Machine Learning in Public Health

Lecture 4: K -Nearest Neighbors Classification and Receiver Operating Characteristics (ROC)

Dr. Yang Feng

Today's agenda

- K-Nearest Neighbors Classification
- Receiver Operating Characteristics (ROC)

Review on K -Nearest Neighbors Regression

- Given the training data $\{(x_i, y_i), i = 1, \cdots, n\}$.
- Main Idea of KNN: given a new observation x_0, find the K **nearest** observations among $\{x_i, i = 1, \cdots, n\}$.
- What is near? Measure by the Euclidean Distance $\|x_i - x_0\|_2^2 = (x_i - x_0)^2$.
- Let's call the set N_0. Then, we have

$$\hat{y}_0 = \frac{1}{K} \sum_{i \in N_0} y_i.$$

K -Nearest Neighbors Classification

- Difference: now, the response is categorical, say $y_i \in \{1, 2, \cdots, L\}$.

- Same process for finding the K **nearest** observations.

- Main difference: we can't simply average the y_{i_j}'s at the end. Instead, we estimate the conditional probability for class j as the fraction of points in N_0 whose response values equal j:

$$P(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j),$$

where $j = 1, \cdots, J$.

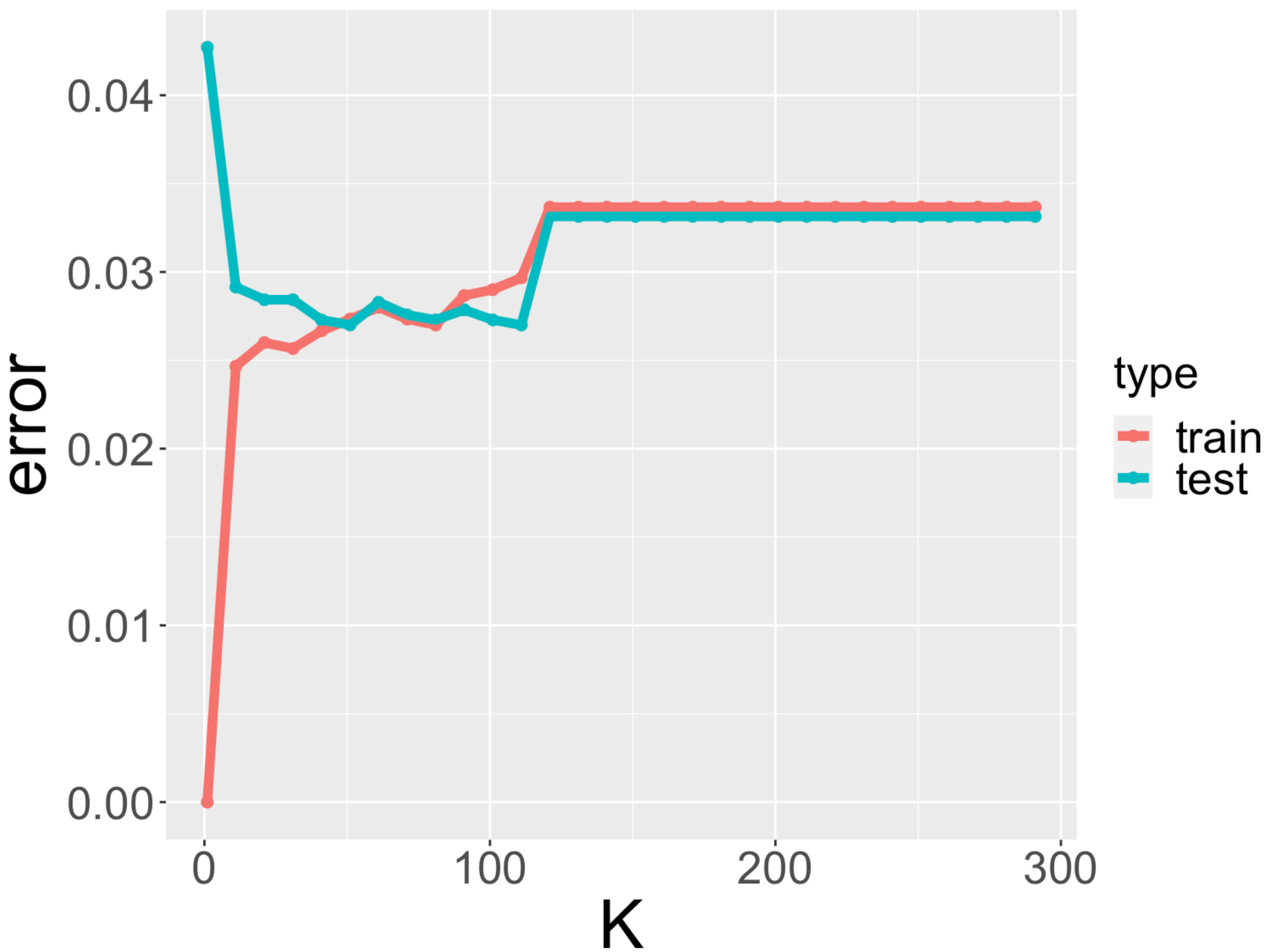
- Then, we choose class that with the largest probability.

$$\hat{y}_0 = \arg\max_j P(Y = j|X = x_0)$$

- Same as Logistic Regression, LDA, and QDA!

Predict `default` using balance

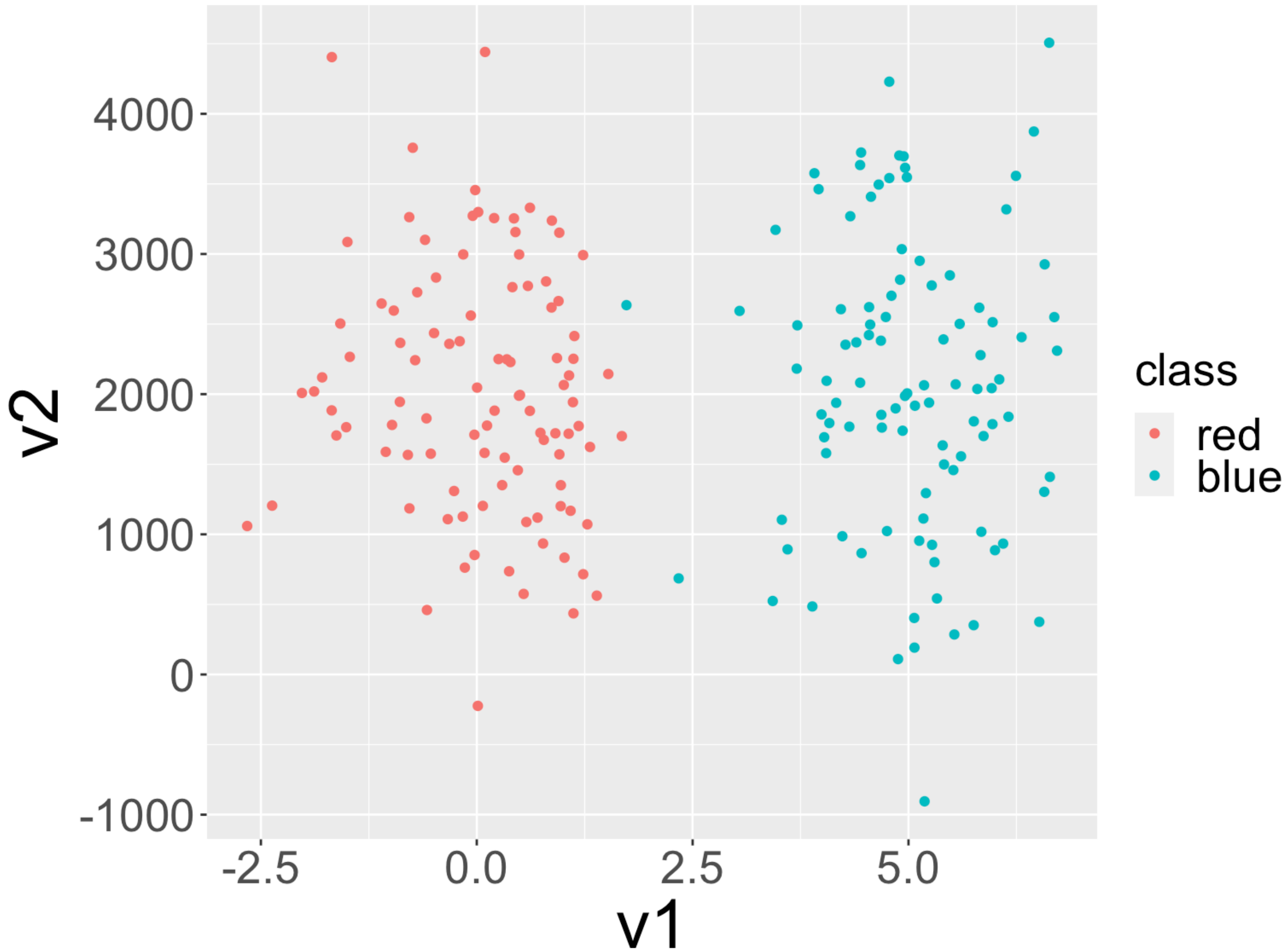
- Prepare the training and test data
- Run KNN on training data with varying K values



Importance of standardization before applying KNN

```r
library(MASS)
n <- 100
set.seed(0)
mu1 <- c(0, 2)
mu2 <- c(5, 2)
Sigma <- matrix(c(1, 0, 0, 1), 2, 2)
x1 <- mvrnorm(n, mu1, Sigma)
x2 <- mvrnorm(n, mu2, Sigma)
dat <- rbind(data.frame(v1 = x1[, 1], v2 = 1000*x1[, 2], class = "red"),
             data.frame(v1 = x2[, 1], v2 = 1000*x2[, 2], class = "blue"))

x1 <- mvrnorm(n, mu1, Sigma)
x2 <- mvrnorm(n, mu2, Sigma)
test_dat <- rbind(data.frame(v1 = x1[, 1], v2 = 1000*x1[, 2], class = "red"),
             data.frame(v1 = x2[, 1], v2 = 1000*x2[, 2], class = "blue"))
ggplot(dat) + geom_point(mapping = aes(x = v1, y = v2, color = class)) + mytheme
```

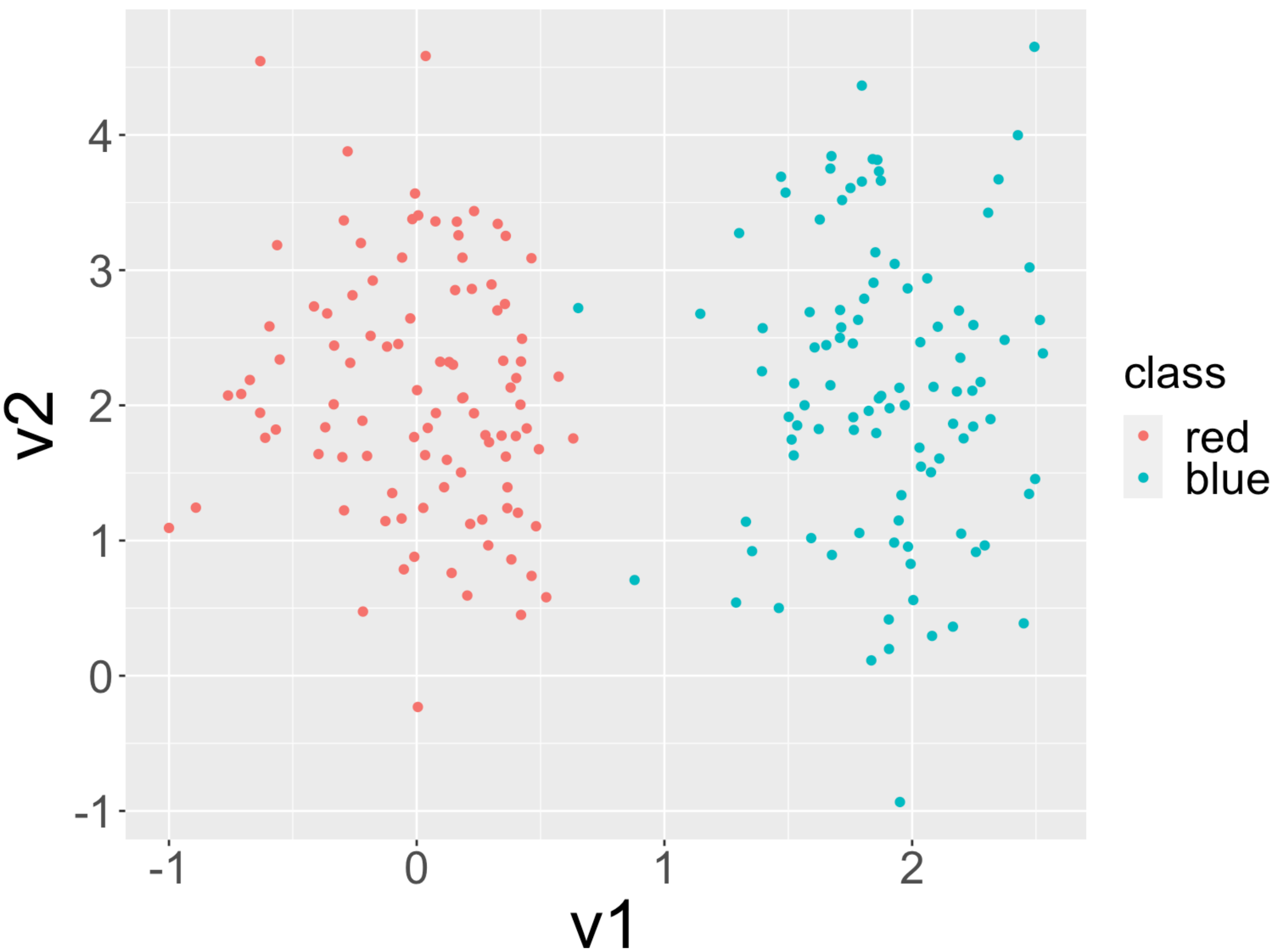



```r
dat$class <- as.factor(dat$class)
test_dat$class <- as.factor(test_dat$class)
fit <- knn3(class ~ ., data = dat, k = 5)
ypred <- predict(fit, newdata = test_dat, type = "class")
mean(ypred != test_dat$class)
```

```
## [1] 0.535
```

After standardization

```
fit_std <- preProcess(dat, method = "scale")
dat_std <- predict(fit_std, newdata = dat)
test_dat_std <- predict(fit_std, newdata = test_dat)
ggplot(dat_std) + geom_point(mapping = aes(x = v1, y = v2, color = class)) + mytheme
```




```
fit <- knn3(class ~ ., data = dat_std, k = 5)
ypred <- predict(fit, newdata = test_dat_std, type = "class")
mean(ypred != test_dat_std$class)
```

```
## [1] 0.005
```

Threshold for Binary Classification

- Recall in logistic regression, once we have the estimated probability $P(Y = 1|X = x_0)$, we compare it with 0.5. Clearly, we can change the threshold 0.5.

- Fit the logistic regression model on the training data.

```
fit_logi <- glm(default ~ balance + income, data = default_tr,
        family='binomial');
pred_train_prob <- predict(fit_logi, type = 'response')
pred_train_label <- ifelse(pred_train_prob > 0.5, 'Yes', 'No')
table(true = default_tr$default, predicted = pred_train_label) #Confusion
        Matrix/Table
```

```
##      predicted
## true    No  Yes
##   No  2888   11
##   Yes   65   36
```

Confusion Matrix/Table

		Predicted class		
		− or Null	+ or Non-null	Total
True class	− or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
	Total	N*	P*	

Some additional measures

Name	Definition	Synonyms
False Pos. rate	FP/N	Type I error, 1−Specificity
True Pos. rate	TP/P	1−Type II error, power, sensitivity, recall
Pos. Pred. value	TP/P*	Precision, 1−false discovery proportion
Neg. Pred. value	TN/N*	

FPR vs. TPR

- Let' compute FPR and TPR

```
FP <- sum(default_tr$default == "No" & pred_train_label == "Yes")
TP <- sum(default_tr$default == "Yes" & pred_train_label == "Yes")
N <- sum(default_tr$default == "No")
P <- sum(default_tr$default == "Yes")
FPR <- FP/N
TPR <- TP/P
FPR
```

```
## [1] 0.003794412
```

```
TPR
```

```
## [1] 0.3564356
```

Adjusting the threshold

- Taking another look at the confusion matrix

```
##      predicted
## true    No  Yes
##   No  2888   11
##   Yes   65   36
```

- We missed too many default cases. Not what the credit card company wants!

- Should we decrease the threshold 0.5 , or increase it?

Change the decision threshold

- Decrease the threshold

```
pred_train_label_2 <- ifelse(pred_train_prob > 0.3, 'Yes', 'No')
table(true = default_tr$default, predicted = pred_train_label_2)
```

```
##       predicted
## true     No  Yes
##    No  2850   49
##    Yes   46   55
```

- Increase the threshold

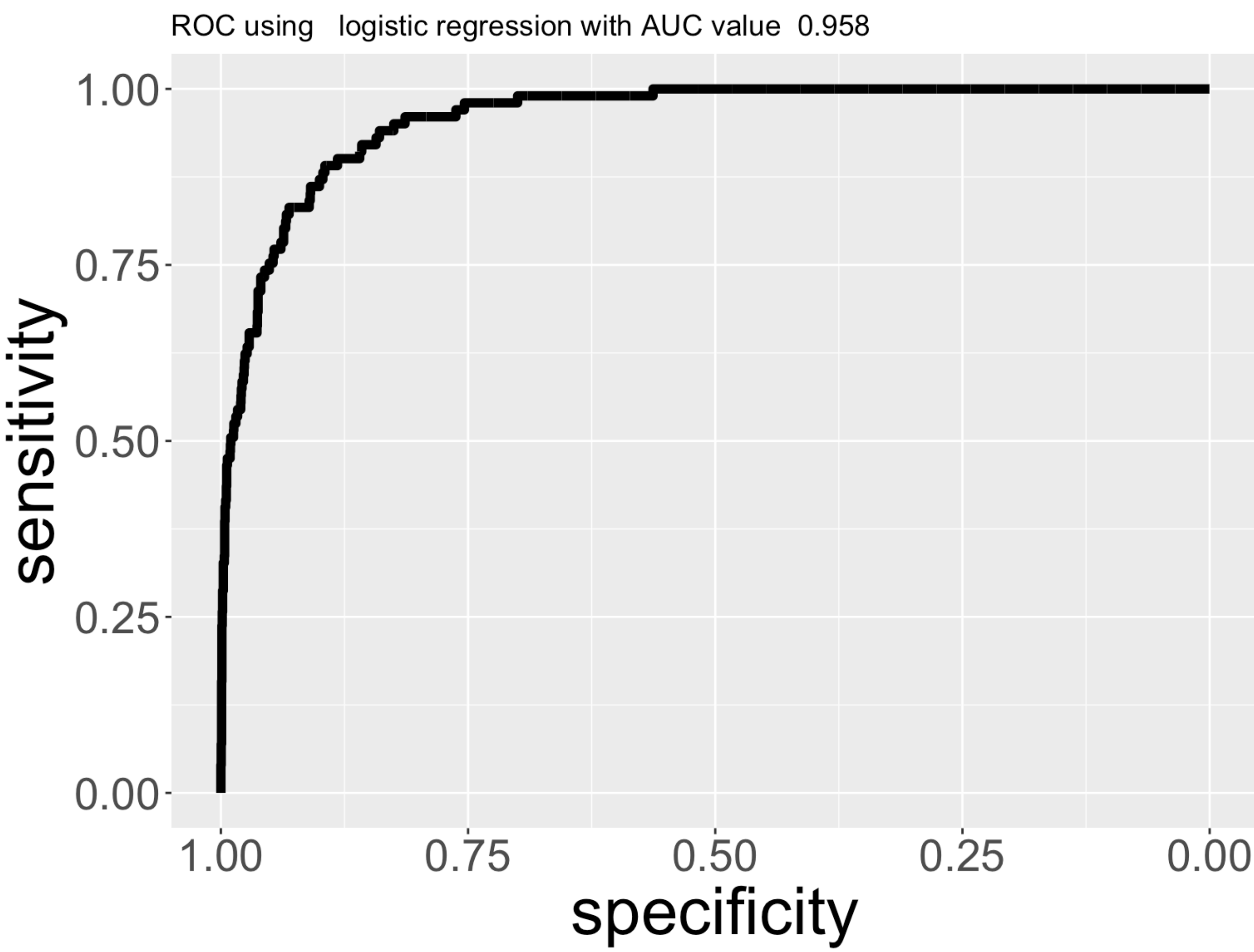
```
pred_train_label_2 <- ifelse(pred_train_prob > 0.7, 'Yes', 'No')
table(true = default_tr$default, predicted = pred_train_label_2)
```

```
##       predicted
## true     No  Yes
##    No  2896    3
##    Yes   80   21
```

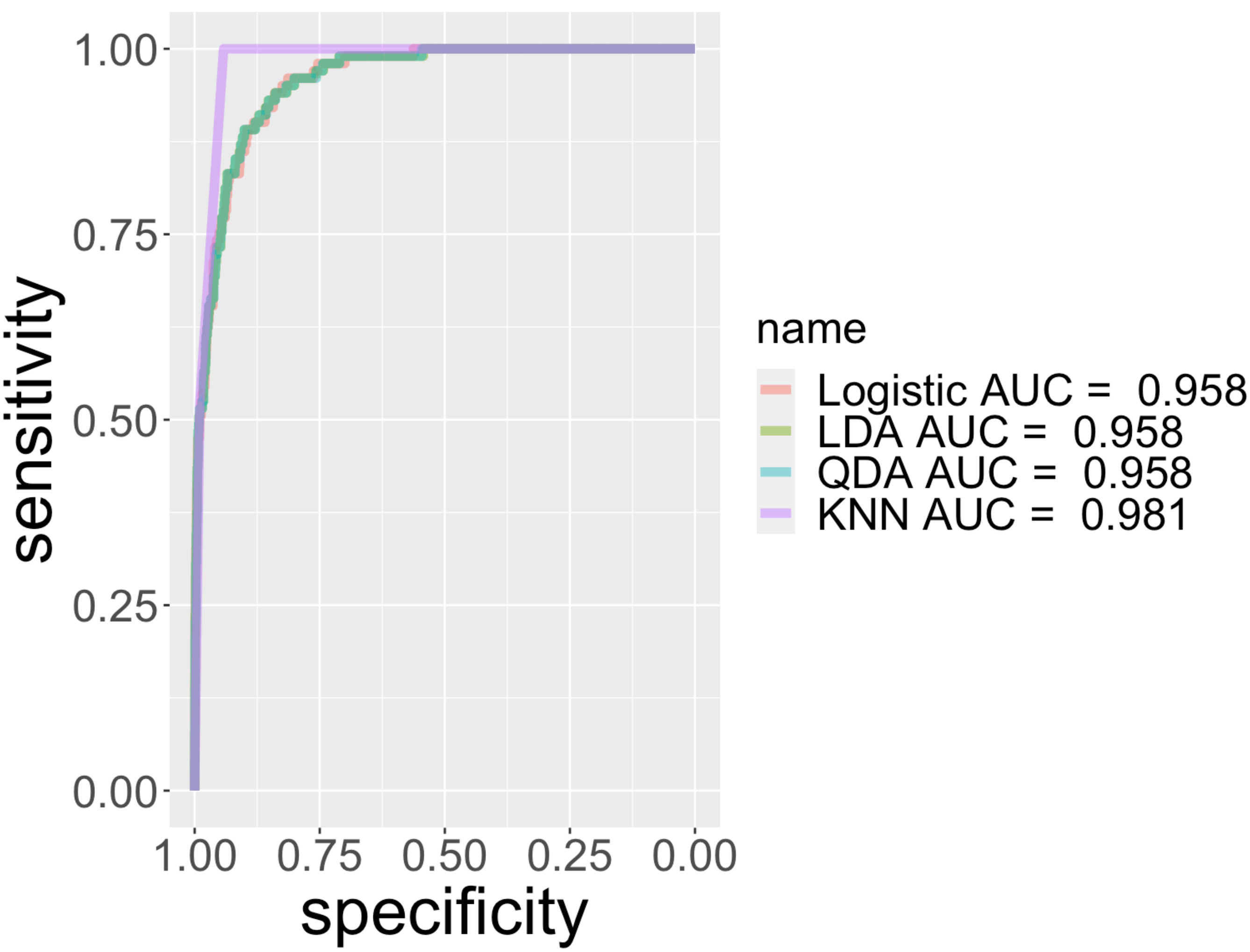
Receiver Operating Characteristics (ROC)

- Most common way to visualize the trade-off between True Positive Rate (TPR) and False Positive Rate (FPR) when the threshold changes.

- Using the **pROC** package.



Comparing different methods via `ggroc()`



Next Class

- Resampling Methods
  - *Cross-Validation*
  - *Bootstrap*