

BIOS 635: K-Nearest Neighbor and Linear Regression

Kevin Donovan

1/28/2021

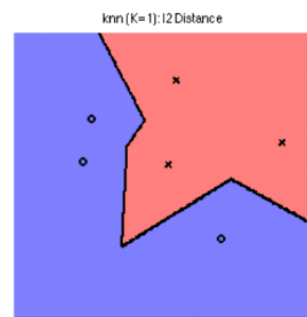
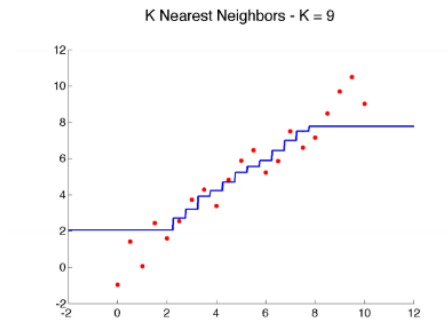
Review

- Homework 1 due on today at 11 PM 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 $\{j_1, \dots, j_K\}$ 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: $\text{sign}(\sum_{k=1}^K y_{j_k})$



KNN Examples: Regression

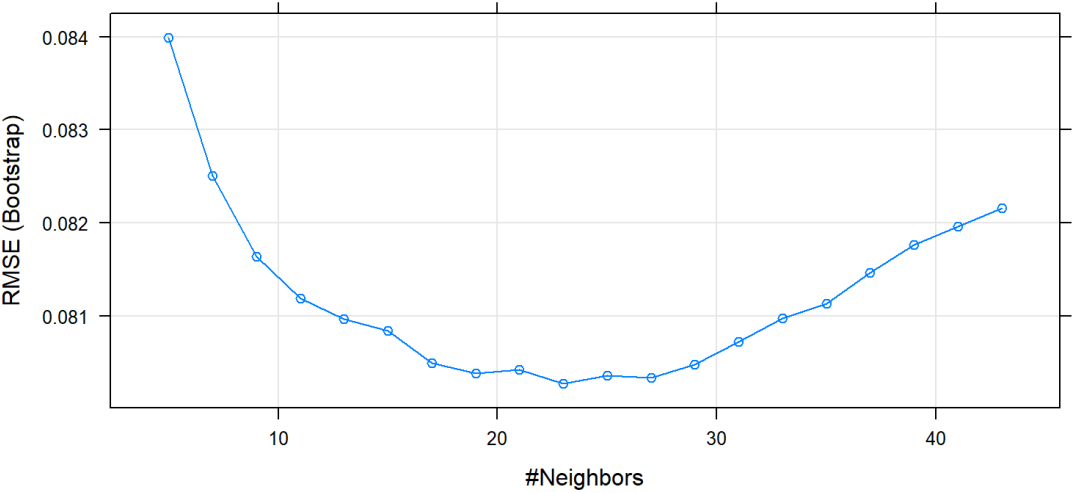
- Suppose we want to predict chance at getting admitted to graduate school based on
 - GRE (GRE Score)
 - GPA (CGPA)
- First, we **don't** split the data

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

# Look at tuning results
knnFit
```

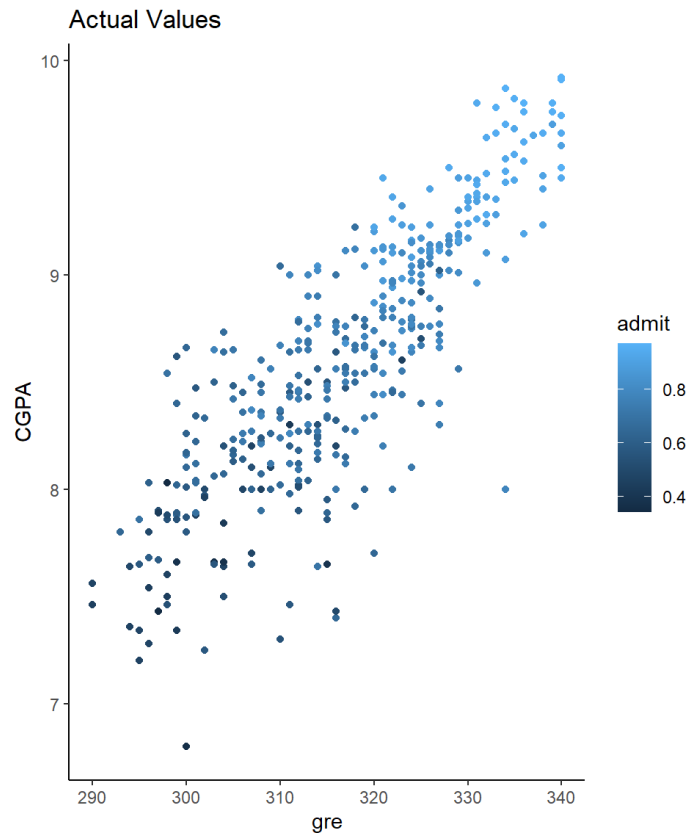
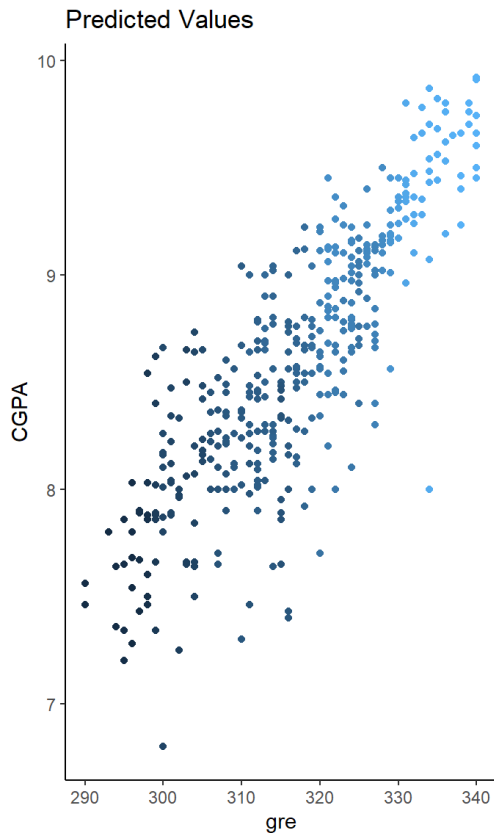
```
## k-Nearest Neighbors
##
## 400 samples
## 2 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 400, 400, 400, 400, 400, 400, ...
## Resampling results across tuning parameters:
##
## k RMSE Rsquared MAE
## 5 0.08399026 0.6562708 0.06198492
## 7 0.08250409 0.6657843 0.06149025
## 9 0.08164273 0.6721247 0.06108271
## 11 0.08119285 0.6745646 0.06083617
## 13 0.08096974 0.6762516 0.06057670
## 15 0.08083990 0.6772038 0.06050660
## 17 0.08049822 0.6797874 0.06026656
## 19 0.08038855 0.6805712 0.06016444
## 21 0.08042498 0.6804126 0.06019089
## 23 0.08027426 0.6818170 0.06004864
## 25 0.08035880 0.6810901 0.06020992
## 27 0.08033956 0.6813598 0.06026175
## 29 0.08048045 0.6803138 0.06043737
## 31 0.08072203 0.6784773 0.06083795
## 33 0.08097813 0.6764790 0.06116968
## 35 0.08113881 0.6753223 0.06121701
## 37 0.08146372 0.6729074 0.06161808
## 39 0.08176444 0.6704537 0.06187339
## 41 0.08196063 0.6690050 0.06220046
## 43 0.08215608 0.6674638 0.06244537
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 23.
```

```
plot(knnFit)
```



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)

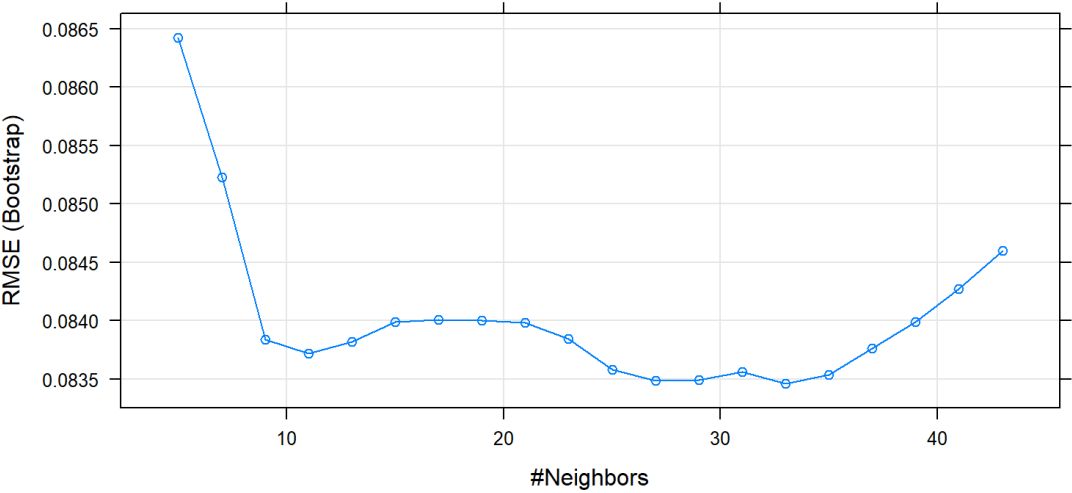
set.seed(12)
student_data_tt_index <- createDataPartition(student_data$admit, p=0.6, list = FALSE)
student_data_train <- student_data[student_data_tt_index,]
student_data_test <- student_data[-student_data_tt_index,]

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

# Look at tuning results
knnFit
```

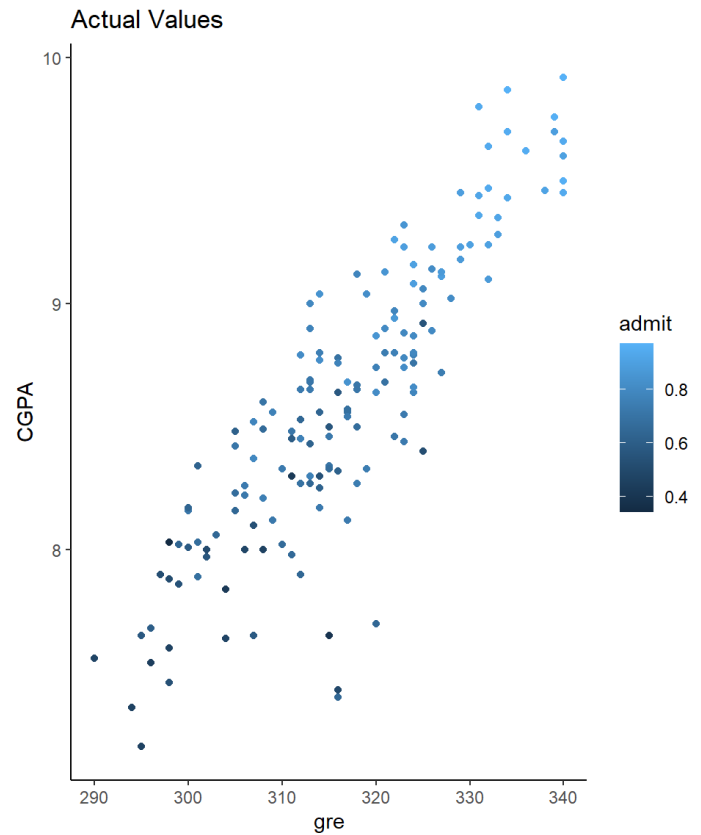
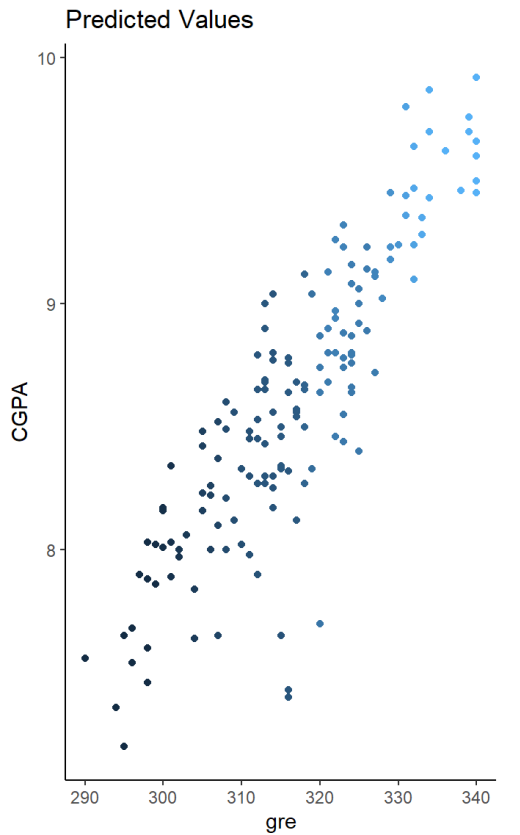
```
## k-Nearest Neighbors
##
## 241 samples
## 2 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 241, 241, 241, 241, 241, 241, ...
## Resampling results across tuning parameters:
##
## k    RMSE      Rsquared    MAE
## 5    0.08642279 0.6348680 0.06442511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



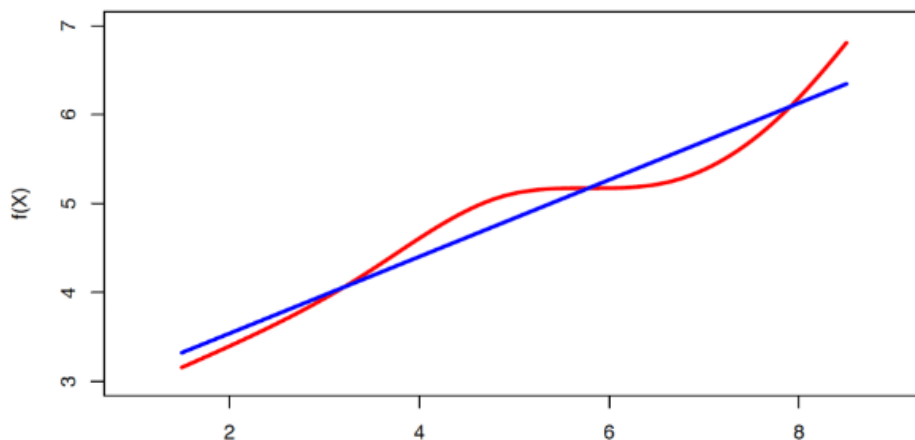
KNN Examples: Regression

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



Linear Regression

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



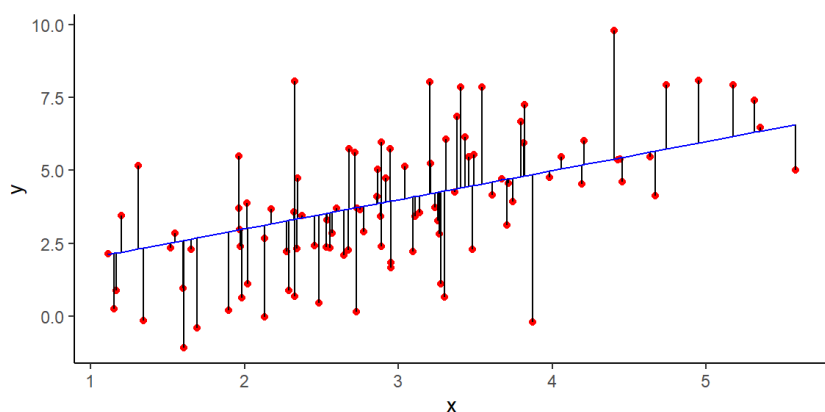
Simple Linear Regression

- **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 $\{(x_1, y_1), \dots, (x_n, y_n)\}$
- 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

$$\begin{aligned}\hat{\beta}_1 &= \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2} \\ &= r_{xy} \frac{\hat{\sigma}_y}{\hat{\sigma}_x}\end{aligned}$$

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$$

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 $\text{Var}(\epsilon) = \sigma^2$

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

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

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

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

- *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 hypothesis* 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 its distribution under H_0
 - For linear regression, test statistic is

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

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

Model Assumptions

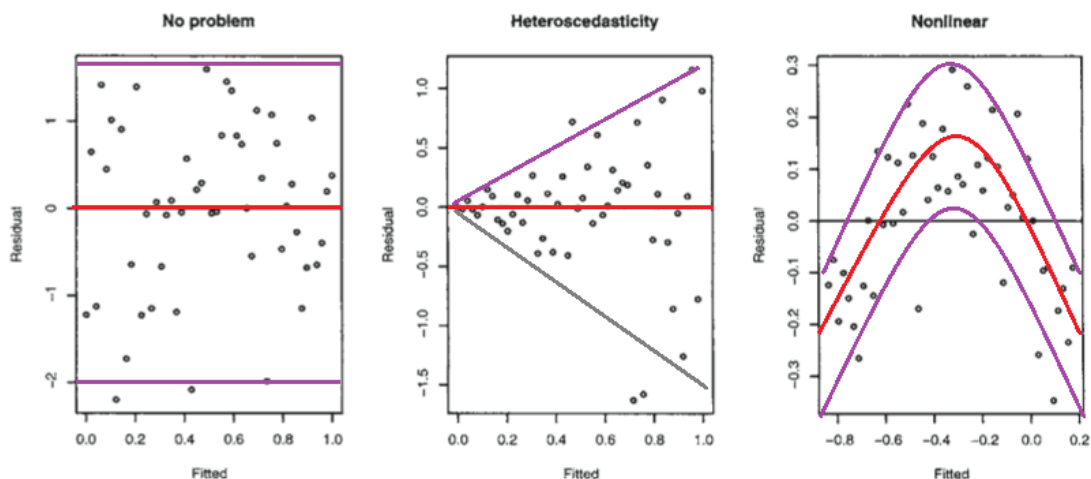
- Note that above **distribution** for test statistic only hold if

1. Usual linear regression **assumptions met**:

- Homoskedasticity
- Independent residuals
- **and** one of the following

2. Normally distributed residuals

3. Large sample approximation



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

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

- **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:

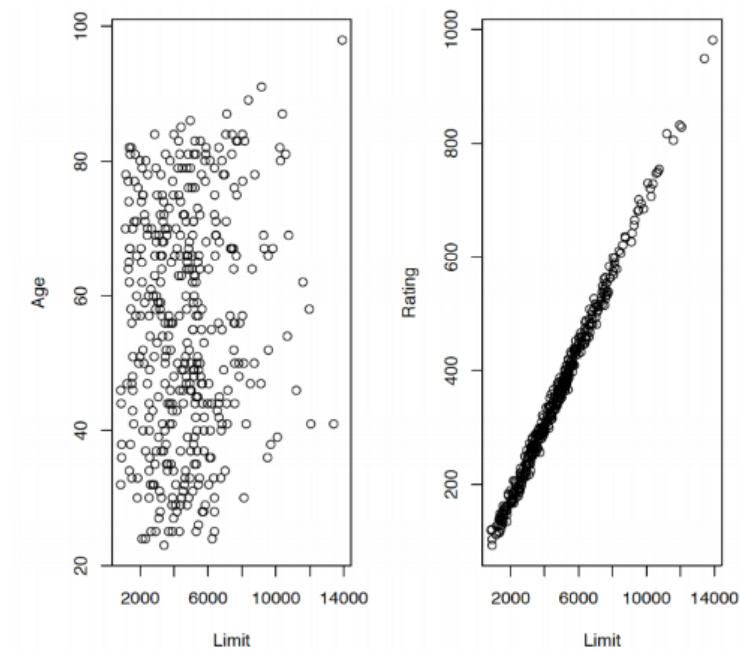
- $admit = \beta_0 + \beta_1 GRE + \beta_2 GPA + \epsilon$

- **Estimation:**

- Again, least squares
- $RSS = \sum_{i=1}^n (e_i)^2 = \sum_{i=1}^n (y - \hat{\beta}_0 - \hat{\beta}_1 X_1 - \dots - \hat{\beta}_p X_p)^2$

Collinearity

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



Example

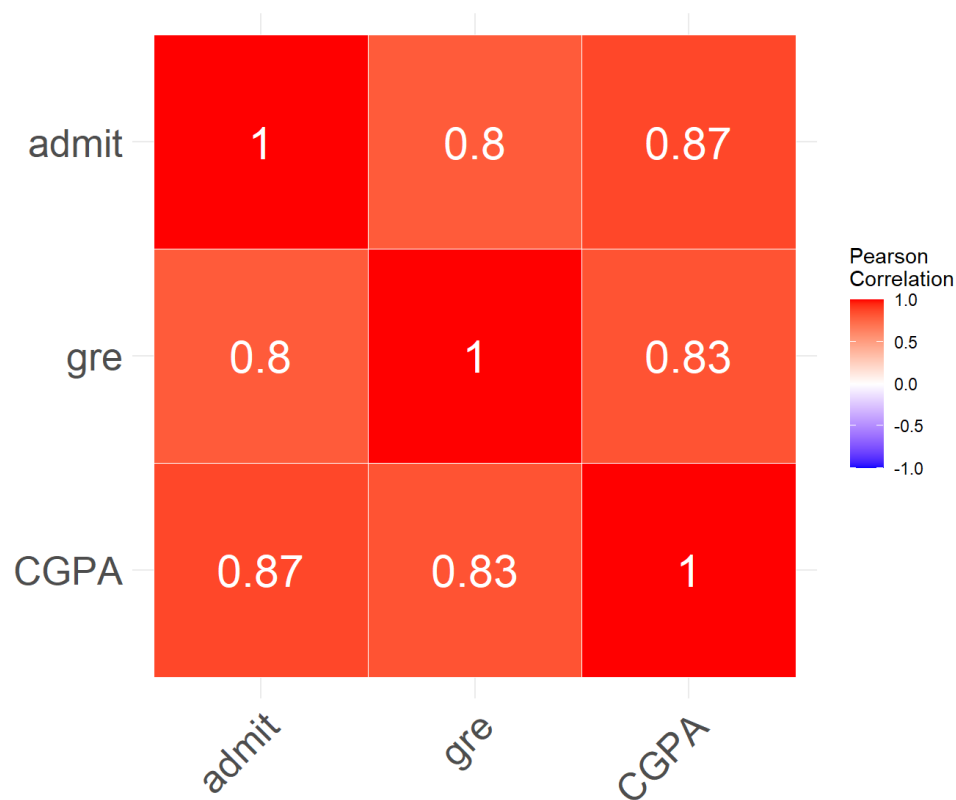
- Let's use regression instead of KNN to predict admission chance
- Model: $\text{admit} = \beta_0 + \beta_1 \text{GRE} + \beta_2 \text{GPA} + \epsilon$

```
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 $(0, 1, \dots, M)$ 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:

$$admit = \beta_0 + \beta_1prestige_1 + \beta_2prestige_2 + \beta_3prestige_3 + \beta_4prestige_4 + \epsilon$$

```
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**

- $admit = \beta_0 + \beta_1 GRE + \beta_2 GPA + \epsilon$
- *Change in average 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:

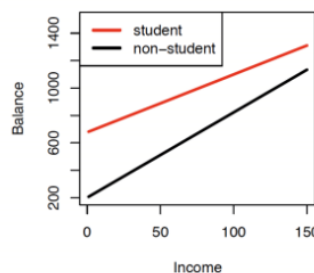
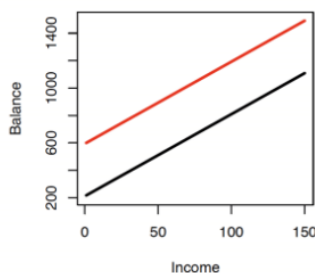
■ Ex. 2: Continuous and categorical feature

- No interaction term

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

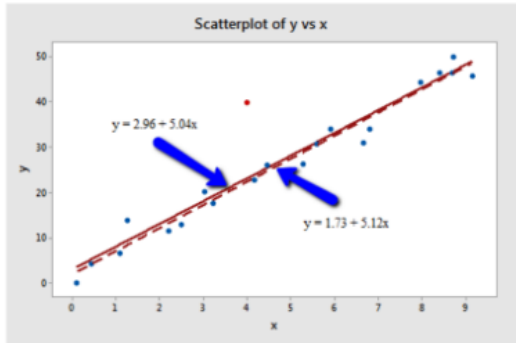
■ With interaction term

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

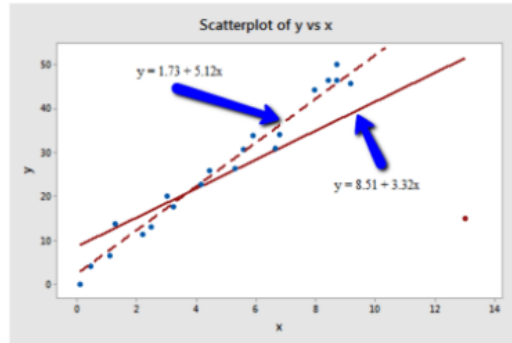


Outliers and High Leverage Points

- 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:
$$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{j=1}^n (x_j - \bar{x})^2}$$