Machine learning for the social sciences

Michele Tizzoni

PhD Programme in Sociology and Social Research

First technical concepts

Overview

- Train and test
- Generalizability
- Performance evaluation

The classic OLS

OLS/logistic regression from a statistics standpoint → we use the entire dataset to estimate our θ.

ML approach ⇒ assuming a dataset with $i.i.d.$ observations, we split it in 2 sets:

- Train Set: set of examples (e.g., pairs (x, y)) used to the train the algorithm ⇒ these inputs are used to fit the parameters through optimization methods (e.g., SGD)
- Test Set: the independent set of unseen examples that is used to evaluate the performance of the algorithm on the task, based on the learned weights in the training phase

+ sometimes we have an intermediate set, called validation set, in which hyperparameters are tuned to further optimize the performance of the algorithm.

Example on a regression task

Let's pretend we have a dataset containing 150,000 obs. measuring individual school performance (y), in the range 0-100.

To train a ML regression model we need to:

- Split the dataset in train (80%?) and test (20%?) sets.
- Fit the model on train set, evaluate error, using, e.g. $\text{MSE}_{train} = \frac{1}{m}\sum_i (\hat{y}^t - y^t)_i^2$
- Fit the model on the test set, evaluate its error and performance ($\text{MSE}_{test} = \frac{1}{m}\sum_i (\hat{y}^{tt} - y^{tt})_i^2$)

What is the performance of our model in predicting individual school performance scores? We answer to this question by looking at the **test** set.

Small datasets: k-fold cross validation

Small datasets (e.g., 100 observations): k-fold cross validation is a good strategy to evaluate and validate model performance. Procedure:

1. Shuffle the observations randomly (we can → i.i.d.)
2. Split dataset into k independent sets.
3. Per each group:
    - Use it as a test test
    - Use the remaining k – 1 groups as training sets
    - Fit a model
4. Assess overall model through evaluation of each sub-model.

Small datasets: k-fold cross validation

Iteration	Train on	Test on	RMSE (Test)
1	*S1, S2, S3, S4*	S5	0.188
2	*S1, S2,S3,S5*	S4	0.211
3	*S1, S2, S4, S5*	S3	0.195
4	*S1, S3, S4, S5*	S2	0.299
5	*S2, S3, S4, S5*	S1	0.201

Overall RMSE= 0.204

Generalizability

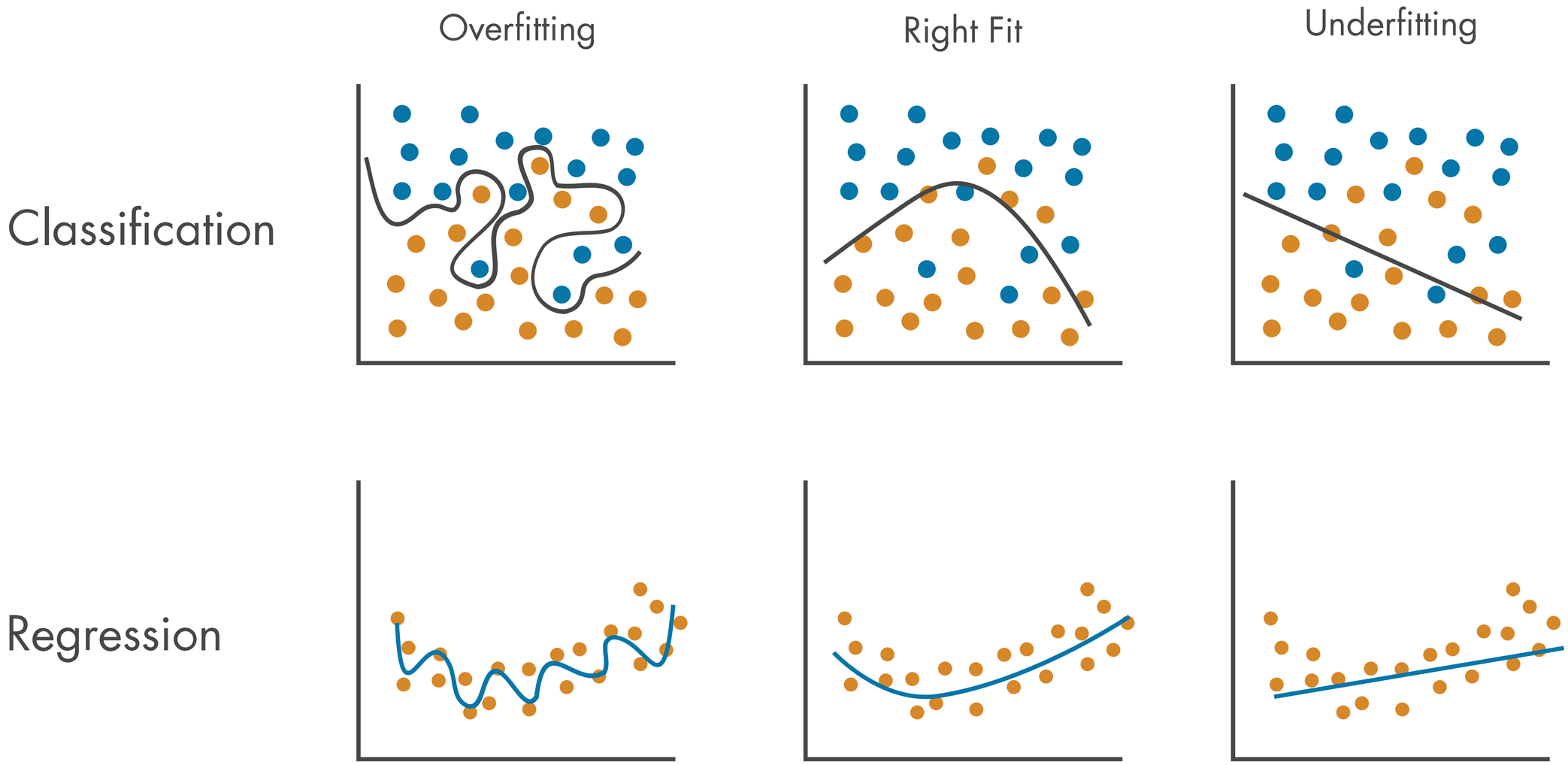
The most important aim of a ML model is **generalizability**.

The chosen algorithm must able to learn patterns that can be useful to capture the dynamic relationships in the test set. To achieve this, one's goals should be to:

- **Minimize the training error** as much as possible.
- **Reduce the gap between training and test error** as much as possible.

⇒ this brings us to two core challenges of ML: **underfitting and overfitting.**

Underfitting and overfitting



Underfitting and overfitting

- **Underfitting**: generally occurs when the model is not able to reach a sufficiently low error in the training set
- **Overfitting**: the gap between the train and test error is particularly large

A complementary explanation is the following, given an hypothesis class *H*:

- **Underfitting**: *H* is less complex than the underlying function(s) governing the data (e.g., fitting a line to data generated from a cubic polynomial)
- **Overfitting**: *H* is more complex than the underlying function(s) governing the data (e.g., fitting a cubic polynomial to data generated from a line)

How to prevent these two phenomena from occurring? ⇒ **assess model capacity**

Approaches

- Traditional statistical approaches are often used off-the-shelf in the social sciences:

  ✳ `rreg income nationality education crimhistory`

- Look for the significance stars, rule out some predictors, try others based on previous multicollinearity tests.

- ML is often extremely different, especially for complex models ⇒ **many decisions to take**, hyperparameters to tune, etc.

Suggestions

R and Python enable you to run a ML model in two lines of code. However, employing ML meaningfully is no easy task.

- Know your data.
- Ground your approach in theory/motivated hypotheses.
- Dig into the technical aspects of the selected approach.
- Engineer your feature space.
- Test, evaluate, experiment.


Evaluating a model

Model evaluation

Evaluating a model is a decisive and complex task (it requires domain expertise, besides technical knowledge).

We need to distinguish between evaluation metrics for

1. Classification
2. Regression

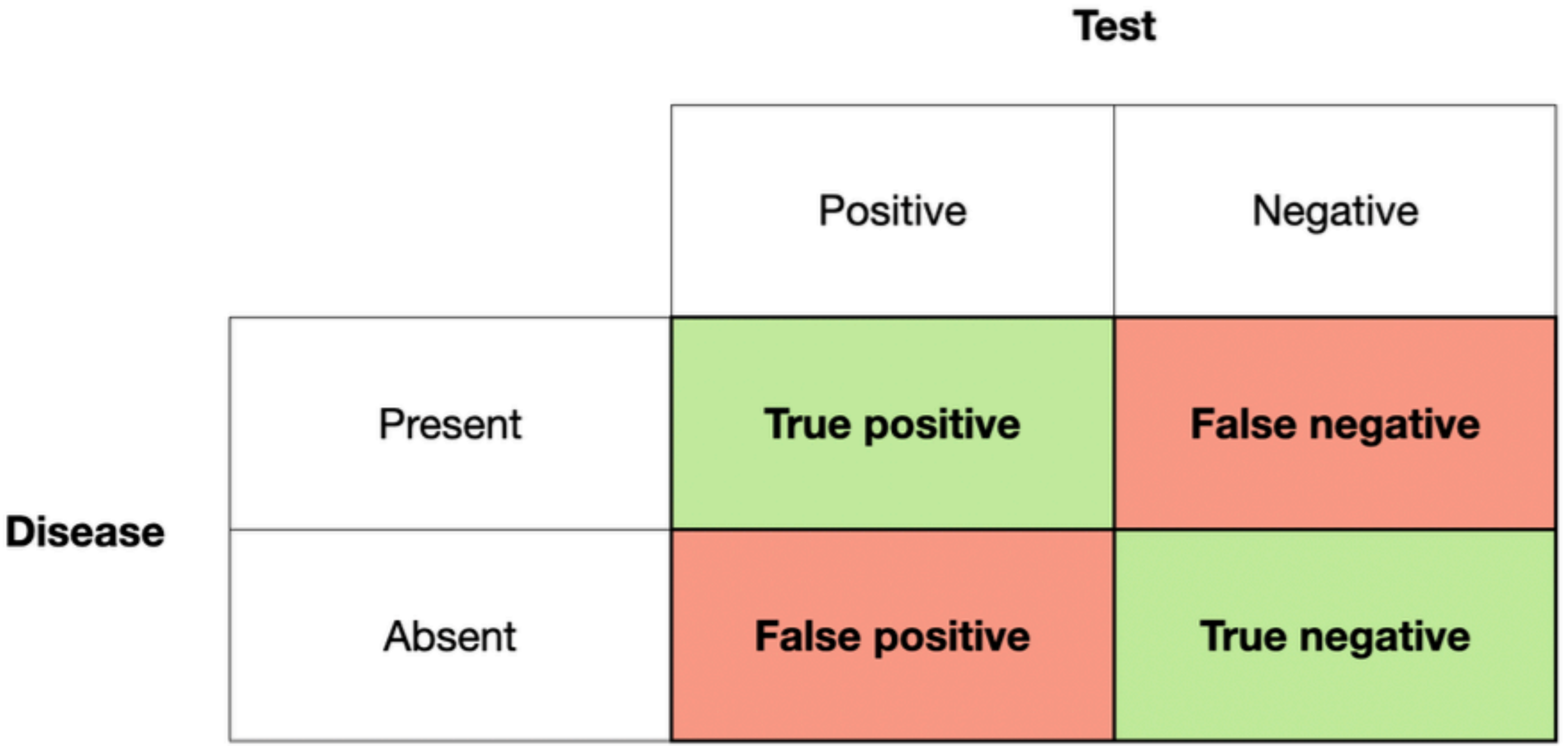
Classification: the confusion matrix

Assume (for the sake of simplicity) to have a binary classification problem.

Let's consider the case of a **health test** (a COVID-19 swab, for instance).

- You have the disease, the test is positive: **true positive**
- You have the disease, but the test is negative: **false negative**
- You don't have the disease, and the test is negative: **true negative**
- You don't have the disease, but the test is positive: **false positive**

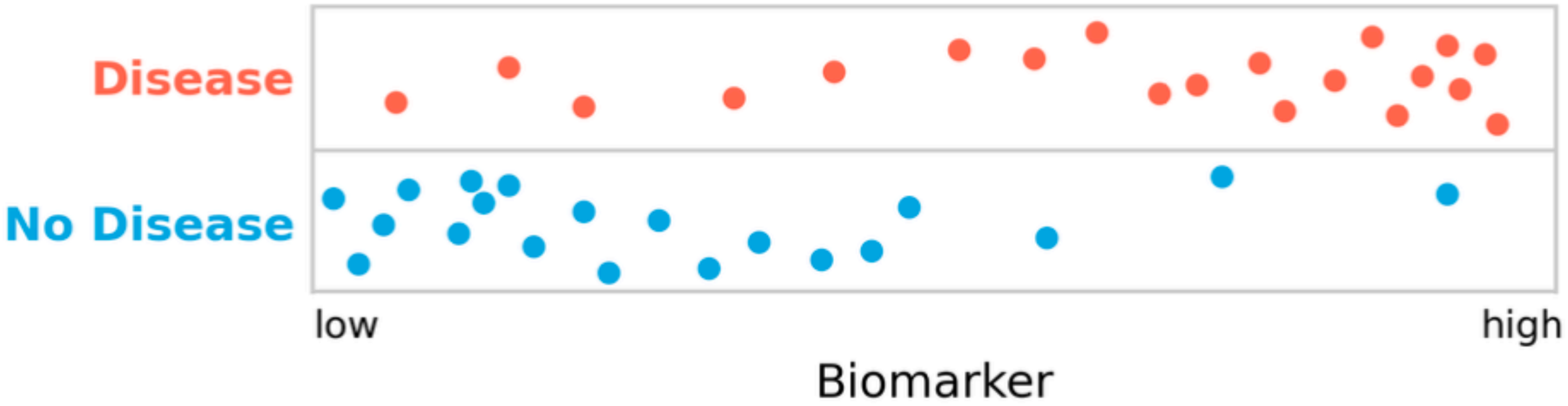
Classification: the confusion matrix



False positives and false negatives

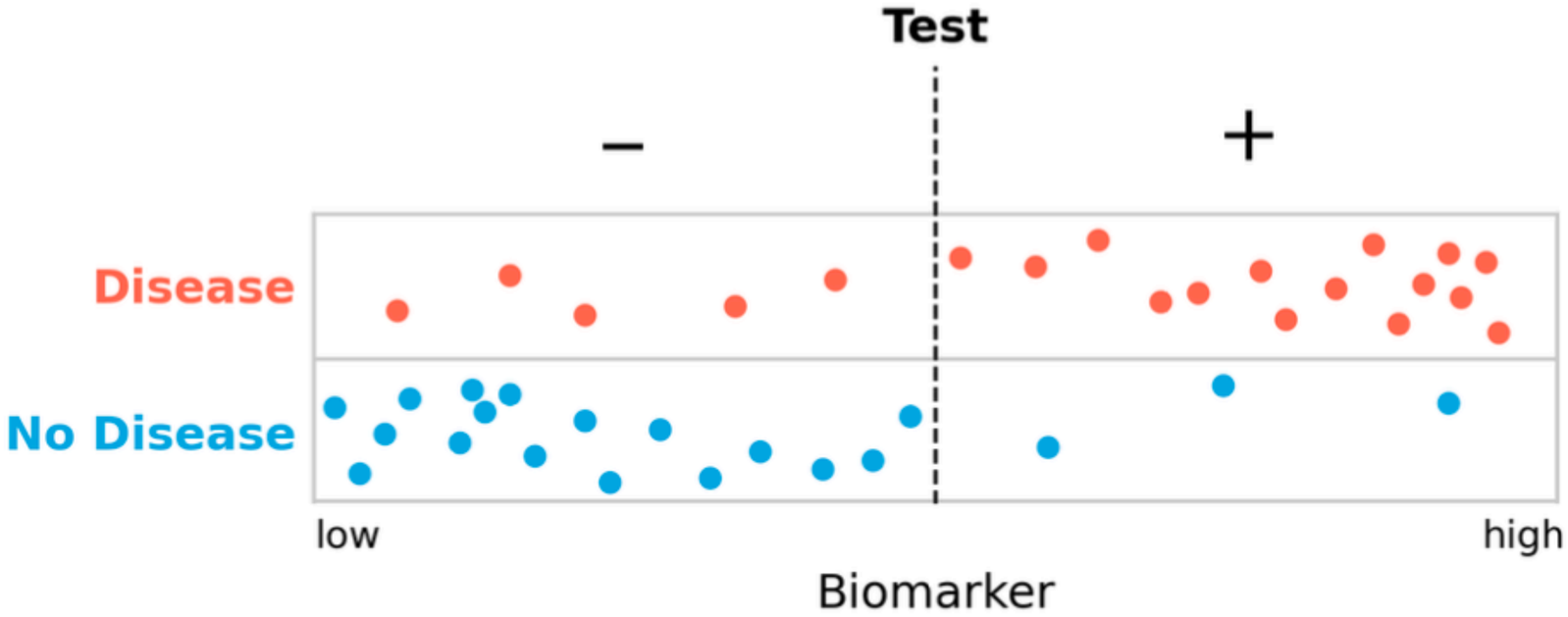
- **Type I error**: you think something is there but it is not - a false alarm.
- **Type II error**: you miss something that is actually there.
- Which error is worse usually depends on the situation.
- For example, a false positive test may lead to a treatment with very strong negative side effects. A rather bad Type I error.
- Missing a disease with a false negative test outcome can be just as bad, as the disease may not get the necessary treatment. A classic Type II error.

Testing



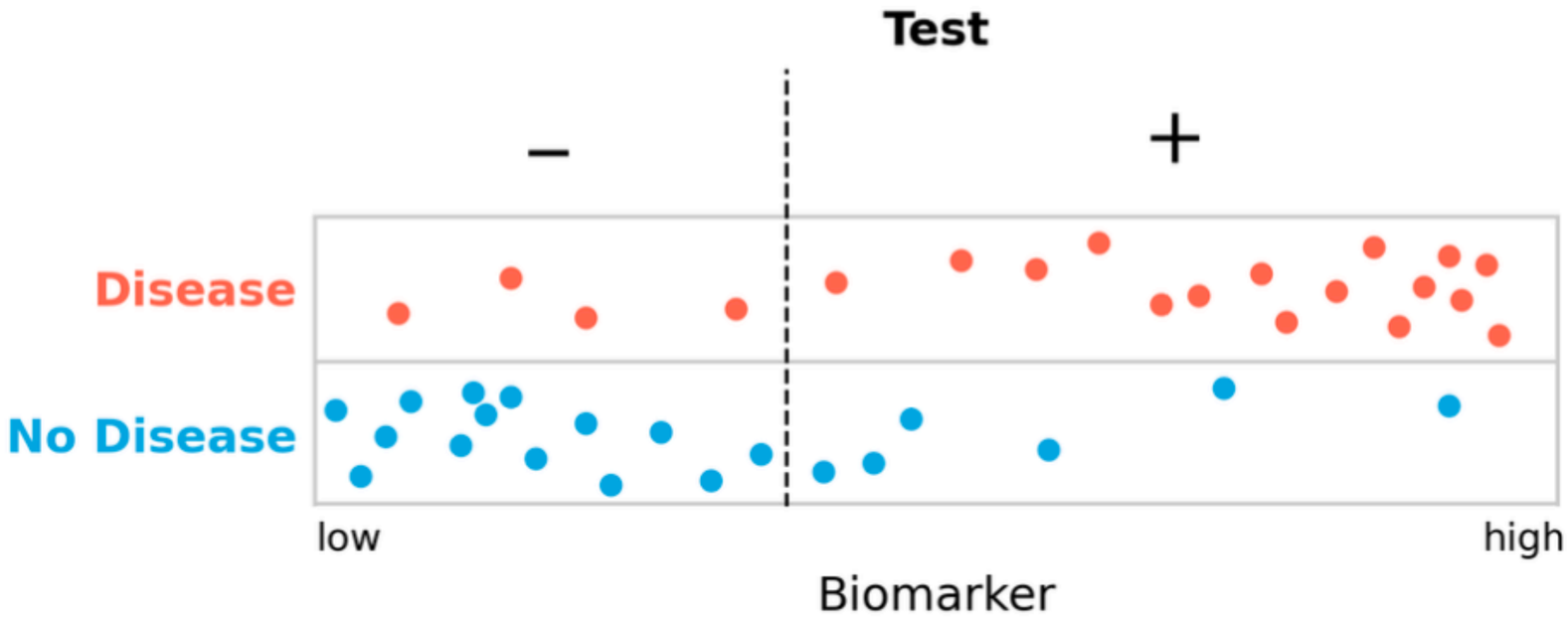
- Many diagnostics tests work by measuring the presence or amount of a **biomarker**.
- In a ML classification model, the biomarker is any input feature of the model, or a feature vector.

Testing



- We need to define a **cutoff**, above which we declare the presence of the disease.
- **There is no perfect cutoff**. The choice will impact the number of false positive and false negatives. We now miss 5 individuals with the disease…
- In a ML model this corresponds to choosing the best parameters for the classification.

Testing



- Let's move the cutoff to the left, to reduce the number of false negatives…
- Of course, the number of false positives will increase…
- We need to a framework to deal with this uncertainty.

Precision and recall

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

The Precision score (also called **Positive Predictive Value**) indicates how many times the predicted positive prediction was actually correct.

In other words, it gives as a measure of the ability of a classifier not to label as positive a sample that is negative.

Precision and recall

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

Recall (also called **Sensitivity**) quantifies the ability of the algorithm to find all the positive samples.

This metric is extremely important, for instance, in cases where a disease has to be detected by a machine.

F1 score

$$\mathbf{F1} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Sometimes, we may be interested in combining together Recall and Precision to evaluate our model performance.

F1-score is generally the solution. F1-score is the harmonic mean of the precision and recall.

This measure however calls for an important caveat: **it does not consider TN in its computation** ⇒ misleading in unbalanced datasets.

Specificity

$$\textbf{Specificity} = \frac{TN}{TN + FP}$$

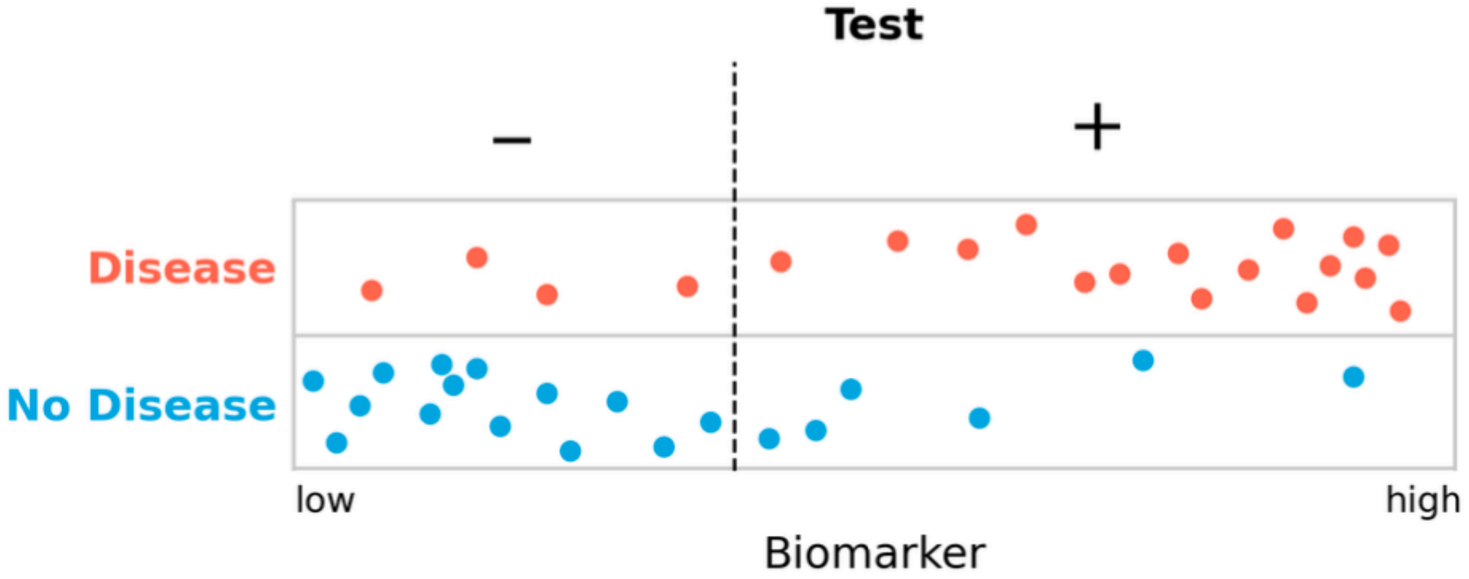
Specificity is very important in medical tests and it measures how good is a model to get **few false positives**.

Specificity is also called the **true negative rate (TNR)** and complements the sensitivity or recall which is the true positive rate (TPR).

Let’s imagine we have a “test" that is always positive. This is useless test but it will have a sensitivity of 100%. Of course, it will have a specificity of 0%, too.

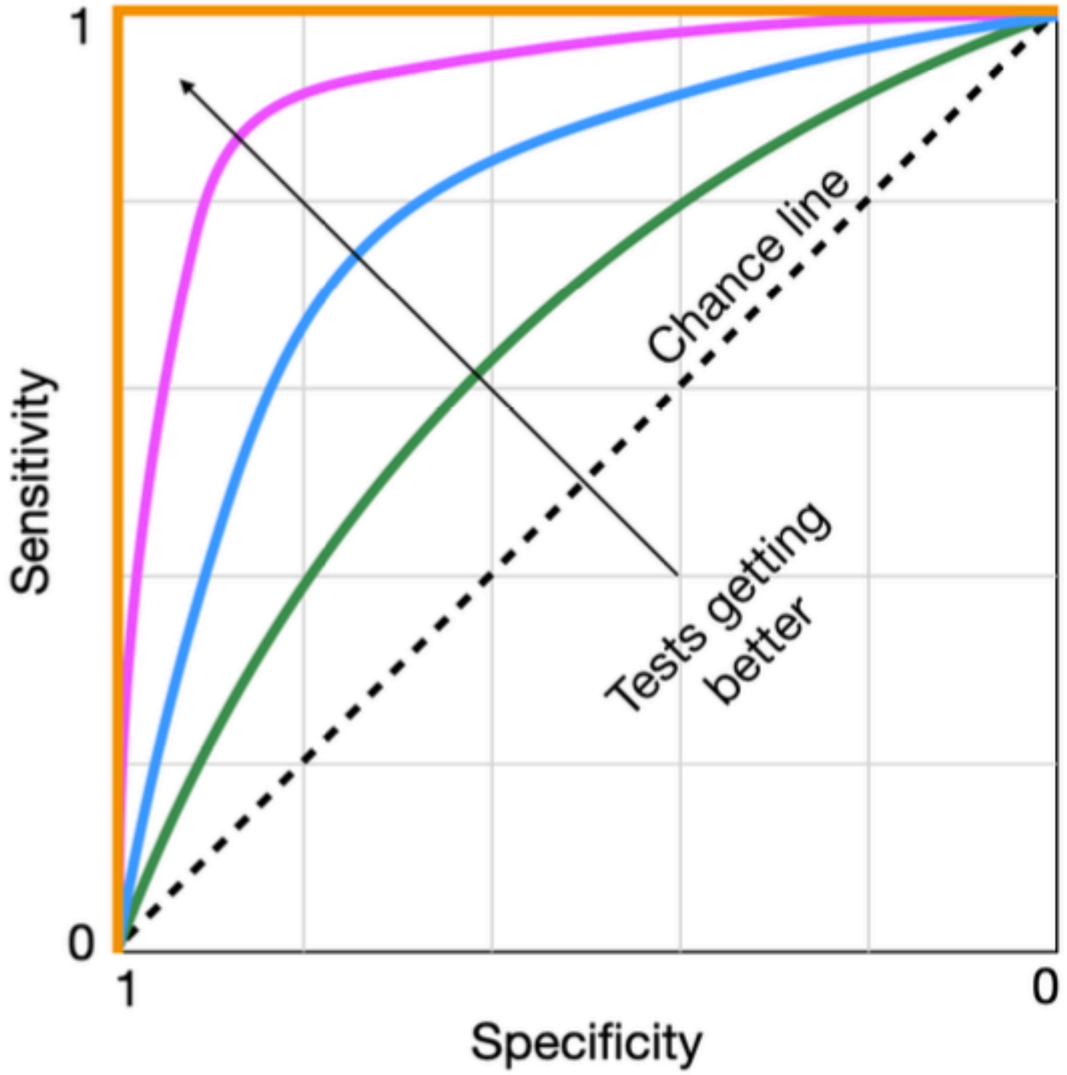
Specificity and sensitivity always go together.

The ROC Curve



- How do we make a decision about the cutoff?
- We try different cutoff and we see what that does to the tradeoff of the sensitivity and specificity of the model.
- The most common way to visualize this is the **ROC curve (receiver operating characteristics).**

The ROC Curve



$$\textbf{Specificity} = \frac{TN}{TN + FP}$$

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

(or sensitivity)

The ROC Curve

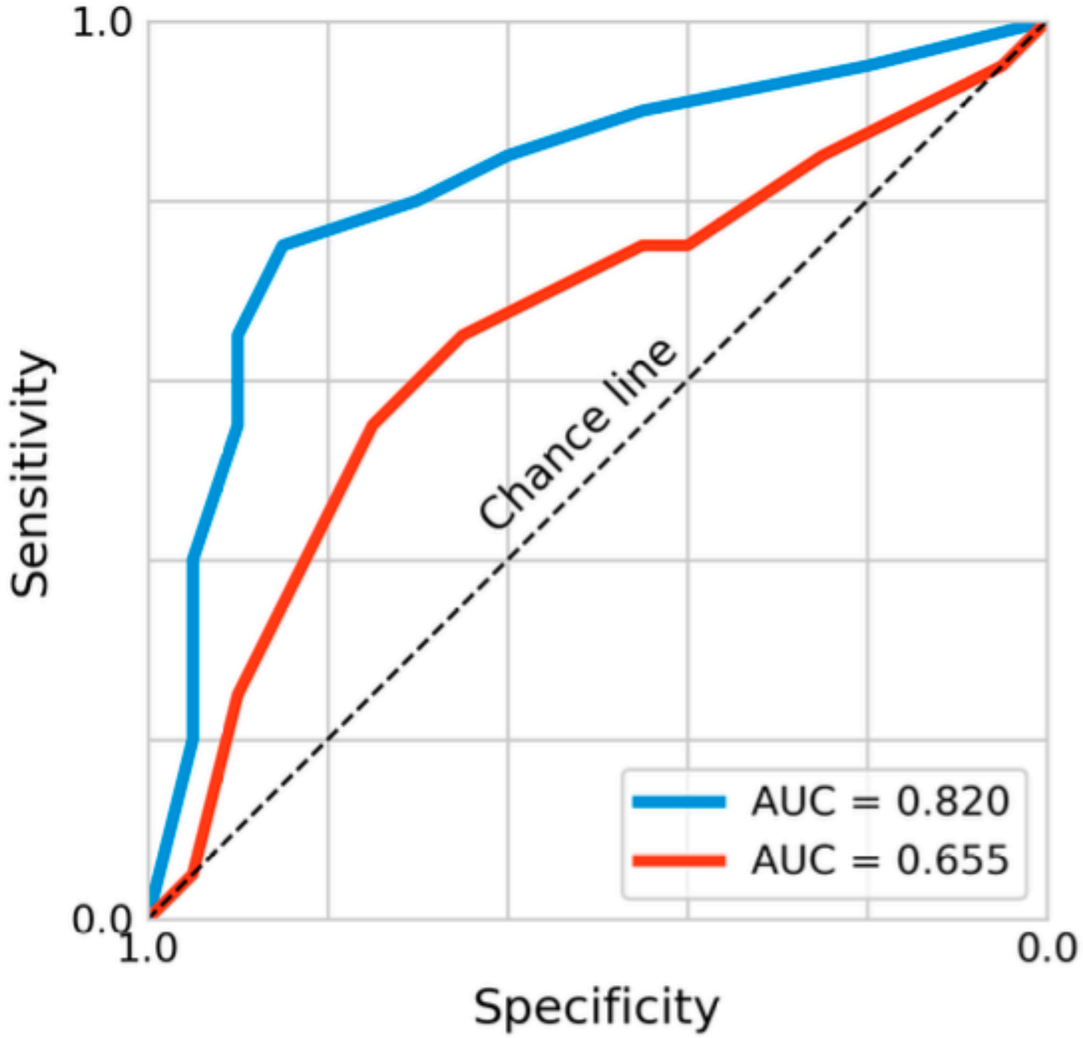
- If we start following a line at the bottom left, we start with highly specific tests that very rapidly also achieve high levels of sensitivity, without having to sacrifice a lot of specificity.

- However, eventually, we reach a point where each **additional small reduction in specificity does not gain us a lot of sensitivity in exchange**, and the tradeoff stops being valuable.

- These points typically sit “on the shoulder” of the curve towards the top left corner. This is our **ideal cutoff**.

The ROC AUC

- The Area Under the Curve (AUC) allows to compare two or more different tests.
- In the figure, two hypothetical tests are compared and the blue one performs better for any given cutoff.



Example: assistance micro targeting

PNAS RESEARCH ARTICLE | ECONOMIC SCIENCES COMPUTER SCIENCES OPEN ACCESS

Geographic microtargeting of social assistance with high-resolution poverty maps

Isabella S. Smythe[a] and Joshua E. Blumenstock[b,1]

Edited by Karen Seto, Yale University, New Haven, CT; received November 9, 2021; accepted June 7, 2022

Hundreds of millions of poor families receive some form of targeted social assistance. Many of these antipoverty programs involve some degree of geographic targeting, where aid is prioritized to the poorest regions of the country. However, policy makers in many low-resource settings lack the disaggregated poverty data required to make effective geographic targeting decisions. Using several independent datasets from Nigeria, this paper shows that high-resolution poverty maps, constructed by applying machine learning algorithms to satellite imagery and other nontraditional geospatial data, can improve the targeting of government cash transfers to poor families. Specifically, we find that geographic targeting relying on machine learning–based poverty maps can reduce errors of exclusion and inclusion relative to geographic targeting based on recent nationally representative survey data. This result holds for antipoverty programs that target both the poor and the extreme poor and for initiatives of varying sizes. We also find no evidence that machine learning–based maps increase targeting disparities by demographic groups, such as gender or religion. Based in part on these findings, the Government of Nigeria used this approach to geographically target emergency cash transfers in response to the COVID-19 pandemic.

Significance

Many antipoverty programs use geographic targeting to prioritize benefits to people living in specific locations. This paper shows that high-resolution poverty maps, constructed with machine learning algorithms from satellite imagery, can improve the geographic targeting of benefits to the poorest members of society. This approach was used by the Nigerian government to

Example: assistance micro targeting

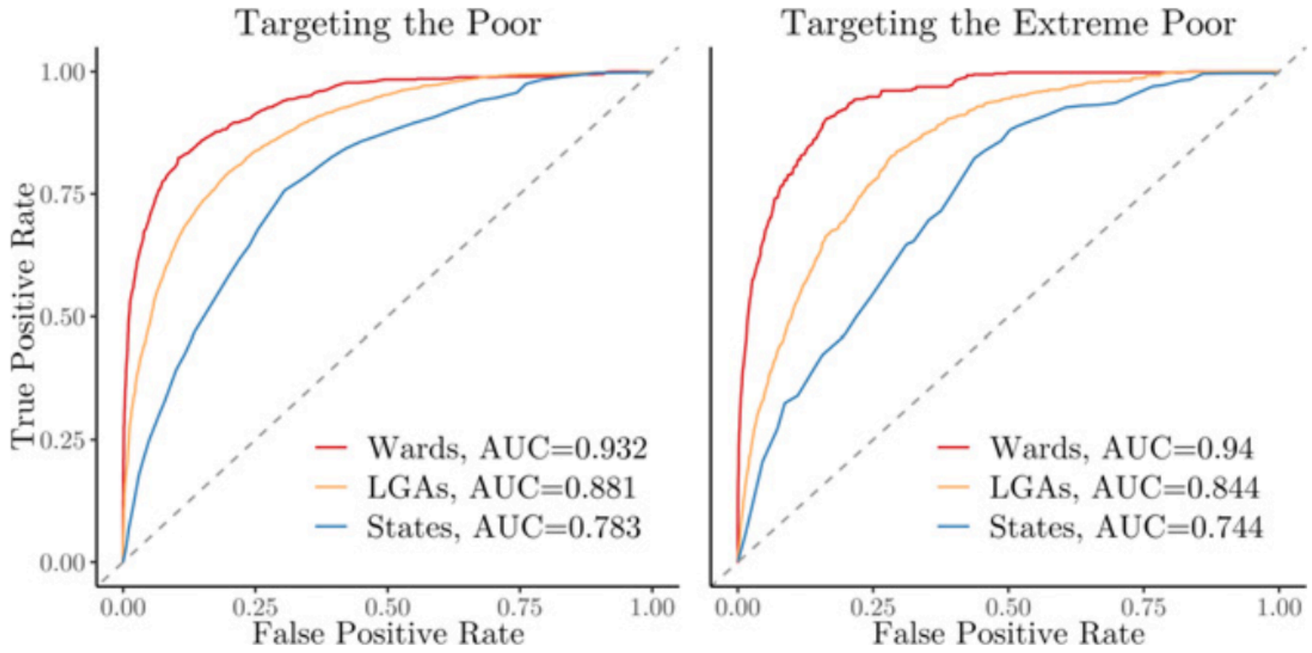


Fig. 1. Targeting performance of policies at different administrative units. ROC curves show the performance of geographic targeting policies designed at the state (Admin-1), LGA (Admin-2), and ward (Admin-3) levels, where all households in a targeted administrative unit receive full benefits and households in untargeted units receive no benefits. The targeting of an administrative unit is determined based on the average wealth of the unit as calculated from NLSS data. True and false positive rates are calculated based on the portion of true poor households that are targeted, where true poverty status is determined based on the NLSS.

Regression: beyond R^2

From a classical statistical standpoint, regression models are usually evaluated using R^2 or similar measures (e.g., Pseudo R^2) ⇒ How much variance in the data is explained by the model?

In ML, **other metrics are instead used** to assess the extent to which our algorithm gets closer to y.

Among these are:

- Mean Absolute Error
- Mean Squared Error
- Root Mean Squared Error

Regression: Mean Absolute Error

MAE is simply the average of the absolute differences between predicted and actual values ⇒ it quantifies the error without capturing the direction of the error.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} |\, y_i - \hat{y}_i \,|$$

MAE is robust to outliers.

Regression: MSE and RMSE

In MSE we take the square of the absolute errors:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

By taking the square of the error, we penalize larger errors more.

A similar measure is Root Mean Squared Error (RMSE) ⇒ how spread out are the errors?

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

Regression

In general, in ML study, all different metrics are used to evaluate a model and a comparison of performance based on all metrics is shown in a single table.

Model	RMSE	CV (RMSE)	MAE
Linear Regression	847.62	1.55	630.76
Ridge	877.35	1.60	655.70
k-Nearest Neighbor	1655.70	3.02	1239.35
Random Forest	539.08	0.98	370.09
Gradient Boosting	1021.55	1.86	746.24
Neural network	2741.91	5.01	2180.89
Extra Trees	466.88	0.85	322.04


Algorithms

Initial caveat

- In this section of the course we are going to cover some of the most popular ML algorithms.
- Fundamental: be aware that **these only represents a minority** of the entire universe of models that are being developed in the community.
- From a theoretical/foundational point of view, ML continuously produces new approaches/methods.

Back to basics: logistic regression

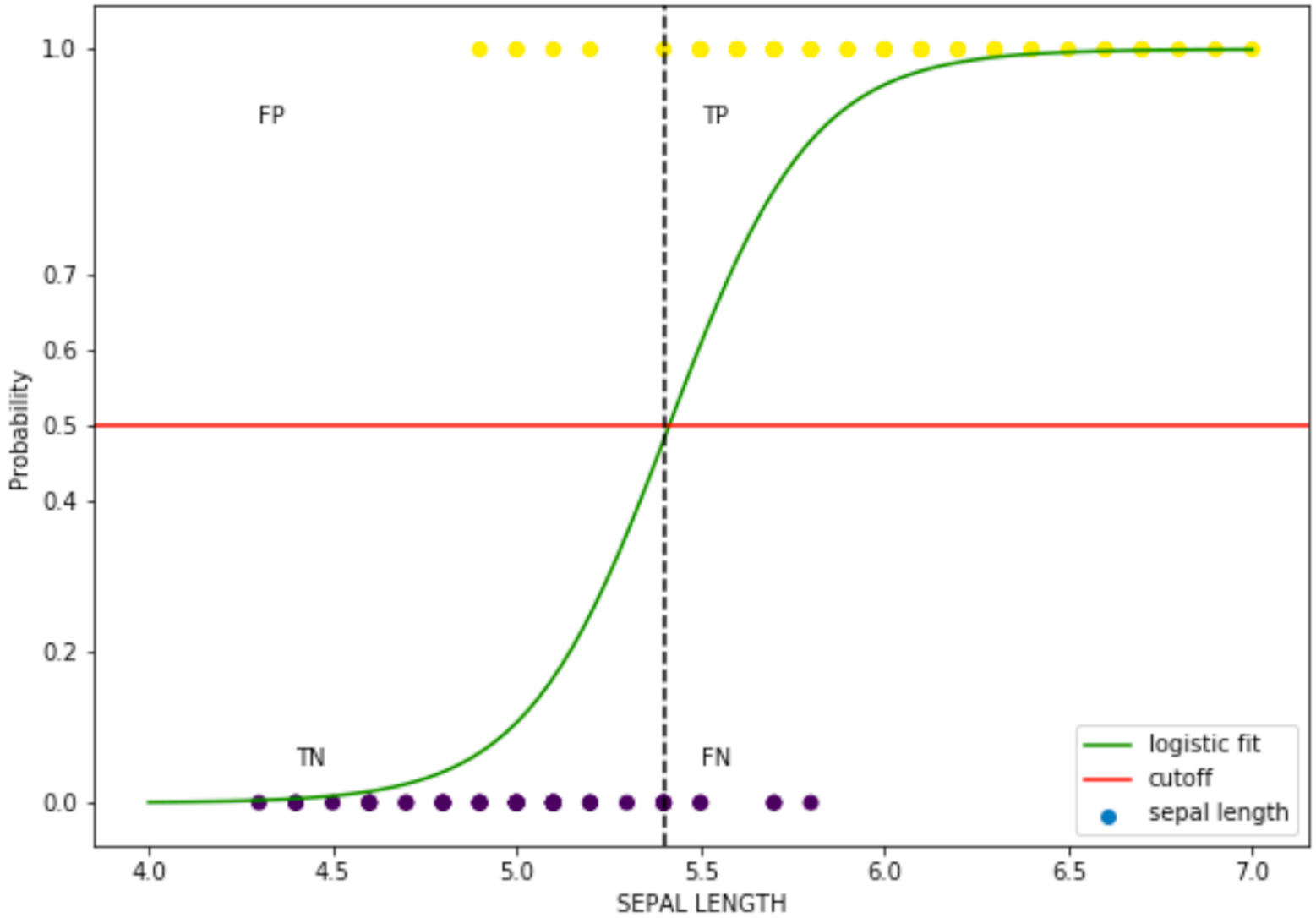
- Logistic regression is considered one of the standard methods for binary classification problems in ML.
- However it is not properly a classification method, but rather a regression method that is **used for classification when coupled with a decision rule**.
- ...What?

Logistic regression

- Logistic regression can be seen as a **probabilistic classification model.**
- Per each instance, the algorithms assign probabilities to each of the classes ⇒

  Decision Boundary: $p \geq 0.5, class = 1 \;\; p < 0.5, class = 0$
- The task becomes a classification one because we represent our target response in a binary state, while it was rather continuous.

Logistic regression



Example based on the Iris dataset:

en.wikipedia.org/wiki/Iris_flower_data_set

Regularization methods

OLS and Logistic regression may suffer from several problems, including **multicollinearity** (good old Stats) and **overfitting/generalizability**.

Regularization methods are useful to deal with such issues: they reduce model dimensionality, increase generalizability and also interpretability.

Among regularization methods the most popular are:

- Ridge regression
- LASSO regression

Back to basics: linear regression

Assume we have a simple linear regression model h(x) = $f(x)$

$$h(x) = \theta^T x = \theta_0 + \sum_{j=1}^{N} \theta_j x_j$$

$\Rightarrow$ p+2 parameters: p-vector of θ-coefficients, one intercept, one variance parameter (Gaussian error). The parameters are generally estimated using OLS:

$$\text{RSS}(\theta) = \sum_{i=1}^{N} \left(y_i - \theta_0 - \sum_{j=1}^{N} \theta_j x_{ij} \right)^2$$

The need for regularization

When the assumptions required by ordinary least squares (OLS) regression are met, the **coefficients produced by OLS are unbiased** and, of all unbiased linear techniques, have the lowest variance.

As the number of features grow, our OLS assumptions typically break down and our models often overfit (aka have high variance) to the training sample.

Regularization methods provide a means to control our regression coefficients, which can reduce the variance and decrease our out of sample error.

Bias-variance trade-off

The bias–variance decomposition forms the **conceptual basis for regression regularization methods**.

Regularization methods introduce bias into the regression solution that can reduce variance considerably relative to the ordinary least squares (OLS) solution.

Although the OLS solution provides non-biased regression estimates, the **lower variance solutions produced by regularization techniques** provide superior MSE performance.

Ridge regression

A method to achieve regularization is Ridge Regression (also called L2 Regression).

$$\mathscr{L}(\theta) = \sum_{i=1}^{N} \left(y_i - \theta_0 - \sum_{j=1}^{p} \theta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \theta_j^2$$

- It penalizes coefficients shrinking them towards 0.
- We introduce a penalty term λ that allows decreasing model complexity + decreasing variance we need to find the optimal value to maximize the bias/ variance trade-off.
- ISSUES: the model may have low estimate precision + difficult interpretability due to the high number of predictors (it does not exclude them!)

Lasso regression

Lasso Regression seeks to solve the limitations of Ridge Regression.

- It minimizes the sum of absolute values of θ
- For high values of λ, certain coefficients are zeroed and thus excluded from the model ⇒ better handling of multicollinearity and interpretability.

The equation of the Loss function in Lasso regression is:

$$\mathscr{L}(\theta) = \sum_{i=1}^{N} \left(y_i - \theta_0 - \sum_{j=1}^{p} \theta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\theta_j|$$

Ridge vs Lasso

There is not a clear and universal answer to the question “what model should I choose?”

If you can, **try to compare them**. In general:

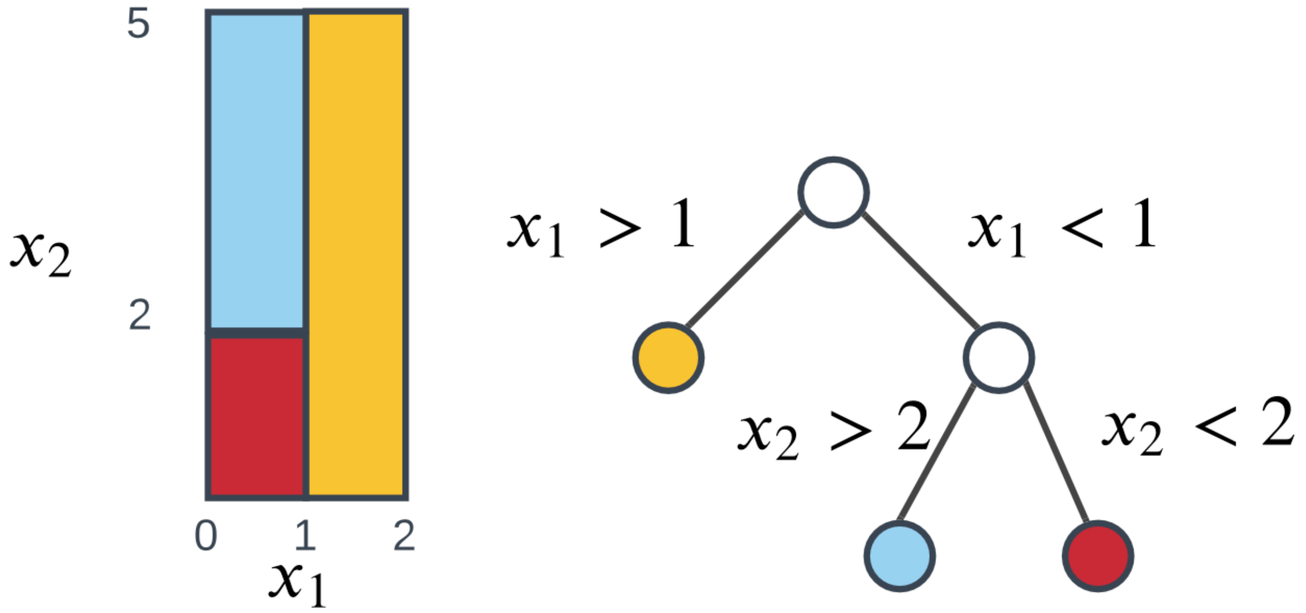
- Ridge regression likely performs better when many predictors p really impact the target/response/dependent variable.

- Lasso regression, conversely, generally outperforms Ridge Regression when we have a high number of p but only a small subset really impact Y.

Decision trees

Decision Trees are a non-parametric supervised learning method that can be used for both regression and classification.

They involve stratification/segmentation of the feature space into a number of finite regions.



Decision tree: how to split?

For classification and regression trees we rely on different criteria to split each node.

- Classification Tree: maximize information gain candidate on each possible split is computed via Entropy.
- Regression Tree: minimize objective function considering quality of partitions.

Decision trees: pros and cons

PROS:

- They are generally good when interpretability is necessary (White box).
- Little data preparation.
- Computational cost: acceptable.

CONS:

- Risk of overfitting/instability.
- Low performance compared to other methods.

Beyond a single decision tree

To improve the predictive performance of single Decision trees there are three main alternatives:

- Bagging
- Random Forests
- Gradient Boosting Machines (XGBoost in Python)

Unsupervised learning

Unsupervised Learning concerns the classes of problem in which there is not outcome/target measure, and the goal is therefore to **find and describe patterns and associations among inputs.**

Unsupervised learning: tasks

Unsupervised Learning cover many different tasks that are often common in data science applications, including:

- **Clustering**: Finding communities/subgroups of objects that are similar according to a certain criterion (generally intra-cluster similarity and inter-cluster dissimilarity). Different families of clustering approaches (e.g., distance-based, density-based, hierarchical...)
- **Anomaly Detection**: Detecting instances that significantly deviate from a learned distribution of previous instances
- **Dimensionality Reduction:** Representing X in a lower-dimension feature vector while preserving key properties of the data.
- **Density Estimation**: Constructing an estimate of the probability density function using a set of data points.

Unsupervised learning: algorithms

- **Clustering**: k-means, DBSCAN, Spectral Clustering, OPTICS, Ward Hierarchical…
- **Anomaly Detection**: One-class Support Vector Machine, Isolation Forest, Local Outlier Factor...
- **Dimensionality Reduction:** Singular Value Decomposition, Principal Component Analysis, Linear Discriminant Analysis, Non-negative Matrix Factorisation…
- **Density Estimation**: Gaussian Mixtures, Kernel Density Estimation...

Clustering

Distance and Similarity are two key concepts in constructing clustering algorithms.

- **Quantitative data** ⇒ *distance* (e.g., Minkowski, Euclidean, Cosine, etc.)

- **Qualitative data** ⇒ *similarity* (e.g., Jaccard, Hamming, etc.)

Evaluation

Internal Evaluation: we do not have a ground-truth (as in supervised learning), we can still rely on indicators/coefficients to measure how well a cluster represents data based on the core underlying aim of clustering ⇒ Maximize intra-cluster similarity, maximize inter-cluster dissimilarity.

External Evaluation: we have a ground truth (labels for each point), but we use it afterwards to evaluate how our groups resemble actual sub-spaces in our data.

Evaluation

Internal Evaluation: besides differences, same intuition: items in the same cluster must be more similar than those in different clusters. Some strategies:

- Davies-Bouldin Index
- Silhouette Score

External Evaluation: compare groupings based on pre-classified items (often done by experts in the application domain). Some strategies:

- Purity Index
- Rand Index

Clustering approaches

We can distinguish clustering algorithms in several categories:

- **Centroid-based:** algorithms organizing data in non-hierarchical clusters, based on closeness of points to the centroid of each cluster.

- **Density-based:** algorithms transforming subspaces with high density into clusters.

- **Hierarchical-based**: algorithms creating trees of clusters (well suited for taxonomies, structured data).

- **Distribution-based**: algorithms assuming data points are associated with certain distributions (e.g., Gaussian).

Examples

We are going to cover three algorithms belonging to as many clustering categories:

- *Centroid-based*: **K-Means**
- *Density-based*: **DBSCAN**
- *Hierarchical-based*: **Hierarchical Agglomerative Clustering**

k-Means

k-Means (Lloyd 1982) is one of the **most popular clustering algorithms**. Four main steps:

- Define/place K centroid into the space represented by the items we want to cluster.
- Assign each item to the cluster with the closest centroid.
- When all objects are assigned, recalculate the positions of the K centroids:

$$c_i = \frac{1}{S_i} \sum_{x \in S_i} x_i$$

- Repeat 2 and 3 until centroids do not move anymore.

k-Means

Steps 2/3 are based on the minimization of the underlying objective function:

being the Euclidean distance between data point

k-Means: pros and cons

PROS:

- Simple to implement.
- Computationally efficient.

CONS:

- Need to set k (suggestion: test for different values like $2 \leq k \leq 10$)
- Highly sensitive to the initial position of centroids. Suggestion: start positioning them very distant from each other as done in **k-means++** (Arthur and Vassilvitskii 2007)
- Difficult to handle outliers/noisy data points.

DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Ester et al. 1996) performs clustering based on the density of the points in the n-dimensional space.

Contrary to k-means, not all points are necessarily assigned to a given cluster

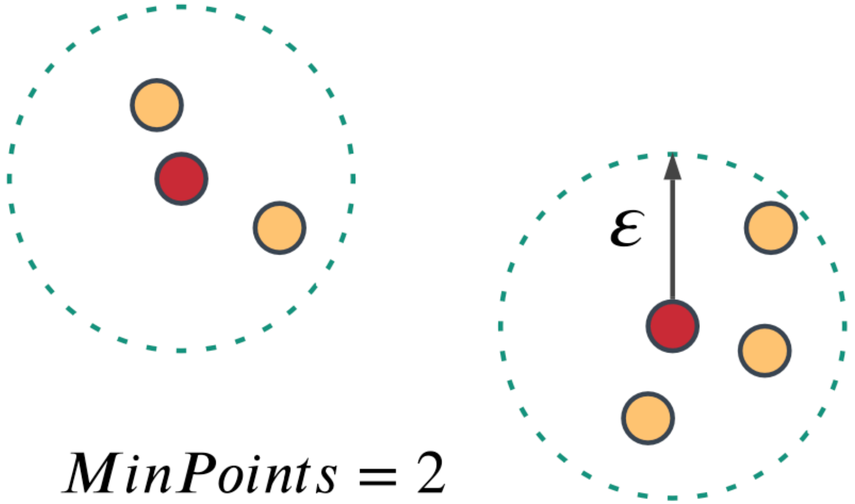
However, similarly to k-means, a point can be at most associated with one cluster

DBSCAN

The algorithm exploits two hyperparameters: $minPts$ and ϵ

- $minPts$: the minimum number of points necessary for a region to be considered dense (larger D, larger $minPts$)

- ε: distance measure that is used to locate points in the neighborhood of any other point



DBSCAN

Steps:

1. Start with an arbitrary (non-visited) starting point.
2. Find the neighborhood of this point using ε.
3. If there are at least $minPoints$ in the radius ε, then these belong to that cluster.
4. The clusters are then expanded through recursive repetition of the neighborhood search for each neighboring point.
5. Procedure stops when all points are visited → the points that after the last steps do not belong to any cluster are noise.

DBSCAN: pros and cons

PROS:

- It does not require to pre-specify number of clusters.
- Ability to handle clusters of arbitrary size and shape.
- Extremely good in separating regions with low density vs regions with high density.
- Robust to outliers.

CONS:

- Extremely weak performance when dealing with clusters that have similar density.
- Computationally expensive.
- Specification of $minPoints$ and ε requires experiments/a priori knowledge.

Hierarchical clustering

Hierarchical Clustering works well when dealing with data that have a hierarchical well-defined structure (e.g., taxonomies, documents → very popular in information retrieval)

It can be divided into two different approaches:

- **Agglomerative (or HAC)**: each observation is initially treated as a single cluster (bottom-up).

- **Divisive**: all observations are initially assigned to a single cluster (top-down).

Agglomerative clustering

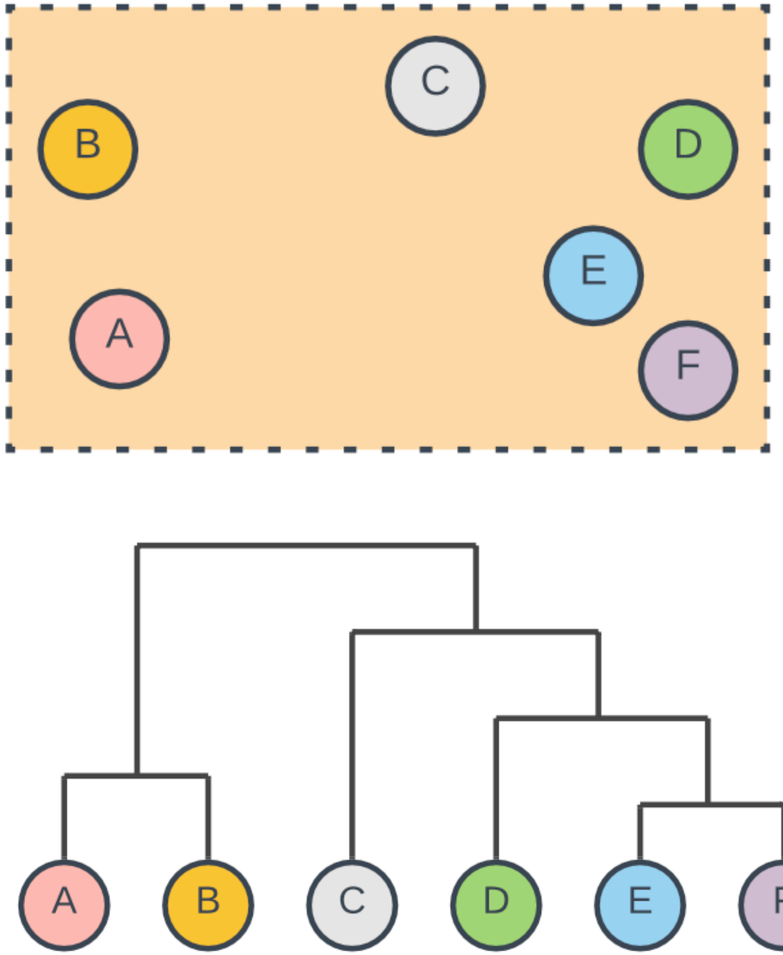
Steps:

1. Create N clusters, with $N = |X|$

2. Combine the two closest data points in a single cluster → forming N – 1 clusters

3. Take the two closest clusters and combine them in a single cluster → forming N – 2 clusters

4. Repeat 3) until left with one single cluster.



Agglomerative clustering

How we define distance/closeness between points/clusters? There exist several linkage methods. The most common are:

- **Complete-linkage**: distance between two clusters is the longest distance between two points in each cluster.

- **Single-linkage**: distance between two clusters is the shortest distance between two points in each cluster (robust against outliers).

- **Average-linkage**: distance between two clusters is the average distance between each point in one cluster to every point in the other cluster.

- **Centroid-linkage**: distance is the difference between centroid of cluster 1 and centroid of cluster 2, and so on.

Agglomerative clustering

How to decide when it is time to stop the merging the clusters? Different options:

- Cut at a pre-specified level of similarity
- Cut the dendrogram where the gap between two successive combination similarities is the largest
- Pre-specify k

HAC: pros and cons

PROS:

- It does not require to pre-specify number of clusters.
- Easy implementation.
- With moderate data, dendrogram is useful in interpreting the data.
- Robust to outliers.

CONS:

- With large datasets, difficult to assess the correct number of clusters.
- Computationally expensive.
- The algorithm doesn’t undo previous steps (errors stay in the loop).

A final example

Mapping socioeconomic status

frontiers | Frontiers in Big Data

TYPE Original Research
PUBLISHED 21 November 2022
DOI 10.3389/fdata.2022.1006352

OPEN ACCESS

EDITED BY
Patrick McSharry,
University of Oxford, United Kingdom

REVIEWED BY
Isabelle Tingzon,
Technical University of Munich, Germany
David Pastor-Escuredo,
LifeD Lab, Spain

*CORRESPONDENCE
Michele Tizzoni
michele.tizzoni@isi.it

SPECIALTY SECTION
This article was submitted to Data Analytics for Social Impact, a section of the journal Frontiers in Big Data

Mapping urban socioeconomic inequalities in developing countries through Facebook advertising data

Simone Piaggesi[1,2], Serena Giurgola[1], Márton Karsai[1,3,4], Yelena Mejova[1], André Panisson[1,5] and Michele Tizzoni[1,6]*

[1]Institute for Scientific Interchange Foundation, Turin, Italy, [2]Dipartimento di Informatica - Scienza e Ingegneria, Alma Mater Studiorum University of Bologna, Bologna, Italy, [3]Department of Network and Data Science, Central European University, Wien, Austria, [4]Alfréd Rényi Institute of Mathematics, Budapest, Hungary, [5]CENTAI Institute, Turin, Italy, [6]Department of Sociology and Social Research, University of Trento, Trento, Italy

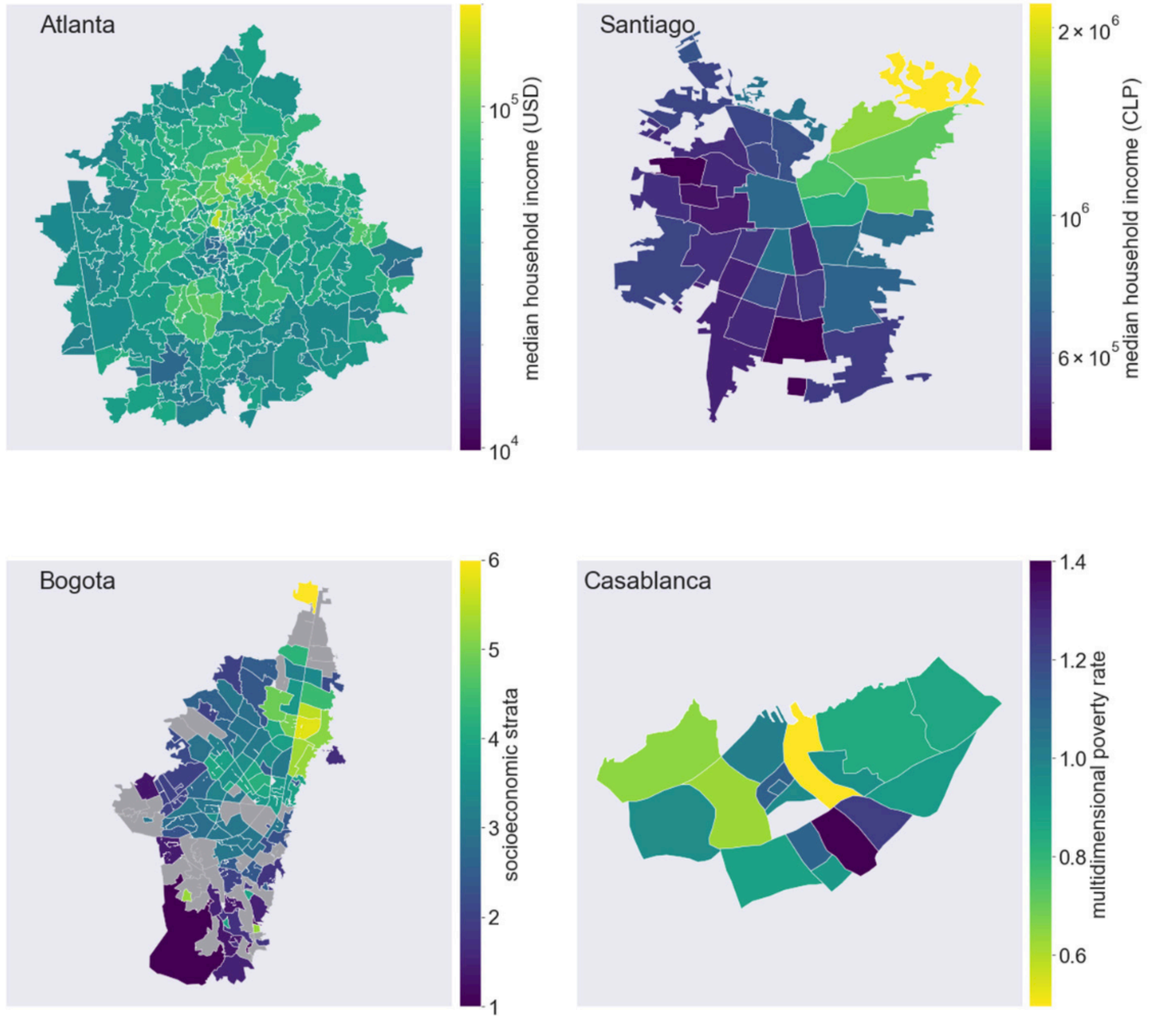
Mapping socioeconomic status

Task: predict socioeconomic status in 4 cities based on input features from Facebook advertising data.

Issue: socioeconomic status is given by different metrics (income, scores, etc.)

Solution: classification task, with different thresholds on SES variables.

ML method: boosted decision trees, k-fold cross validation, training and testing on different cities.



Artwork reference

- E. Hopper, *Lighthouse hill* (1927)
- U. Boccioni, *Quelli che vanno (Stati d'animo)* (1911)
- G. Balla, *Pessimismo e ottimismo* (1923)
- U. Boccioni, *Quelli che vanno (Stati d'animo, prima versione)* (1910)
- T. Signorini, *L'alzaia* (1864)