Intro to Data Science - part III

Predictive Modelling

Sotirios Damouras

24-11-2023

The Data Science and Statistics Society

Table of contents

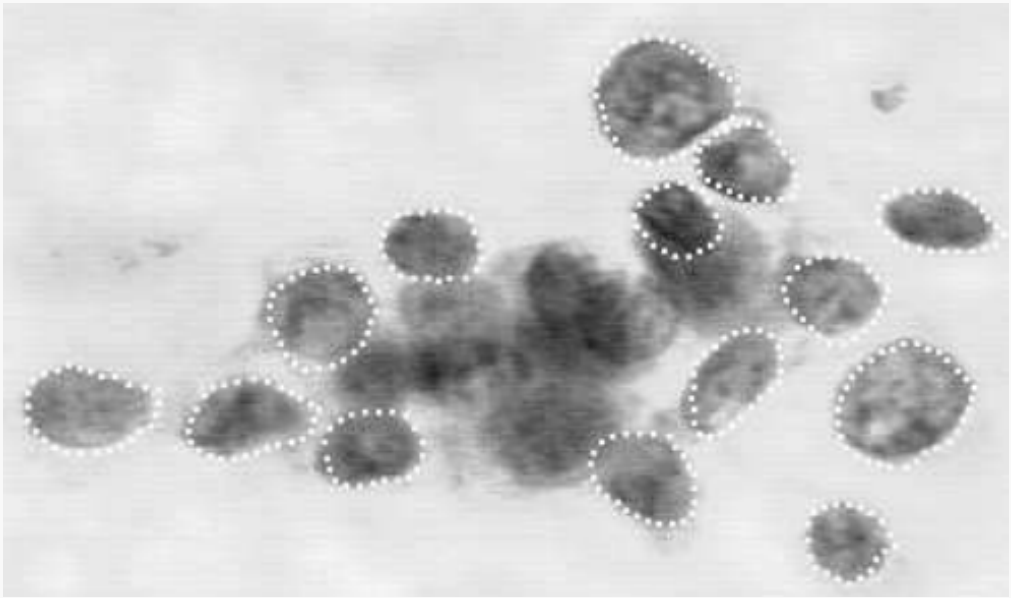
Exploring the Data Set

Wisconsin Diagnostic Breast Cancer [WDBC] Data

- Fine-Needle Aspiration (FNA) data on 569 patients
  - 212 w/ malignant & 357 w/ benign tumors
  - available from UCI data repository
- FNA biopsy withdraws small amount of tissue/fluid from susipcious area; The sample is then checked for cancer cells

WDBC Data

- Calculate tumor cell *features* based on FNA images



WDBC Features

For each cell, calculate

1. *radius* (mean distance from center to perimeter)
2.
3.
4.
5.
6.
7.
8.

WDBC Features

For each cell, calculate

1. *radius* (mean distance from center to perimeter)
2. *texture* (standard deviation of gray-scale values)
3.
4.
5.
6.
7.
8.

WDBC Features

For each cell, calculate

1. *radius* (mean distance from center to perimeter)
2. *texture* (standard deviation of gray-scale values)
3. *perimeter, area*
4.
5.
6.
7.
8.

WDBC Features

For each cell, calculate

1. *radius* (mean distance from center to perimeter)
2. *texture* (standard deviation of gray-scale values)
3. *perimeter, area*
4. *smoothness* (local variation in radius lengths)
5.
6.
7.
8.

WDBC Features

For each cell, calculate

1. *radius* (mean distance from center to perimeter)
2. *texture* (standard deviation of gray-scale values)
3. *perimeter, area*
4. *smoothness* (local variation in radius lengths)
5. *compactness* ($\frac{\text{perimeter}^2}{\text{area}} - 1$)
6.
7.
8.

WDBC Features

For each cell, calculate

1. *radius* (mean distance from center to perimeter)
2. *texture* (standard deviation of gray-scale values)
3. *perimeter, area*
4. *smoothness* (local variation in radius lengths)
5. *compactness* ($\frac{\text{perimeter}^2}{\text{area}} - 1$)
6. *concavity* (severity of concave portions of contour
7.
8.

WDBC Features

For each cell, calculate

1. *radius* (mean distance from center to perimeter)
2. *texture* (standard deviation of gray-scale values)
3. *perimeter, area*
4. *smoothness* (local variation in radius lengths)
5. *compactness* ($\frac{\text{perimeter}^2}{\text{area}} - 1$)
6. *concavity* (severity of concave portions of contour
7. *concave points* (number of concave portions of contour)
8.

WDBC Features

For each cell, calculate

1. *radius* (mean distance from center to perimeter)
2. *texture* (standard deviation of gray-scale values)
3. *perimeter, area*
4. *smoothness* (local variation in radius lengths)
5. *compactness* ($\frac{\text{perimeter}^2}{\text{area}} - 1$)
6. *concavity* (severity of concave portions of contour
7. *concave points* (number of concave portions of contour)
8. *symmetry, fractal dimension*

WDBC Features

- For each image (group of cells), report
  1. *mean* (.m)
  2. *standard deviation* (.se)
  3. *worst* value (.w)
- In total, $3 \times 10 = 30$ features

WDBC Data

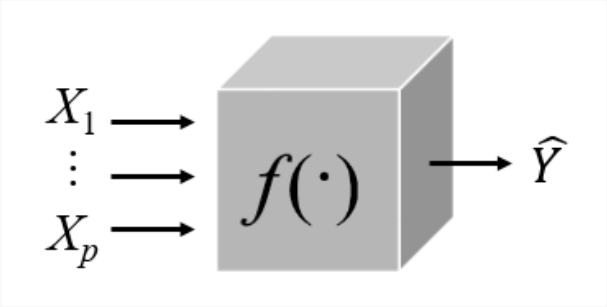
Lets run the read chunk in the starter file!

```
wdbc = read_csv("data/edbc.csv")
glimpse(wdbc)
```

Classification

Classification

- Create system that predicts *label* Y based on *feature* variables $X_1, ..., X_p$
- Called *binary* classification when Y takes only 2 values
- Find function $f(\cdot)$ such that $f(X_1, ..., X_p) = \hat{Y} \approx Y$



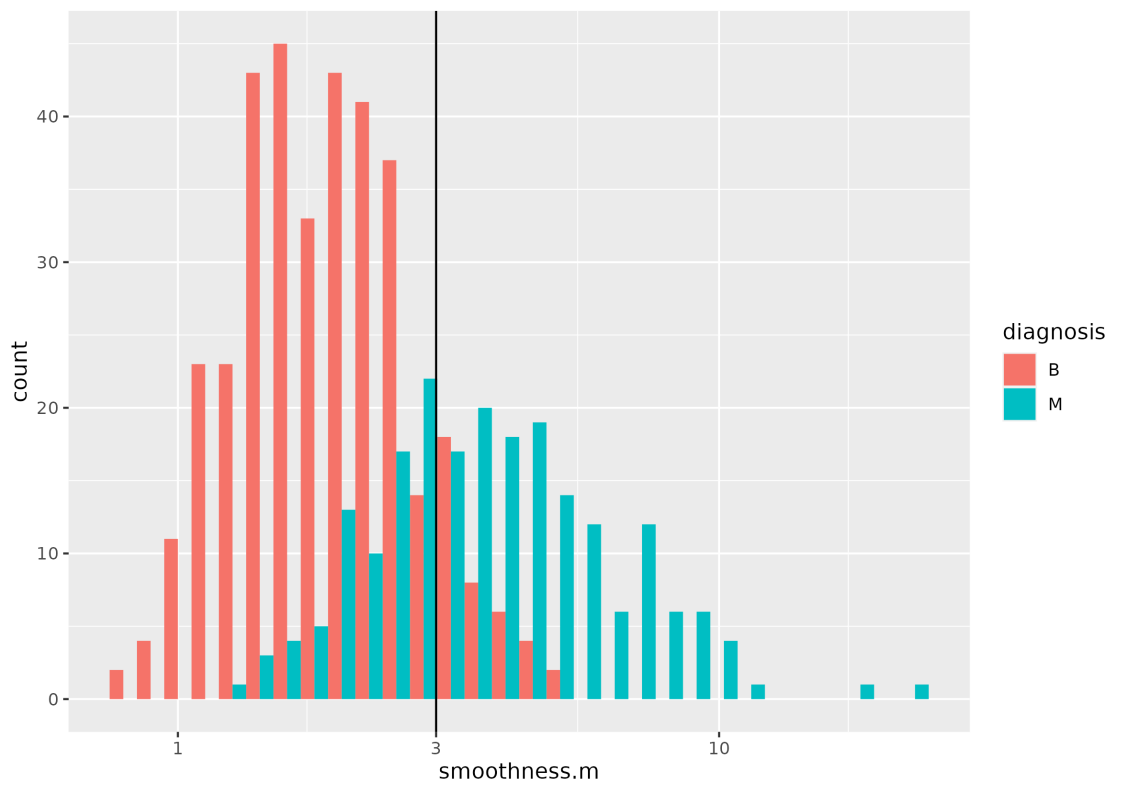
- Several methods to arrive at $f(\cdot)$; Some work better on certain problems than others

Threshold Classification

- Mean cell smoothness is indicative of cancer

```
wdbc %>% ggplot(aes(x = smoothness.m, fill = diagnosis )) +
geom_histogram(position = "dodge", bins=30) +
geom_vline(xintercept = 3) + scale_x_log10()
```

Threshold Classification



Classification Accuracy

Accuracy : Proportion of *correct* predictions

- Does not differentiate classes (all equally important)
- Compare to *naive* majority classifier
  - For WDBC, predict benign

```
wdbc %>% mutate( naive = "B",
predicted = ifelse( smoothness.m > 3, "M", "B")) %>%
summarise( acc.pred = mean(predicted == diagnosis),
           acc.naiv = mean(naive == diagnosis) )
```

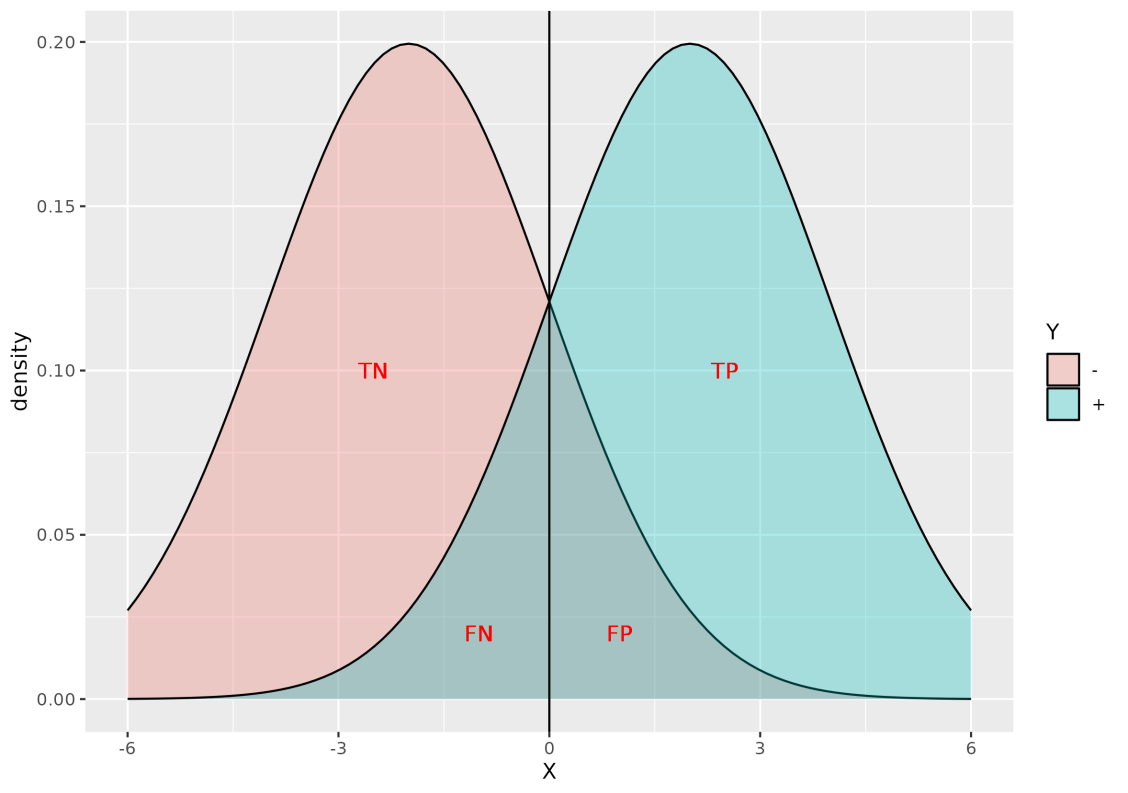
Confusion Matrix

- Confusion Matrix is a more *nuanced* performance measure
- Assuming class of interest is *positive*

	Actual Positive	Actual Negative	Sum
Predict Positive	True Positive (*TP*)	False Positive (*FP*)	$PP = TP + FP$
Predict Negative	False Negative (*FN*)	True Negative (*TN*)	$PN = FN + TN$
Sum	$P = TP + FN$	$N = FP + TN$	

- What would be Type I/II Error in hypothesis testing?

Confusion Matrix



Example

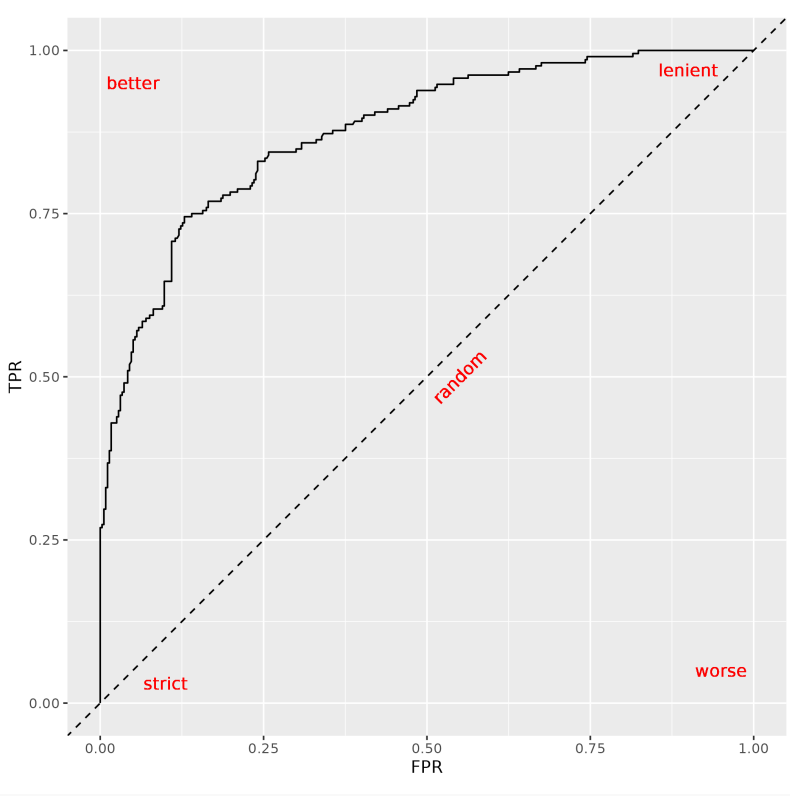
```
wdbc %>%
mutate( predicted = ifelse( smoothness.m > 3, "M", "B") ) %>%
mutate( predicted = fct_relevel(predicted, "M"),
        diagnosis = fct_relevel(diagnosis, "M") ) %>%
xtabs( ~ predicted + diagnosis, data = .) %>% addmargins()
```

Classifier Performance

Sensitivity/Recall/True Positive Rate (TPR) : *TP/P*
Precision/Positive Predicitive Value (PPV) : *TP/PP*
Specificity/True Negative Rate (TNR) : *TN/N*
False Positive Rate (FPR) : *FP/N = 1 - TPR*
F1-measure : $\left(2 \times \frac{\text{Sens.} \times \text{Prec.}}{\text{Sens.} + \text{Prec.}}\right)$ *Closer to 1 is better*

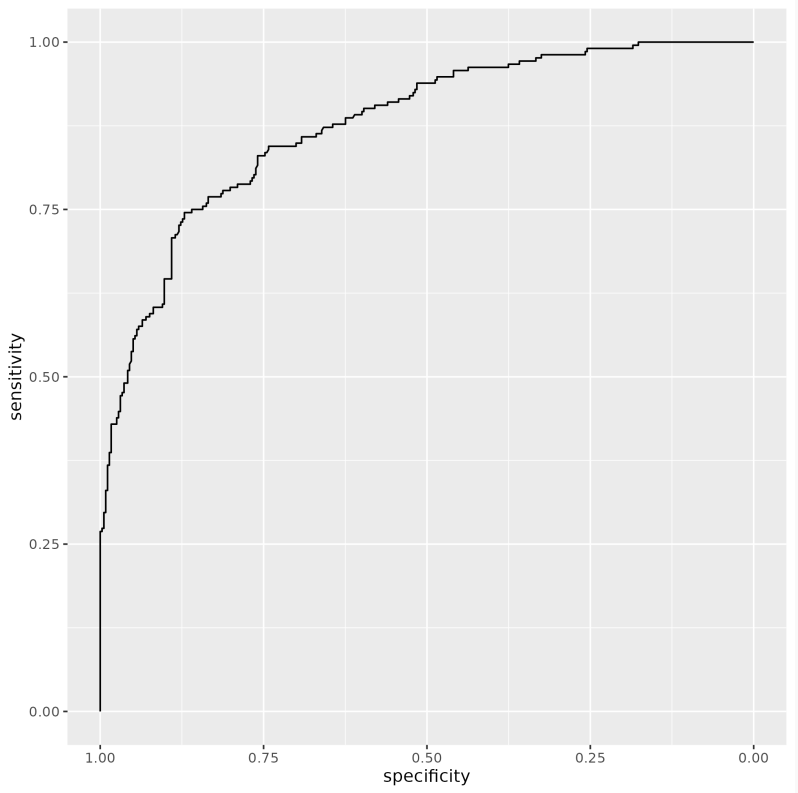
ROC curve

ROC curve : plot of FPR vs TPR for *all possible thresholds* (configurations) of binary classifier



Example

```
ROC_out = roc(diagnosis ~
smoothness.m, data = wdbc)
ggroc(ROC_out)
```



ROC curve

- Classifiers with ROC curve *above and to the left* are *better*

  – Compare classifiers *irrespective* of threshold – Difficult to compare curves that cross
- *Area under curve* (AUC) used as a proxy for classifier performance

```
auc(diagnosis ~ smoothness.m,  data = wdbc) # auc(ROC_out)
```

Multivariate Classification

Multiple Features

- Information can be reflected in multiple variables

Feature Engineering : Which features should be *generated* from available information?

- E.g. in WDBC, features represent cell morphology

Feature Selection : Which features from *fixed* collection should one use?

- Not always optimal to use *all* features

WDBC Data

- Thresholding *individual* variables gives OK classifier
- Can it improve by combining *multiple* variables?

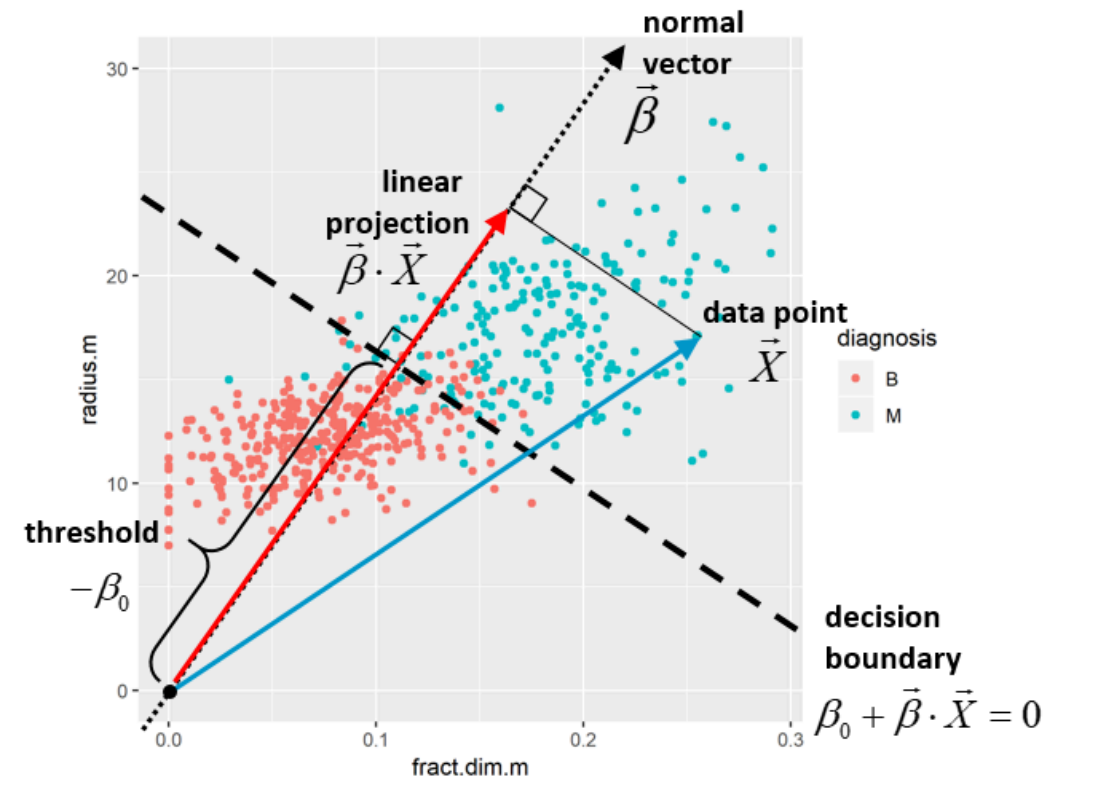
Linear Classifiers

- Linear (binary) classifier uses *linear decision boundary* (i.e. threshold) to classify observations
  - Boundary is line in 2D, plane in 3D feature space
- Class determined by distance (+/-) from boundary
  - Distance expressed as *linear function* of features
- Classification function reduces to

$$f(X_1, ..., X_p) = \text{sign}(\beta_0 + \beta_1 X_1 + ... + \beta_p X_p)$$

  - Parameter β_0 controls distance from *boundary*

Linear Classifiers



Linear Classifiers

- Several ways to find linear decision boundary
  - Logistic Regression
  - Linear Discriminant Analysis (LDA)
  - Support Vector Machines (SVM)
- ALl methosd produce *normal vector* of linear coefficients ($\vec{\beta}$)
- We will focus on logistic regression

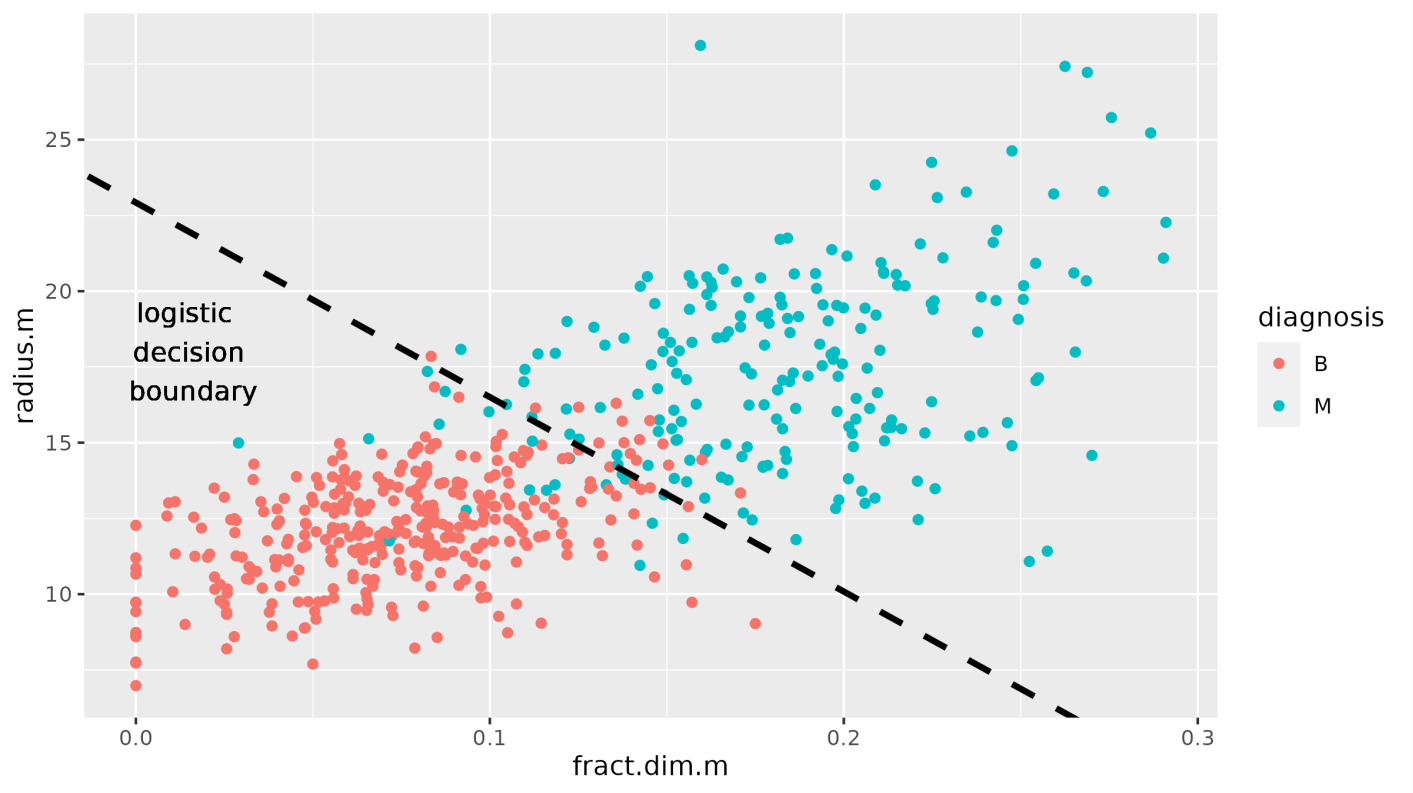
Logistic Regression

- Fit logistic regression with `glm()`

```
glm_out = glm( diagnosis ~ fract.dim.m + radius.m, family = "binomial",
data = wdbc %>% mutate( diagnosis = factor(diagnosis)) )
broom::tidy(glm_out)
```

- Decision boundary $-18.59 + 52.06X_1 + 0.81X_2 = 0$

Logistic Regression



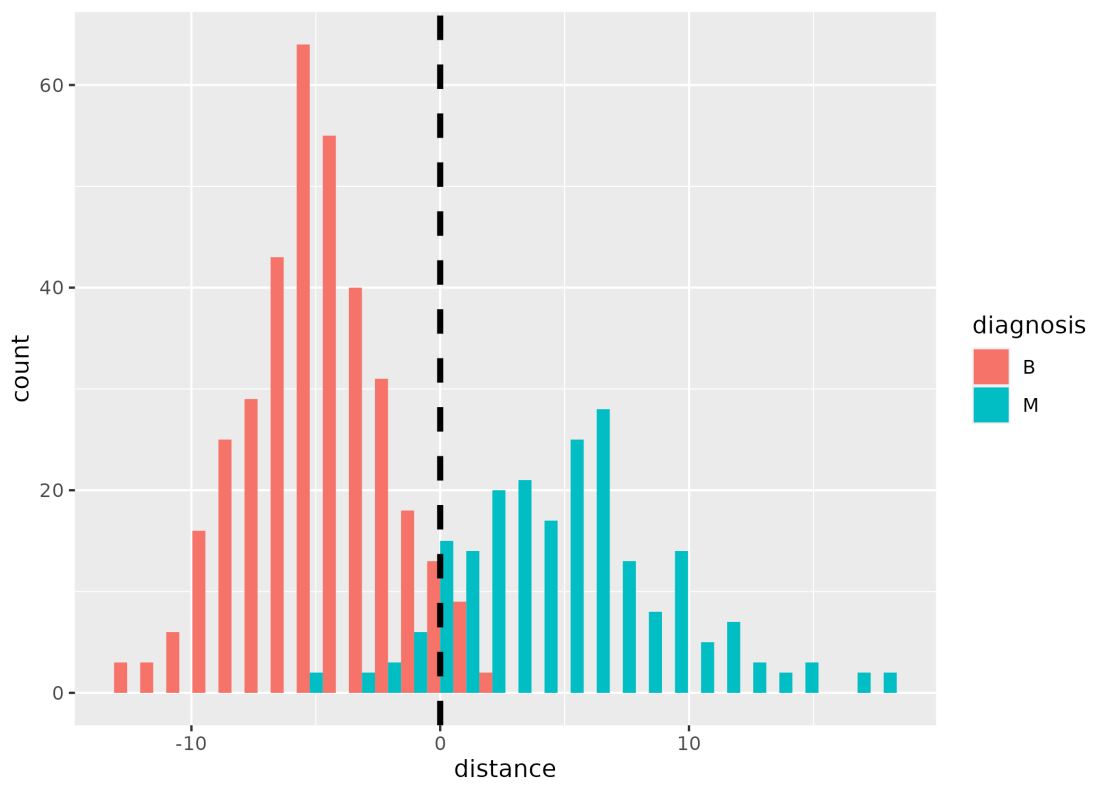
Logistic Regression

- `predict()` returns *distance* from *decision boundary*
  - i.e. projection+threshold

```
( glm_pred = predict(glm_out) ) %>% sample(5)
```

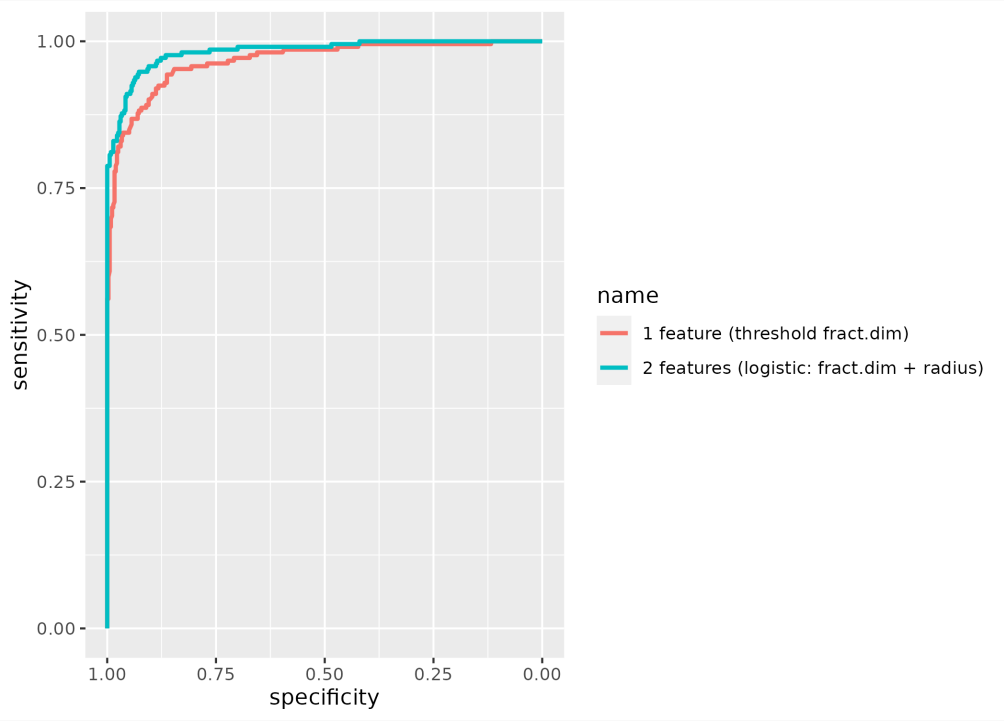
```
wdbc %>% modelr::add_predictions(glm_out, var = "distance") %>%
mutate( predicted = ifelse(distance < 0, "B", "M")  ) %>%
xtabs(~ predicted + diagnosis, data = .) %>% prop.table()
```

Logistic Regression



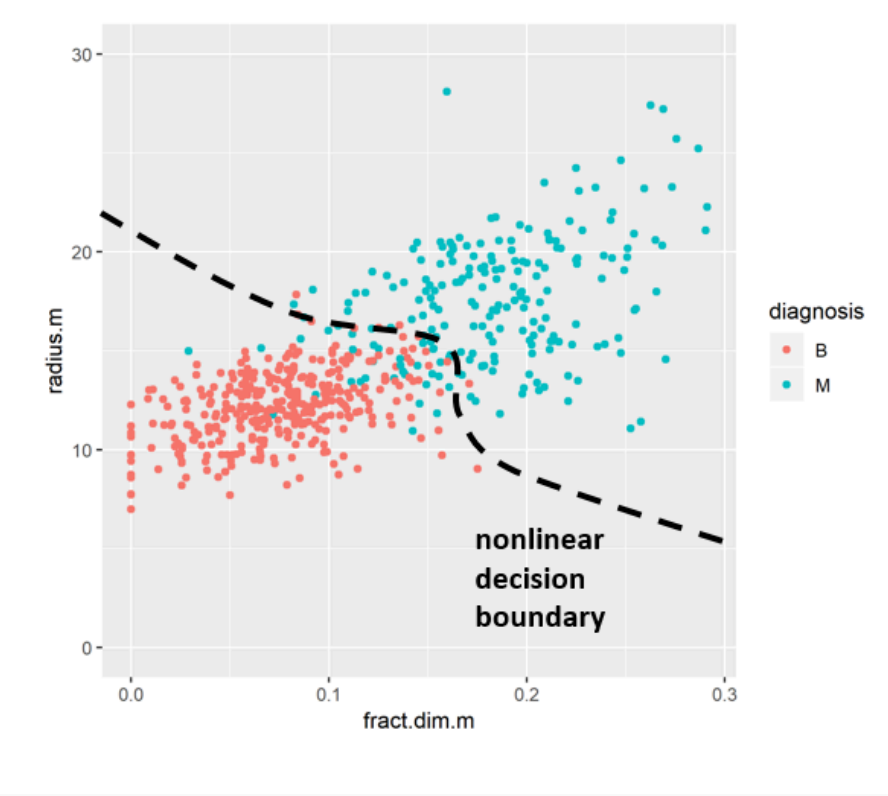
Single vs Multiple Features

- Compare ROC curves



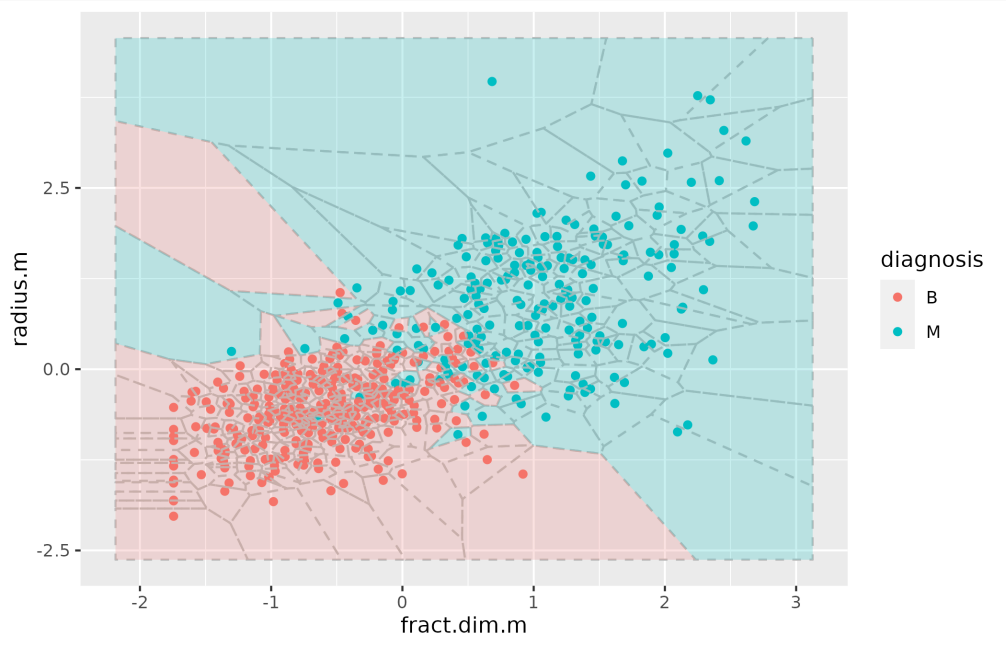
Nonlinear Classifiers

- Classifier with *nonlinear* decision boundaries



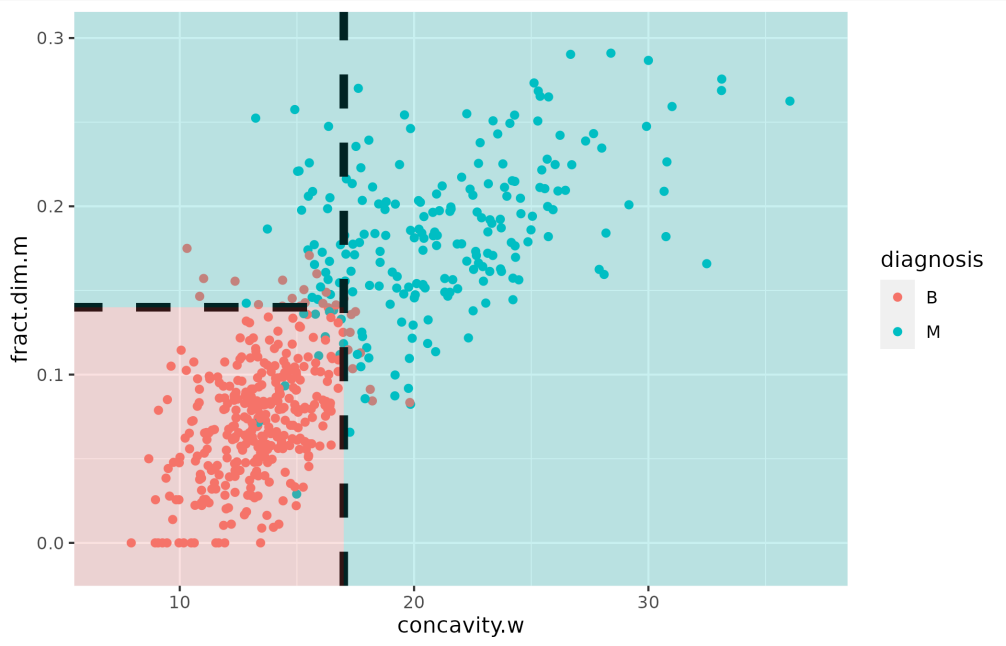
Nearest Neighbor (NN)

- Basics nonlinear classifier: classify each point like its *nearest neighbor*



Classification Tree

- Threshold different variables in *nested/hierarchical* manner



Classification Tree

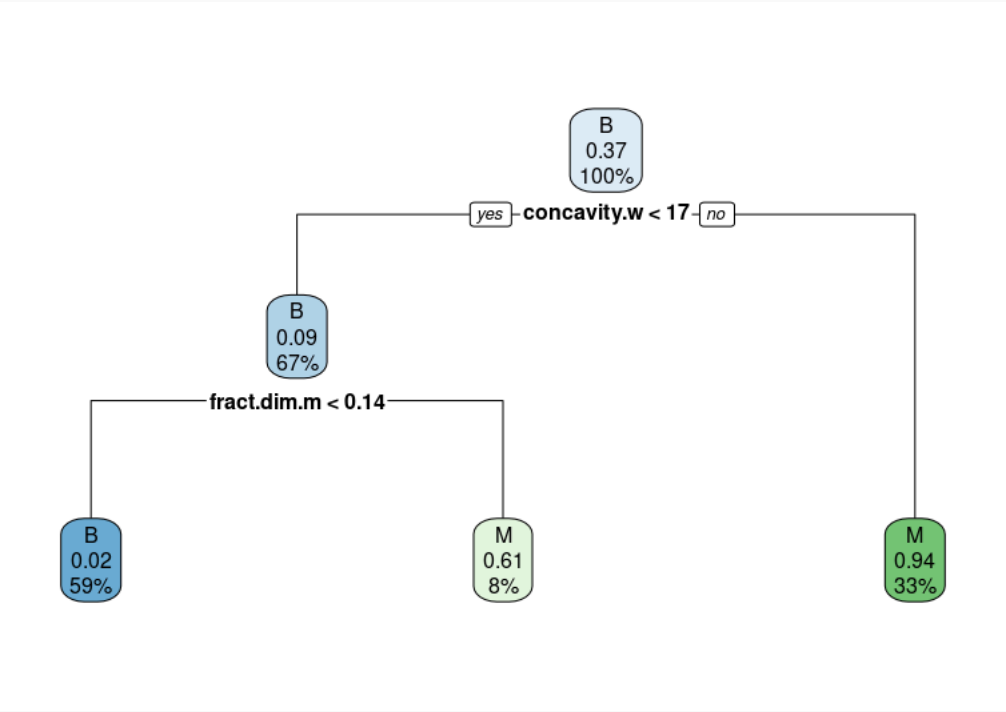
- Fit tree using library `rpart` (recursive partitioning)

```
rpart_out = rpart( diagnosis ~ . - id, data = wdbc,
method = "class", control = rpart.control(minsplit=50) )
rpart_out
```

Classification Tree

- Plot tree using library `rpart.plot`

```
rpart.plot(rpart_out)
```



Classification Tree Predictions

- `predict()` gives class *probabilities*
  - Use `type = "class"` to get classes

```
predict(rpart_out) %>% head(2)
```

```
wdbc %>% modelr::add_predictions( rpart_out, type = "class" ) %>%
xtabs( ~ pred + diagnosis, data = .) %>% prop.table()
```

Thank you!