Quantitative Methods

Intro to Machine Learning

Dr. Yves-Alexandre de Montjoye

Associate Professor in Dept. of Computing joint with the Data Science Institute

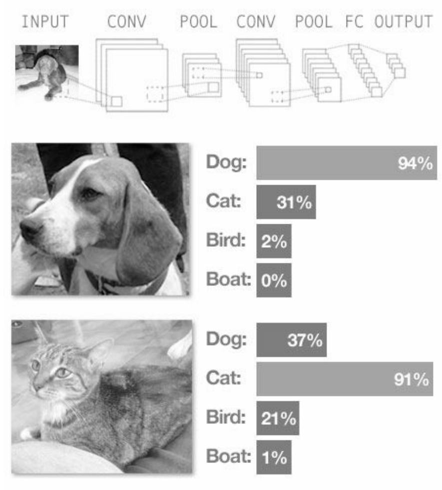
- Postdoc at Harvard
- PhD from Massachusetts Institute of Technology
- MSc in Applied Mathematics from Université catholique de Louvain, Ecole Centrale Paris, and Katholieke Universiteit Leuven
- BSc in Engineering from Louvain

Head of the Computational Privacy Group
Director of the Algorithmic Society Lab

Teach CO408 (Privacy Engineering) and DE-BD2 (Big Data) at Imperial College London

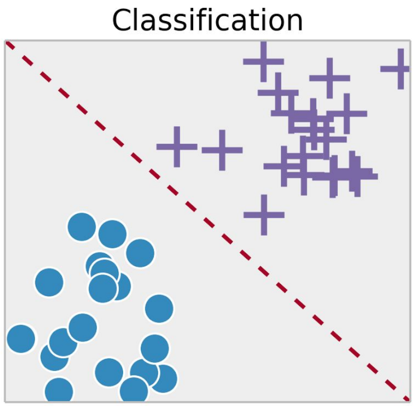
Intro to Machine Learning: Classifiers

We will here focus on predictive analytics: training a model on labeled data (“where we know the right answer, e.g. dog or cat”) to then guess the answer in another similar* dataset



Examples:

- Predicting whether a picture is a picture of a cat or a dog to prevent spam on social network for cat owners
- Predicting if a mushroom is toxic or not based on a picture
- Predicting if a person is a republican or a democrat based on demographics
- or...



* Similar is a big necessary assumption here, the model learns from the data so if the data is not representative the model will not work well or even completely wrong


NATIONAL
GEOGRAPHIC
CHANNEL

Titanic dataset

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew.

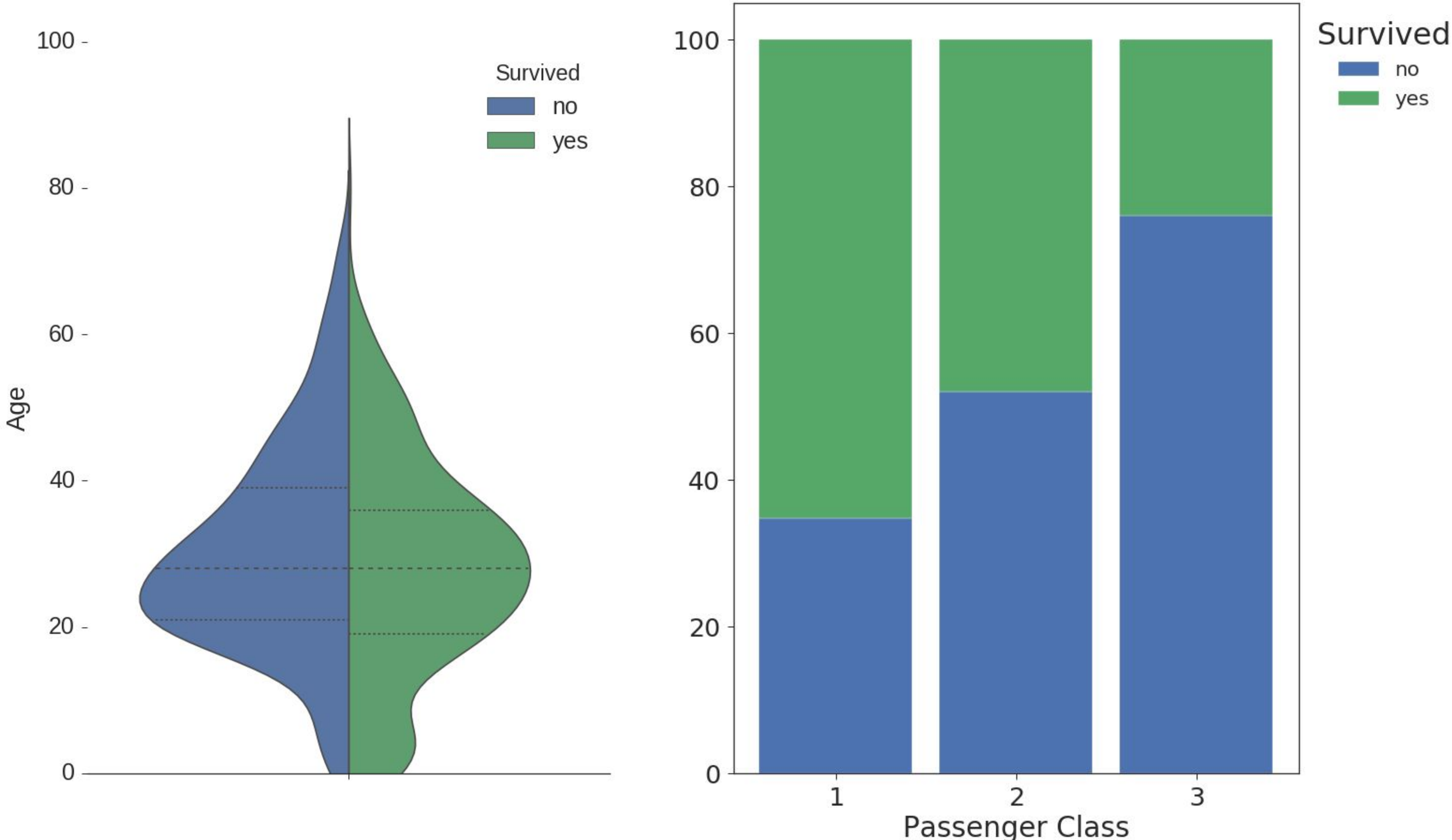
Although there is (as always) some element of luck involved in surviving the sinking of a ship, were some people more likely to survive than others?

And, if yes, I could have used this to make a prediction and used this prediction to price travel insurance in 1913?

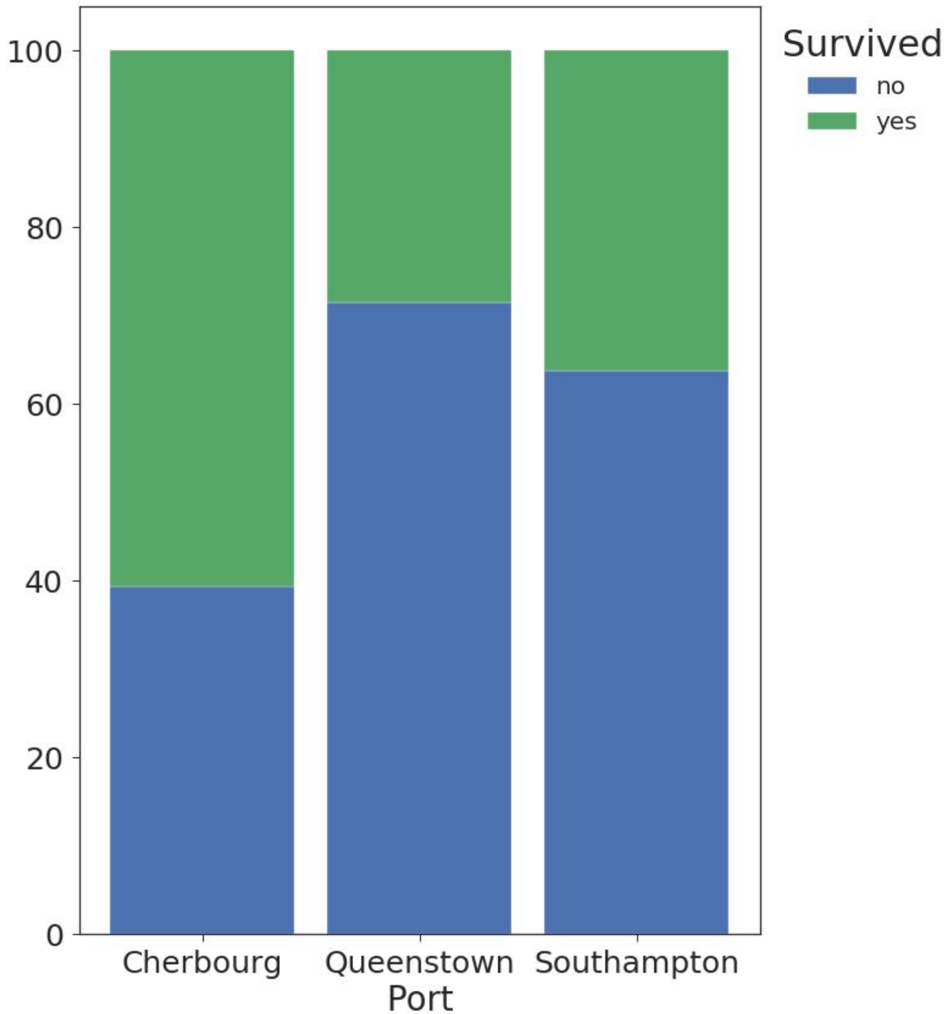
Variables:

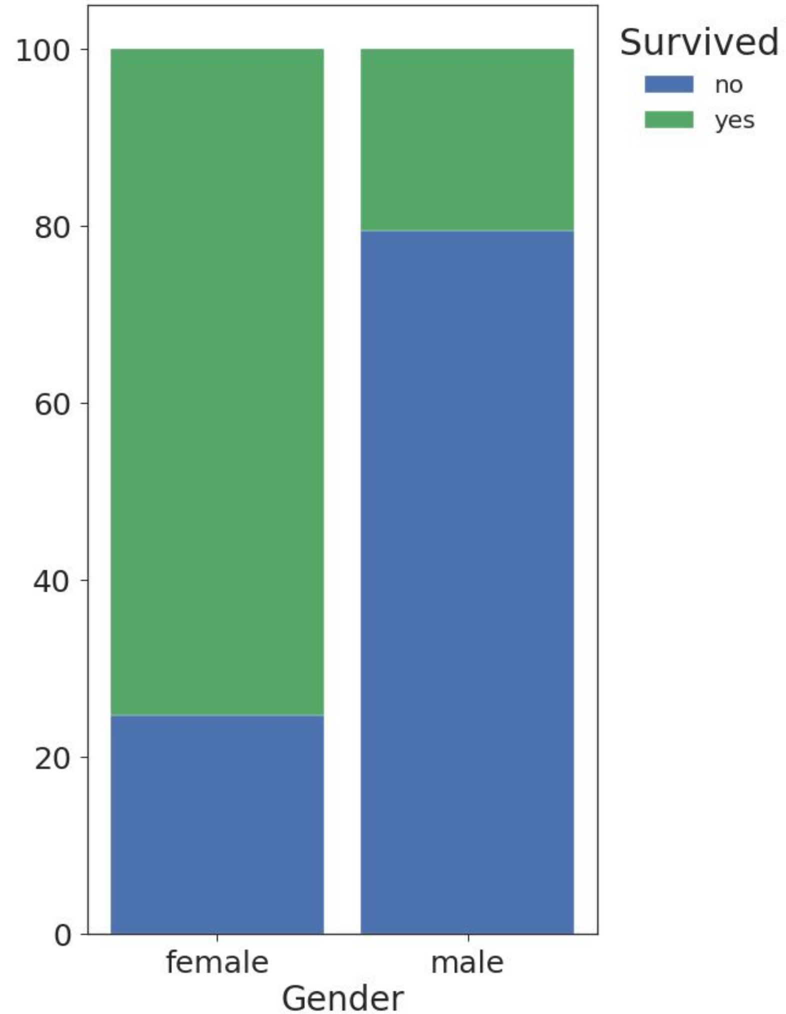
- *Survival*: (0 = No, 1 = Yes)
- *Pclass:* Ticket class (1st, 2nd, 3rd)
- *Sex*: Sex (male/female)
- *age:* Age [years]
- *fare:* Passenger fare in Pre-1970 British Pounds
- *embarked:* Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)
- Etc

Which feature might help?



Which feature might help?





Your first classifier: Logistic regression

```
titanic <- read.csv("titanic.csv")                    # load the titanic dataset
```

Your first classifier: Logistic regression

```
titanic <- read.csv("titanic.csv")                          # load the titanic dataset
logistic.mod1 <- glm(Survived ~ Fare, data = titanic,      # estimate a generalized linear model
family = binomial(logit))                                   # for logistic model
```

Your first classifier: Logistic regression

```
titanic <- read.csv("titanic.csv")                          # load the titanic dataset
logistic.mod1 <- glm(Survived ~ Fare, data = titanic,       # estimate a generalized linear model
family = binomial(logit))                                   # for logistic model
summary(logistic.mod1)                                      # summary of regression

Call:
glm(formula = Survived ~ Fare, family = binomial(logit), data = titanic)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5623  -0.9077  -0.8716   1.3412   1.5731

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.894502   0.107385  -8.330  < 2e-16 ***
Fare         0.015738   0.002489   6.323 2.57e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Your first classifier: Logistic regression

```
titanic <- read.csv("titanic.csv")                        # load the titanic dataset
logistic.mod1 <- glm(Survived ~ Fare, data = titanic,     # estimate a generalized linear model
family = binomial(logit))                                 # for logistic model
summary(logistic.mod1)                                    # summary of regression

Call:
glm(formula = Survived ~ Fare, family = binomial(logit), data = titanic)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5623  -0.9077  -0.8716   1.3412   1.5731

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.894502   0.107385  -8.330  < 2e-16 ***
Fare         0.015738   0.002489   6.323 2.57e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

$$P_{surviving} = \frac{1}{(1 + \exp(-(-0.8945 + 0.0157 * fare)))}$$

We won’t spend too much time on this but 1) you can interpret significance the way you did with linear models, 2) you can interpret coefficients wrt direction and size (exact meaning is beyond scope)

Which you can then use to make predictions and compute the accuracy of your model

```
library(caret)
```

First use function predict to predict the probabilities for each person to have survived

```
fitted.results <- predict(logistic.mod1, newdata = titanic, type = "response")
# fitted.results = [0.6, 0.2, 0.2, 0.1, 0.8, ...]
```

Which you can then use to make predictions and compute the accuracy of your model

```
library(caret)
```

First use function predict to predict the probabilities for each person to have survived

```
fitted.results <- predict(logistic.mod1, newdata = titanic, type = "response")
# fitted.results = [0.6, 0.2, 0.2, 0.1, 0.8, ...]
```

Then you need to convert the probabilities to binary decisions (1 if greater then 0.5 and 0 otherwise)

```
fitted.results <- ifelse(fitted.results > 0.5, 1, 0) # convert to binary
# fitted.results = [1, 0, 0, 0, 1, ....]
```

Which you can then use to make predictions and compute the accuracy of your model

```
library(caret)
```

First use function predict to predict the probabilities for each person to have survived

```
fitted.results <- predict(logistic.mod1, newdata = titanic, type = "response")
# fitted.results = [0.6, 0.2, 0.2, 0.1, 0.8, ...]
```

Then you need to convert the probabilities to binary decisions (1 if greater then 0.5 and 0 otherwise)

```
fitted.results <- ifelse(fitted.results > 0.5, 1, 0) # convert to binary
# fitted.results = [1, 0, 0, 0, 1, ....]
```

After that convert it to factor with same categories as the original variable (“No” and “Yes”)

```
fitted.results <- factor(fitted.results, levels = c(0,1), labels = c("No", "Yes"))
# fitted.results = [yes, no, no, no, yes, ....]
```

Which you can then use to make predictions and compute the accuracy of your model

```
library(caret)
```

First use function predict to predict the probabilities for each person to have survived

```
fitted.results <- predict(logistic.mod1, newdata = titanic, type = "response")
# fitted.results = [0.6, 0.2, 0.2, 0.1, 0.8, ...]
```

Then you need to convert the probabilities to binary decisions (1 if greater then 0.5 and 0 otherwise)

```
fitted.results <- ifelse(fitted.results > 0.5, 1, 0) # convert to binary
# fitted.results = [1, 0, 0, 0, 1, ....]
```

After that convert it to factor with same categories as the original variable (“No” and “Yes”)

```
fitted.results <- factor(fitted.results, levels = c(0,1), labels = c("No", "Yes"))
# fitted.results = [yes, no, no, no, yes, ....]
```

Now we can compute the accuracy (more on the confusion matrix later)

```
confusionMatrix(fitted.results, titanic[,"Survived"])$overall[1] # to calculate model accuracy
# fitted.results =       [yes, no,  no, no, yes, ....]
# titanic[,"Survived"]) = [yes, yes, no, no, no,  ....]

> 0.666
```

Survived ~ Fare + Sex

```
Call:
glm(formula = Survived ~ Fare + Sex, family = binomial(logit),
    data = titanic)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2652  -0.6465  -0.6014   0.8010   1.9381

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.659002   0.167464   3.935 8.31e-05 ***
Fare         0.012052   0.002623   4.595 4.33e-06 ***
Sexmale     -2.371126   0.189333 -12.524  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

As expected, being a men strongly decreases the likelihood of survival according to our model (and significant)

```
fitted.results <- factor(ifelse( predict(logistic.mod2, newdata = titanic, type = "response") >
                         0.5, 1, 0), levels = c(0,1), labels = c("No", "Yes"))
confusionMatrix(fitted.results, titanic[,"Survived"])$overall[1] # to calculate model accuracy
> 0.777
```

And adding gender increases accuracy

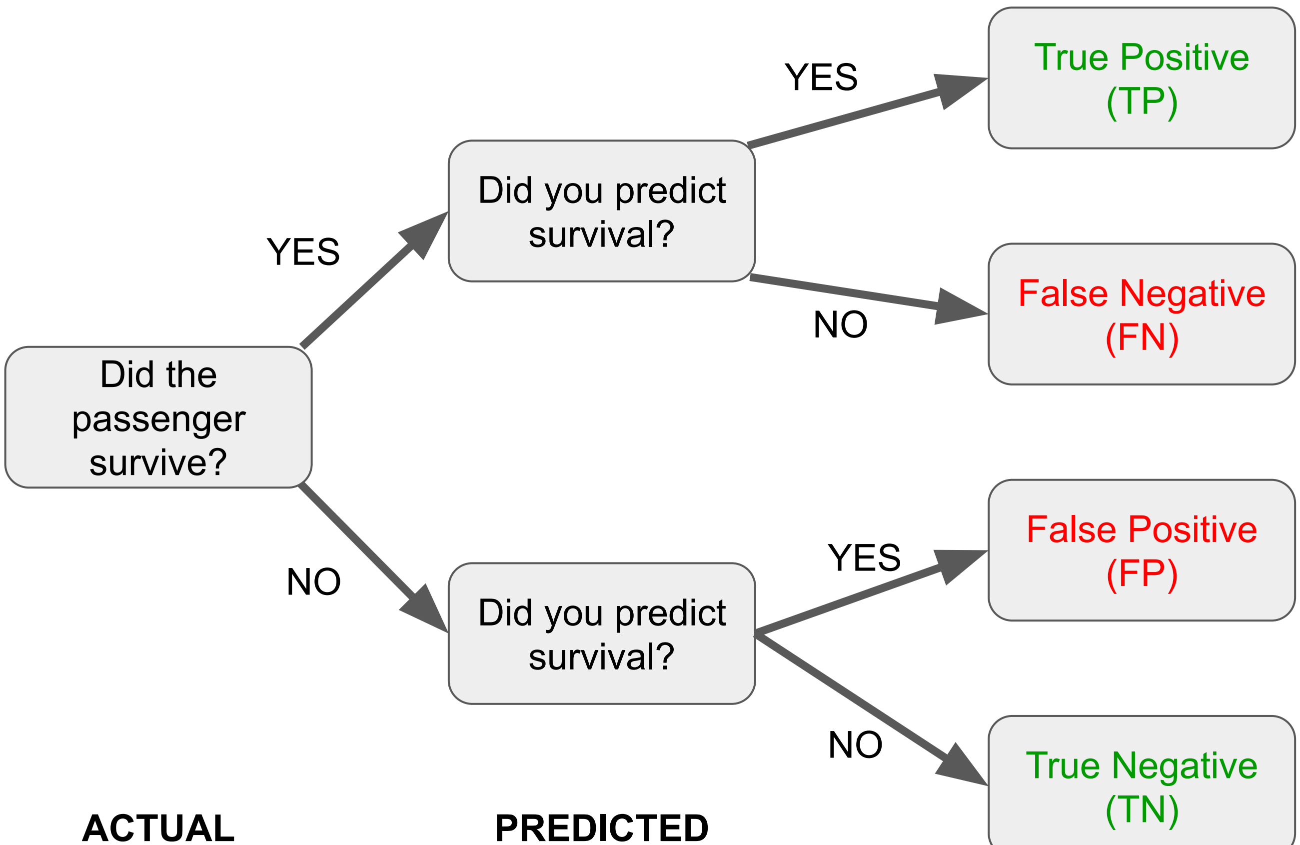
Survived ~ Fare + Age + Pclass + Sex

```
Call:
glm(formula = Survived ~ Fare + Age + Pclass + Sex, family = binomial(logit),
    data = titanic)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7363  -0.6810  -0.3965   0.6558   2.4640

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      3.7149937  0.4644987   7.998 1.27e-15 ***
Fare             0.0005189  0.0022553   0.230    0.818
Age             -0.0369401  0.0077460  -4.769 1.85e-06 ***
Pclass2         -1.2682002  0.3127441  -4.055 5.01e-05 ***
Pclass3         -2.5335614  0.3278321  -7.728 1.09e-14 ***
Sexmale         -2.5096331  0.2084270 -12.041  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

confusionMatrix(fitted.results, titanic[,"Survived"])$overall[1] # to calculate model accuracy
> 0.792
```

Pclass can take 3 values, 2 and 3 are the effect of being in 2nd and 3rd class as opposed to being in 1st which is the default value (same as men in the previous slide and here)

Beyond accuracy: Evaluating how often and how is my model wrong

There are four possible outcomes of a prediction I (well, my model) made:



My model predicted Positive and was right

My model predicted Negative and was wrong

My model predicted Positive and was wrong

My model predicted Negative and was right

This is called a confusion matrix

This is usually summarised in a matrix:

	Prediction: Yes - Survived	Prediction: No - Died
Actual: Yes - Survived	True Positives (TP)	False Negatives (FN)
Actual: No - Died	False Positives (FP)	True Negatives (TN)

Clearly, we would like to get as many True Negatives and True Positives as possible (my model was **right**) and as few False Positives and False Negatives as possible (my model was **wrong**).

From this I can compute accuracy which is $Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$

Confusion matrix and performance measures in *R*

We can extract all of these metrics from the confusion matrix in R

```
confusionMatrix(fitted.results, titanic[,"Survived"])$table

>           Reference
Prediction  No  Yes
      No   357   81
      Yes   67  207
```

The accuracy:

```
confusionMatrix(fitted.results, titanic[,"Survived"])$overall[1]
> 0.792
```

Let’s compete

Competition: best accuracy

For this exercise you will need to estimate a logistic regression on a crime dataset.

The variable of interest is larcenies per capita (larcPerPop). The variable is equal to one when larcenies per capita are high, and equal to zero when larcenies per capita are low.

The variables you can use are listed in “List of variables.txt”, e.g.:
-- **population**: population for community: (numeric - expected to be integer)
-- **householdsize**: mean people per household (numeric - decimal)
-- **racepctblack**: percentage of population that is african american (numeric - decimal)
-- **racePctWhite**: percentage of population that is caucasian (numeric - decimal)
-- **racePctAsian**: percentage of population that is of asian heritage (numeric - decimal)
-- **racePctHisp**: percentage of population that is of hispanic heritage (numeric - decimal)

Setup: 3 files

You will need to use the following files:
- ‘template_1_XX.R’
- ‘train_1.R’
- ‘crime_competition.csv’

Setup: 3 files

You will need to use the following files:
- ‘template_1_XX.R’ (**pick the one with the letter of your group**)
- ‘train_1.R’
- ‘crime_competition.csv’

Put all three files in the same folder, open RStudio and set the working directory in the folder where you put the above files.

How to do it

Use the file ‘template_1_XX.R’ to specify the formula of your model, i.e. the add/modify the covariates you want to use, and save the changes.

Example:
formula <- "larcPerPop ~ householdsize ***+ population"***

Note, the independent var must be larcPerPop, only add/modify covariates

To see the name of the variables you can look at “List of variables.txt” or open ‘crime_competition.csv’ in Excel

Be careful!

DO NOT write additional lines of code in that file, only the formula, as shown in the example below.

Example:

```
formula <- "larcPerPop ~ householdsize + population"
```

You do not need to estimate and validate the model yourself!

Train your model and see its accuracy

To train your model and see it's accuracy, use the file 'train_1.R'

This file contains a function called 'train_1'.

<u>Select that function and run it with CTRL+ENTER (or CMD+ENTER)</u>

Doing so will add the function 'train_1' to your environment.

Then to see the results, open **another R script** write and run:

> ***train_1(group = 'xx')***
>
> With replace xx with the letter of your group, e.g., if you are group B write and run train_1(groups = 'B')

Summary

1. Open the file ‘template_1_XX.R’ (the one with the letter of your team)

2. Modify the formula in the ‘template_1_XX.R’ file by adding variables then save the file

   /!\ do not add anything else than the name of the variables you want to add with a symbol + between them

3. Open the file ‘train_1.R’ in RStudio

4. CTRL+ENTER on the function ‘train_1’ inside the file ‘train_1.R’

5. In **another R script** write and run:
   train_1(group = ‘xx’) # replace xx with the letter of your group

6. Send your template file to yvesalexandre@demontjoye.com

Yves-Alexandre de Montjoye
Imperial College London
Computational Privacy Group
deMontjoye@imperial.ac.uk