BTS | Barcelona Technology School

Classical Data Analysis

Master in Big Data Solutions 2019-2020

Matteo Manca

matteo.manca@bts.tech

Today's Objective

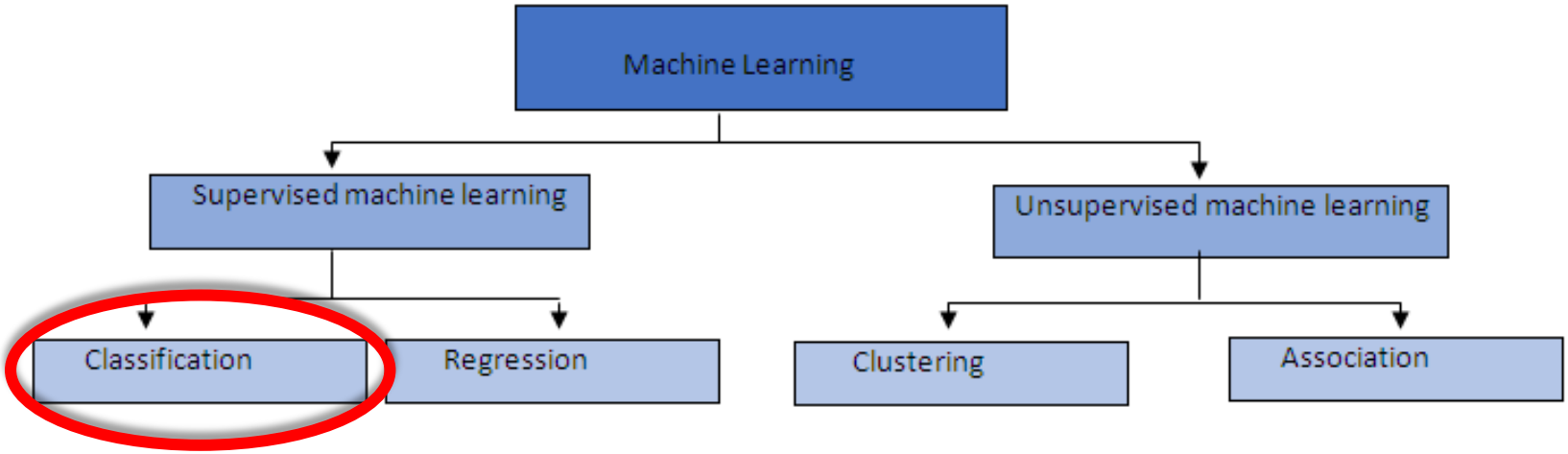
- Review concepts from previous sessions
- Regression analysis (Polynomial, Ridge, Lasso, Elastic Net)
  - Theory
  - Regularized linear regression with Python

Contents

- Introduction to data analysis
- Linear Regression
- Logistic Regression
- Regression analysis (Polynomial, Ridge, Lasso, etc.)
- Neaural Networks
- Support Vector Machines (SVM)
- Decision Trees
- Ensamble Methods
- Other Classifier (KNN, Naïve Bayes)
- K-Means
- PCA
- Hierarchical Clustering
- Recommender Systems

Previous Session: Logistic Regression

Supervised learning: Classification



The Logistic Regression algorithm solve **classification** problems. Don't get confused by the name "Regression"!

Logistic Regression

- The main goal of logistic regression is to perform binary classification but it can be extended for multiclass problems

- Examples of applications:
  - In medical research, in the field of predictive food microbiology, to describe bacterial growth/no growth interface, in the 0–1 format.
  - Predict the risk of developing a given disease based on observed characteristics of the patient.
  - In credit risk modeling in identifying potential loan defaulters.

All the above problems are **binary** classification problems.

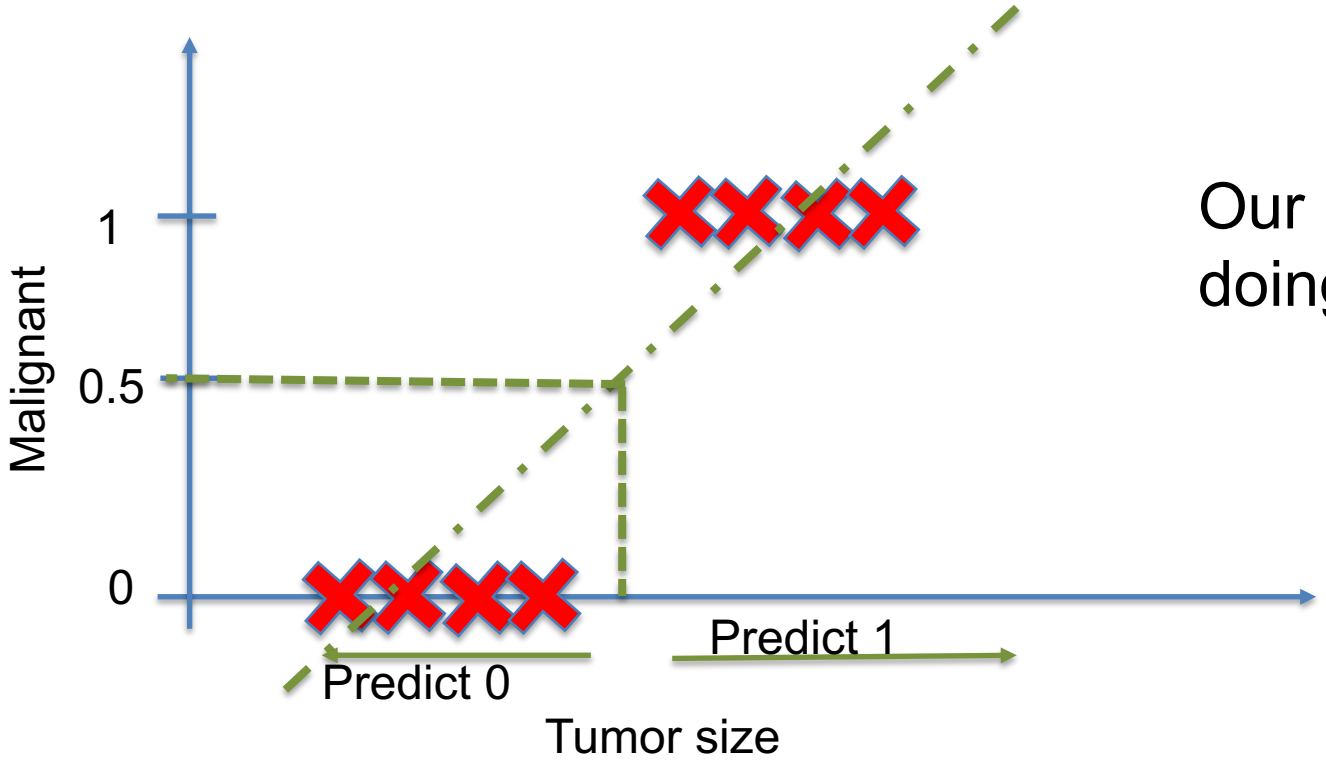
Possible results:

- y = 1
- Y = 0

- Feature Engineering plays an important role in regards to the performance of Logistic

Logistic Regression

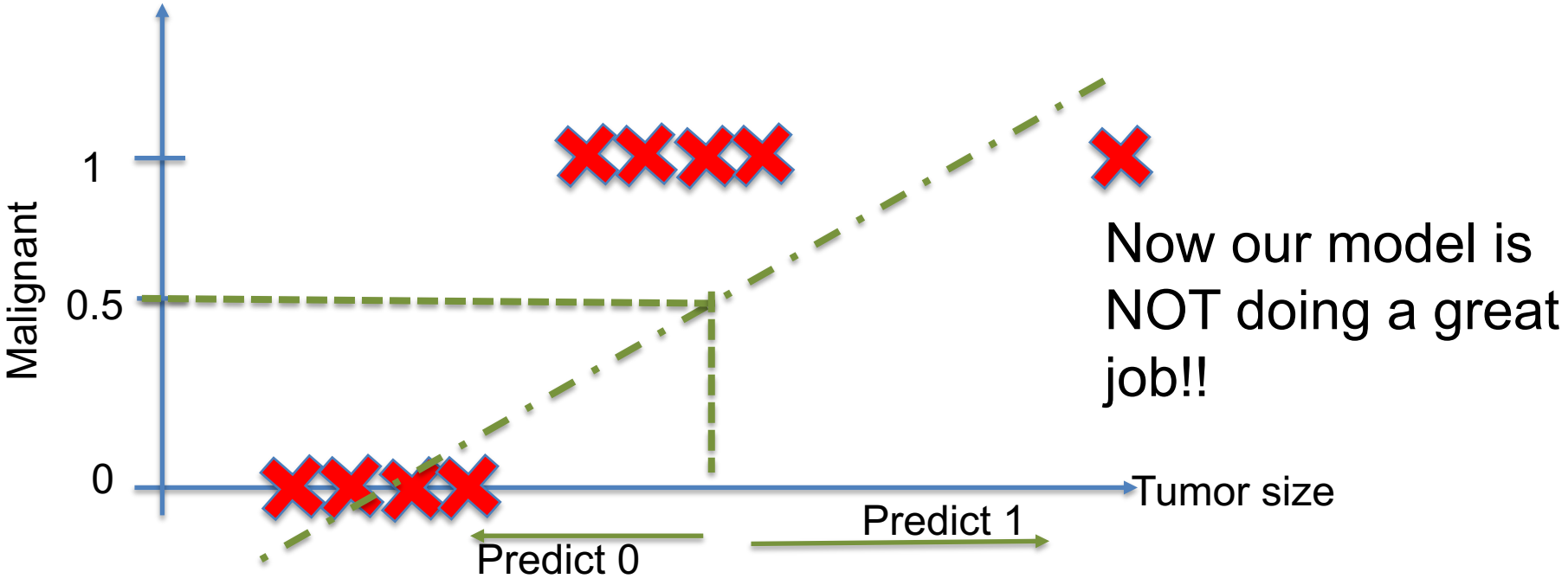
- How to develop a classification algorithm?
  - Let's try applying a linear regression model to classify a tumor as malignant or not!



Our model is doing a great job!!

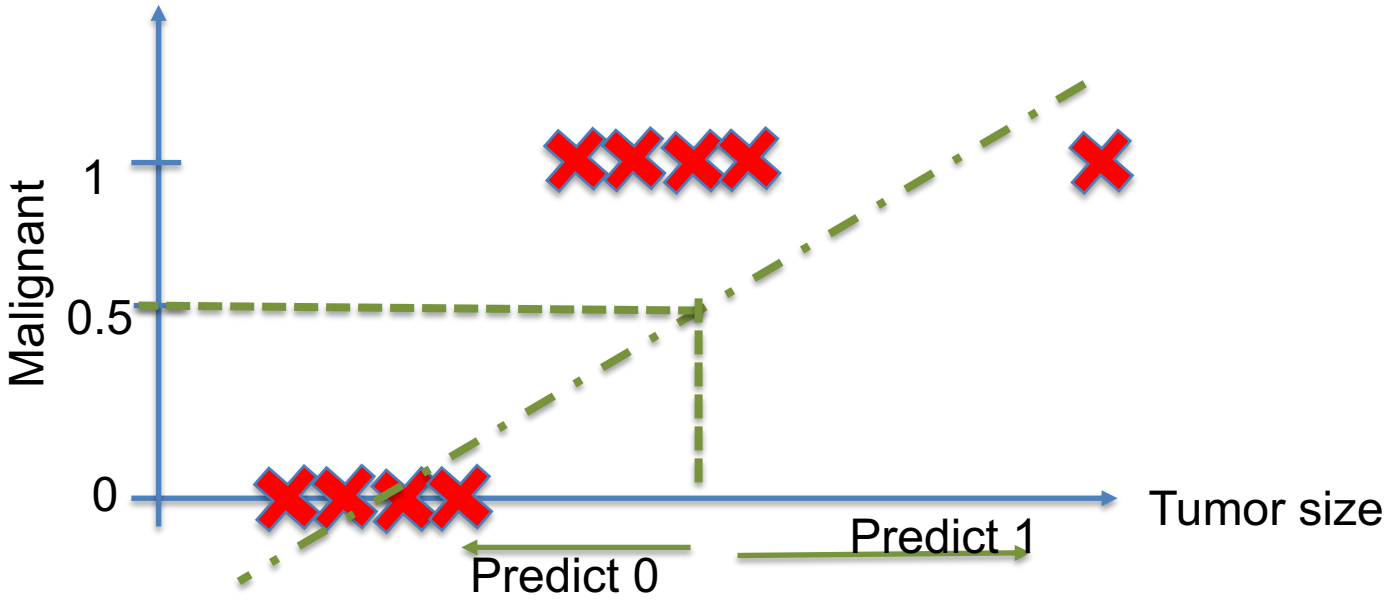
Logistic Regression

- How to develop a classification algorithm?
  - Let's try applying a linear regression model to classify a tumor as malignant or not!
  - Let'a add an additional example



Many positive examples are now classified as negative.

Logistic Regression



- Many positive examples are now classified as negative.
- It also doesn't make sense for $h_\theta(x)$ to take values larger than 1 or smaller than 0 when we know that $y \in \{0, 1\}$.

Logistic Regression

Linear regression hipothesis function:

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1 + \ ... + \theta_n x_n$$

In Logistic regression we want the output of the hypothesis function to be between 0 an 1. We apply the **sigmoid function (aka logistic function)** to the output of the linear regression, obtaining the hypothesis function for the logistic regression:

$$h_\theta(x) = g(\theta^T x) = g(\theta_0 + \theta_1 x_1 + \ ... + \theta_n x_n)$$

Logistic Regression

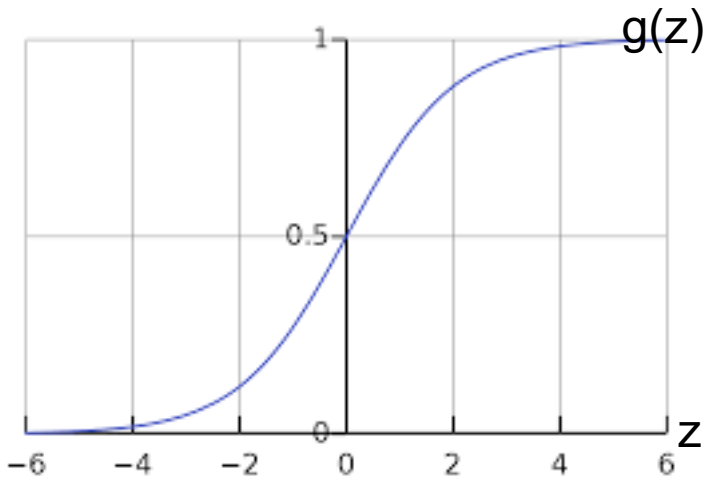
- Sigmoid function:

$$g(z) = \frac{1}{1 + e - z} \quad \Rightarrow \quad \frac{1}{1 + e^{-(\Theta^T X)}}$$

Logistic Regression

Logistic Regression hypothesis function:

$$g(z) = \frac{1}{1 + e - ^{z}}$$



When our hypothesis $h_\theta(x)$ outputs a number, we treat that value as the estimated probability that y=1 on input x

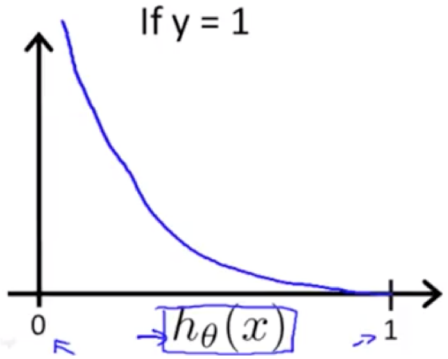
Notice that g(z) tends towards 1 as $z \to \infty$, and g(z) tends towards 0 as $z \to -\infty$. Moreover, g(z), and hence also h(x), is always bounded between 0 and 1.

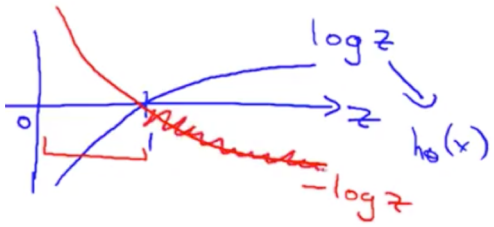
Logistic Regression

Cost function

- Given the logistic regression model, how do we fit the θ parameters?

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$





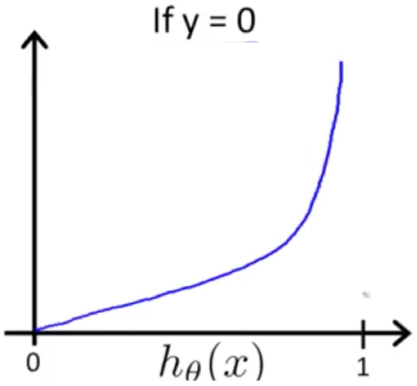
- if the hypothesis predicts that y = 1 and y = 1 cost = 0.
- if the hypothesis predicts that y = 0 and y = 1 cost -> inf.

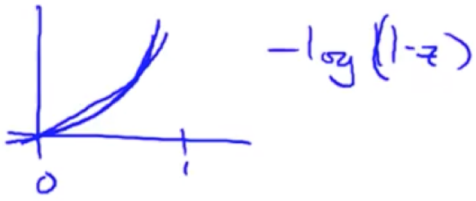
Logistic Regression

Cost function

- Given the logistic regression model, how do we fit the θ parameters?

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$





- if the hypothesis predicts that y = 0 and y = 0 cost = 0.
- if the hypothesis predicts that y = 1 and y = 0 cost -> inf.

Logistic Regression

Cost function

- Given the logistic regression model, how do we fit the θ parameters?

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

- This cost function can be written as follow

$$-\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

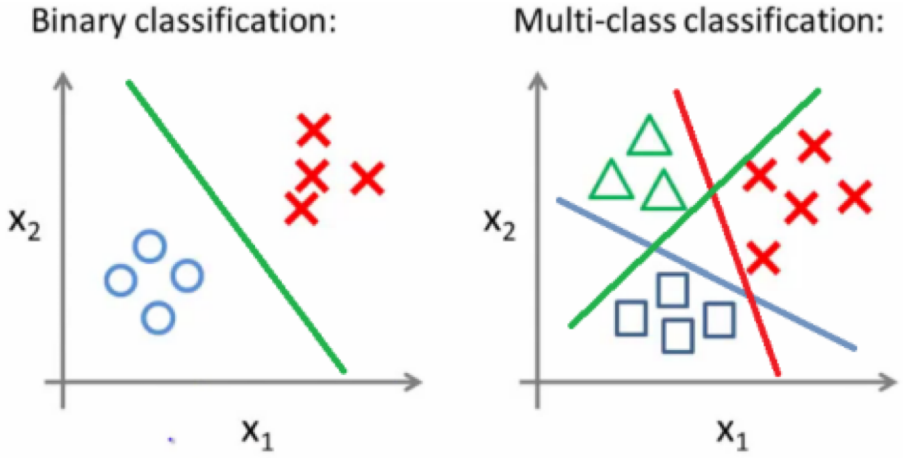
- This cost function can be derived from statistics using the principle of maximum likelihood estimation

- We can apply the **gradient descent** algorithm to find the parameter that minimize the cost function.

Logistic Regression

Multi-class Classification

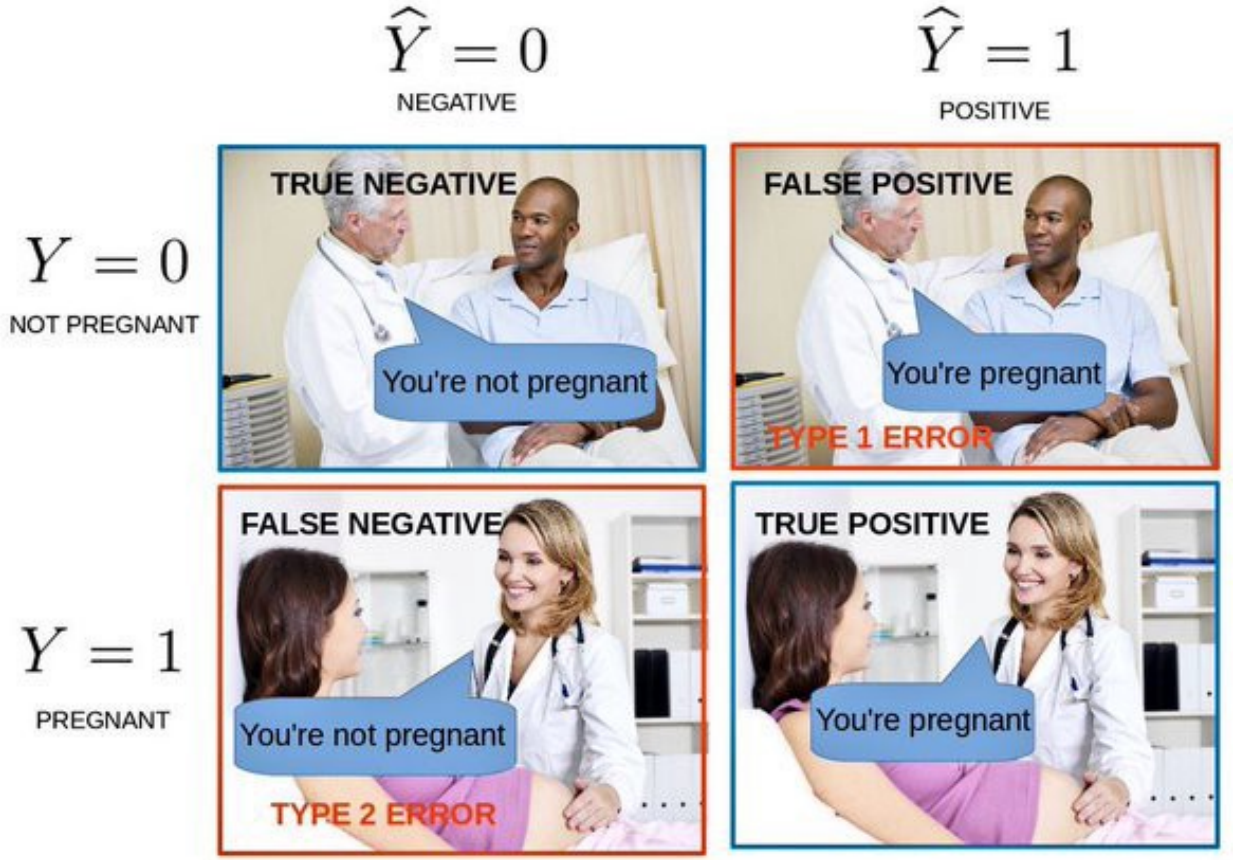
- **One vs. all classification**
  - classification problem with N distinct classes
  - Train N distinct binary classifiers, each designed for recognizing a particular class



Logistic Regression

Evaluation Metrics

- **Confusion Matrix**



Logistic Regression

Evaluation Metrics

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Accuracy:

- Overall, how often is it **correct**?
- (TP + TN) / total = 150/165 = 0.91

- It may not be reliable, especially in case of imbalanced dataset.

Logistic Regression

Evaluation Metrics

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Misclassification Rate (Error Rate):

- Overall, how often is it **wrong**?
- (FP + FN) / total = 15/165 = 0.09

Logistic Regression

Evaluation Metrics

- **Precision:** percentage of your results which are relevant. Among all the elements that your model classify as positive, how many are really positive examples?

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

- **Recall:** percentage of total relevant results correctly classified by your algorithm.

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

Logistic Regression

Evaluation Metrics

- **F1-Score:** It is the weighted average of the Precision and the recall scores.

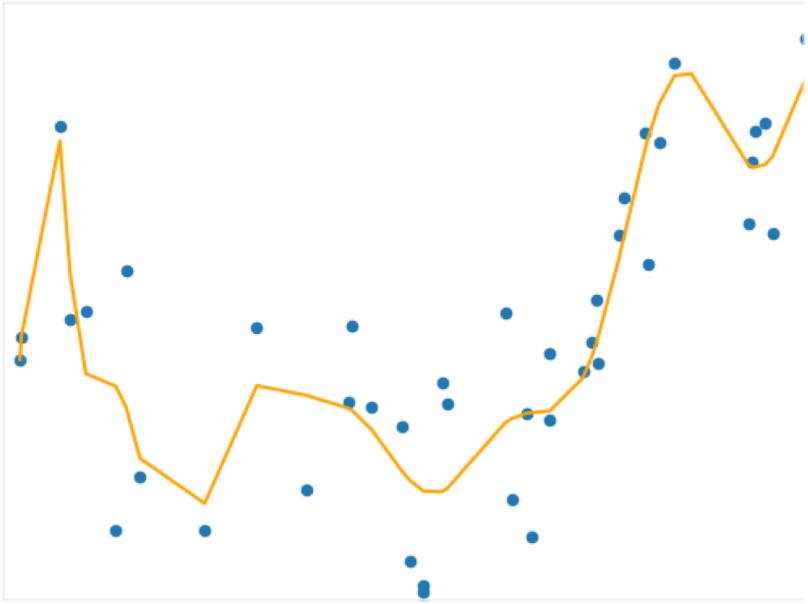
$$F1Score = 2(\frac{Precision \times Recal}{Precision + Recal})$$

Polynomial Regression

Polynomial regression

Limitation of linear Regression

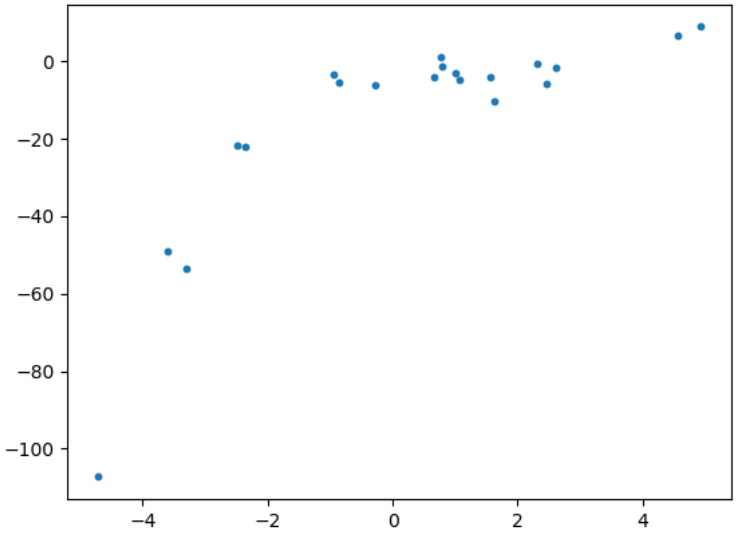
- **Linear regression** requires the **relation** between the dependent variable and the independent variable to be **linear**.



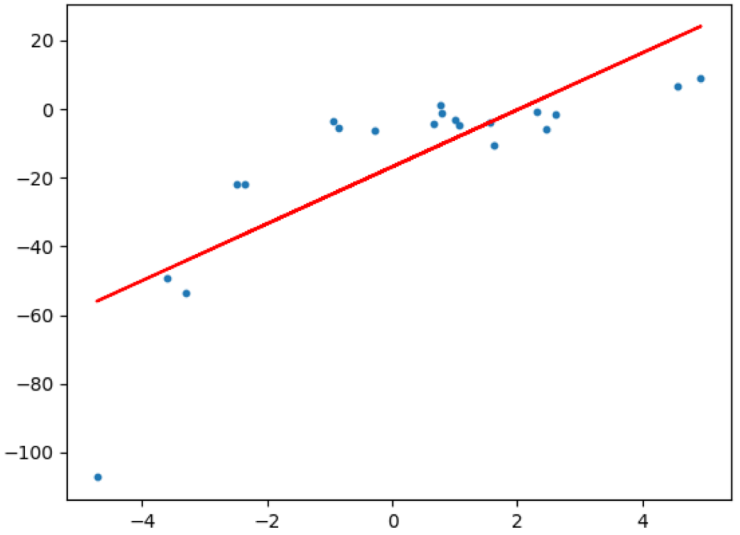
Can linear models be used to fit non-linear data?

Polynomial regression

Why Polynomial Regression?



Linear regression model

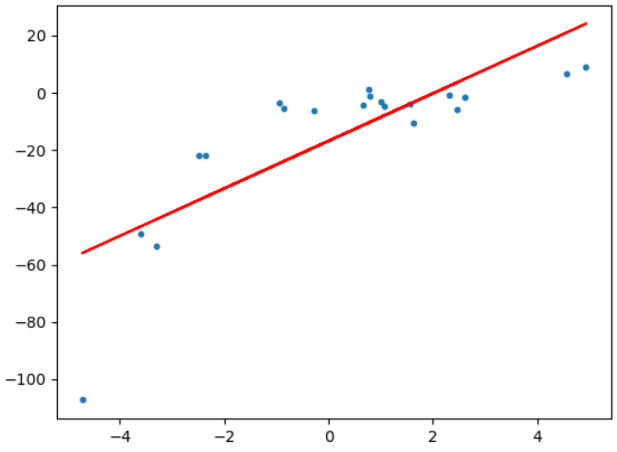


The model is unable to capture the patterns in the data. This is an example of **under-fitting**.

Polynomial regression

Why Polynomial Regression?

Linear regression model



How to overcome **under-fitting**?

Increase the complexity of the model.
We need to add more features to the data

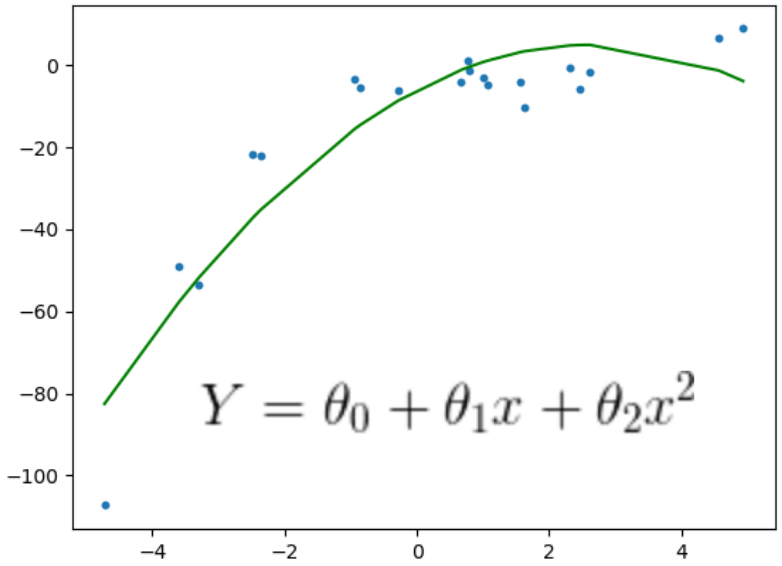
$$Y = \theta_0 + \theta_1 x \quad \Rightarrow \quad Y = \theta_0 + \theta_1 x + \theta_2 x^2$$

This is still considered to be a **linear model** as the coefficients/weights associated with the features are still linear. x^2 is only a new feature.

Polynomial regression

Why Polynomial Regression?

Linear regression model 2nd degree polynomial

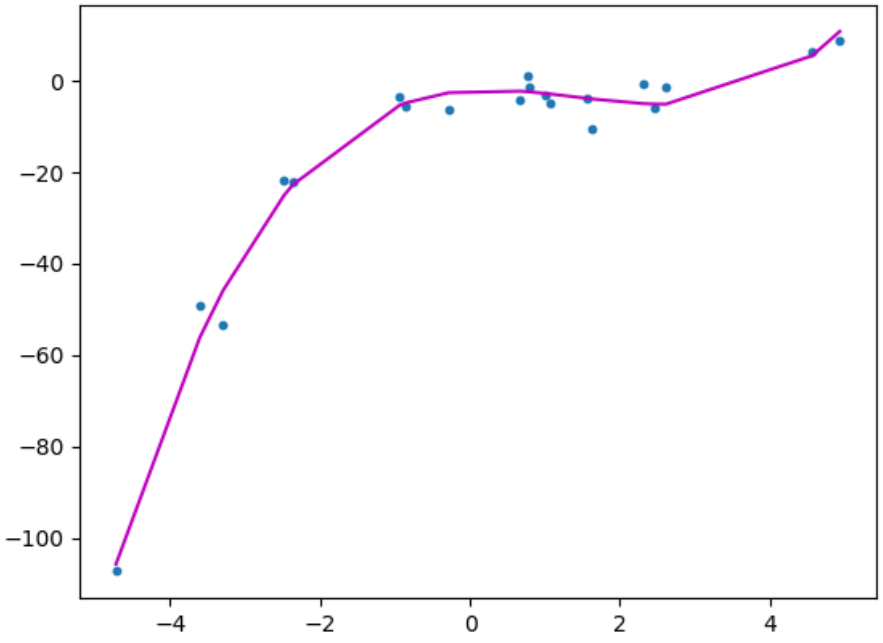


It is clear from the plot that the quadratic curve (2nd degree polynomial) is able to fit the data better than the linear line.

Polynomial regression

Why Polynomial Regression?

Linear regression model polynomial of degree 3

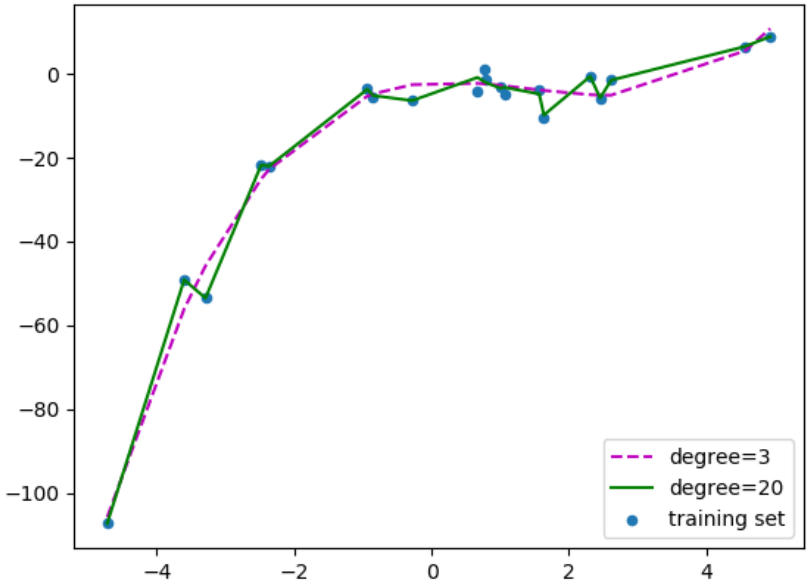


It passes through more data points than the quadratic and the linear plots.

Polynomial regression

Why Polynomial Regression?

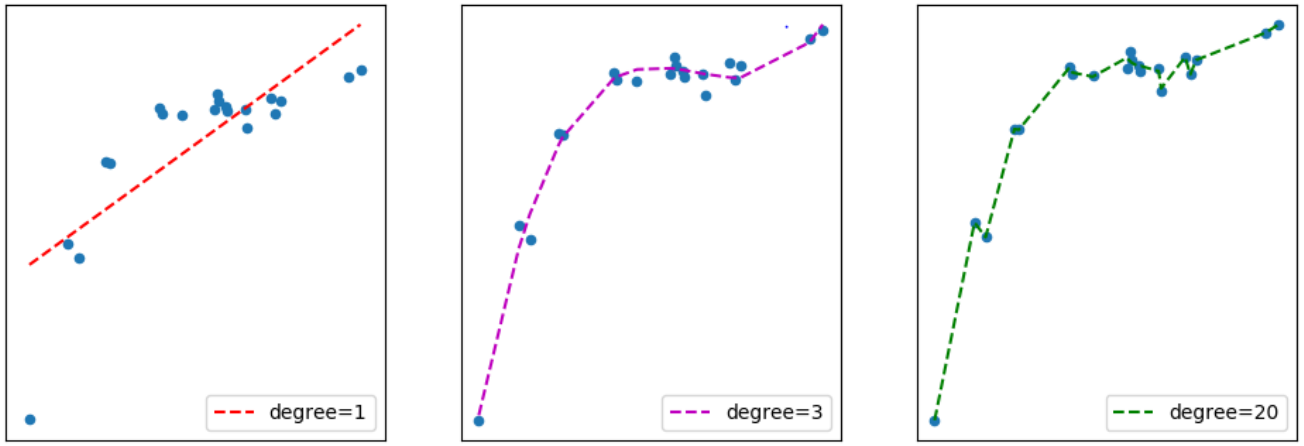
Linear regression model polynomial of degree 20



It passes through more data points but here we have a problem: **Overfitting**, i.e., the model is also capturing the noise in the data and even if it works well on the training set it will fail to generalize (the performance in the test set will be poor).

Bias vs Variance trade-off

- **Bias**: error due to the model's simplistic assumptions in fitting the data. A **high bias** means that the model is unable to capture the patterns in the data and this results in **underfitting**.
- **Variance:** error due to the complex model trying to fit the data. **High variance** results in **over-fitting** the data. Variance refers to how much the model is dependent on the training data.

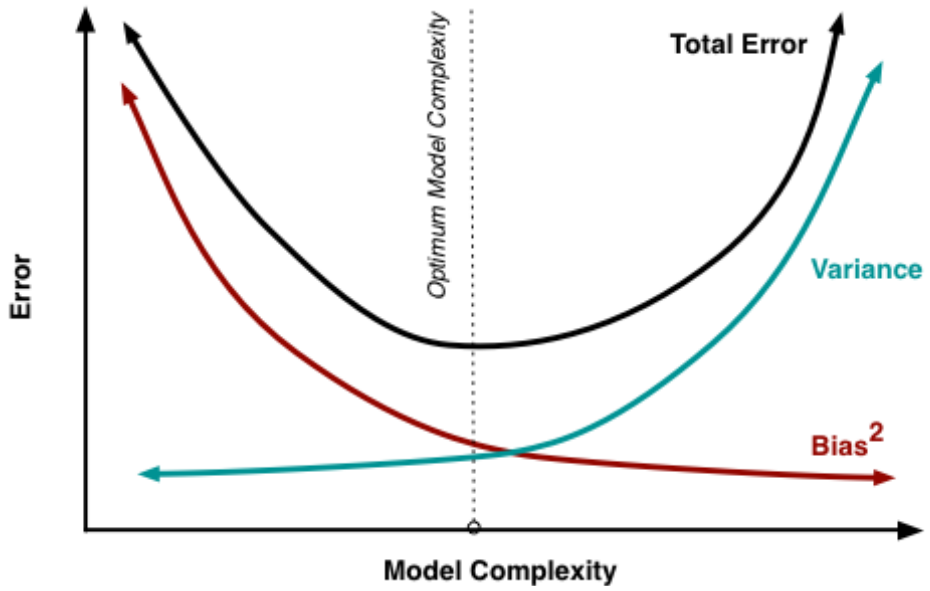


Underfit
High Bias
Low Variance

Correct Fit
Low Bias
Low Variance

Overfit
Low Bias
High Variance

Bias vs Variance trade-off



Regularizatioin: Ridge, Lasse and Elastic Net

Regularization

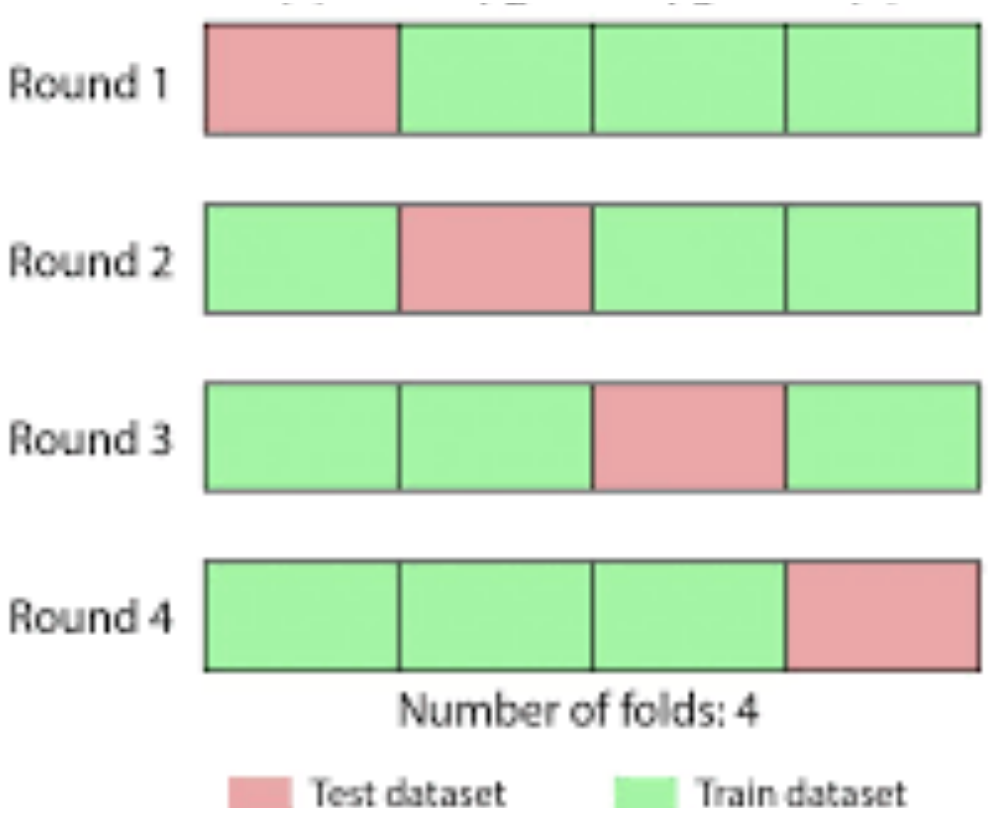
How to address overfitting

- In classical linear regression if the model feels like one particular feature is particularly important, the model may place a large weight to the feature. This might lead to **overfitting** in small datasets.

- Reduce Number of feature (Manually or Model selection algorithm)
- **Regularization**
- Collect More data
- **Cross Validation**
- Ensambling (Combine prediction from multiple separate models)

Regularization

Cross Validation



Regularization

How to address overfitting

- In classical linear regression if the model feels like one particular feature is particularly important, the model may place a large weight to the feature. This might lead to **overfitting** in small datasets.

- **Regularization techniques consist of modifying the cost function, adding a penalty term, to restrict the values of our coefficients.**

- Regularization techniques:
  - **Lasso**, also called L1-Norm
  - **Ridge**, also called L2-Norm
  - **Elastic Net**: is a combination of Lasso and Ridge

Regularization

LASSO

- Adds an additional term to the cost function, adding the sum of the coefficient values (the L-1 norm) multiplied by a constant lambda.

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

- Lambda set the coefficients of the bad predictors mentioned above 0 (**feature selection**).
- If lambda=0, we effectively have no regularization.
- Large lambda leads coefficients to 0.

Regularization

RIDGE

- Sums the **squares of coefficient** values (the L-2 norm) and multiplies it by some constant lambda.

$$\hat{\beta}^{ridge} = \underset{\beta \in \mathbb{R}}{argmin} \|y - XB\|_2^2 + \lambda \|B\|_2^2$$

- will decrease the values of coefficients, but is unable to force a coefficient to exactly 0 (**No feature selection**).
- If lambda=0, we effectively have no regularization
- It will also select groups of colinear features (grouping effect)

Analysis of both Lasso and Ridge regression has shown that neither technique is consistently better than the other; try both methods to determine which to use!

Regularization

Elastic Net

- Includes both L-1 and L-2 norm regularization terms.

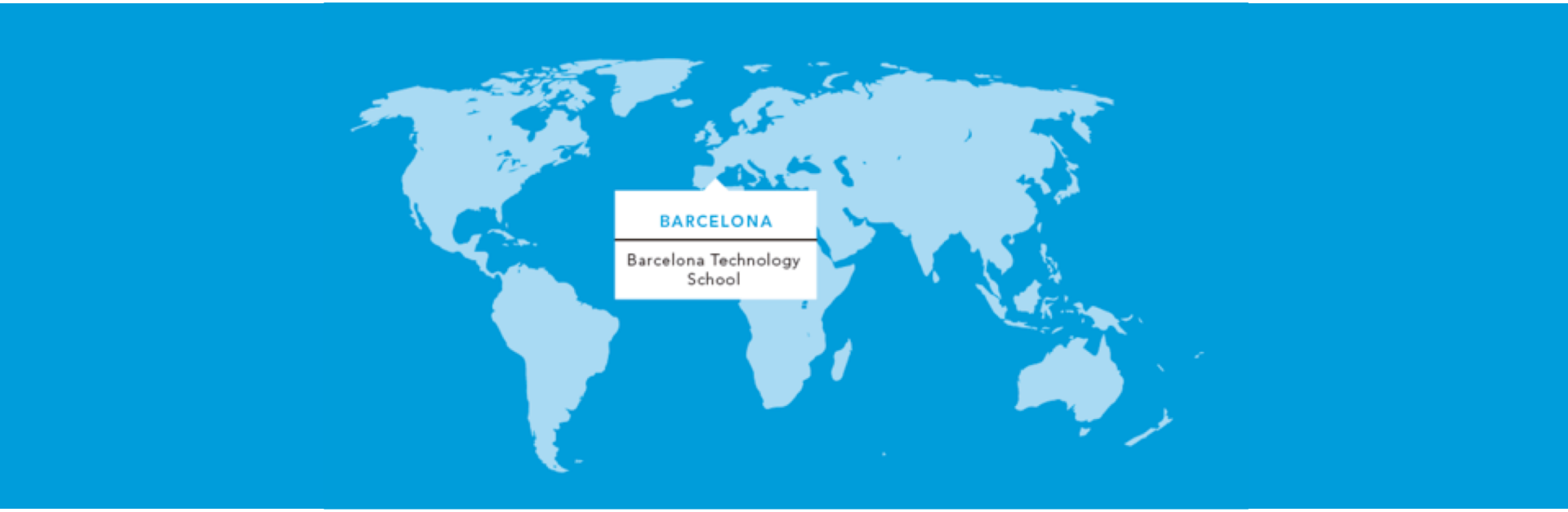
$$\hat{\beta} \equiv \underset{\beta}{\operatorname{argmin}}(\|y - X\beta\|^2 + \lambda_2 \|\beta\|^2 + \lambda_1 \|\beta\|_1).$$

- It seems to have the predictive power better than Lasso, while still performing feature selection. We therefore get the best of both worlds, performing feature selection of Lasso with the feature-group selection of Ridge.

Resources

- Tan, Pang-Ning. *Introduction to data mining*. Pearson Education India, 2006.

- Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. New York: Springer series in statistics, 2001.


BARCELONA
Barcelona Technology School