# INTRO TO DATA SCIENCE
## LECTURE 8: IMBALANCED CLASSES/EVALUATION METRICS

Francesco Mosconi
DAT16 SF // August 24, 2015

# DATA SCIENCE IN THE NEWS

# DATA SCIENCE IN THE NEWS

August 12, 2015

## Inside the Zestimate: Data Science at Zillow

Alex Woodie

If you're like most homeowners, you probably sneak a peek at your 'Zestimate' from time to time to see how your home's value might have changed. Getting a Zestimate is very easy and straightforward for users, but behind the scenes, there's a hefty amount of data science that goes into the equation.

The Zestimate is a core part of Zillow's offering, and is critical for the company's business model. The figure is an estimated market value that's based on a number of public and user-submitted data, including physical attributes, like location, lot size, square footage, and number of bedrooms and bathrooms. Historical data like real-estate transfers and tax information is also factored in, as are sales of comparable houses in a neighborhood.
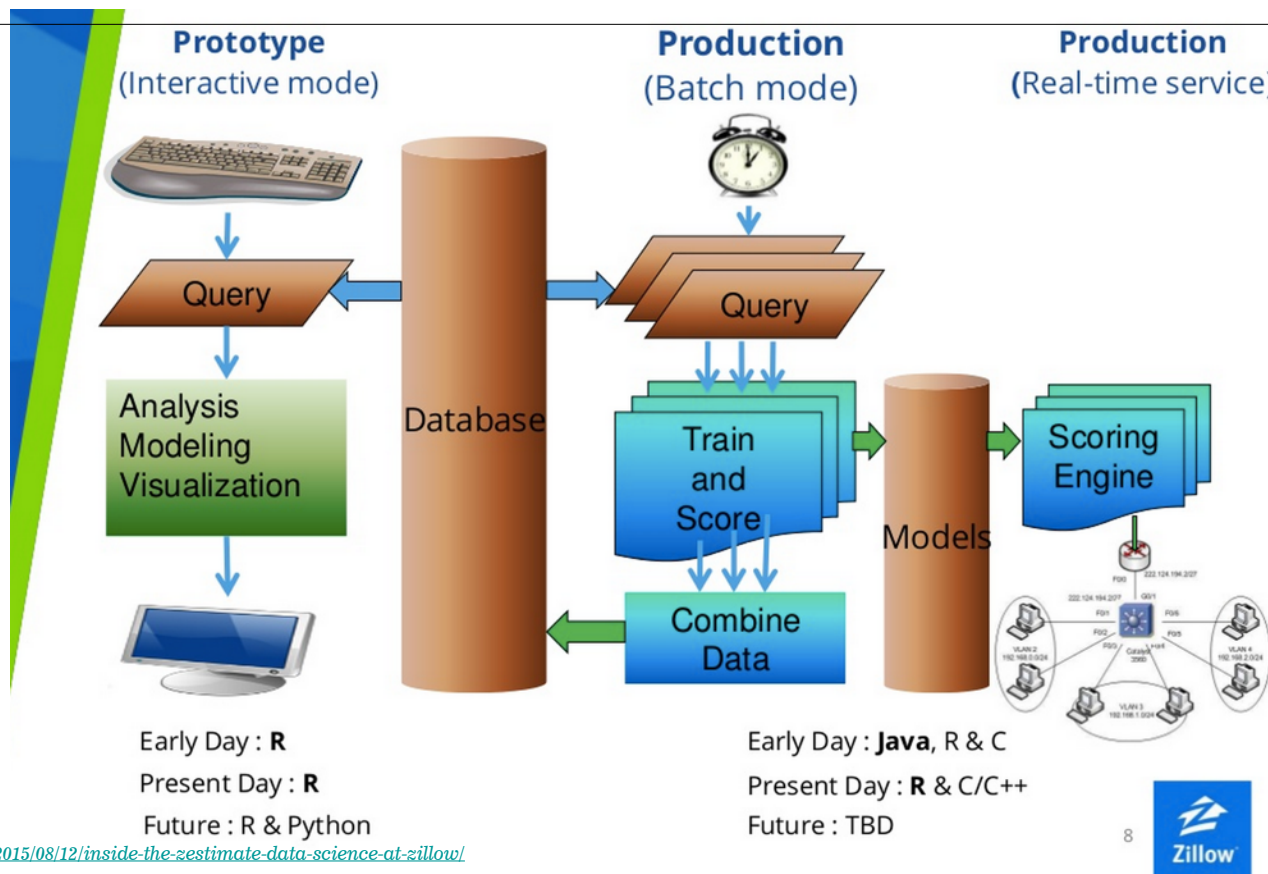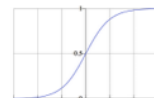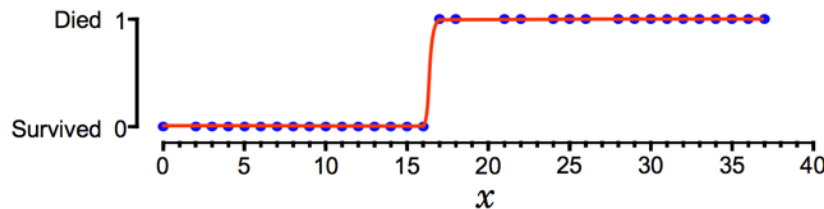
# DATA SCIENCE IN THE NEWS
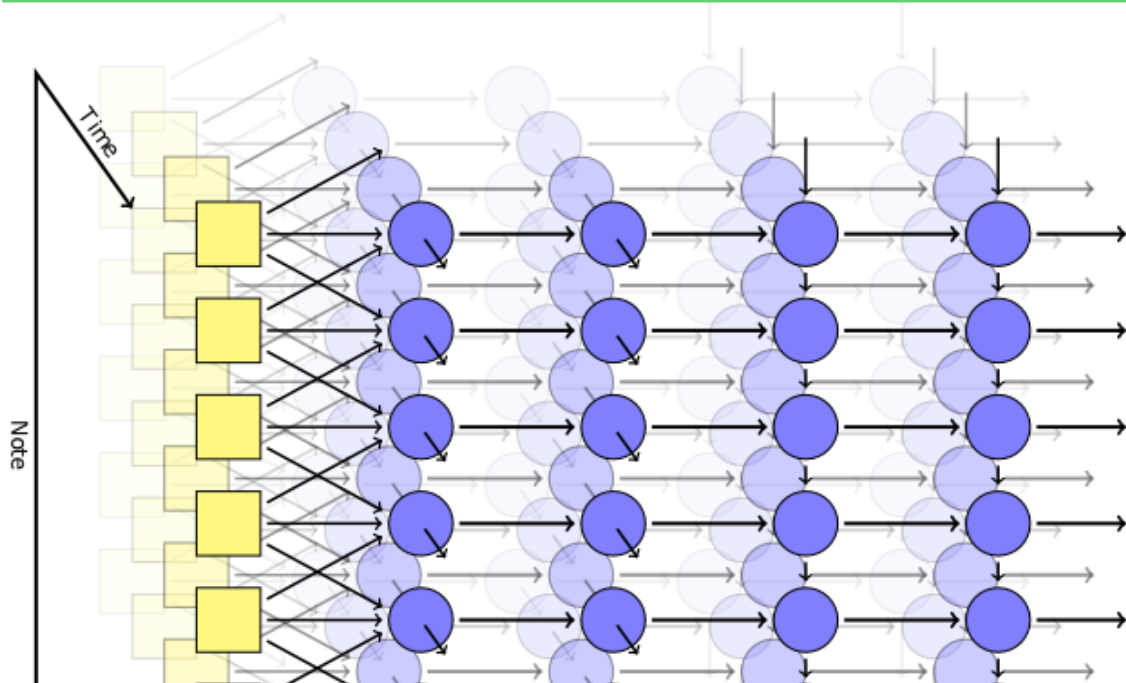
# LAST TIME:

# I. LOGISTIC REGRESSION
# II. OUTCOME VARIABLES
# III. ERROR TERMS
# IV. INTERPRETING RESULTS

# LAB: IMPLEMENTING LOGISTIC REGRESSION IN PYTHON

# QUESTIONS?

# QUESTIONS?

## WHAT WAS THE MOST INTERESTING THING YOU LEARNT?

## WHAT WAS THE HARDEST TO GRASP?

# I. IMBALANCED CLASSES
# II. ERROR RATES
# III. ROC CURVES
# IV. EVALUATION METRICS
# V. COST FUNCTION
# VI. VALIDATION CURVES AND LEARNING CURVES

## KEY OBJECTIVES
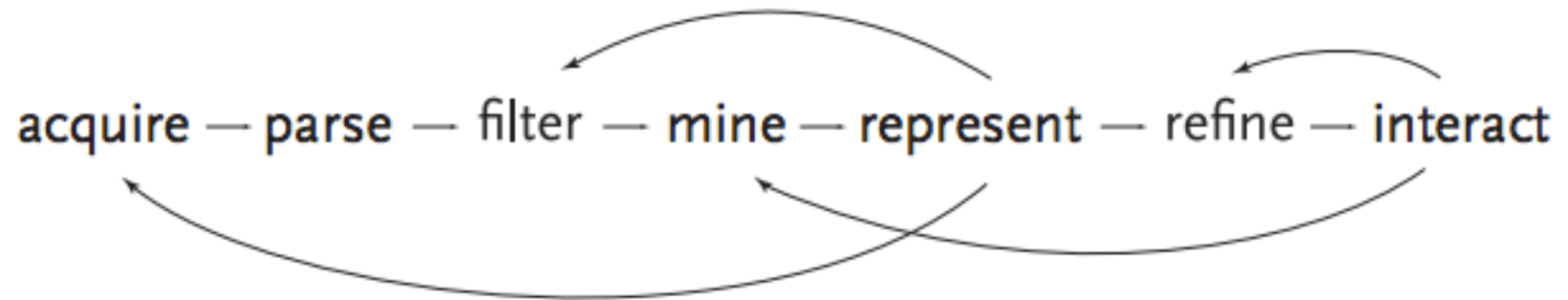
- RECOGNIZE IMBALANCED CLASSES

- TREAT DATASETS INVOLVING IMBALANCED CLASSES

- KNOW VARIOUS EVALUATION METRICS FOR REGRESSION AND CLASSIFICATION

- USE DIFFERENT EVALUATION METRICS TO JUDGE QUALITY OF A MODEL

- UNDERSTAND AND USE ROC CURVES IN BINARY CLASSIFICATION PROBLEMS

- UNDERSTAND AND USE LEARNING CURVES

- GENERALIZE ML PROCESS USING COST FUNCTION FRAMEWORK

- IMPLEMENT ALL THE ABOVE IN PYTHON

# A BRIEF RECAP
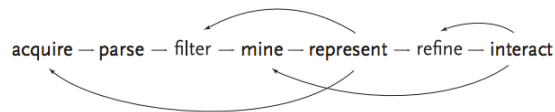
acquire — parse — filter — mine — represent — refine — interact

acquire — parse — filter — mine — represent — refine — interact

|  | **Continuous** | **Categorical** |
|---|---|---|
| **Supervised** | regression ✅ | classification ✅ |
| **Unsupervised** | dimension reduction | clustering |

acquire — parse — filter — mine — represent — refine — interact

| | Continuous | Categorical |
|---|---|---|
| **Supervised** | regression ✓ | classification ✓ |
| **Unsupervised** | dimension reduction | clustering |

# Regression & Regularization

acquire — parse — filter — mine — represent — refine — interact

|  | Continuous | Categorical |
|---|---|---|
| **Supervised** | regression ✅ | classification ✅ |
| **Unsupervised** | dimension reduction | clustering |

Regression & Regularization

# 3 Classification methods

KNN

Naïve Bayes

Logistic Regression

acquire — parse — filter — mine — represent — refine — interact

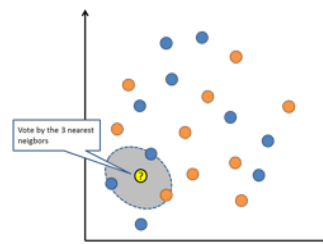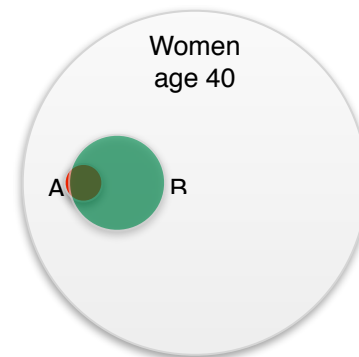|  | Continuous | Categorical |
|---|---|---|
| Supervised | regression ✓ | classification ✓ |
| Unsupervised | dimension reduction | clustering |

Regression & Regularization

3 Classification methods

KNN

Women age 40

A    B

Naïve Bayes

Logistic Regression

Train - Test Split

training set

test set

dataset

new data

model

predictions

acquire — parse — filter — mine — represent — refine — interact

|  | Continuous | Categorical |
|---|---|---|
| **Supervised** | regression ✓ | classification ✓ |
| **Unsupervised** | dimension reduction | clustering |

Regression & Regularization

3 Classification methods
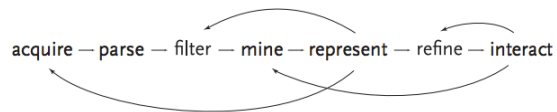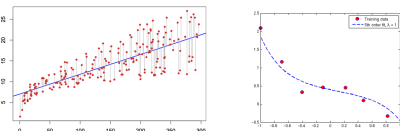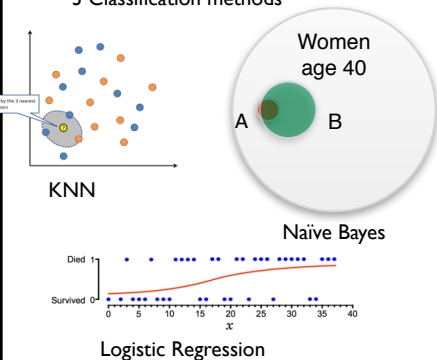
KNN

Women age 40

A    B

Naïve Bayes

Logistic Regression

Train - Test Split

new data

training set

model

test set

predictions

dataset

## Cross Validation

1. Randomly split training instances into *K* equal-sized subsets

fold 1
fold 2
...
fold i
...
fold K

Features & **Target**

For i in 1:K

training: all folds except fold i

fold i

2. Train a ML **model** on the *training* subset

ignore Target when predicting

model

3. Make predictions on *fold i* subset

*fold i* predictions

4. Store *fold i* predictions in the CV predictions array

5. Compare CV predictions to **Target** to assess accuracy

CV predictions

*fold i* predictions

acquire — parse — filter — mine — represent — refine — interact

| | Continuous | Categorical |
|---|---|---|
| **Supervised** | regression ✓ | classification ✓ |
| **Unsupervised** | dimension reduction | clustering |

Train - Test Split

**Regression & Regularization**



new data

training set

model

**3 Classification methods**

test set

predictions

dataset

Women age 40

A    B

**KNN**

Cross Validation

**Naïve Bayes**

1. Randomly split training instances into K equal-sized subsets

For i in 1:K

training: all folds except fold i

fold 1
fold 2
...
fold i
...
fold K

2. Train a ML model on the training subset

model

3. Make predictions on fold i subset

fold i predictions

fold i

4. Store fold i predictions in the CV predictions array

Features & Target

Died 1
Survived 0

x

**Logistic Regression**

5. Compare CV predictions to Target to assess accuracy

CV predictions    fold i predictions

# What else?

acquire — parse — filter — mine — represent — refine — interact

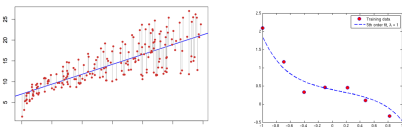| | Continuous | Categorical |
|---|---|---|
| **Supervised** | regression ✓ | classification ✓ |
| **Unsupervised** | dimension reduction | clustering |

Regression & Regularization

3 Classification methods
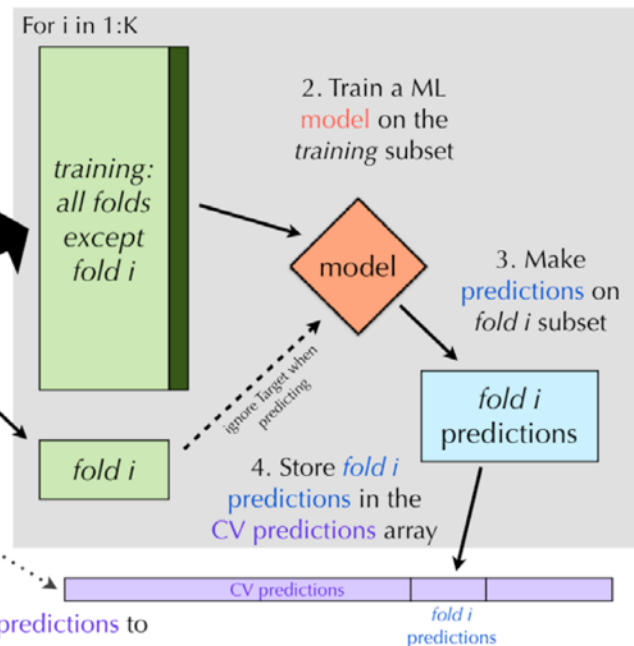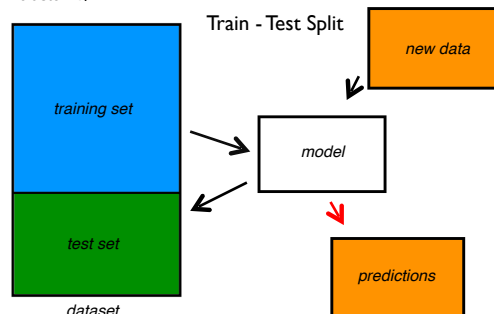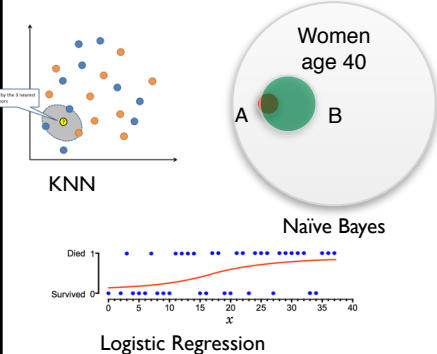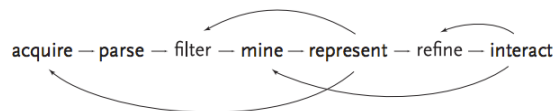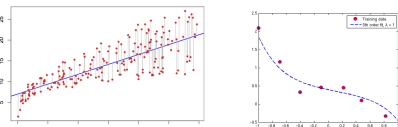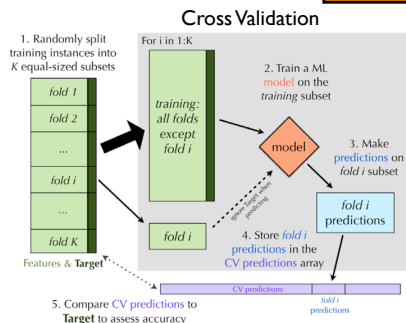
KNN

Women age 40

A    B

Naïve Bayes

Logistic Regression

Train - Test Split

new data

training set

model

test set

dataset

predictions

Cross Validation

1. Randomly split training instances into K equal-sized subsets

For i in 1:K

training: all folds except fold i

fold 1
fold 2
...
fold i
...
fold K

2. Train a ML model on the training subset

model

3. Make predictions on fold i subset

fold i

fold i predictions

4. Store fold i predictions in the CV predictions array

Features & Target

CV predictions | fold i predictions

5. Compare CV predictions to Target to assess accuracy

# What else?
- Probability
- Bayes Theorem
- GLMs
- ...

# Today we consolidate and expand

# A MOTIVATION EXAMPLE

Cancer Screen => classify cancer scans for doctor to review

No Cancer                                                Cancer
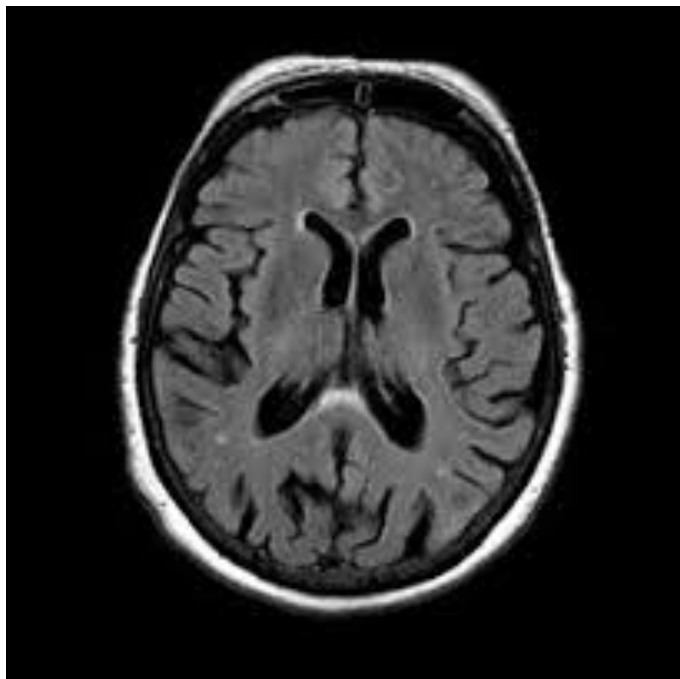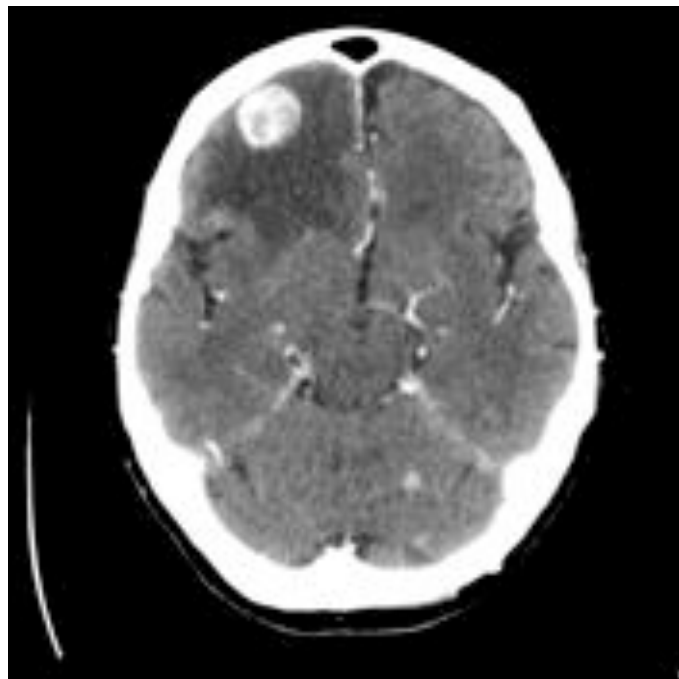
## ISSUE 1: Many more healthy brain scans

- Imbalance confuses classifiers => only perform well on dominant class
- Situation is very common in other fields (e.g. fraud detection)
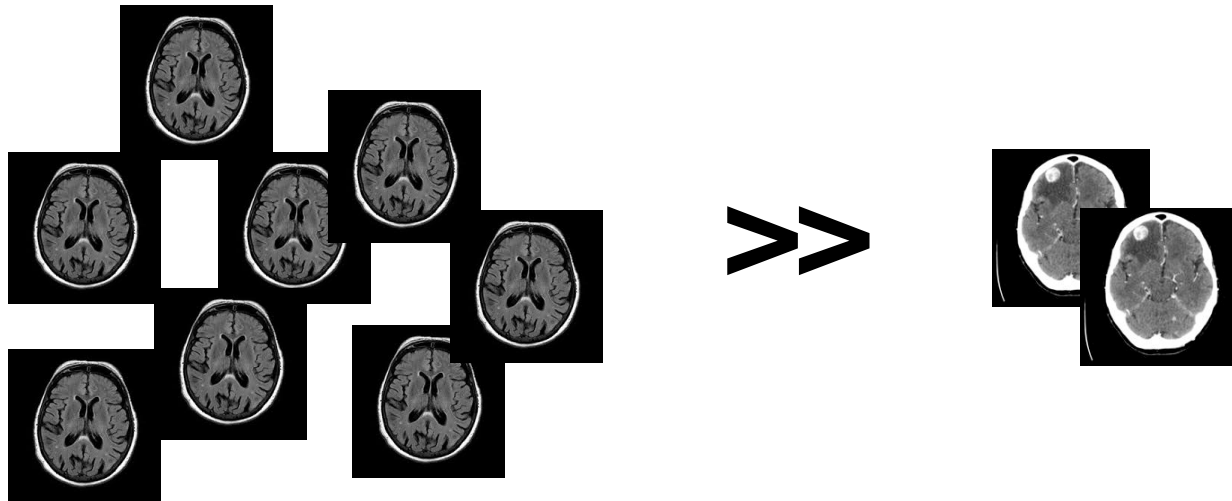
## ISSUE 2: Not all errors are equal…
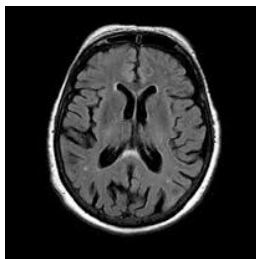


Error 1    Classifier Label: Cancerous    Permissable, because a physician will review it
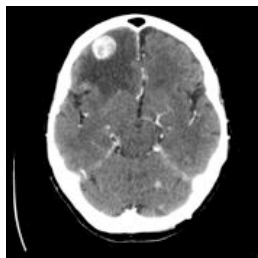


Error 2    Classifier Label: Non-Cancerous    Not permissable, because this data will be discarded

# IMBALANCED CLASSES

Imbalanced classes can be re-balanced in several ways
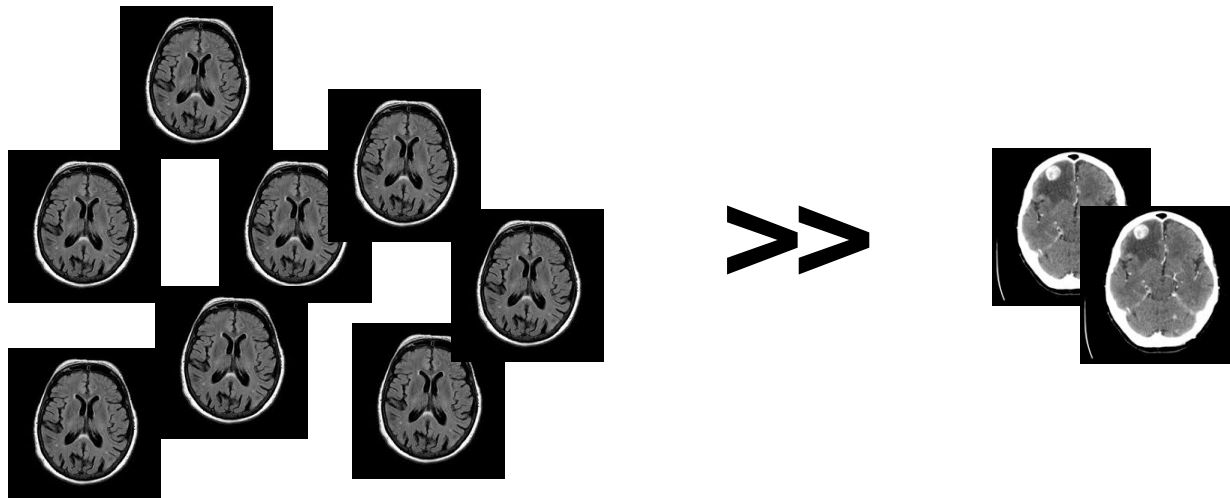
Imbalanced classes can be re-balanced in several ways

1. **Undersampling** the dominant class -  remove some the majority class so it has less weight

Imbalanced classes can be re-balanced in several ways

1. **Undersampling** the dominant class - remove some the majority class so it has less weight

2. **Oversampling** the minority class - add more of the minority class so it has more weight.

Imbalanced classes can be re-balanced in several ways

1. **Undersampling** the dominant class -  remove some the majority class so it has less weight

2. **Oversampling** the minority class - add more of the minority class so it has more weight.

3. **Hybrid** - doing both

Imbalanced classes can be re-balanced in several ways

1. **Undersampling** the dominant class -  remove some the majority class so it has less weight

2. **Oversampling** the minority class - add more of the minority class so it has more weight.
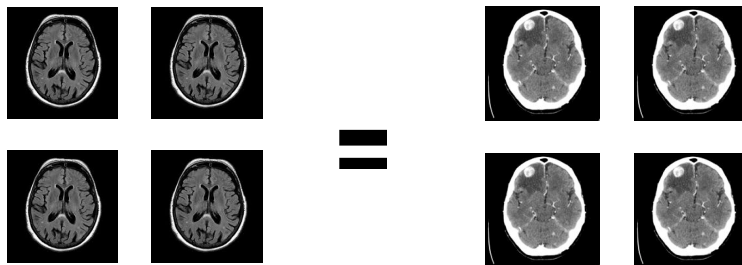
3. **Hybrid** - doing both

# Undersampling

Randomly remove elements from the
majority class.
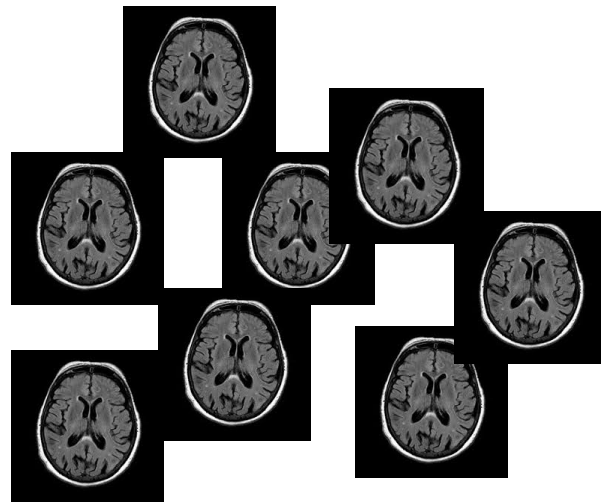
# Undersampling

Randomly remove elements from the majority class.
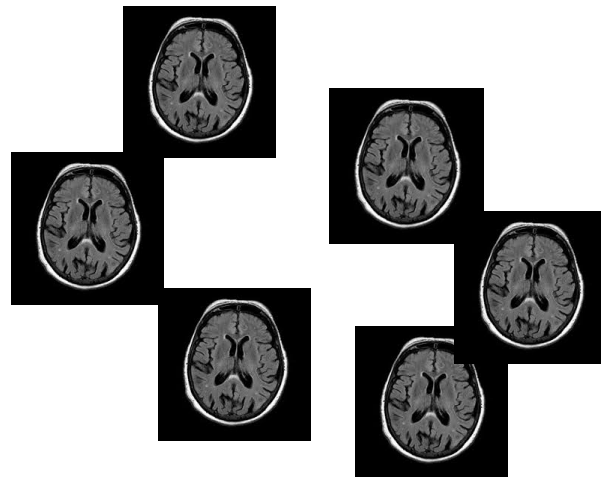
# Undersampling

Randomly remove elements from the majority class.
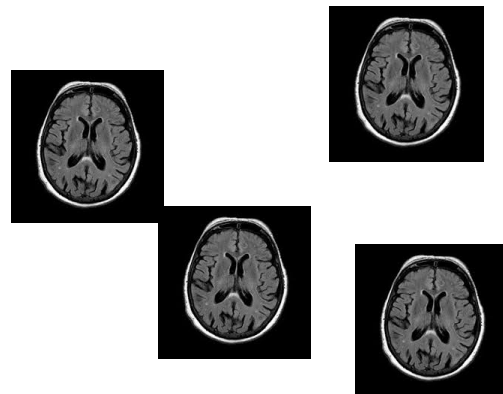
## Undersampling

Randomly remove elements from the majority class.

Drawback: Removing data points could lose important information
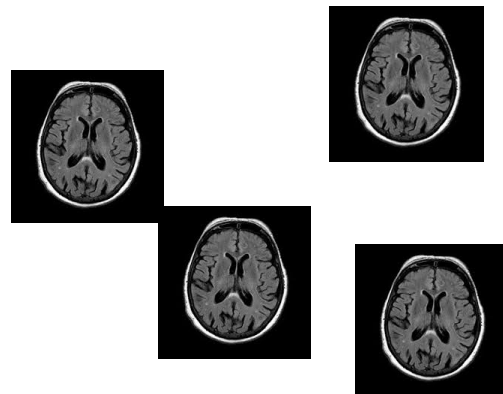
## Oversampling
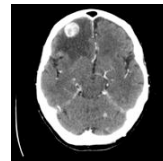


Duplicate elements of your minority class

# Oversampling

Duplicate elements of your minority class
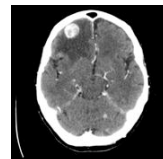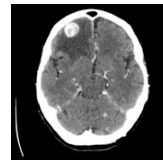
# Oversampling

Duplicate elements of your minority class
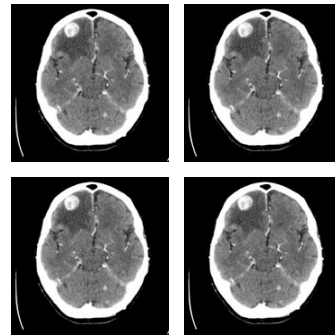
# Oversampling

Duplicate elements of your minority class
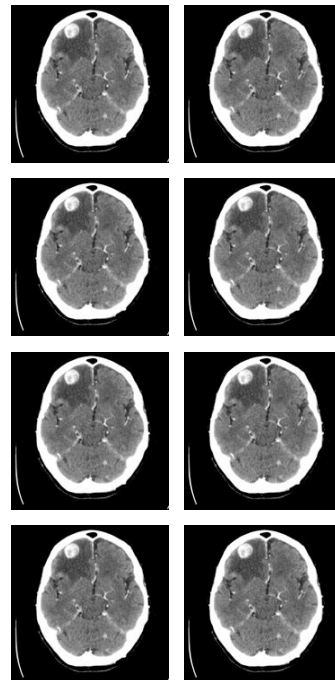
# Oversampling

Duplicate elements of your minority class

<span style="color:red">Drawback: Just replicating randomly minority classes could cause overfit</span>
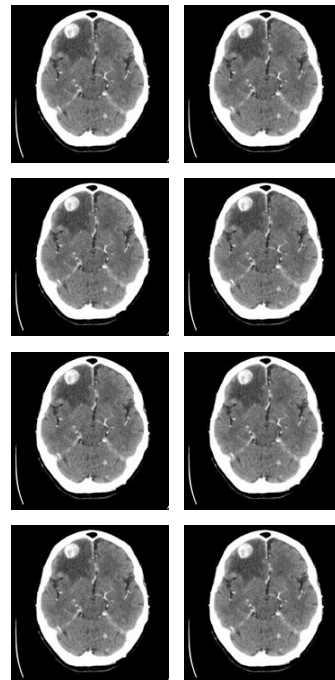
# SMOTE: generate synthetic minority examples
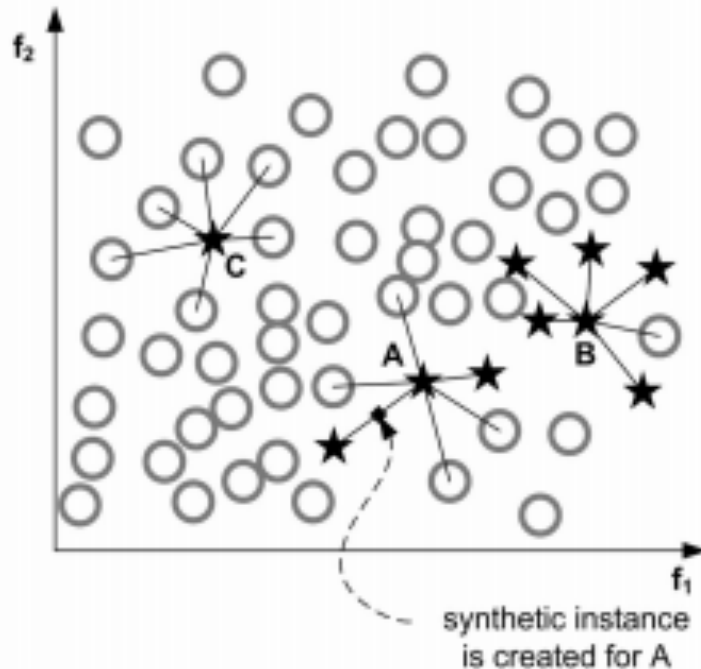


Consider 6-nearest neighbor: m=6

For A:  number of minority instance: 2
        number of majority instance: 4      →     "DANGER" instance

For B:  number of minority instance: 5
        number of majority instance: 1      →     "SAFE" instance

For C:  number of minority instance: 6
        number of majority instance: 0      →     "NOISE" instance

synthetic instance
is created for A

http://egr.uri.edu/wp-uploads/he/learning-from-imbalanced-data.pdf

**SMOTE:** generate synthetic minority examples

For every minority example:
1) find k nearest neighbors in the minority class
2) randomly select one of these neighbors
3) generate new synthetic examples along the line between the minority example and the selected neighbor.

There are even more sophisticated methods, for a review see:

http://egr.uri.edu/wp-uploads/he/learning-from-imbalanced-data.pdf

# ERROR RATES

To deal with issue 2 we need a more sophisticated definition of error rates in a binary classification problem

**True Positive**: An Example that is **positive** and is classified as **positive**
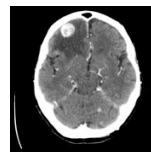


Label: **positive**

To deal with issue 2 we need a more sophisticated definition of error rates in a binary classification problem

**True Positive**: An Example that is **positive** and is classified as **positive**



Label: **positive**

**True Negative**: An Example that is **negative** and is classified as **negative**
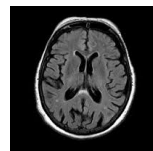


Label: **negative**

To deal with issue 2 we need a more sophisticated definition of error rates in a binary classification problem

**True Positive**: An Example that is **positive** and is classified as **positive**



Label: **positive**

**True Negative**: An Example that is **negative** and is classified as **negative**



Label: **negative**

**False Positive**: An Example that is **negative** and is classified as **positive**



Label: **positive**

To deal with issue 2 we need a more sophisticated definition of error rates in a binary classification problem

**True Positive**: An Example that is **positive** and is classified as **positive**



Label: **positive**

**True Negative**: An Example that is **negative** and is classified as **negative**



Label: **negative**

**False Positive**: An Example that is **negative** and is classified as **positive**



Label: **positive**

**False Negative**: An Example that is **positive** and is classified as **negative**
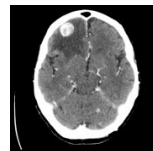


Label: **negative**

# Confusion Matrix

|  | Condition Positive | Condition Negative |
|---|---|---|
| **Test Positive** | **TRUE POSITIVE** | **FALSE POSITIVE** (Type I error) |
| **Test Negative** | **FALSE NEGATIVE** (Type II error) | **TRUE NEGATIVE** |

# Confusion Matrix

| n = 165 | Condition Positive | Condition Negative |
|---|---|---|
| Test Positive | 100 | 10 |
| Test Negative | 5 | 50 |

How many classes are there?
How many patients?
How many times is disease predicted?
How many patients actually have the disease?

# Confusion Matrix

| | Total population | Condition positive | Condition negative | Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$ | |
|---|---|---|---|---|---|
| Test outcome positive | Test outcome positive | **True positive** | **False positive** (Type I error) | Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$ | False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$ |
| Test outcome | Test outcome negative | **False negative** (Type II error) | **True negative** | False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$ | Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$ |
| | Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$ | True positive rate (TPR), Sensitivity, Recall $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$ | False positive rate (FPR), Fall-out $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$ | Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$ | Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR}-}$ |
| | | False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$ | True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$ | Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$ | |

Condition (as determined by "Gold standard")

| n = 165 | Condition Positive | Condition Negative |
|---|---|---|
| **Test Positive** | 100 | 10 |
| **Test Negative** | 5 | 50 |

**Precision:**
When test is positive, how often is prediction correct?
TP / test yes = 100/110 = 0.91

**Sensitivity/Recall/TPR**:
When actual value is positive, how often is prediction correct?
TP / actual yes = 100/105 = 0.95

**Specificity/TNR:**
When actual value is negative, how often is prediction correct?
TN / actual no = 50/60 = 0.83

**Accuracy**:
Overall, how often is it **correct**?

(TP + TN) / total = 150/165 = 0.91

| n = 165 | Condition Positive | Condition Negative |
|---|---|---|
| Test Positive | 100 | 10 |
| Test Negative | 5 | 50 |

**Precision:**
When test is positive, how often is prediction correct?
TP / test yes = 100/110 = 0.91

**Sensitivity/Recall/TPR**:
When actual value is positive, how often is prediction correct?
TP / actual yes = 100/105 = 0.95

**F score**

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Confusion matrix
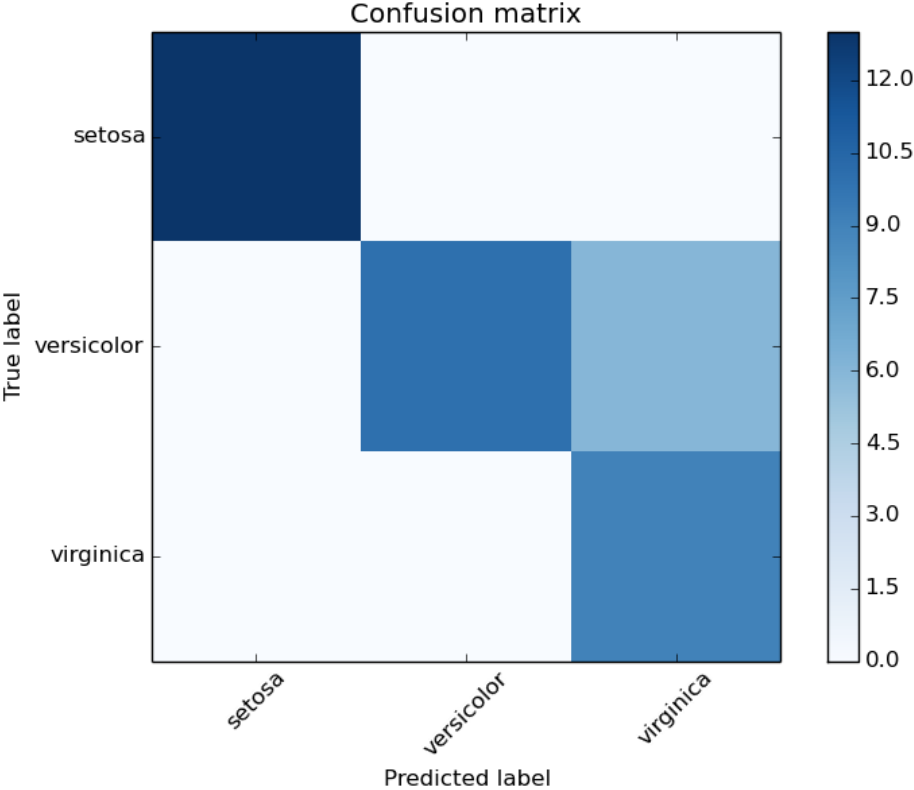
# ROC CURVES
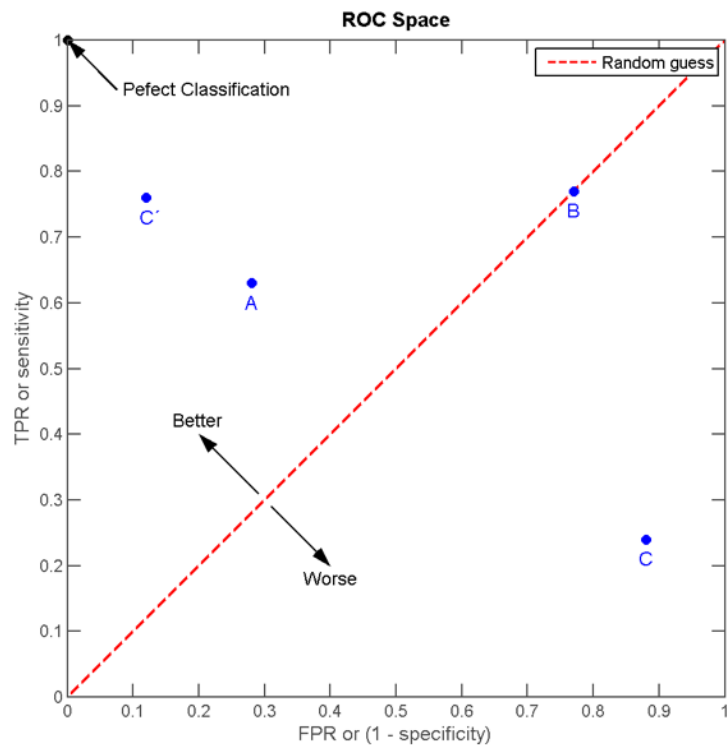
TP Rate = True Positives / All positives

FP Rate = False Positives / All Negatives

| Email Number | Score | True Label |
|---|---|---|
| 5 | 0.99 | Spam |
| 8 | 0.82 | Spam |
| 2 | 0.60 | Spam |
| 1 | 0.60 | Ham |
| 7 | 0.48 | Spam |
| 3 | 0.22 | Ham |
| 4 | 0.10 | Ham |
| 6 | 0.02 | Ham |

Every email is assigned a "spamminess" score by our classification algorithm. To actually make our predictions, we choose a numeric cutoff for classifying as spam.

An ROC Curve will help us to visualize how well our classifier is doing without having to choose a cutoff!

| Email Number | Score | True Label |
|---|---|---|
| 5 | 0.99 | Spam |
| 8 | 0.82 | Spam |
| 2 | 0.60 | Spam |
| 1 | 0.60 | Ham |
| 7 | 0.48 | Spam |
| 3 | 0.22 | Ham |
| 4 | 0.10 | Ham |
| 6 | 0.02 | Ham |

Every email is assigned a "spamminess" score by our classification algorithm. To actually make our predictions, we choose a numeric cutoff for classifying as spam.

An ROC Curve will help us to visualize how well our classifier is doing without having to choose a cutoff!

| Cut off | TPR (y) | FPR (x) | Cut off | TPR (y) | FPR (x) |
|---|---|---|---|---|---|
| 0 | | | 0.50 | | |
| 0.05 | | | 0.65 | | |
| 0.15 | | | 0.85 | | |
| 0.25 | | | 1 | | |

| Email Number | Score | True Label |
|---|---|---|
| 5 | 0.99 | Spam |
| 8 | 0.82 | Spam |
| 2 | 0.60 | Spam |
| 1 | 0.60 | Ham |
| 7 | 0.48 | Spam |
| 3 | 0.22 | Ham |
| 4 | 0.10 | Ham |
| 6 | 0.02 | Ham |

Every email is assigned a "spamminess" score by our classification algorithm. To actually make our predictions, we choose a numeric cutoff for classifying as spam.

An ROC Curve will help us to visualize how well our classifier is doing without having to choose a cutoff!

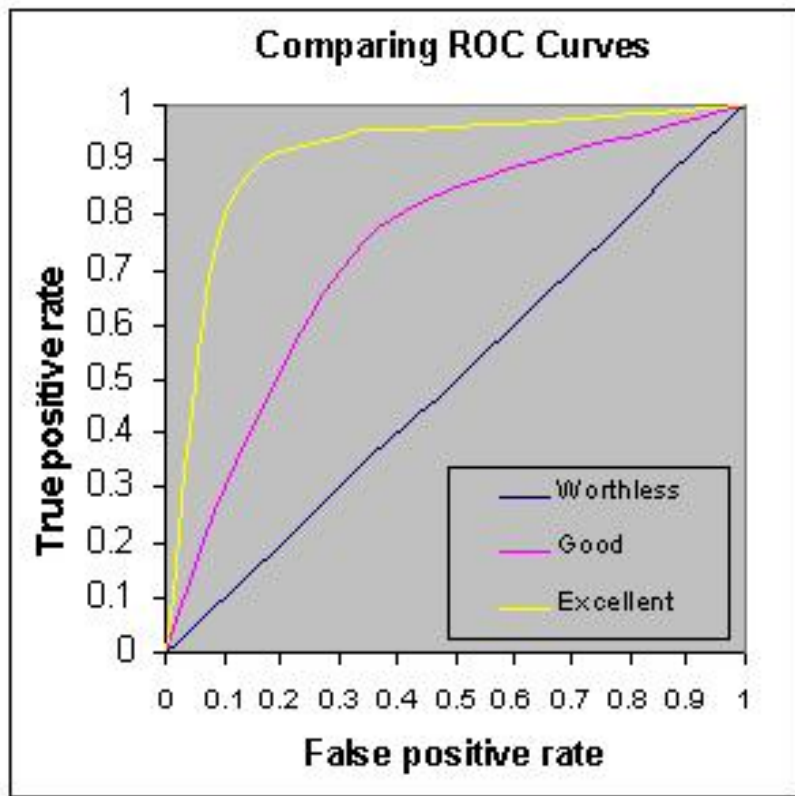| Cut off | TPR (y) | FPR (x) | Cut off | TPR (y) | FPR (x) |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 0.50 | 0.75 | 0.25 |
| 0.05 | 1 | 0.75 | 0.65 | 0.5 | 0 |
| 0.15 | 1 | 0.5 | 0.85 | 0.25 | 0 |
| 0.25 | 1 | 0.25 | 1 | 0 | 0 |

Comparing ROC Curves

**ROC Curves** show the relationship between the TP Rate and the FP Rate as we vary the decision threshold for the classifier

| Cut off | TPR (y) | FPR (x) | Cut off | TPR (y) | FPR (x) |
|---------|---------|---------|---------|---------|---------|
| 0 | 1 | 1 | 0.50 | 0.75 | 0.25 |
| 0.05 | 1 | 0.75 | 0.65 | 0.5 | 0 |
| 0.15 | 1 | 0.5 | 0.85 | 0.25 | 0 |
| 0.25 | 1 | 0.25 | 1 | 0 | 0 |

## Area Under the Curve (AUC)

We evaluate a classifier by measuring the Area Under the Curve for its ROC curve. The Greater area under the curve, the more effective the classifier.

Then for our chosen classifer, we pick an appropriate decision threshold. In general, we pick the decision threshold that gets us closest to the upper left corner

Q: Would the ROC Curve (and AUC) change if the scores changed but the ordering remained the same?

A: Not at all! The ROC Curve is only sensitive to rank ordering and does not require calibrated scores.

**Area Under the Curve (AUC)**

We evaluate a classifier by measuring the Area Under the Curve for its ROC curve. The Greater area under the curve, the more effective the classifier.

Then for our chosen classifer, we pick an appropriate decision threshold. In general, we pick the decision threshold that gets us closest to the upper left corner

# OTHER EVALUATION METRICS

# REGRESSION METRICS

## RMSE

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

- Used for regression problems
- Square root of the mean of the squared errors
- Easily interpretable (in the "y" units)
- "Punishes" larger errors

RMSE

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Example:
y_true = [100, 50, 30]
y_preds = [90, 50, 50]
RMSE = np.sqrt((10**2 + 0**2 + 20**2) / 3) = 12.88

# EXPLAINED VARIANCE

$$\texttt{explained\_variance}(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}}$$

Example:
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
explained_variance(y_true, y_pred) = 0.957

## Mean Absolute Error

$$\text{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} |y_i - \hat{y}_i|.$$

Example:
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
mean_absolute_error(y_true, y_pred) = 0.5

Median Absolute Error

$$\text{MedAE}(y, \hat{y}) = \text{median}(\mid y_1 - \hat{y}_1 \mid, \ldots, \mid y_n - \hat{y}_n \mid).$$

Particularly interesting because it's robust to outliers.

# Classification Metrics

Accuracy Score

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

i.e. the relative frequency of accurate predictions.

## Log Loss

$$L_{\log}(y, p) = -\log \Pr(y|p) = -(y \log p) + (1 - y) \log(1 - p))$$

It is commonly used in (multinomial) logistic regression and neural networks, as well as in some variants of expectation-maximization

# COST FUNCTION

# Linear Regression Example

| Size in feet² (x) | Price in 1000$ (y) |
| --- | --- |
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| ... | ... |

Linear Regression Example

Hypothesis

$h(x) = \beta_0 + \beta_1 x$

| Size in feet² (x) | Price in 1000$ (y) |
|---|---|
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| ... | ... |

Linear Regression Example

| Size in feet² (x) | Price in 1000$ (y) |
|:---:|:---:|
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| ... | ... |

Hypothesis

$$h(x) = \beta_0 + \beta_1 x$$

How do I determine the $\beta$s?

Linear Regression Example

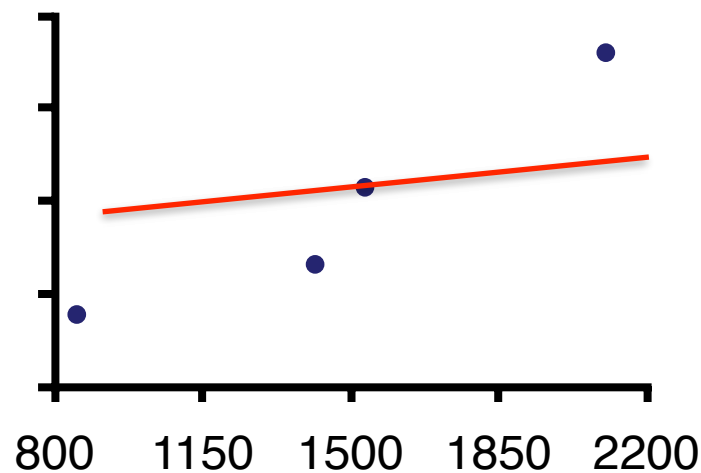| Size in feet² (x) | Price in 1000$ (y) |
| --- | --- |
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| ... | ... |

Hypothesis

$$h(x) = \beta_0 + \beta_1 x$$

How do I determine the $\beta$s?

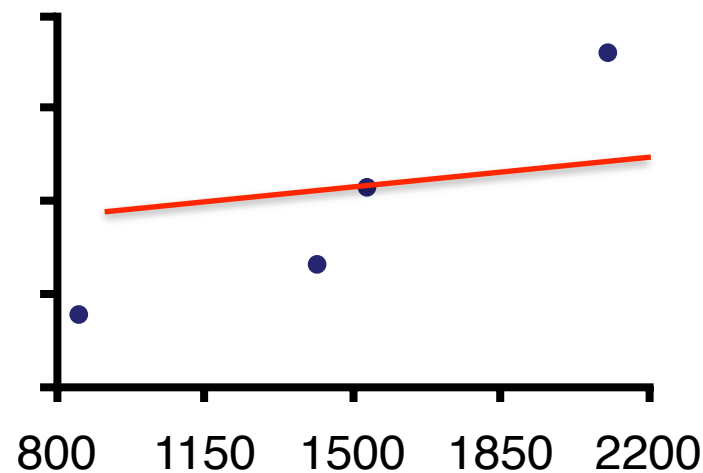IDEA: choose the $\beta$s to minimize distance of h(x) from training data (x, y)

Linear Regression Example

In this case the distance used is the Squared Error

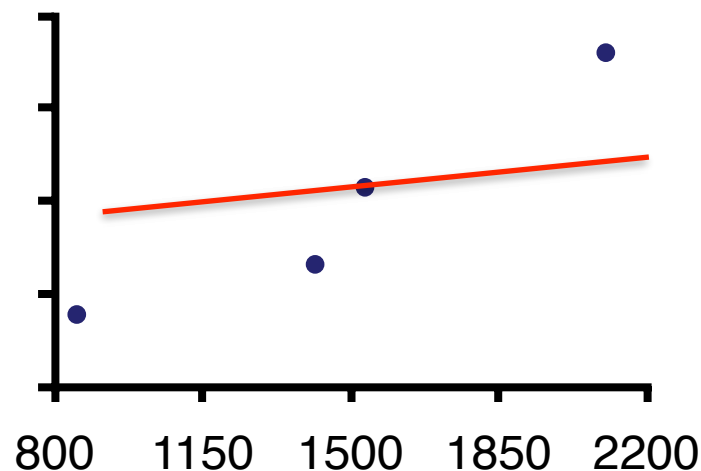| Size in feet² (x) | Price in 1000$ (y) |
|:---:|:---:|
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| ... | ... |

Linear Regression Example

In this case the distance used is the Squared Error:

$$J(\beta_0, \beta_1) = \frac{1}{m} \sum_{i=0}^{m} (h_\beta(x_i) - y_i)^2$$

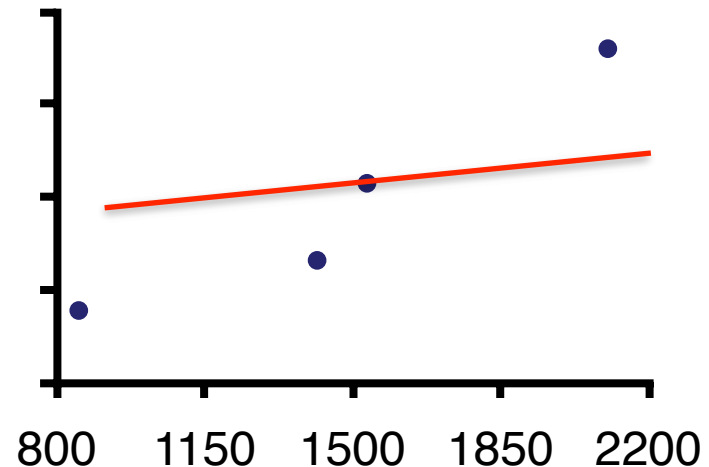| Size in feet² (x) | Price in 1000$ (y) |
|:---:|:---:|
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| ... | ... |

Linear Regression Example

In this case the distance used is
the Squared Error:

$$J(\beta_0, \beta_1) = \frac{1}{m} \sum_{i=0}^{m} (h_\beta(x_i) - y_i)^2$$
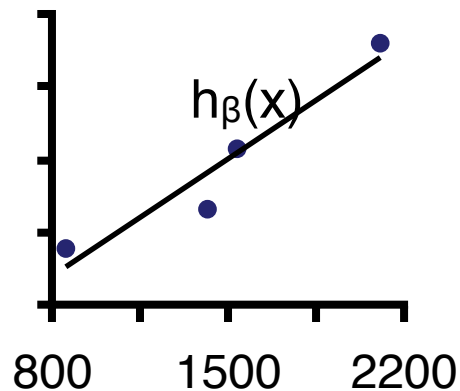
Which is just an example of

COST FUNCTION

| Size in feet² (x) | Price in 1000$ (y) |
|:---:|:---:|
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| ... | ... |

We can generalize and say:

Training set : (x, y)

Hypothesis Function : $h_\beta(x)$

$h_\beta(x)$

800     1500    2200

We can generalize and say:

Training set : (x, y)

Hypothesis Function : $h_\beta(x)$

$h_\beta(x)$

$J(\beta)$

800  1500  2200

$\beta$

Cost Function: $J(\beta)$

# GOAL
## Find the values for the parameters β that minimize the cost function over the set of training data
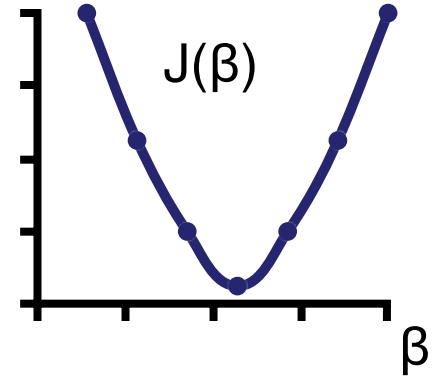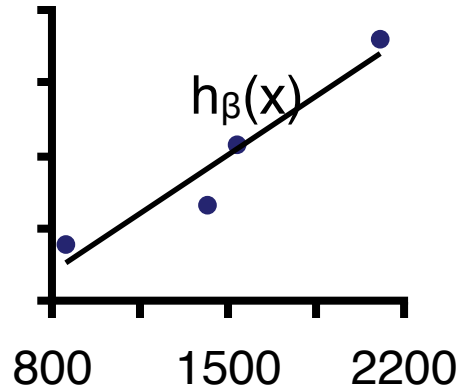
We can generalize and say:

Training set : (x, y)

Hypothesis Function : $h_\beta(x)$

Cost Function: $J(\beta)$

And this is approach can be generalized to any ML problem, with any dimension and any goal (regression, classification)

## GOAL
Find the values for the parameters β that minimize the cost function over the set of training data

# Machine learning problems can be re-stated as:

1. Define a hypothesis function over the training set

2. Choose an appropriate cost function that depends on parameters

3. Find the combination of parameters that minimize the cost of the hypothesis over the training set

# VALIDATION CURVES AND LEARNING CURVES

ML models depend on
Hyperparameters

ML models depend on
Hyperparameters

Not the βs, but e.g.
regularization strength

ML models depend on
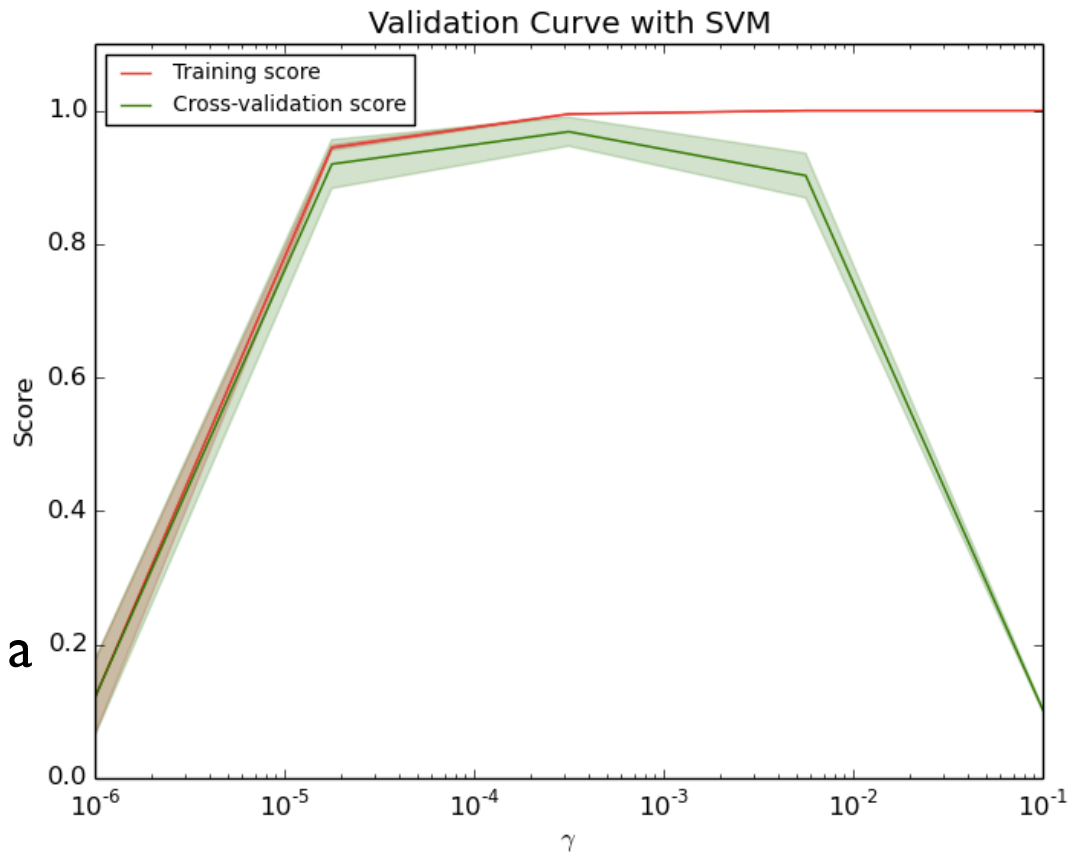Hyperparameters

Not the βs, but e.g.
regularization strength

How do I tune them? What
value to choose?

ML models depend on
Hyperparameters

Not the βs, but e.g.
regularization strength

How do I tune them? What
value to choose?

Plot Training and CV score as a
function of the Hyper params

How do I know if I have enough data?

How do I know if I have enough data?

Plot Training and CV score as a function of the Hyper params