BIOS 635: K-Nearest Neighbor and Linear Regression

Kevin Donovan

1/28/2021

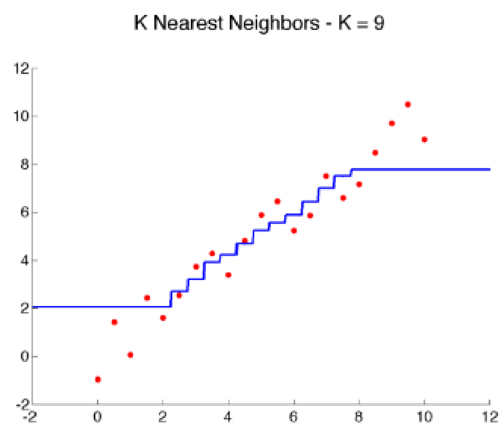
Review

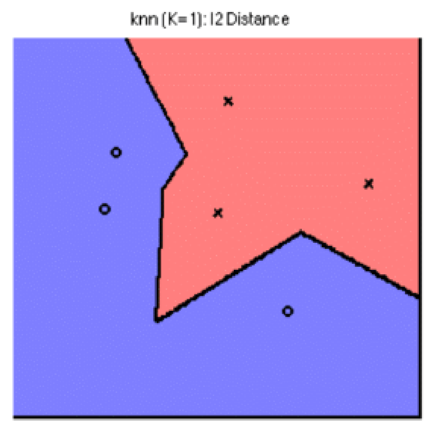
- Homework 1 due on today at 11PM through GitHub Classroom
- Article Evaluation 1 assigned, due on 2/9 through GitHub Classroom
- Last lecture: discussed bias and variance trade-off, classification, k-nearest neighbor

K-nearest Neighbor (KNN)

Simple and **flexible** algorithm:

1. Given training data $D = \{\mathbf{x}_i, y_i\}$, distance function $d(\cdot, \cdot)$ and input $\mathbf{x}$
2. Find $\{j_1, \ldots, j_K\}$ closest examples with respect to $d(\mathbf{x}, \cdot)$
   - (regression) if $y \in \mathbf{R}$, return average: $\frac{1}{K}\sum_{k=1}^{K} y_{j_k}$
   - (classification) if $y \in \pm 1$, return majority: $sign(\sum_{k=1}^{K} y_{j_k})$





KNN Examples: Regression

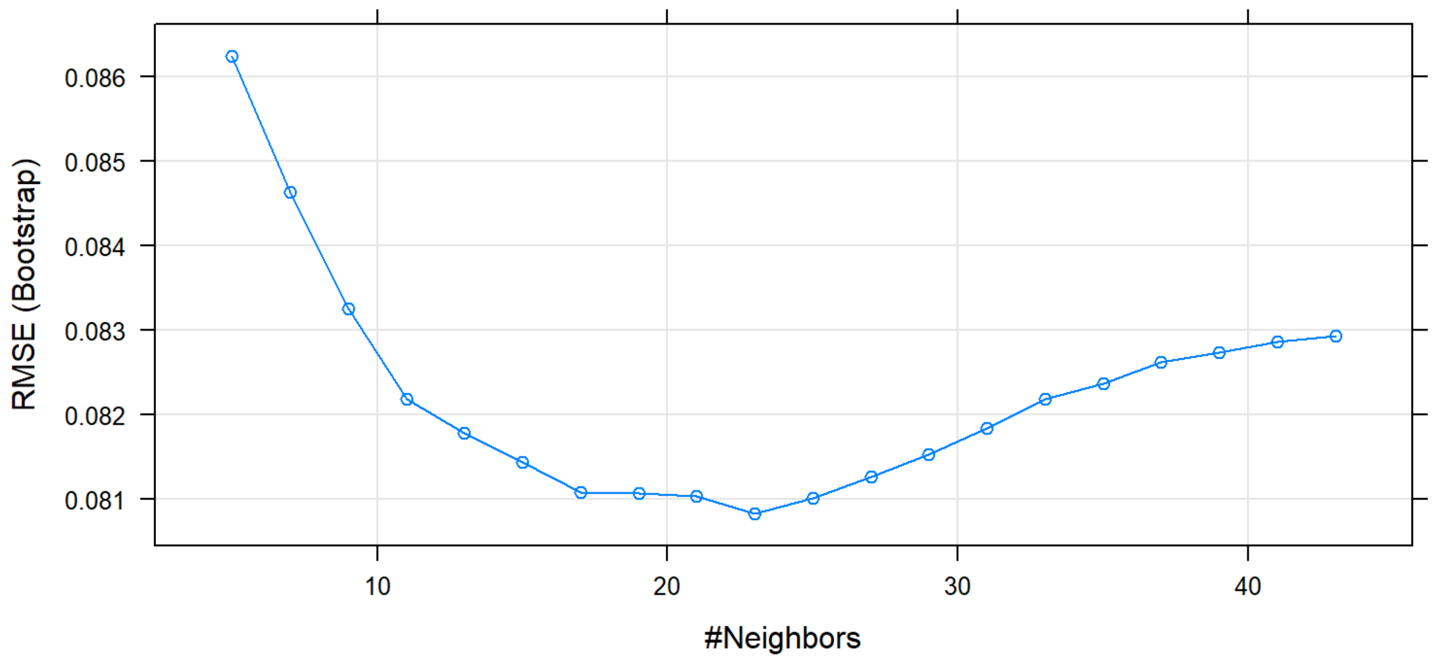
- Suppose we want to predict chance at getting admitted to graduate school based on
  - *GRE (GRE Score)*
  - *GPA (CGPA)*
- First, we **don't split the data**

```
knnFit <- train(admit ~ ., data = student_data, method = "knn", tuneLength = 20)

# Look at tuning results
knnFit
```

```
## k-Nearest Neighbors
##
## 400 samples
##   2 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 400, 400, 400, 400, 400, 400, ...
## Resampling results across tuning parameters:
##
##   k   RMSE        Rsquared   MAE
##    5  0.08624068  0.6407232  0.06399691
```

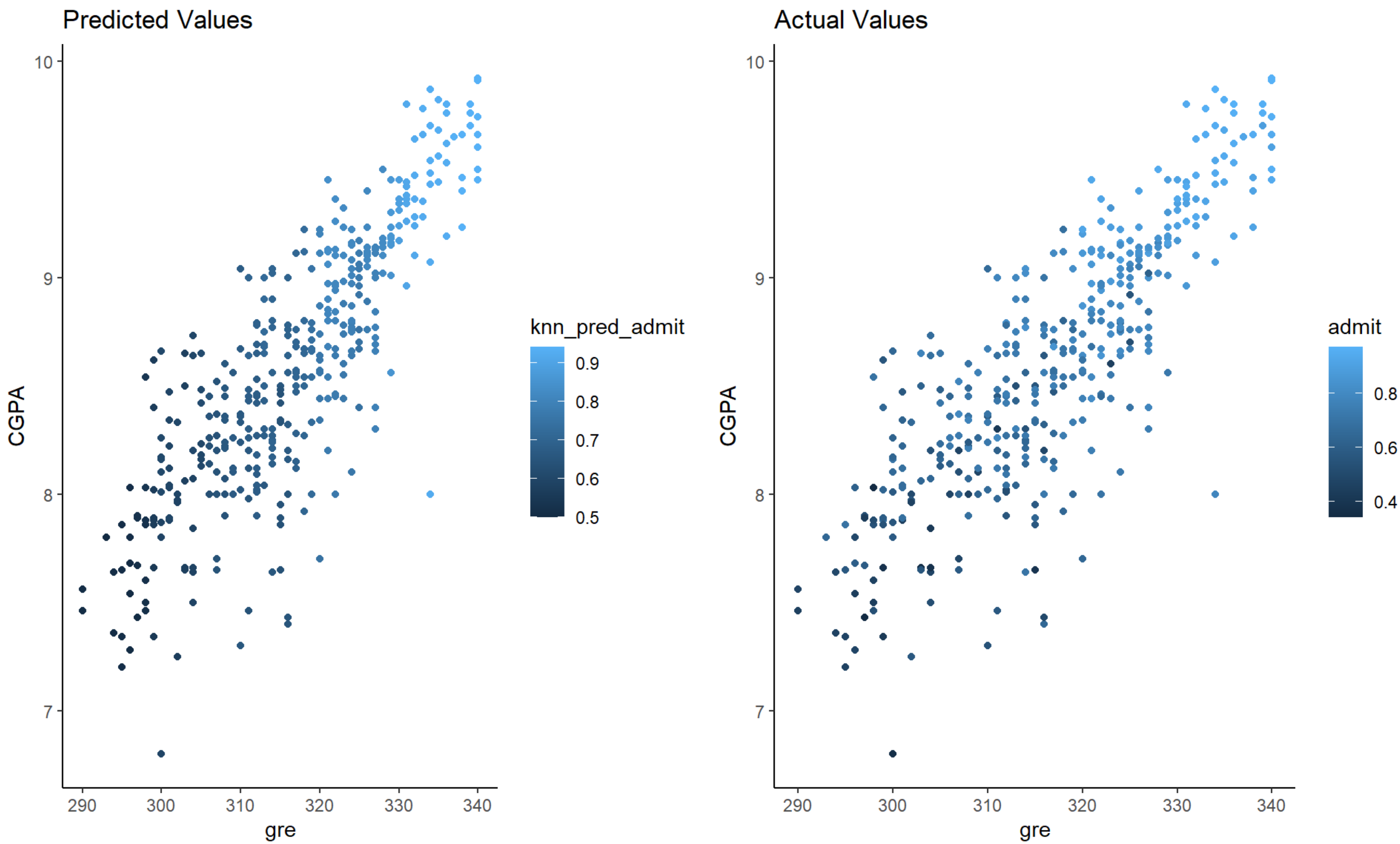
```
##    7  0.08463256  0.6509904  0.06307299
##    9  0.08325199  0.6602502  0.06208702
##   11  0.08218289  0.6683209  0.06131491
##   13  0.08178742  0.6711018  0.06107814
##   15  0.08143417  0.6733520  0.06058410
##   17  0.08108427  0.6761255  0.06035442
##   19  0.08106587  0.6761504  0.06032425
##   21  0.08103356  0.6763025  0.06025458
##   23  0.08082788  0.6780733  0.06021622
##   25  0.08101433  0.6764256  0.06039879
##   27  0.08126620  0.6744632  0.06066101
##   29  0.08152713  0.6723189  0.06103786
##   31  0.08184195  0.6699480  0.06149335
##   33  0.08218699  0.6675157  0.06194097
##   35  0.08236770  0.6662938  0.06215607
##   37  0.08261692  0.6644788  0.06245605
##   39  0.08273686  0.6637750  0.06255669
##   41  0.08286002  0.6628480  0.06280799
##   43  0.08293707  0.6621550  0.06294731
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 23.
```

```
plot(knnFit)
```


0.086
0.085
0.084
0.083
0.082
0.081
RMSE (Bootstrap)
10
20
30
40
#Neighbors

KNN Examples: Regression

```
## [1] "RMSE=0.0754339"
```



KNN Examples: Regression

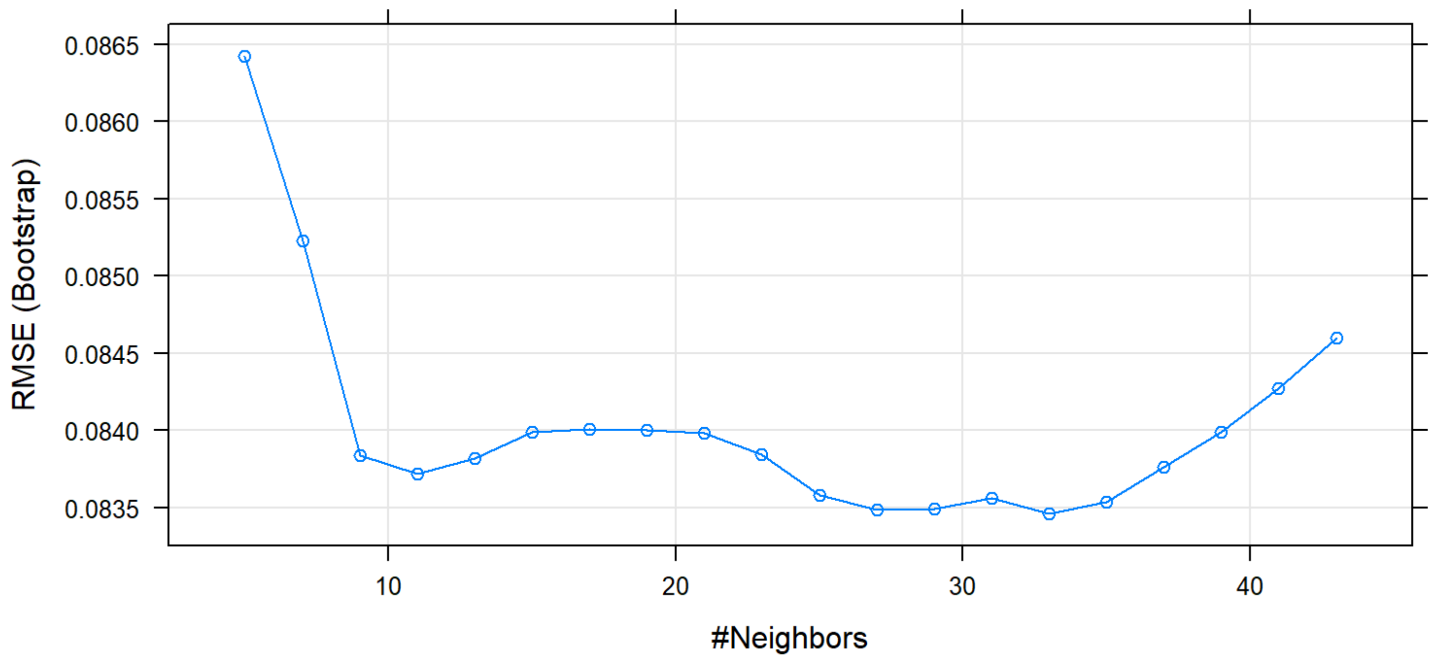
- Split the data into training and testing portion
- 60:40 split used

```
student_data <- student_data %>% select(-knn_pred_admit)

set.seed(12)
student_data_tt_index <- createDataPartition(student_data$admit, p=0.6, list = FALSE)
student_data_train <- student_data[student_data_tt_index,]
student_data_test <- student_data[-student_data_tt_index,]

knnFit <- train(admit ~ ., data = student_data_train, method = "knn", tuneLength = 20)

# Look at tuning results
knnFit
```

```
## k-Nearest Neighbors
##
## 241 samples
##   2 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 241, 241, 241, 241, 241, 241, ...
## Resampling results across tuning parameters:
##
##   k   RMSE        Rsquared   MAE
##    5  0.08642279  0.6348680  0.06442511
```

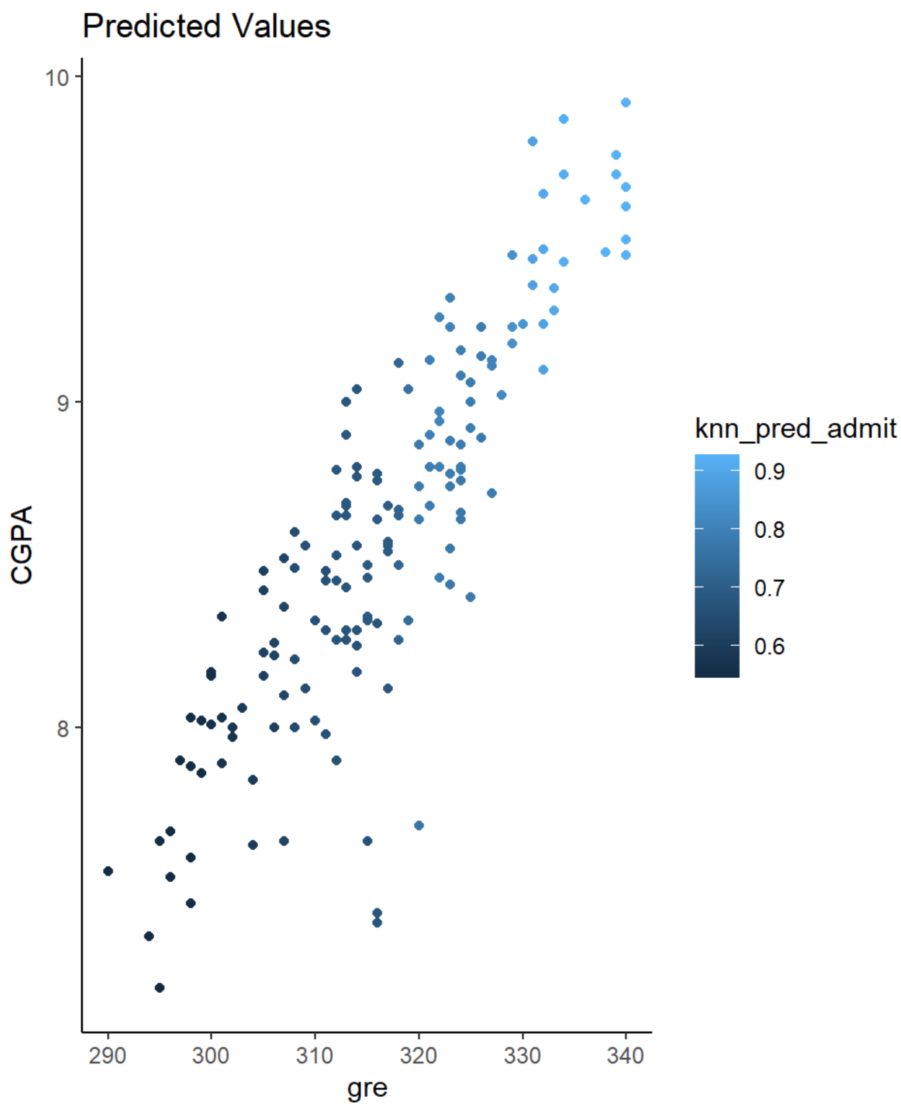
```
##    7  0.08523053  0.6409723  0.06380427
##    9  0.08383730  0.6504916  0.06227605
##   11  0.08371969  0.6520071  0.06207500
##   13  0.08381847  0.6517754  0.06229965
##   15  0.08398719  0.6493221  0.06242865
##   17  0.08400965  0.6487351  0.06258471
##   19  0.08400019  0.6483039  0.06272774
##   21  0.08398394  0.6484643  0.06296187
##   23  0.08384457  0.6496890  0.06313776
##   25  0.08358056  0.6517596  0.06331797
##   27  0.08348635  0.6527741  0.06334313
##   29  0.08349258  0.6532058  0.06354299
##   31  0.08356277  0.6529664  0.06376675
##   33  0.08346069  0.6545094  0.06377156
##   35  0.08353767  0.6545371  0.06389530
##   37  0.08376537  0.6532999  0.06416951
##   39  0.08399193  0.6529646  0.06463158
##   41  0.08427033  0.6518218  0.06504143
##   43  0.08459924  0.6500339  0.06540996
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 33.
```

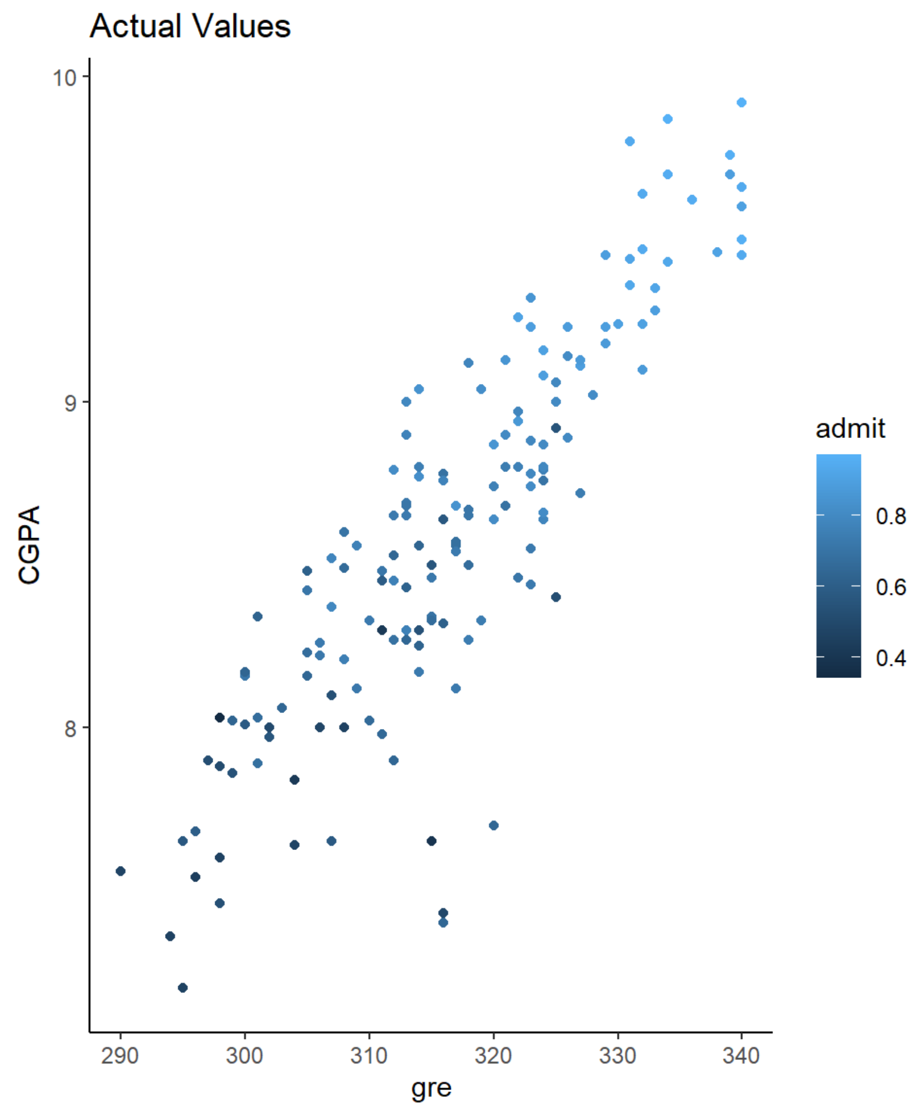
```
plot(knnFit)
```


RMSE (Bootstrap)
0.0865
0.0860
0.0855
0.0850
0.0845
0.0840
0.0835
10
20
30
40
#Neighbors

KNN Examples: Regression

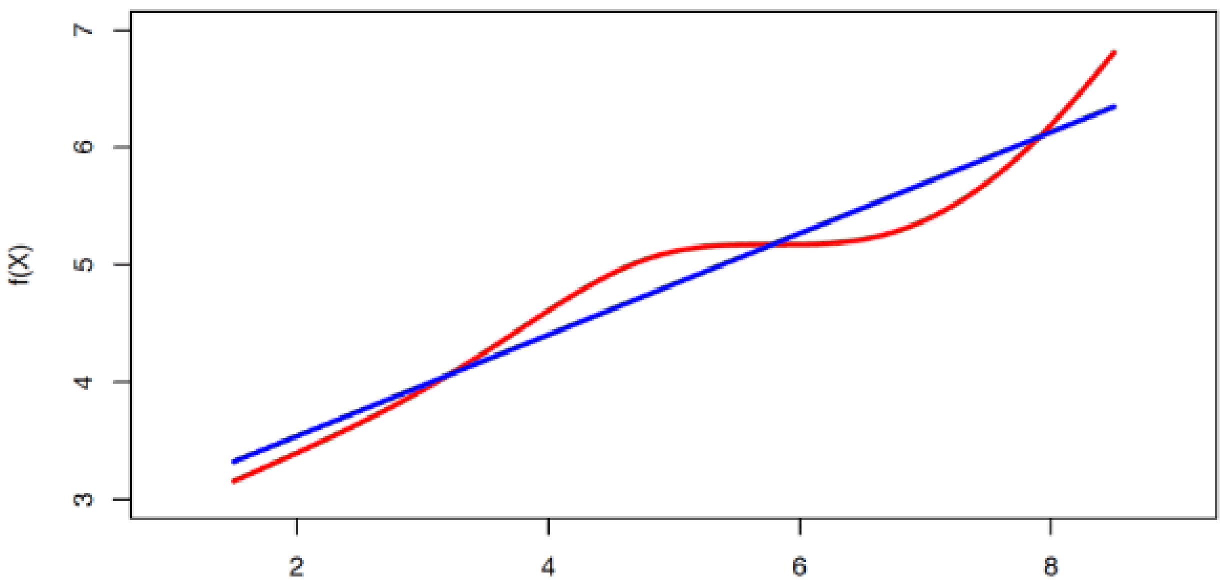
```
## [1] "Test RMSE=0.0825106"
```





Linear Regression

- Let's model the relationship between Y and $X_1, \ldots, X_p$
  - *Model:* $Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \epsilon$
  - *Simple, but very flexible and generalizable*
  - *True trend never linear, but may be* ***good approximation***

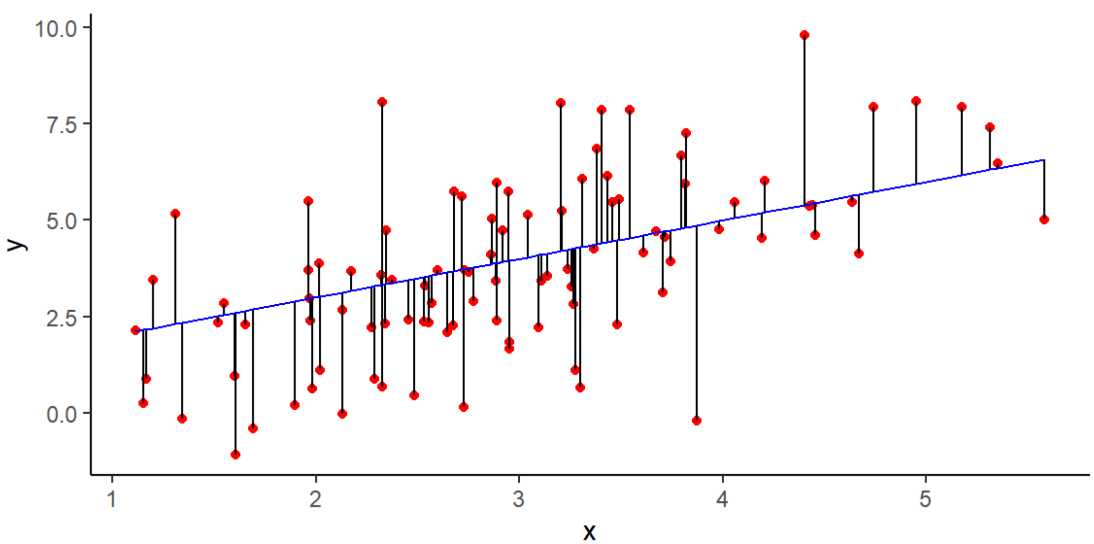


Simple Linear Regression

- **Model**: $Y = \beta_0 + \beta_1 X + \epsilon$
  - β_0, β_1 *are coefficients or parameters*
  - ϵ *is error*
- **Idea**: estimate trend by estimating parameters
  - *Estimates =* $\hat{\beta}_0, \hat{\beta}_1$
  - *Predict* Y *using estimated trend:* $\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x$
  - *How to estimate?*

Simple Linear Regression: Estimation

- Estimation by **least squares**
  - *Suppose* $\hat{\beta}_0, \hat{\beta}_1$ *is a possible estimate*
  - *Suppose we observe data* $\{(x_1, y_1), \ldots, (x_n, y_n)\}$
  - *For above estimates, define residuals* $e_i = y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)$
  - *Define residual sum of squares (RSS) as* $RSS = \sum_{i=1}^{n} (e_i)^2$
  - ***Goal**: Choose* $\hat{\beta}_0, \hat{\beta}_1$ *which minimizes* RSS



Simple Linear Regression: Estimation

- In case with one feature, can show

$$
\begin{aligned}
\hat{\beta_1} &= \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})} \\
&= r_{xy}\frac{\hat{\sigma_y}}{\hat{\sigma_x}}
\end{aligned}
$$

$$
\hat{\beta_0} = \bar{Y} - \hat{\beta_1}\bar{X}
$$

where $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$ and $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$

Assessing Variability of Estimates

- *Standard error*(SE) of estimator = variability across repeated samples
- Denoting $\mathrm{Var}(\epsilon) = \sigma^2$

$$\mathrm{SE}(\hat{\beta}_1) = \sqrt{\frac{\sigma^2}{\sum_{i=1}^{n}(X_i - \bar{X})}}$$

$$\mathrm{SE}(\hat{\beta}_0) = \sqrt{\sigma^2\Big[\frac{1}{n} + \frac{\sigma^2}{\sum_{i=1}^{n}(X_i - \bar{X})}\Big]}$$

- *Confidence intervals* (CI): for β_j for $j = 0, 1$

$$\hat{\beta}_j \pm z_{1-\alpha/2} * \mathrm{SE}(\hat{\beta}_j)$$

  - *Ex. For 95% CI, $\alpha = 0.05$ with $z_{0.975} = 1.96$*
  - *Never know σ^2 thus need to replace in formulas with estimate $\hat{\sigma^2}$*
  - *Results in **estimates** $\hat{SE}(\hat{\beta}_j)$ with $t_{n-2,1-\alpha/2}$ used*

Hypothesis Testing

- SEs also used to perform *hypothesis tests* on coefficients
- Ex. Test the *null hypthesis* of
  - H_0*: There is no relationship between* X *and* Y
  - *Alternative* H_1*: There is* ***some*** *relationship between* X *and* Y
- Mathematically, this corresponds to
  - H_0: $\beta_1 = 0$
  - H_1: $\beta_1 \neq 0$

Test Statistic

- To test null vs alternative, need to aggregate data into *test statistic*
  - *Then compare statistic value to it's distribution under* H_0
  - *For linear regression, test statistic is*

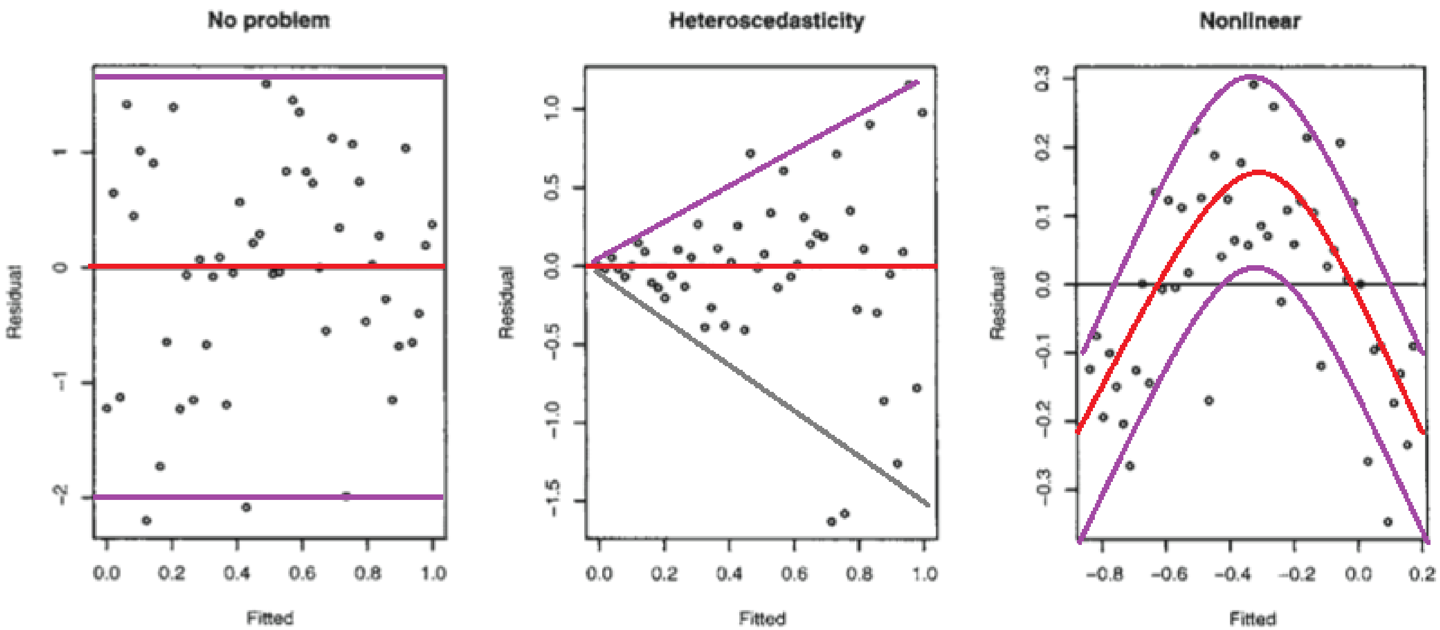
$$T = \frac{\hat{\beta}_1 - 0}{S\hat{E}(\hat{\beta}_1)}$$

- T has a t-distriubtion with $n-2$ degrees of freedom under null
- Can compute probability of observing $|T| \geq |t|$ in sample
  - *Denoted as p-value*

Model Assumptions

- Note that above **distribution** for test statistic only hold if

    1. *Usual linear regression* ***assumptions met****:*
        - Homoskedasicity
        - Independent residuals
        - **and** one of the following
    2. *Normally distributed residuals*
    3. *Large sample approximation*


No problem
Heteroscedasticity
Nonlinear
Residual
Fitted
0.0
0.2
0.4
0.6
0.8
1.0
-0.8
-0.6
-0.4
-0.2
0.2
0.3
0.1
-0.1
-0.3
0.5
1.0
-0.5
-1.0
-1.5
1
0
-1
-2

Assessing Model Accuracy

- *Residual Standard Error*

$$RSE = \sqrt{\frac{1}{n-2} RSS}$$

$$\text{where } RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- *R*^2 or "fraction of variance accounted for"

$$R^2 = \frac{TSS - RSS}{TSS}$$

$$\text{where } TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

TSS is the *total sum of squares*

Multiple Linear Regression

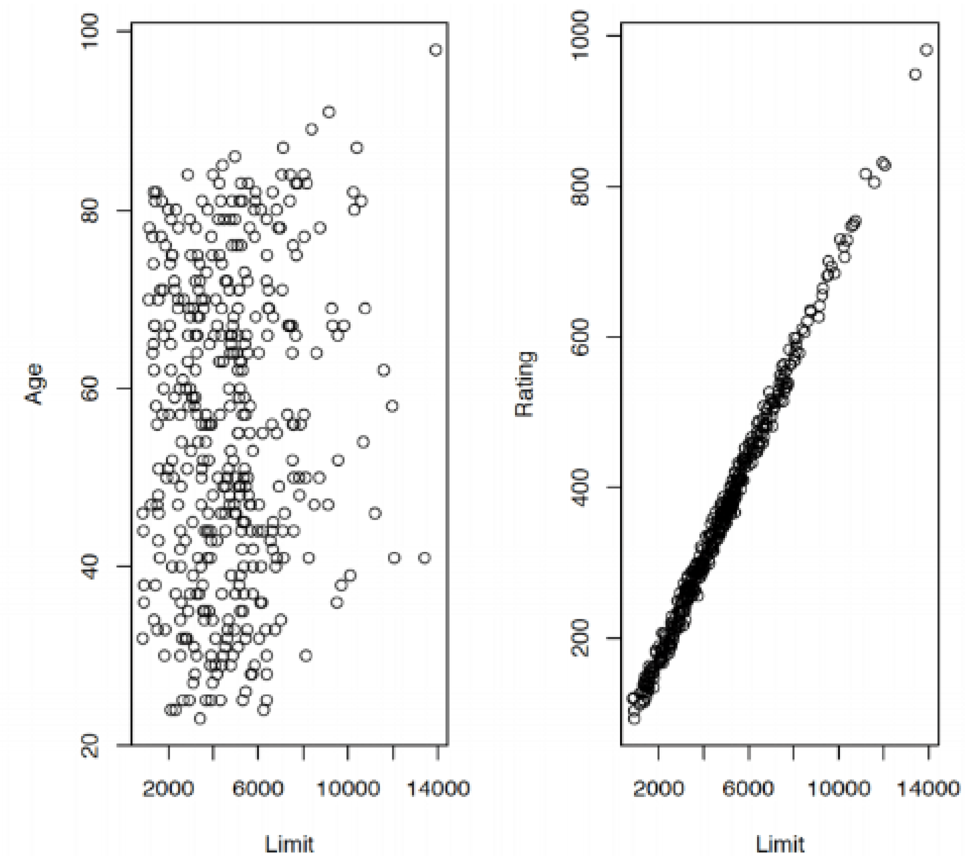
- Model is

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \epsilon$$

- **Interpretation**:
  - β_j *is mean change in* Y *after a one unit increase in* X_j*, holding other* X *fixed*
- For the graduate student admissions example:
  - *`admit`* = β_0 + β_1*GRE* + β_2*GPA* + ϵ
- **Estimation**:
  - *Again, least squares*
  - $RSS = \sum_{i=1}^{n} (e_i)^2 = \sum_{i=1}^{n} (y - \hat{\beta}_0 - \hat{\beta}_1 X_1 - \ldots - \hat{\beta}_p X_p)^2$

Collinearity

- Two or more predictors have **high correlation** with each other
- Implying holding one constant while varying others not possible in data
- Can result in **poor estimation**



Example

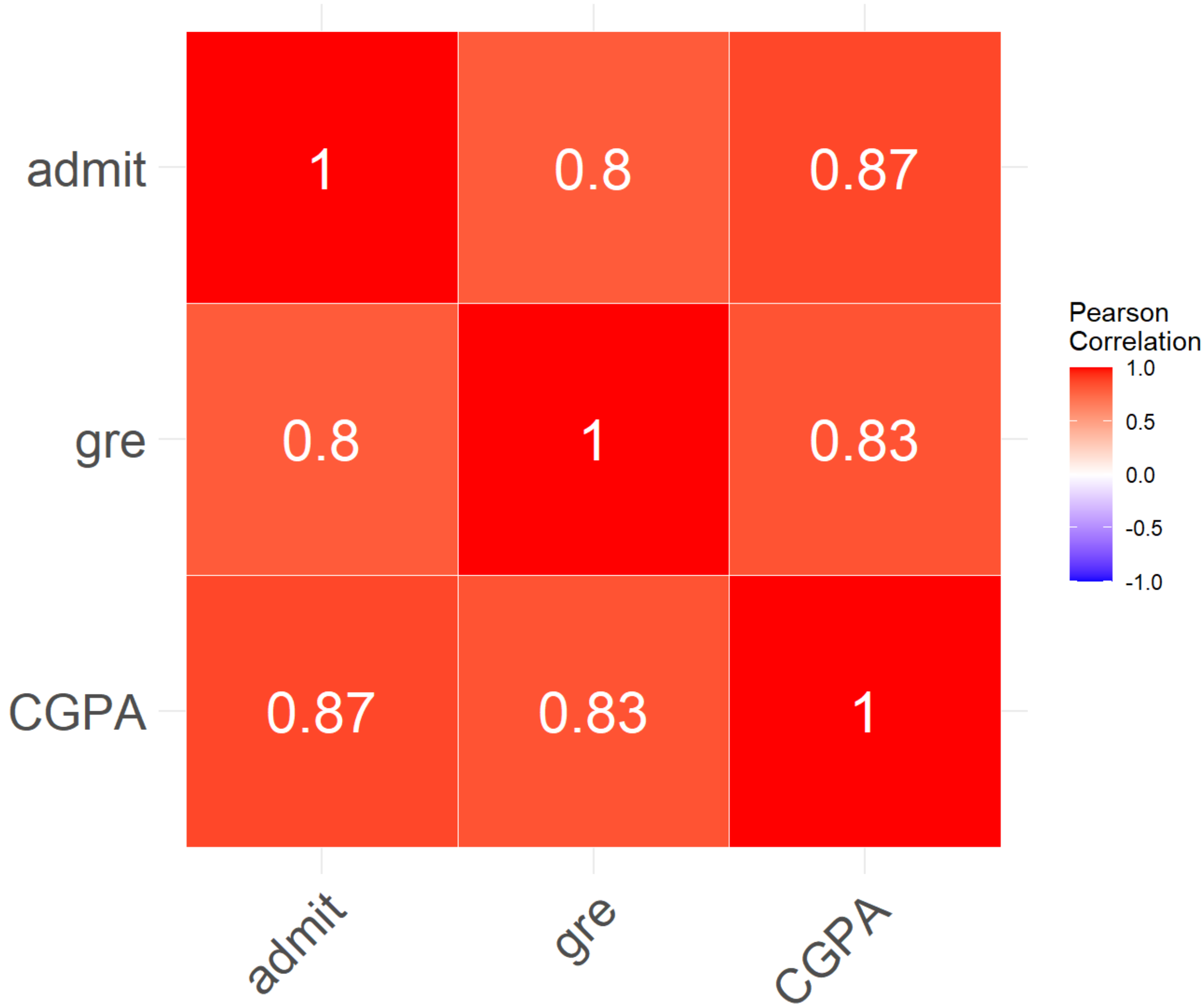
- Let's use regression instead of KNN to predict admission chance
- Model: $\texttt{admit} = \beta_0 + \beta_1\texttt{GRE} + \beta_2\texttt{GPA} + \epsilon$

```
lm_fit <- lm(formula = admit~., data=student_data)
tidy(lm_fit) %>%
  mutate(p.value=ifelse(p.value<0.005, "<0.005",
                        as.character(round(p.value, 2)))) %>%
  flextable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	-1.6175	0.10612	-15.2	<0.005
gre	0.0031	0.00053	5.8	<0.005
CGPA	0.1599	0.01015	15.8	<0.005

Example

- Let's look at the correlation matrix to assess **collinearity**



Qualitative Predictors

- **Quantitative**: β assess change in mean Y when X **changes by 1 unit**
- How do this work with variables indicating **categories**?
- Ex. Consider $X = 0$ or 1
    - *Model:* $Y = \beta_0 + \beta_1 X$
    - *Change in X of 1 unit = change from category "0" to category "1"*
- What about multi-category X?
    - *Ex. Consider $X = 0, 1$ or 2*
    - *Change in X of 1 unit = change from category "x" to category "x+1"*
    - *Assumes same level of change from 0 to 1, 1 to 2*
    - *Unrealistic for **nominal** variables*

Qualitative Predictors: Dummy Coding

- X with M categories $(0, 1, \ldots, M)$ create $M-1$ features
- One is *reference level*, denoted by **all dummy features = 0**
- Ex. Graduate Admissions
  - *Model:* $admit = \beta_0 + \beta_1 prestige + \epsilon$
  - *Suppose* $prestige$ *is categorical ("low", "medium", "high")*
  - *Model:* $admit = \beta_0 + \beta_1 prestige_1 + \beta_2 prestige_2 + \epsilon$

$$prestige_1 = \begin{cases} 1 & X = \text{medium} \\ 0 & \text{else} \end{cases}$$
$$prestige_2 = \begin{cases} 1 & X = \text{high} \\ 0 & \text{else} \end{cases}$$

Qualitative Predictors: Dummy Coding

- Ex. Graduate admission data
  - *Model:*
    $$admit = \beta_0 + \beta_1 prestige_1 + \beta_2 prestige_2 + \beta_3 prestige_3 + \beta_4 prestige_4 + \epsilon$$

```r
student_data <- student_data_full %>%
  mutate(`University Rating`=factor(`University Rating`-1)) %>%
  plyr::rename(c("Chance of Admit"="admit",
                 "University Rating"="prestige"))

lm_fit <- lm(formula = admit~prestige, data=student_data)
tidy(lm_fit) %>%
  mutate(p.value=ifelse(p.value<0.005, "<0.005",
                        as.character(round(p.value, 2)))) %>%
  flextable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.548	0.020	27.8	<0.005
prestige1	0.078	0.022	3.5	<0.005
prestige2	0.164	0.022	7.6	<0.005
prestige3	0.270	0.023	11.8	<0.005
prestige4	0.340	0.024	14.4	<0.005

Interactions:

- **Recall**: In the graduate admission data model, we assumed GPA and GRE effected admissions **independently**
  - $\texttt{admit} = \beta_0 + \beta_1 GRE + \beta_2 GPA + \epsilon$
  - *Change in average* $\texttt{admit}$ *from one unit change in GRE always* β_1
  - *Perhaps GRE matters more for admission chance among certain GPA ranges?*
  - *Referred to as an interaction between these features*

Interactions:

- Ex. 1: Two continuous features
  - *Model:*

$$\begin{aligned} admit &= \beta_0 + \beta_1 GRE + \beta_2 GPA + \beta_3 GRE * GPA + \epsilon \\ &= \beta_0 + (\beta_1 + \beta_3 GPA) * GRE + \beta_2 GPA + \epsilon \\ &= \beta_0 + \beta_1 GRE + (\beta_2 + \beta_3 GRE) * GPA + \epsilon \end{aligned}$$

- Ex. 2: Continuous and categorical feature
  - *Model: Suppose we look at research experience ("No" or "Yes") and GPA*
  - $admit = \beta_0 + \beta_1 Research + \beta_2 GPA + \beta_3 Research * GPA + \epsilon$

```
# Center CGPA
student_data <- student_data %>%
  mutate(CGPA_c = CGPA - mean(student_data$CGPA))

lm_fit <- lm(formula = admit~Research+CGPA_c+Research*CGPA_c, data=student_data)
tidy(lm_fit) %>%
  mutate(p.value=ifelse(p.value<0.005, "<0.005",
                        as.character(round(p.value, 2)))) %>%
  flextable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.699	0.0062	112.1	<0.005
Research	0.041	0.0081	5.0	<0.005
CGPA_c	0.178	0.0108	16.5	<0.005
Research:CGPA_c	0.022	0.0137	1.6	0.1

```
# Compare R2 without interaction
lm_fit_no_int <- lm(formula = admit~Research+CGPA_c, data=student_data)
r2_no_int <- round(summary(lm_fit_no_int)$r.squared, 3)
r2_int <- round(summary(lm_fit)$r.squared, 3)
```

- R^2 for interaction model is 0.777 compared to 0.776 without the interaction

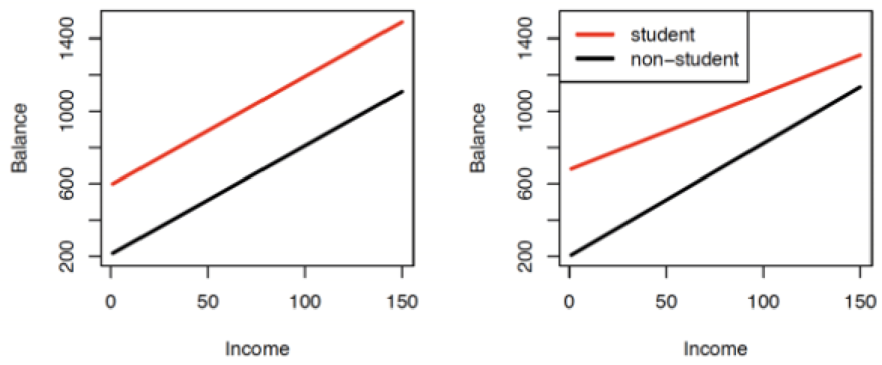
Interactions:

- Ex. 2: Continuous and categorical feature
  - *No interaction term*

$$admit \approx \beta_0 + \beta_1 Research + \beta_2 GPA$$
$$= \beta_2 GPA + \begin{cases} \beta_0 & Research = \text{No} \\ \beta_0 + \beta_1 & Research = \text{Yes} \end{cases}$$

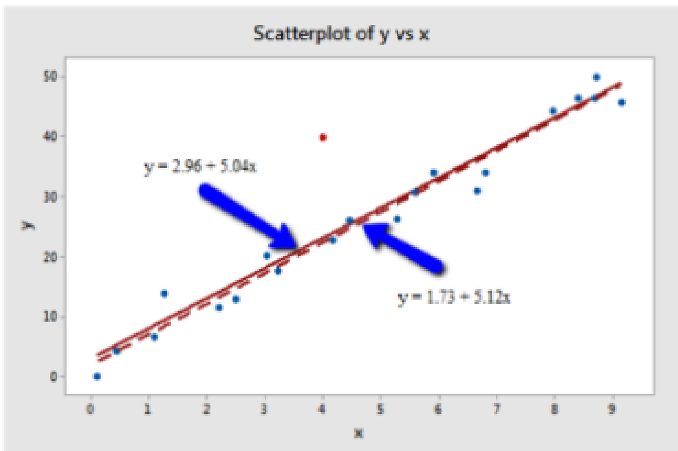
- With interaction term

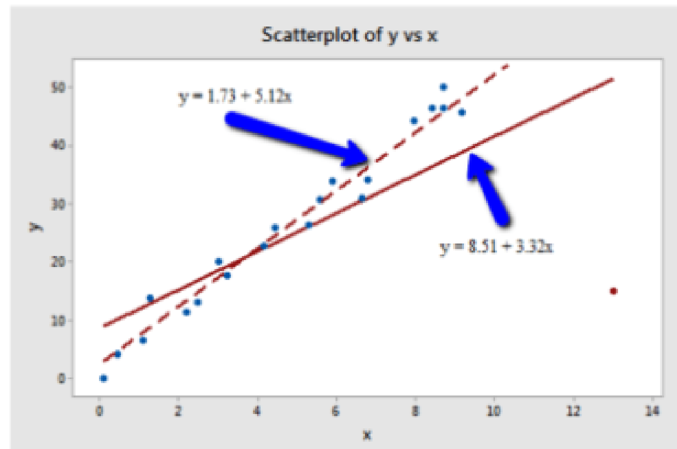
$$admit \approx \beta_0 + \beta_1 Research + \beta_2 GPA + \beta_3 Research * GPA$$
$$= \begin{cases} \beta_0 + \beta_2 GPA & Research = \text{No} \\ (\beta_0 + \beta_1) + (\beta_2 + \beta_3) GPA & Research = \text{Yes} \end{cases}$$


Balance
200
600
1000
1400
0
50
100
150
Income
student
non−student

Outliers and High Leverage Points

- An **outlier** is a data point whose response *y* does not follow the general trend of the rest of the data.
- A data point has high **leverage** if it has "extreme" predictor *x* values.





Residual plot can be used to identify outliers.

Leverage statistic: $h_i = \frac{1}{n} + \frac{(x_i - \bar{x})}{\sum_{j=1}^{n}(x_j - \bar{x})}$