

---

# Classification II

Introduction to data analysis: Lecture 11

Ori Plonsky

Spring 2023

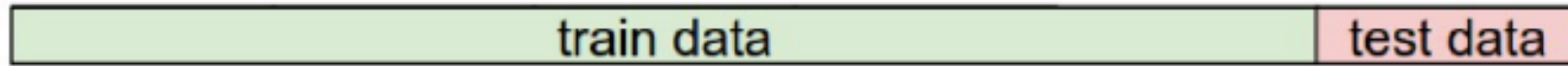


Source: xkcd

---

# Evaluation

- Train on training data, test on test data



- Training data: train classifier
- Test data: measure performance
- (for all algorithms, not just for kNN)

---

# Evaluation

- We want to know how accurate our classifier is

---

# Evaluation

- We want to know how accurate our classifier is
- Accuracy of  $k$ NN on train data?

---

# Evaluation

- We want to know how accurate our classifier is
- Accuracy of  $k$ NN on train data?
- We want to know how accurate our classifier **will be on new data**

---

# Evaluation

- We want to know how accurate our classifier is
- Accuracy of  $k$ NN on train data?
- We want to know how accurate our classifier **will be on new data**

## Test data:

- We “pretend” we do not have some of the data
- If **new data** will come from the same distribution as the **test data**, the accuracy on the **test data** will be similar to accuracy on **new data**

---

# Evaluation

- We want to know how accurate our classifier is
- Accuracy of  $k$ NN on train data?
- We want to know how accurate our classifier **will be on new data**

## Test data:

- We “pretend” we do not have some of the data
  - We therefore cannot use it at any point during training!
- If **new data** will come from the same distribution as the **test data**, the accuracy on the **test data** will be similar to accuracy on **new data**

---

# Evaluation

- We want to know how accurate our classifier is
- Accuracy of  $k$ NN on train data?
- We want to know how accurate our classifier **will be on new data**

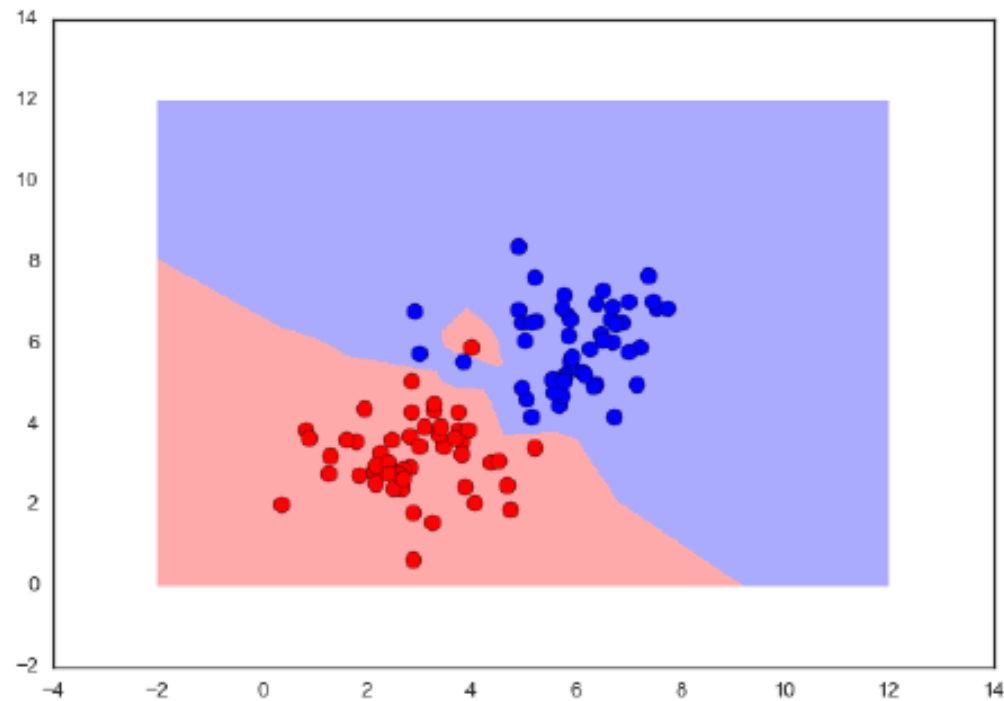
## Test data:

- We “pretend” we do not have some of the data
  - We therefore cannot use it at any point during training!
  - **Beware data leakage:** never use for training any information you would not have had you did not actually have access to the test data (e.g., scale using only train data values)
- If **new data** will come from the same distribution as the **test data**, the accuracy on the **test data** will be similar to accuracy on **new data**

(notebook)

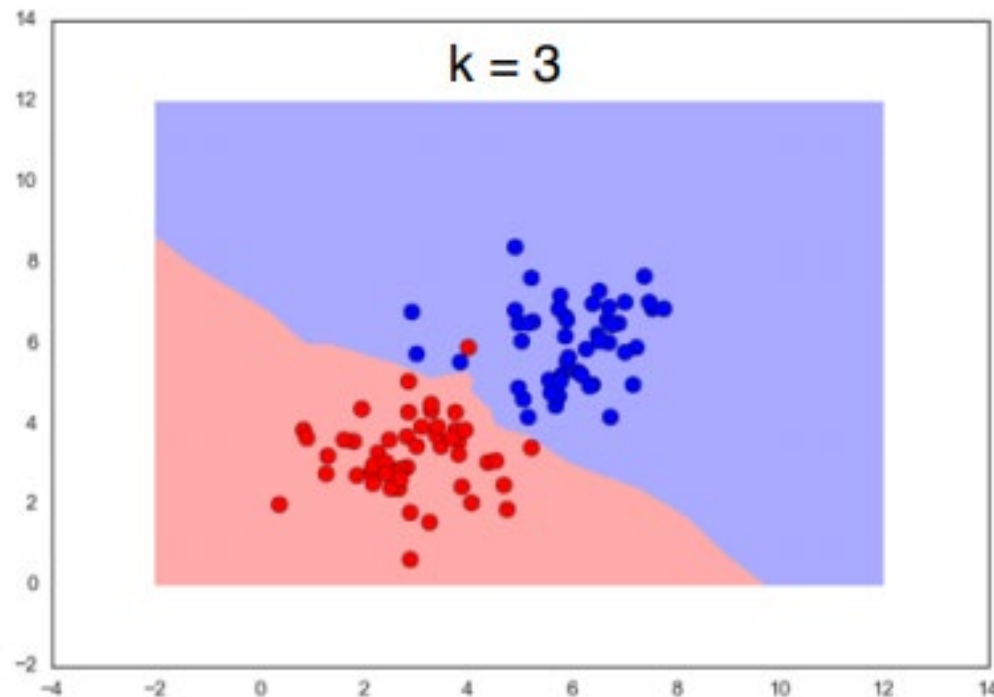


# 1-NN: Problem?

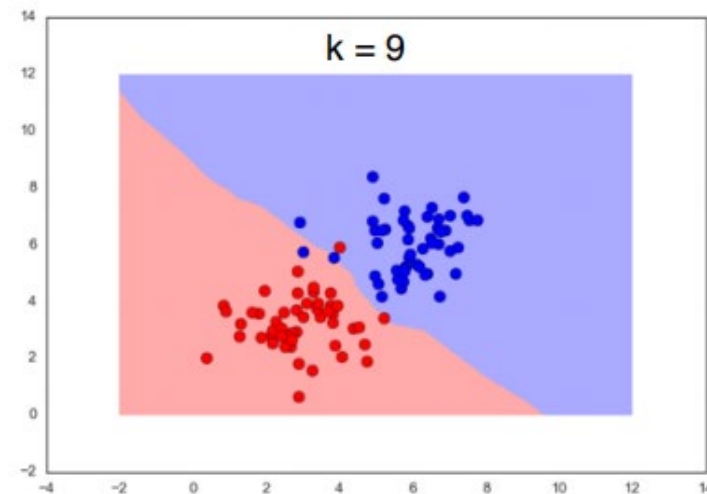
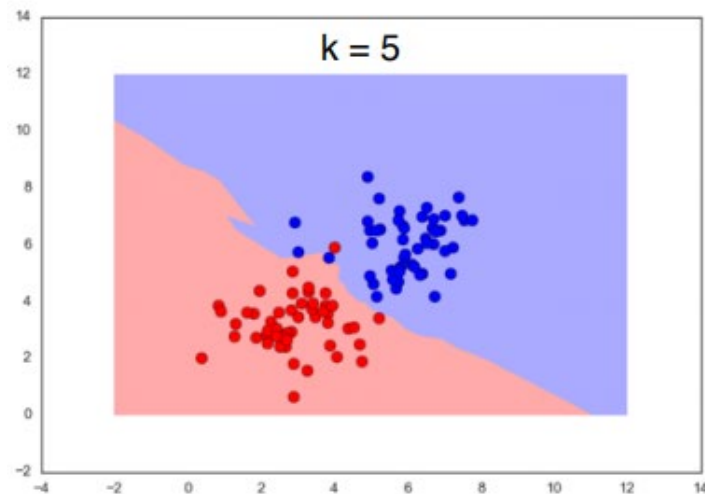
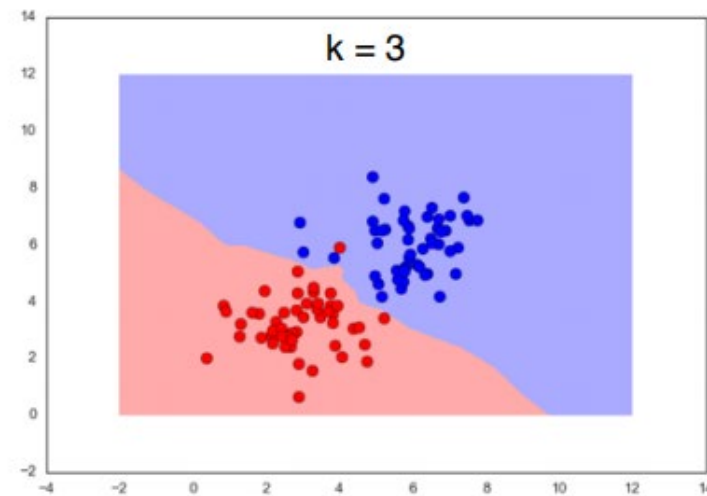
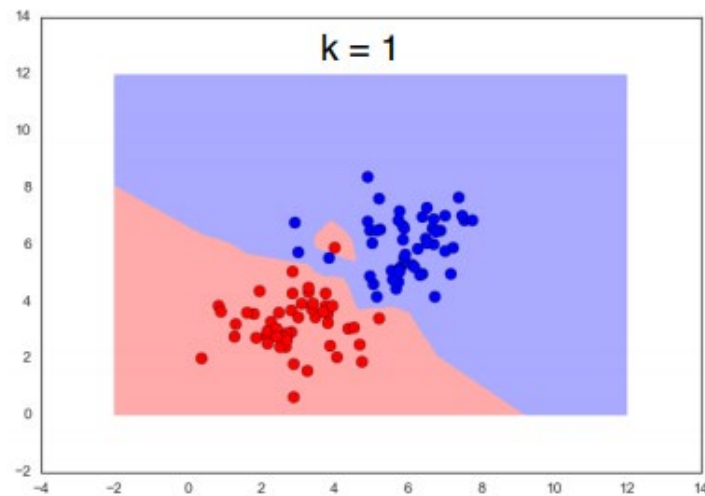


# kNN classification

- Predict class of new data point by majority vote of  $k$  nearest neighbors in training set

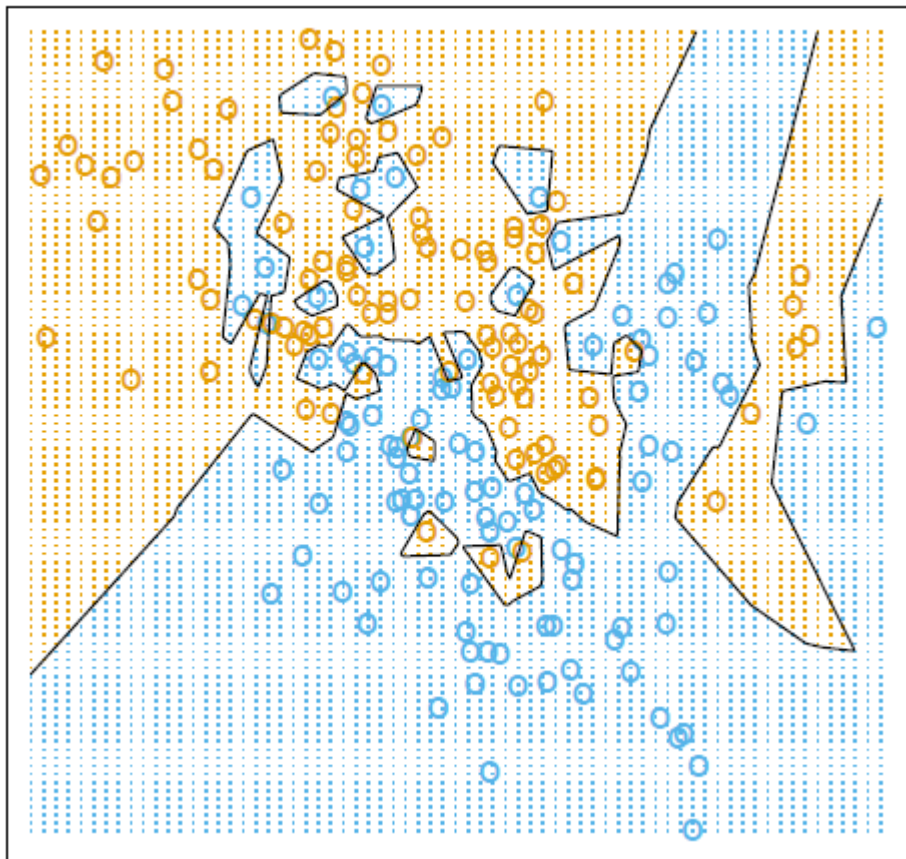


# Impact of $k$

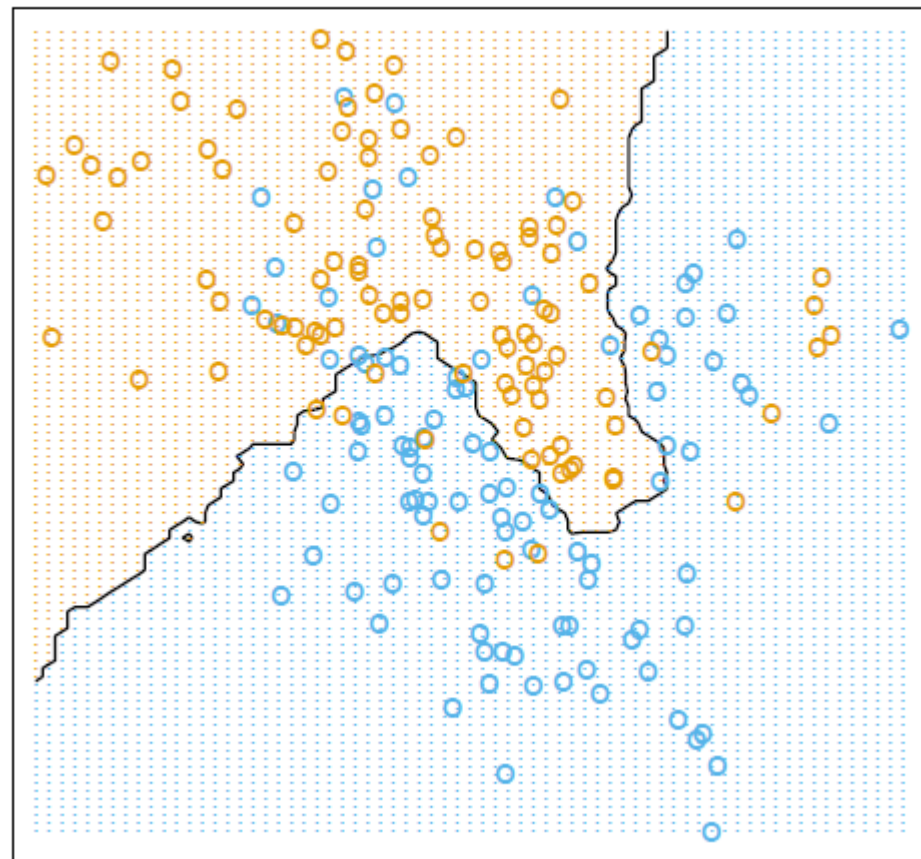


# Impact of $k$

1-Nearest Neighbor Classifier



15-Nearest Neighbor Classifier



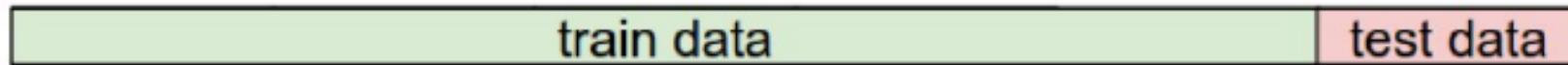
(notebook)

Source: Hastie et al., 2009. The elements of statistical learning 2<sup>nd</sup> ed.

---

# How to choose $k$ ?

- Validation:
  - Train on training data, test on test data
  - Choose  $k$  with lowest test error

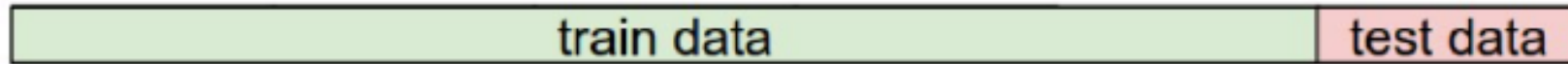


- Training data: train classifier
- Test data: measure performance

---

# How to choose $k$ ?

- Validation:
  - Train on training data, test on test data
  - Choose  $k$  with lowest test error



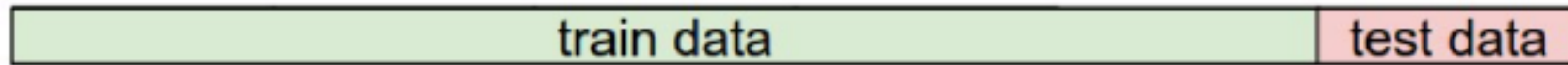
## Problem

- Test data is supposed to estimate classifier performance on **new data**
- We choose the algorithm according to how it predicts the test data
  - Performance measure is no longer independent of our training procedure

---

# How to choose $k$ ?

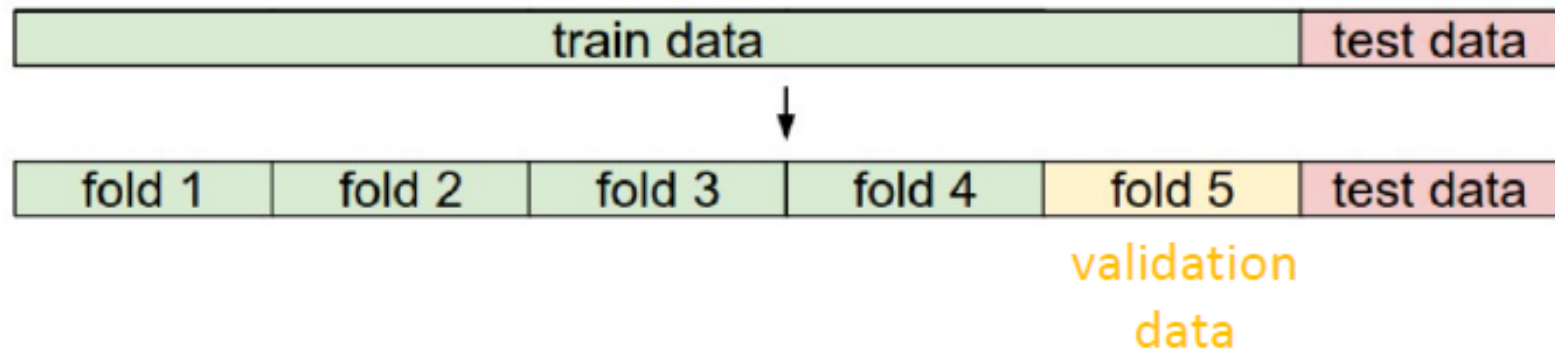
- Validation:
  - Train on training data, test on test data
  - Choose  $k$  with lowest test error



## Problem

- Test data is supposed to estimate classifier performance on **new data**
- We choose the algorithm according to how it predicts the test data
  - Performance measure is no longer independent of our training procedure
- Best  $k$  on the test set may be different than best  $k$  on (really) new data

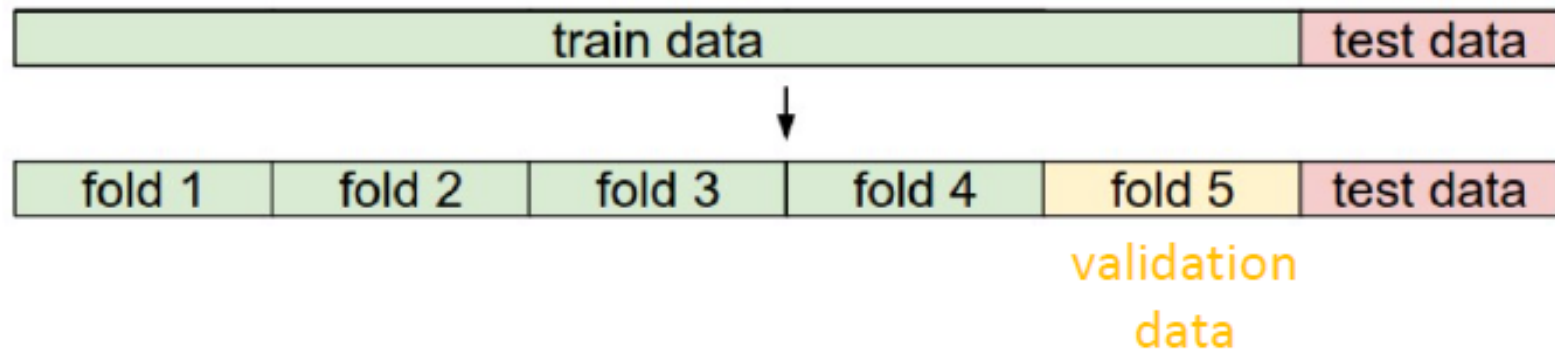
# Cross validation



- Training data: train classifier
- Validation data: estimate parameters ( $k$  for  $k$ NN)
- Test data: measure performance



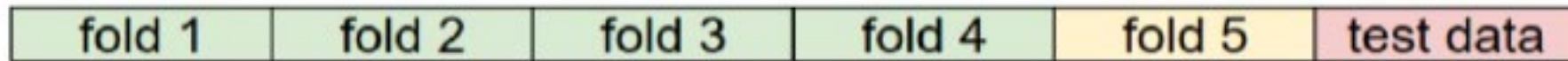
# Cross validation



- Training data: train classifier
- Validation data: estimate parameters ( $k$  for  $k$ NN)
- Test data: measure performance
- This procedure is also called  $K$ -fold CV. The  $K$  here has nothing to do with the  $k$  in  $k$ NN
  - Here we have 5-fold CV

---

# Cross validation



---

# Cross validation



1. Iterate over choice of validation fold

---

# Cross validation



1. Iterate over choice of validation fold
2. For all parameter values:
  1. Train with ( $K-1$  folds of) training data
  2. Validate (i.e. get performance estimate) with validation data

---

# Cross validation



1. Iterate over choice of validation fold
2. For all parameter values:
  1. Train with ( $K-1$  folds of) training data
  2. Validate (i.e. get performance estimate) with validation data
3. Pick parameter values with best performance on validation data

# Cross validation



1. Iterate over choice of validation fold
2. For all parameter values:
  1. Train with ( $K-1$  folds of) training data
  2. Validate (i.e. get performance estimate) with validation data
3. Pick parameter values with best performance on validation data

The test data is never used to determine the parameters!

---

# Cross validation

- Take best parameters based on validation data
- After best parameter is set:  
Retrain classifier on training data and validation data together
  - All  $K$  folds
- Test performance on test data
  - Evaluate on the test set only a single time, at the very end!

---

# kNN – more things to consider

- Breaking ties
  - Two classes: use uneven  $k$
  - At random?
    - More than two classes: shuffle training data first



---

# kNN – more things to consider

- Breaking ties
  - Two classes: use uneven  $k$
  - At random?
    - More than two classes: shuffle training data first
- Weighting of “nearer” vs. “less near” neighbors?

---

# kNN – more things to consider

- Breaking ties
  - Two classes: use uneven  $k$
  - At random?
    - More than two classes: shuffle training data first
- Weighting of “nearer” vs. “less near” neighbors?
- Many features: “nearest” neighbor?

---

# kNN – more things to consider

- Breaking ties
  - Two classes: use uneven  $k$
  - At random?
    - More than two classes: shuffle training data first
- Weighting of “nearer” vs. “less near” neighbors?
- Many features: “nearest” neighbor?
- Categorical features

---

# Categorical features

- kNN relies on numeric distance
- What is the distance between “yes” and “no” or “normal” and “abnormal”?
  - Categorical features do not appear in our heatmap
- We can code them into numerical values and use these values for distance calculations
  - E.g. 0 for “normal” and 1 for “abnormal”
- But what if we have more than 2 categories?
  - “Red”, “Green”, “Blue”
  - What is the problem of coding them numerically: 0=“red”, 1=“green”, 2=“blue”?
  - What if the categories are “bad”, “average”, “good”?

---

# One-hot encoding

- We use a vector size  $m$  to represent a categorical variable with  $m$  categories
- For each category, one of the elements in the vector is “hot” (1) and the others are not (0)

Blue	Green	Red
0	0	1
0	1	0
1	0	0

- Red is coded as 1,0,0
- Green as 0,1,0
- Blue as 0,0,1

(notebook)

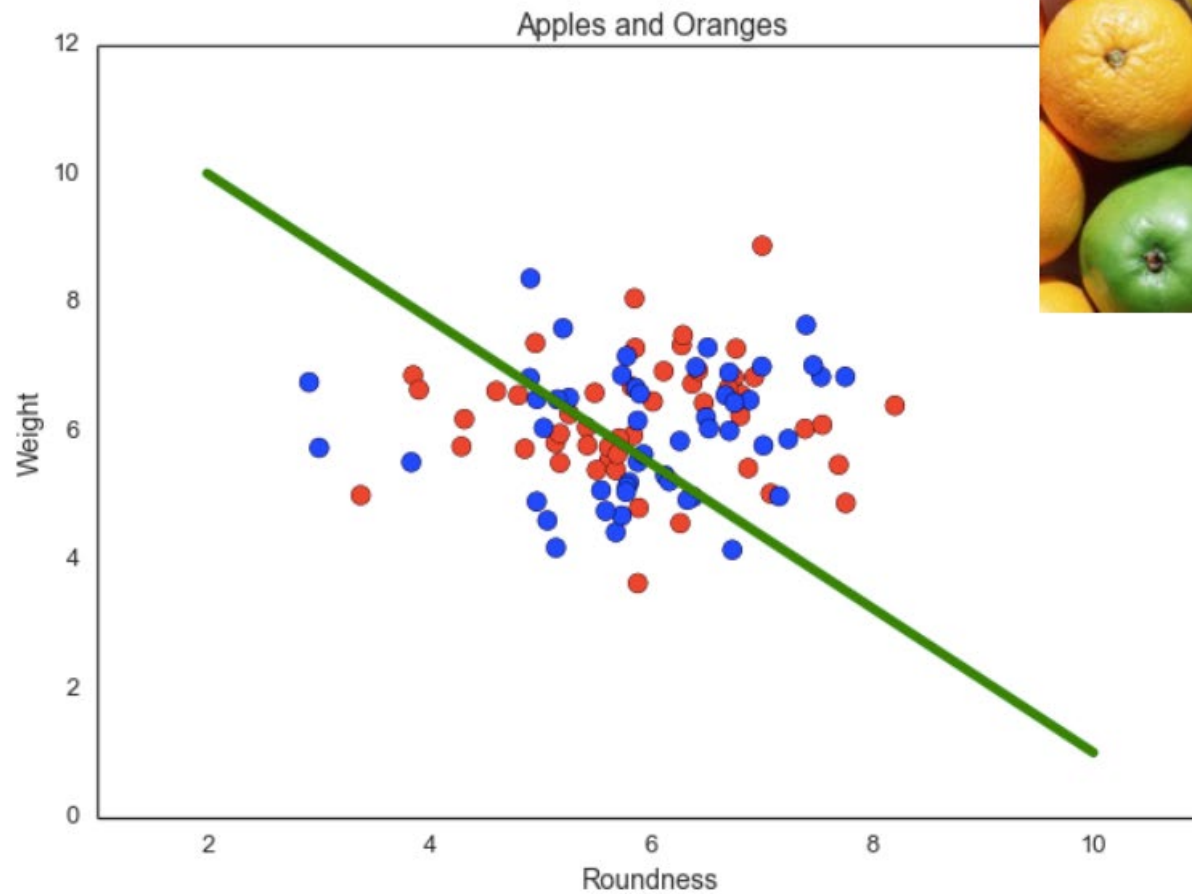
# Machine learning traps

---

# Some machine learning traps...

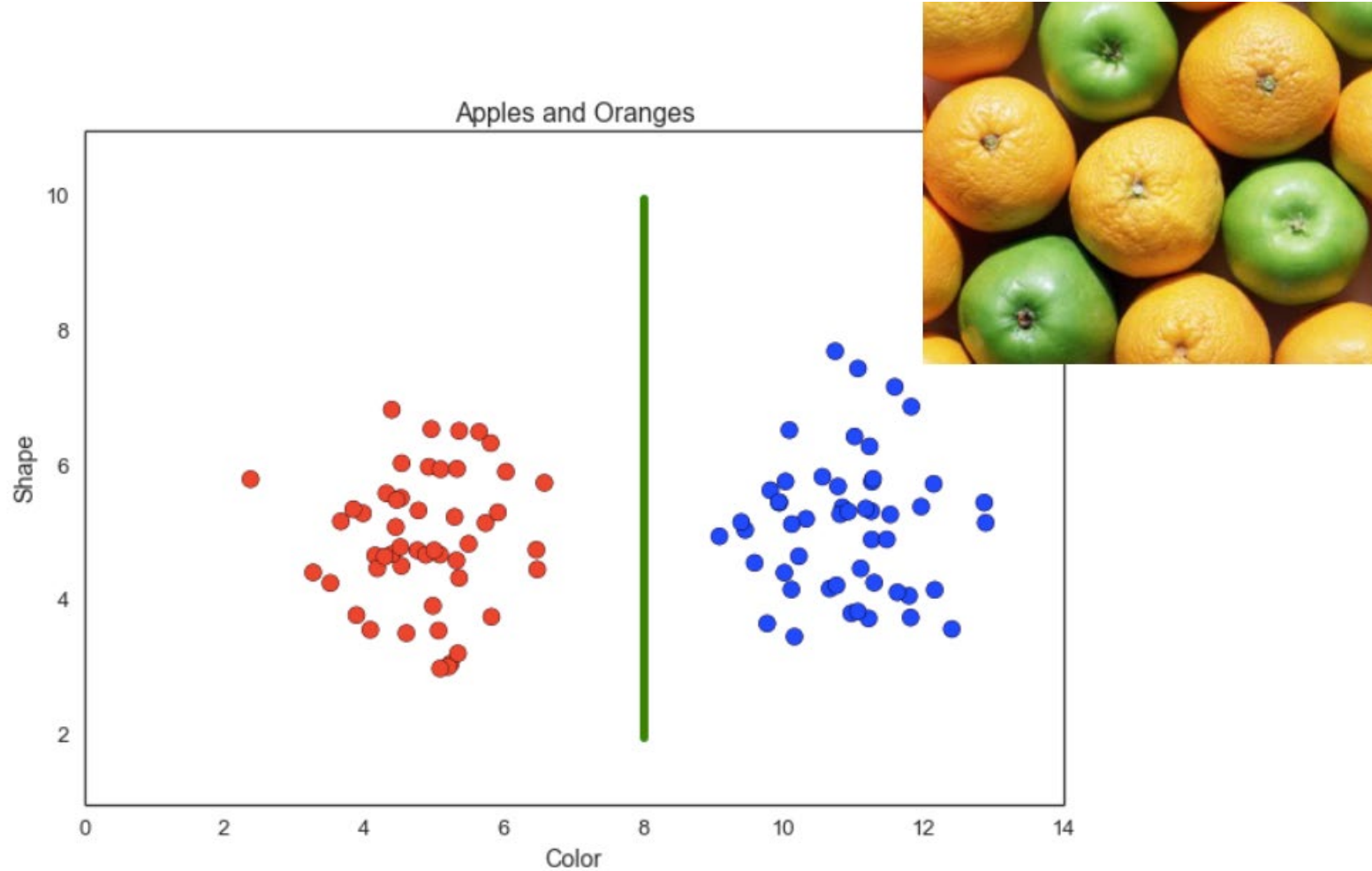
- Feature selection
- High-dimensional data (many features)
- Overfitting
- Data leakage
- Imbalanced data

# Feature selection





# Feature selection



---

# Feature selection: thought exercise

Data regarding people

- Predictors:
  - Weight
  - Gender
  - Age
  - Zip code
- Class to predict: Height over 170 cm (Yes/No)

---

# Feature selection: thought exercise

Data regarding people

- Predictors:
  - Weight
  - Gender
  - Age
  - Zip code
- Class to predict: Height over 170 cm (Yes/No)
- What happens if we use all predictors?
- Can we do better than use all predictors?

---

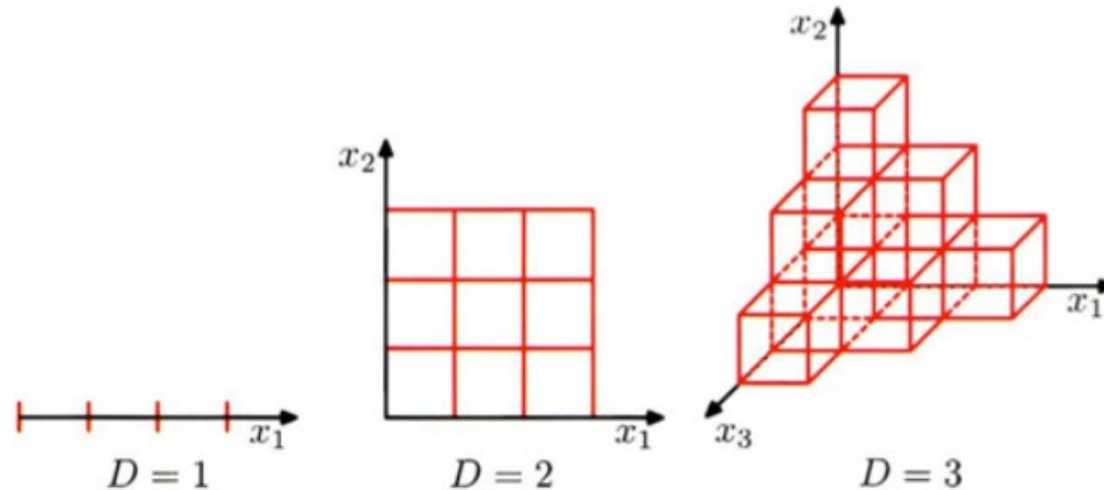
# Feature selection

- There are different methods for feature selection
- Basic approach:  
use features that are highly correlated (in absolute value) with the response
  - Can (should?) also first remove features that are highly correlated with each other

(notebook)

# Curse of dimensionality

- When dimensionality increases, the volume of the space increases so fast that the available data becomes *sparse*
- Statistically sound result requires the sample size  $N$  to grow exponentially with the dimension  $D$
- There are dimensionality reductions algorithms
  - Not in this course



---

# Overfitting

- Task: classify images to Gray Wolf or Siberian Husky



Wolf

Husky

---

# Overfitting

Training data:



Wolf



Husky



# Overfitting



Wolf



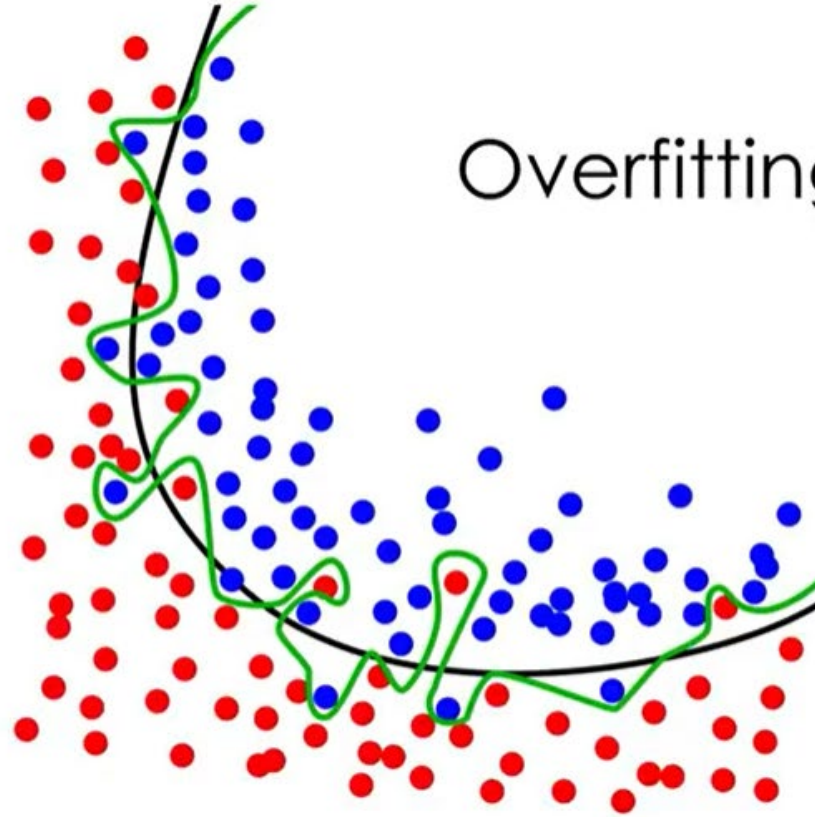
Husky

Test case:





Overfitting



# Google flu predictions



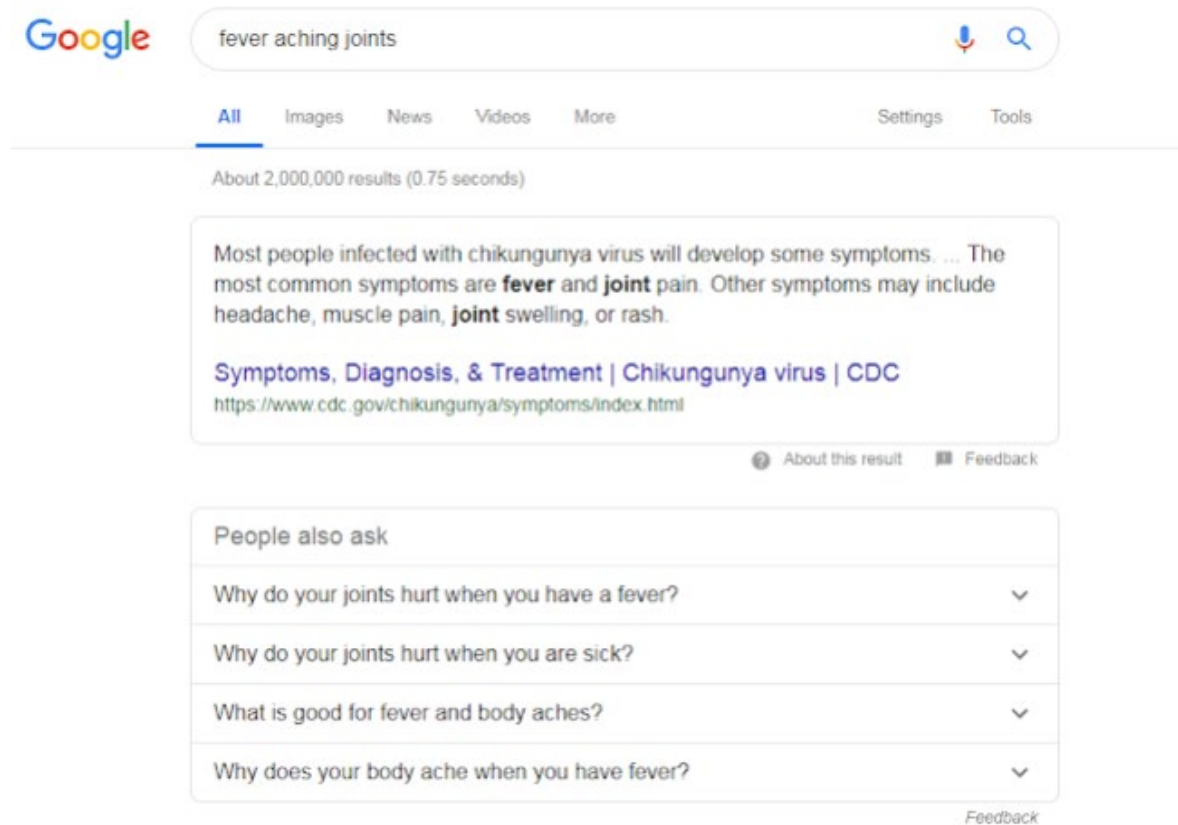
Letter | Published: 19 November 2008

## Detecting influenza epidemics using search engine query data

Jeremy Ginsberg, Matthew H. Mohebbi, Rajan S. Patel, Lynnette Brammer, Mark S. Smolinski & Larry Brilliant



# Predicting based on search queries



The image shows a Google search interface. At the top, the Google logo is on the left, and a search bar contains the text "fever aching joints". To the right of the search bar are icons for voice search and a magnifying glass. Below the search bar, there are tabs for "All", "Images", "News", "Videos", and "More". The "All" tab is selected and underlined. To the right of these tabs are links for "Settings" and "Tools". Below the tabs, it says "About 2,000,000 results (0.75 seconds)".

The main search result is a snippet from the CDC website. It reads: "Most people infected with chikungunya virus will develop some symptoms. ... The most common symptoms are **fever** and **joint** pain. Other symptoms may include headache, muscle pain, **joint** swelling, or rash." Below this snippet is a link titled "Symptoms, Diagnosis, & Treatment | Chikungunya virus | CDC" with the URL "https://www.cdc.gov/chikungunya/symptoms/index.html". To the right of the link are icons for "About this result" and "Feedback".

Below the main result is a section titled "People also ask". It contains four questions, each with a downward arrow icon to its right:

- Why do your joints hurt when you have a fever?
- Why do your joints hurt when you are sick?
- What is good for fever and body aches?
- Why does your body ache when you have fever?

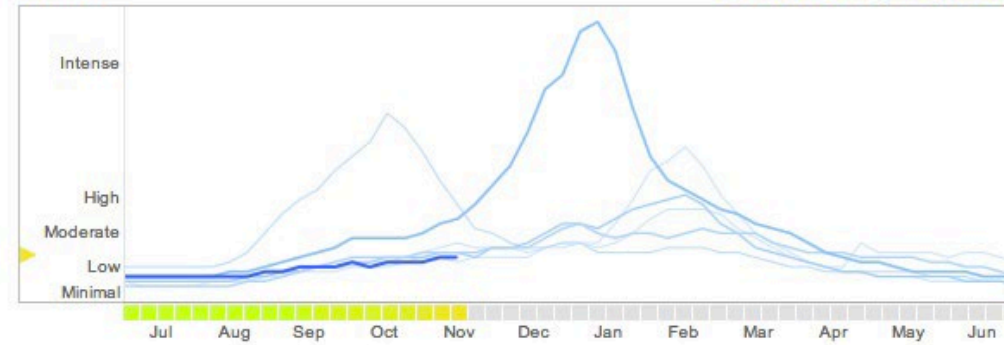
At the bottom right of the "People also ask" section is a "Feedback" link.

## Explore flu trends - United States

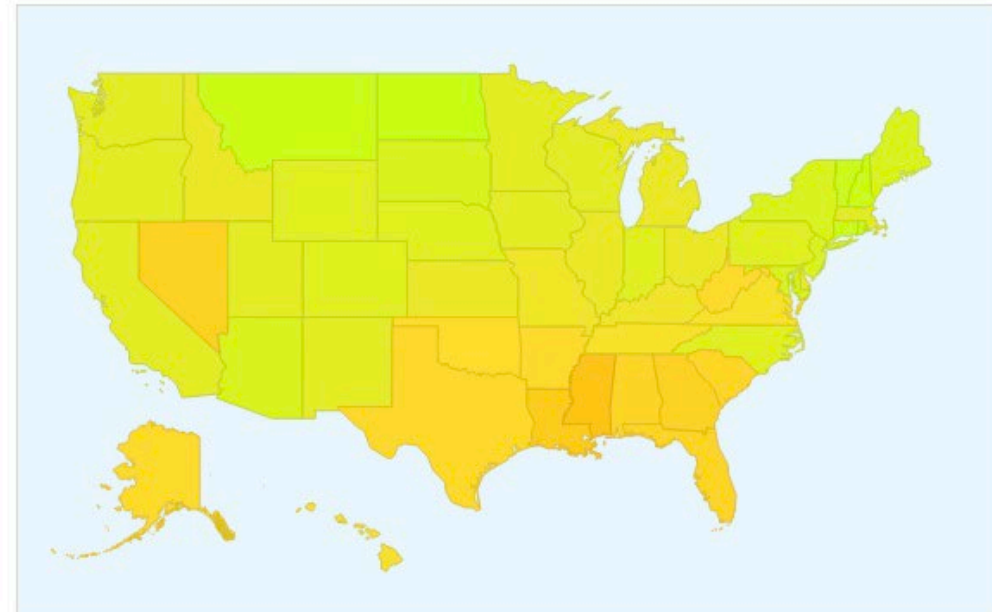
We've found that certain search terms are good indicators of flu activity. Google Flu Trends uses aggregated Google search data to estimate flu activity. [Learn more »](#)

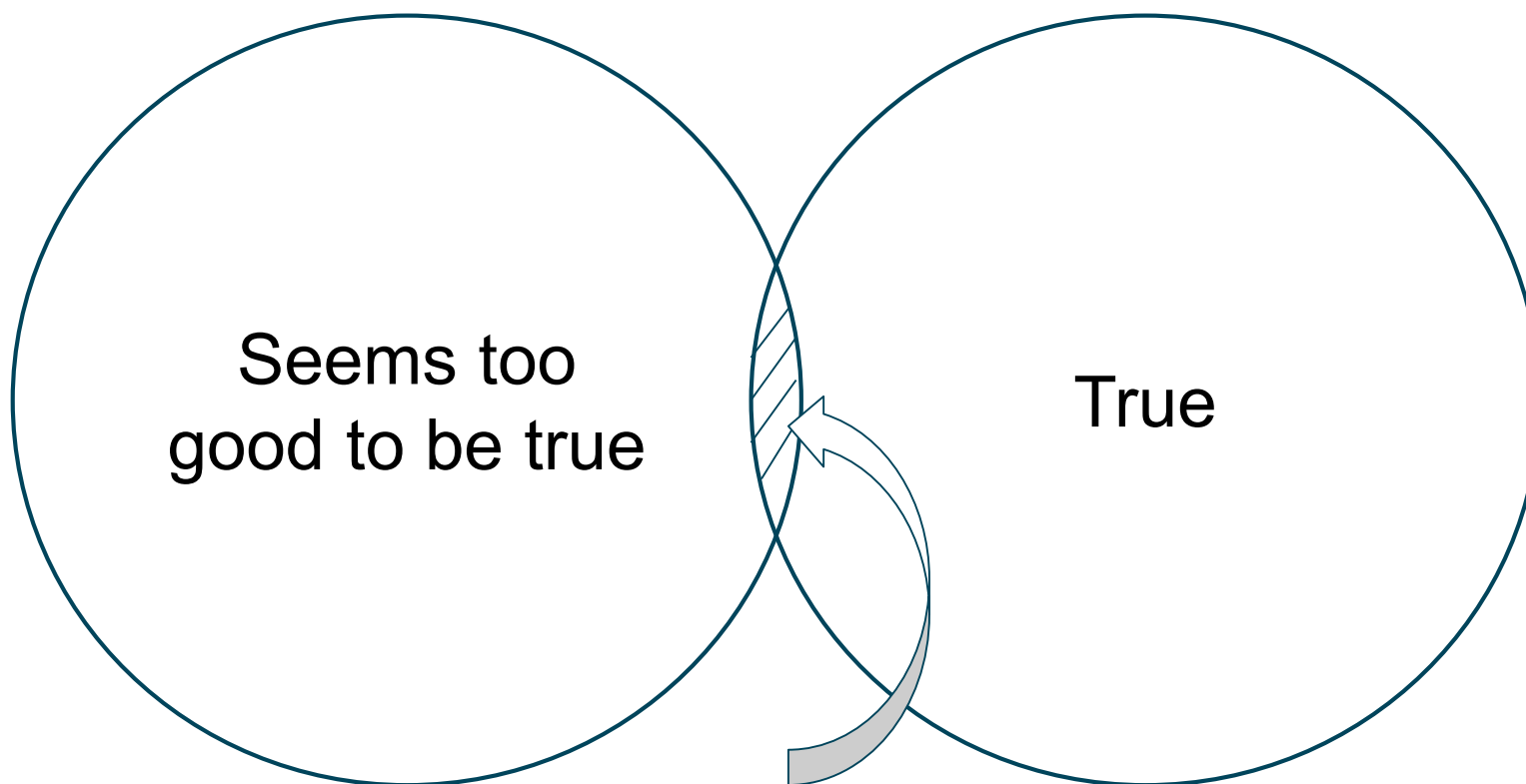
### National

● 2013-2014 ● [Past years](#) ▼



### States | [Cities](#) (Experimental)





*You wish...*



### Thank you for stopping by.

Google Flu Trends and Google Dengue Trends are [no longer publishing](#) current estimates of Flu and Dengue fever based on search patterns. The historic estimates produced by Google Flu Trends and Google Dengue Trends are available below. It is still early days for nowcasting and similar tools for understanding the spread of diseases like flu and dengue – we're excited to see what comes next. Academic research groups interested in working with us should fill out this [form](#).

Sincerely,

The Google Flu and Dengue Trends Team.

### Google Flu Trends Data:

You can also see this data in [Public Data Explorer](#)

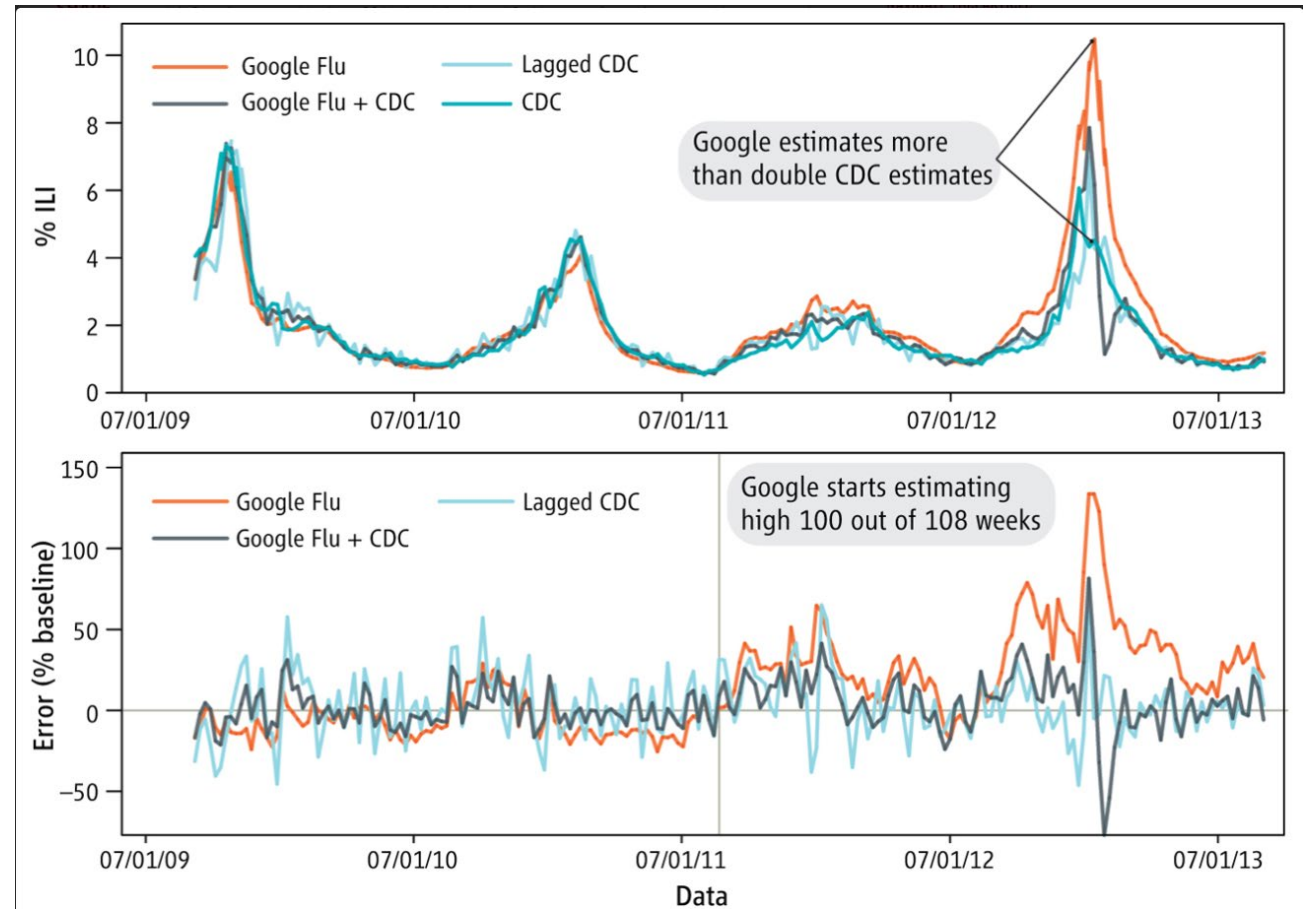
- [World](#)
- [Argentina](#)
- [Australia](#)
- [Austria](#)
- [Belgium](#)
- [Bolivia](#)
- [Brazil](#)
- [Bulgaria](#)
- [Canada](#)
- [Chile](#)
- [France](#)
- [Germany](#)
- [Hungary](#)

# What went wrong?

BIG DATA

## The Parable of Google Flu: Traps in Big Data Analysis

David Lazer,<sup>1,2\*</sup> Ryan Kennedy,<sup>1,3,4</sup> Gary King,<sup>3</sup> Alessandro Vespignani<sup>5,6,3</sup>



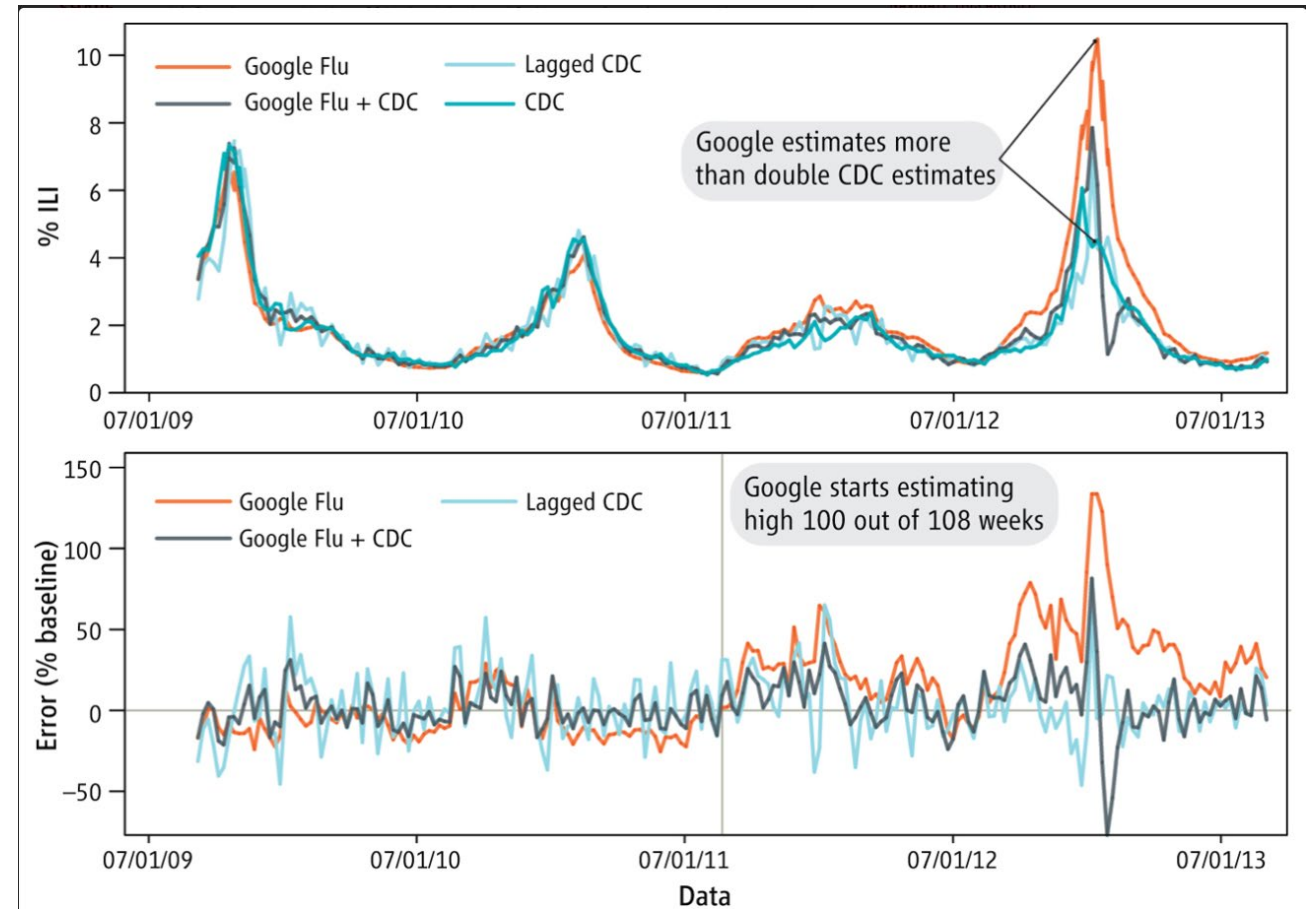
# What went wrong?

BIG DATA

## The Parable of Google Flu: Traps in Big Data Analysis

David Lazer,<sup>1,2\*</sup> Ryan Kennedy,<sup>1,3,4</sup> Gary King,<sup>3</sup> Alessandro Vespignani<sup>5,6,3</sup>

“In short, the initial version of GFT was part flu detector, part winter detector”





---

# Data leakage

- Data leakage is the use of information in the model training process which would not be expected to be available at prediction time
- Example: using the test set to determine  $k$  (instead of CV)
- Example: imagine we are Netflix, and we want to predict whether or not a user will like a certain mini-series. The data includes a “like/dislike” rating for everything the user watched, by episode.
  - What will happen if we randomly split the data to a train set and a test set?

---

# Imbalanced data

- Sometimes, we might have much more data representing one class compared to another class
- For example: when learning to classify spam email, we might have very few examples of spam email compared to non-spam emails

---

# The metric trap

- Classify spam:
  - 95% in test are non-spam
  - 5% in test are spam
- What is the accuracy of a classifier that always classifies “not spam”?

---

# Confusion matrix

Here's a more detailed way to look at our classifier's performance

	True label: Spam	True label: Not spam
Predicted label: Spam	0	0
Predicted label: Not spam	50	950

---

# Confusion matrix

Here's a more detailed way to look at our classifier's performance

	True label: Spam	True label: Not spam
Predicted label: Spam	0 (True Positive)	0 (False Positive)
Predicted label: Not spam	50 (False Negative)	950 (True Negative)

- True positive rate (TPR):  $TP/(TP+FN) = TP/P = 0/50 = 0$
- False positive rate (FPR):  $FP/(FP+TN) = FP/N = 0/950 = 0$

---

# Beyond accuracy

- Precision = Fraction of “true positives” among total predicted positives
  - How “useful” is our classifier in recovering true cases when predicting a positive label
- Recall = Fraction of “true positives” among total *ground-truth* positives
  - How “complete” is our classifier in recovering positive labels; TPR

# Beyond accuracy

- Precision = Fraction of “true positives” among total predicted positives
  - How “useful” is our classifier in recovering true cases when predicting a positive label
- Recall = Fraction of “true positives” among total *ground-truth* positives
  - How “complete” is our classifier in recovering positive labels; TPR

	True label: Spam	True label: Not spam
Predicted label: Spam	0 (True Positive)	0 (False Positive)
Predicted label: Not spam	50 (False Negative)	950 (True Negative)

# Beyond accuracy

- Precision = Fraction of “true positives” among total predicted positives
  - How “useful” is our classifier in recovering true cases when predicting a positive label
- Recall = Fraction of “true positives” among total *ground-truth* positives
  - How “complete” is our classifier in recovering positive labels; TPR

	True label: Spam	True label: Not spam
Predicted label: Spam	0 (True Positive)	0 (False Positive)
Predicted label: Not spam	50 (False Negative)	950 (True Negative)

- Precision =  $TP/(TP+FP) = 0/0$  (undefined)
- Recall =  $TP/(TP+FN) = TPR = 0/50 = 0$



---

# **F<sub>1</sub> score**

- Considers both precision and recall to assess classifier

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- F<sub>1</sub> is between 0 (worst) and 1 (best)
  - If precision or recall are undefined, we can set F<sub>1</sub> to 0

---

# F<sub>1</sub> score

- Considers both precision and recall to assess classifier

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- F<sub>1</sub> is between 0 (worst) and 1 (best)
  - If precision or recall are undefined, we can set F<sub>1</sub> to 0
- F1 problems
  - Does not consider true negatives
  - Weights precision and recall similarly, but this is an application decision

# Another classifier

- Let's say we have another classifier with the following confusion matrix:

	True label: Spam	True label: Not spam
Predicted label: Spam	50 (true positive)	50 (false positive)
Predicted label: Not spam	0 (false negative)	900 (true negative)

- Accuracy?
- TPR? FPR?
- Precision? Recall?  $F_1$ ?
  - Is it good or bad?

---

# Sensitivity and specificity

- Sensitivity = Fraction of “true positives” among total predicted positives
  - Probability of positive classification given true label is positive (=recall=TPR)
- Specificity = Fraction of “true negatives” among total *ground-truth* negatives
  - Probability of negative classification given true label is negative (=TNR)

# Sensitivity and specificity

- Sensitivity = Fraction of “true positives” among total predicted positives
  - Probability of positive classification given true label is positive (=recall=TPR)
- Specificity = Fraction of “true negatives” among total *ground-truth* negatives
  - Probability of negative classification given true label is negative (=TNR)

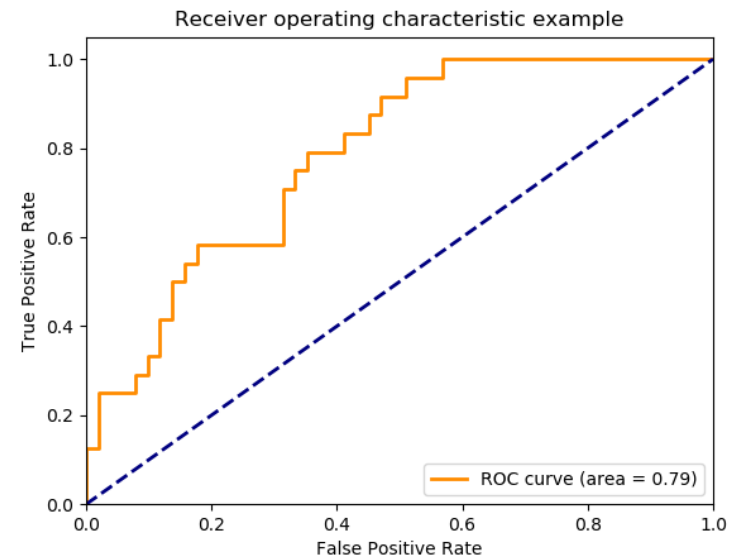
	True label: Spam	True label: Not spam
Predicted label: Spam	0 (True Positive)	0 (False Positive)
Predicted label: Not spam	50 (False Negative)	950 (True Negative)

- Sensitivity =  $TP/(TP+FN) = 0/50 = 0$
- Specificity =  $TN/(TN+FP) = 950/950 = 1$

# Other metrics

- Balanced accuracy: mean of the true positive rate (TPR) and the true negative rate (TNR)
  - Average of sensitivity and specificity
- ROC curve: Sensitivity (TPR) vs. (1-Specificity) (FPR)
  - The more sensitive a classifier becomes, the less specific it usually is

See `sklearn.metrics`



---

# When should we train a classifier?

- In general, we will train a classifier when we want to learn a way to predict the value of a target variable based on a set of available features (e.g., predict who will drop out of university and offer help)
- Note that in some cases we don't actually need a classifier to predict the value of a certain target variable. Examples:
  - Predict the neighborhood from latitude and longitude coordinates
  - Predict the severity of a crime based on offense type