -0.502850   0.151783  -3.313 0.000923 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 350.14  on 279  degrees of freedom
## Residual deviance: 328.00  on 276  degrees of freedom
## AIC: 336
## 
## Number of Fisher Scoring iterations: 4
```

–>

The variable gpa is significant in this case because of its high number. We know it is also significant because of is low p-value being less then 0.05.

Confusion Matrix

```
pred_probs <- predict(log_model, test_data, type = "response")
pred_probs
```

```
##          1          2          3          4          5          6          7 
## 0.50255500 0.41833016 0.65336864 0.32112262 0.42944420 0.44975111 0.27707527 
##          8          9         10         11         12         13         14 
## 0.26051657 0.26533610 0.45282273 0.31728436 0.35540324 0.35591118 0.38310518 
##         15         16         17         18         19         20         21 
## 0.16100557 0.37345143 0.22545878 0.55186631 0.09775218 0.19719993 0.40030852 
##         22         23         24         25         26         27         28 
## 0.41594052 0.41603520 0.62729297 0.48223650 0.21646491 0.28784142 0.35342894 
##         29         30         31         32         33         34         35 
## 0.50124046 0.54682585 0.14548052 0.15673567 0.45011807 0.36616137 0.45748900 
##         36         37         38         39         40         41         42 
## 0.18188874 0.16620824 0.52220721 0.32474883 0.27302768 0.27475891 0.16222961 
##         43         44         45         46         47         48         49 
## 0.49267914 0.34580865 0.49493724 0.31753440 0.39877016 0.35825003 0.64229527 
##         50         51         52         53         54         55         56 
## 0.30057900 0.18099880 0.36554962 0.23492998 0.45227361 0.26378809 0.28421887 
##         57         58         59         60         61         62         63 
## 0.33498815 0.38291375 0.27631965 0.32359683 0.64229527 0.55046532 0.21271526 
##         64         65         66         67         68         69         70 
## 0.18436050 0.29820320 0.35196500 0.39525180 0.24018039 0.32377416 0.10766102 
##         71         72         73         74         75         76         77 
## 0.20164373 0.33473111 0.39859059 0.29366482 0.32897800 0.15425676 0.37045913 
##         78         79         80         81         82         83         84 
## 0.46563156 0.28993116 0.34156775 0.15148060 0.39615461 0.16014125 0.23956584 
##         85         86         87         88         89         90         91 
## 0.44230705 0.29773635 0.52718383 0.20507335 0.49040227 0.16656337 0.28115656 
##         92         93         94         95         96         97         98 
## 0.30702380 0.51205841 0.13404852 0.18121870 0.42704887 0.30303498 0.17477703 
##         99        100        101        102        103        104        105 
## 0.36967341 0.17646667 0.23276922 0.45152019 0.61746982 0.30510513 0.28261856 
##        106        107        108        109        110        111        112 
## 0.14422801 0.24412856 0.32336099 0.30710998 0.32946747 0.35033683 0.47350476 
##        113        114        115        116        117        118        119 
## 0.15440538 0.43666086 0.29772038 0.22030813 0.37573969 0.63482684 0.37133651 
##        120 
## 0.39922284
```

```
pred_classes <- ifelse(pred_probs > 0.5, 1, 0)
```

```
pred_classes <- as.factor(pred_classes)
```

```
head(pred_probs)
```

```
##         1         2         3         4         5         6 
## 0.5025550 0.4183302 0.6533686 0.3211226 0.4294442 0.4497511
```

```
head(pred_classes)
```

```
## 1 2 3 4 5 6 
## 1 0 1 0 0 0 
## Levels: 0 1
```

```
do.call(rbind, Map(data.frame, predicted_classes=pred_classes, admit=test_data$admit))
```

	predicted_classes <fct>	admit <dbl>
1	1	0
2	0	0
3	1	1
4	0	0
5	0	1
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0

1-10 of 120 rows   Previous 1 2 3 4 5 6 ... 12 Next

```
conf_matrix <- table(Predicted = pred_classes, Actual = test_data$admit)
conf_matrix
```

```
##          Actual
## Predicted  0  1
##         0 77 29
##         1  5  9
```

```
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
accuracy
```

```
## [1] 0.7166667
```

```
print(conf_matrix)
```

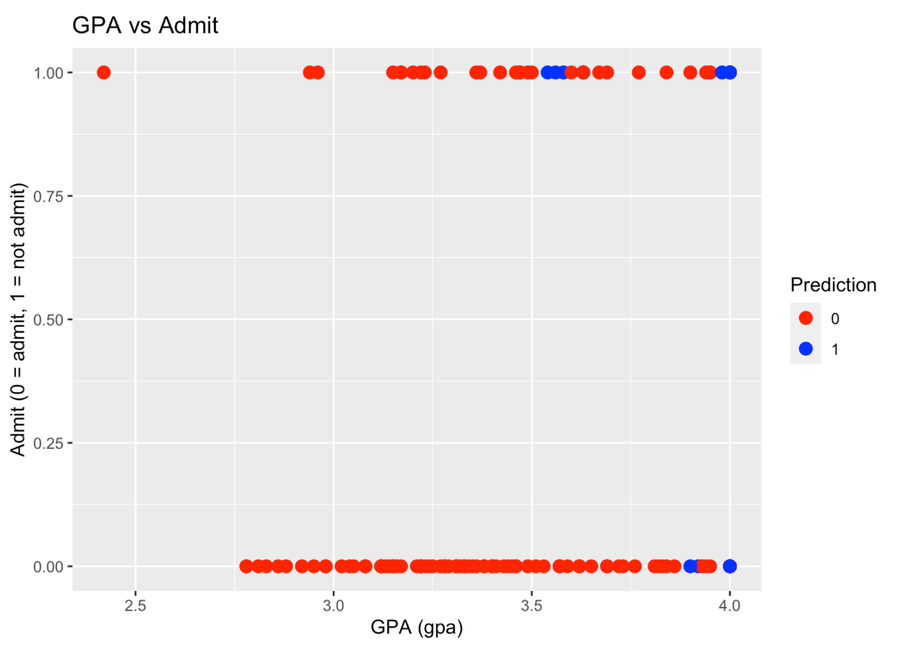
```
##          Actual
## Predicted  0  1
##         0 77 29
##         1  5  9
```

```
print(paste("Accuracy:", round(accuracy, 4)))
```

```
## [1] "Accuracy: 0.7167"
```

–> The accurary of the model is 71%.

```
ggplot(test_data, aes(x = gpa, y = as.numeric(as.character(admit)), color = as.factor(pred_classes))) +
  geom_point(size = 3) +
  labs(title = "GPA vs Admit",
       x = "GPA (gpa)",
       y = "Admit (0 = admit, 1 = not admit)") +
  scale_color_manual(values = c("red", "blue"), name = "Prediction")
```



–> We can conclude that GPA is the most important for predicting admission status

1. **Email** gtorressanchez01@hamline.edu. **Position** Student↩