Data Management

Supervised learning

Malka Guillot

HEC Liège | ECON2306

Table of Contents

Prologue

Reference:

- JWHT, chap 3, 6.2
- Geron, chapter 2

Linear Regression as a Predictive Model

Linear Regression

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

= one of the simplest algorithms for doing supervised learning

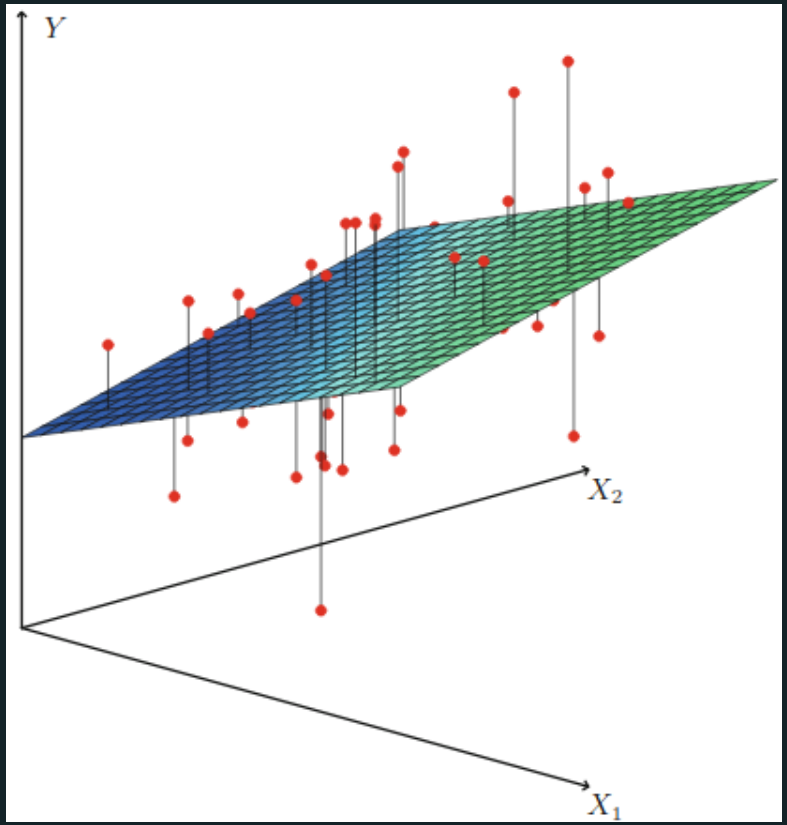
A good starting point before studying more complex learning methods

Estimation by Ordinary Least Squares

$RSS = \text{Residual sum of squares} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Minimizing RSS gives a **closed form solution** for the $\hat{\beta}_1, \cdots \hat{\beta}_p$

Most ML models **do not** have a a colsed form solution


Y
X_2
X_1

Extensions of the Linear Model

Going further model's assumptions:

- the **additive**: the effect of changes in a predictor X_j on the response Y is independent of the values of the other predictors
- **linearity**: the change in the response Y due to a one-unit change in X_j is constant

Interactions

- Adding interacted variable can help
- Should respect the **hierarchy principle**:
  - if an interaction is included, the model should always include the main effects as well

Non Linearity

- Include transformed versions of the predictors in the model

$\Rightarrow$ Including polynomials in X may provide a better fit

Linear Models: pros and cons

- Pros:
  - Interpretability
  - Good predictive performance
  - Accuracy measures for
    - coefficient estimates (standard errors and confidence intervals)
    - the model
- Cons:
  - When $p > n$
  - Tend to over-fit training data.
  - Cannot handle multicollinearity.

Generalization of the Linear Models

- **Classification problems**: logistic regression, support vector machines
- **Non-linearity**: nearest neighbor methods
- **Interactions**: Tree-based methods, random forests and boosting
- **Regularized fitting**: ridge regression and lasso

Regularized Regressions

Why Regularization?

- Solution against **over-fitting**
- Allow High-Dimensional Predictors
  - $p >> n$: OLS no longer has a unique solution
  - x_i "high-dimensional" i.e. very many regressors
    - pixels on a picture

Adding a Regularization Term to the Loss Function $L(.)$

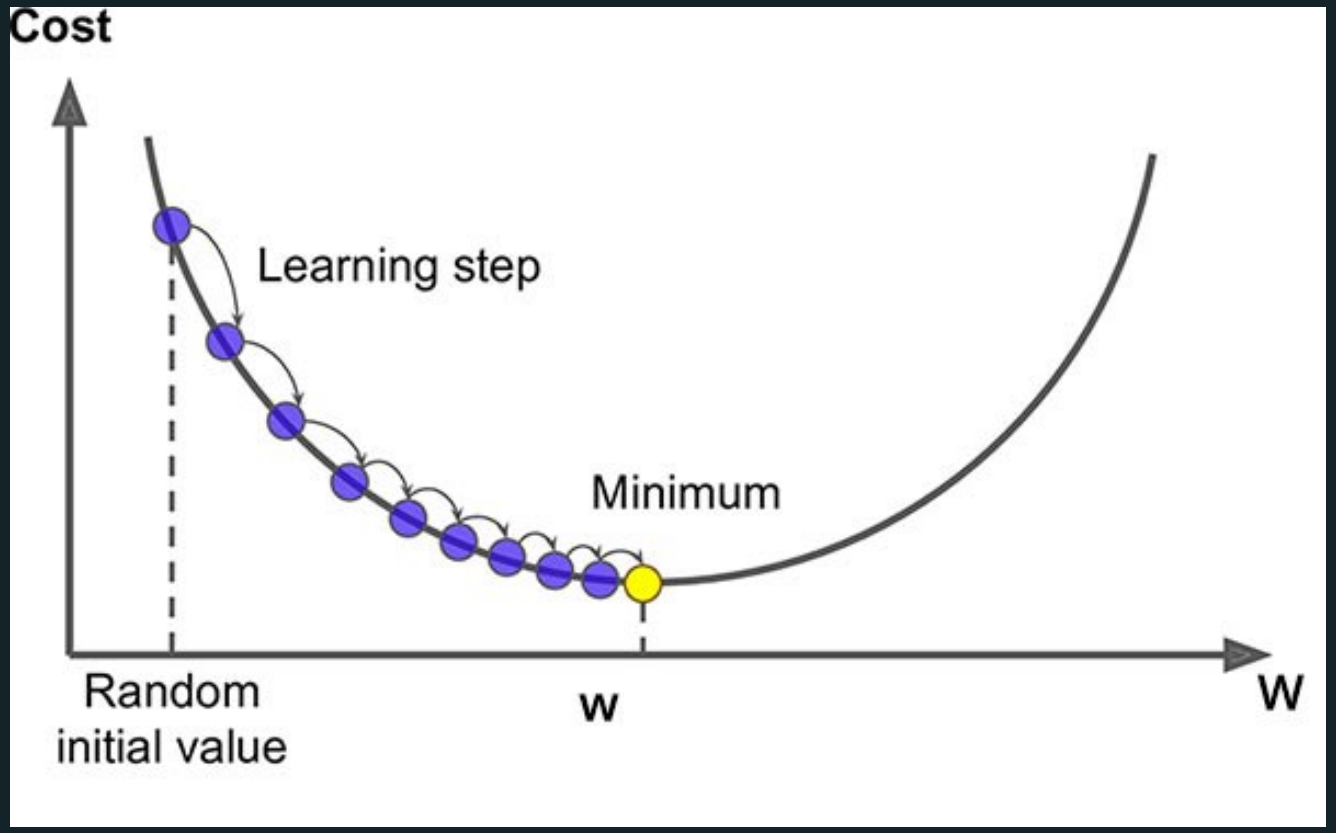
$$\hat{\beta} = argmin_{\beta} \frac{1}{n} \sum_{i=1}^{n} L(h(x_i, \beta), y_i) + \lambda R(\beta)$$

- $R(\beta)$ = **regularization function**
  - $R(\beta) = \sum_{i=1}^{n} p(\beta_i)$ for $p(.)$ the penalty function
- λ is a **hyperparameter** where higher values increase regularization

Different Penalty Functions $p()$

- **Ridge (L2)**: $p(\beta_j) = \beta_j^2$
- **LASSO (L1)**: $p(\beta_j) = |\beta_j|$
- **Elastic Net**: $p(\beta_j) = \alpha|\beta_j| + (1-\alpha)\beta_j^2$
- **Subset selection**: $p(\beta_j) = 1\{\beta_j \neq 0\}$

How to Solve Without a Closed-form Solution? Gradient Descent



Gradient descent measures the local gradient of the error function, and then steps in that direction.

→ Minimum in 0

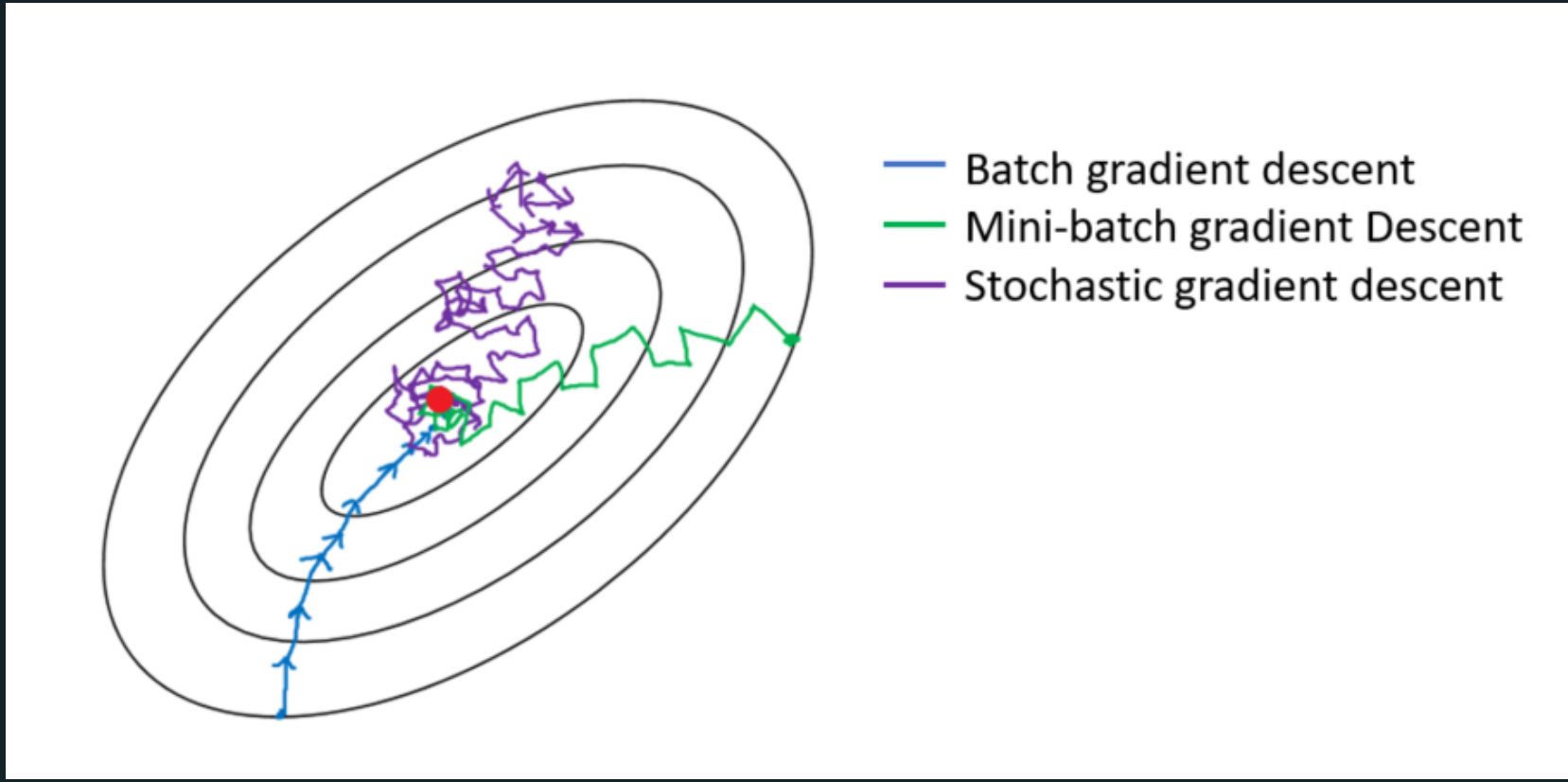
Stochastic Gradient Descent

1. Picks a **random instance** in the training set
2. Computes the gradient only for that single instance

- **Pro**: SGD is much faster to train,
- **Cons**: bounces around even after it is close to the minimum.

→ Compromise: **mini-batch gradient descent**, selects a sample of rows (a "mini-batch") for gradient compute

Varients of Gradient Descent



Ridge Regression

$min_\beta \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$

Where

- $\lambda > 0$ = penalty parameter
- covariates can be high-dimensionnal $p >> N$

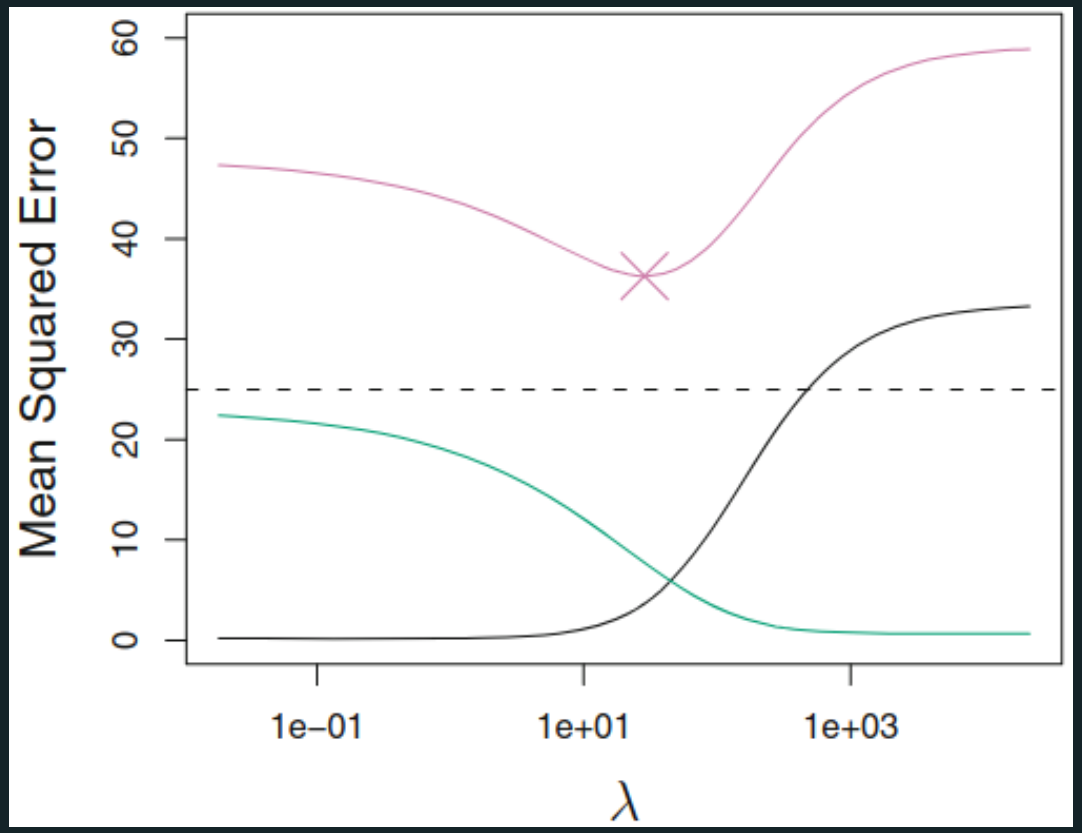
Trade-off, from the minimization of the sum of

1. RSS
2. shrinkage penalty: decreases with β_j

$\rightarrow$ relative importance given by λ

Ridge Regression: shrinkage to 0

Ridge: Variance-Bias Trade-Off



Squared bias (black), variance (green), [test] MSE (red)

Ridge vs. Linear Models

- when outcome and predictors are close to having a linear relationship, the OLS will have low bias but potentially high variance
  - small change in the training data → large change in the estimates
  - worse with p close tp n
  - if $p > n$, OLS do not have a unique solution

→ ridge regression works best in situations where the least squares estimates have high variance

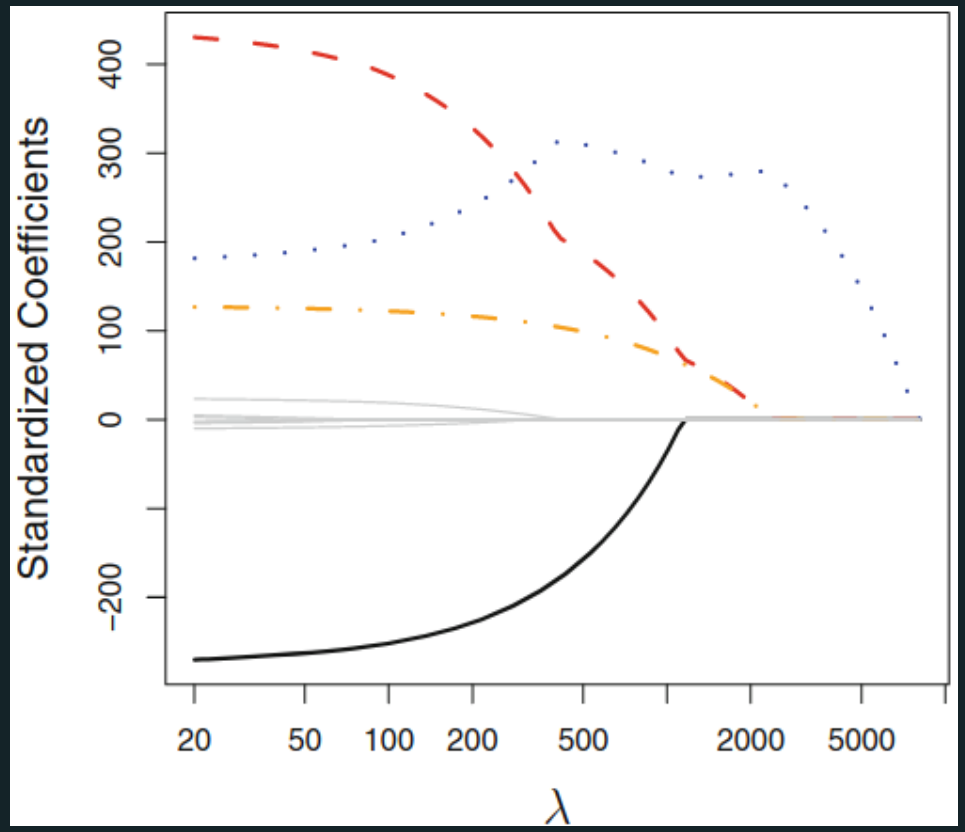
LASSO

Overcome an important drawback of Ridge (all p predictors are included in the final model)

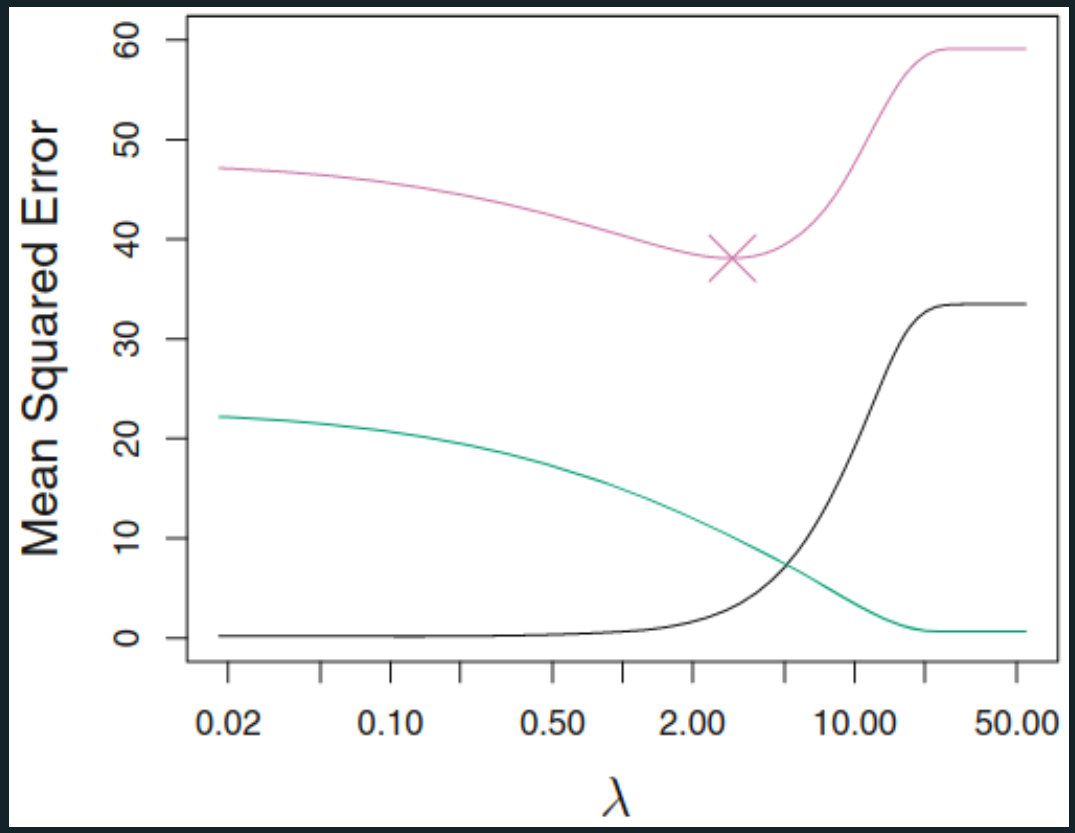
LASSO proposes a method to build a model which just **includes the most important predictors**.

Better for interpretability than Ridge!

Lasso Coefficients



Lasso: Variance-Bias Trade-Off



Squared bias (black), variance (green), [test] MSE (red)

Constrained Regression

The minimization problem can be written as follow:

$$\sum_{i=1}^{n}(y_i - x_i'\beta)^2 \text{ s.t. } \sum_{j=1}^{p} p(\beta_j) \leq s,$$

Where

- Ridge: $\sum_{j=1}^{p} \beta_j^2 < s \rightarrow$ equation of a **circle**
- Lasso: $\sum_{j=1}^{p} |\beta_j| < s \rightarrow$ equation of a **diamond**

Constraint Regions

Lasso **Ridge**

Elastic Net = Lasso + Ridge

$$MSE(\beta) + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2$$

$\lambda_1, \lambda_2 =$ strength of L1 (Lasso) penalty and L2 (Ridge) penalty

Selecting Elastic Net Hyperparameters

- Elastic net **hyperparameters** should be selected to optimize out-of-sample fit (measured by mean squared error or MSE).
- **"Grid search"**
  - scans over the hyperparameter space ($\lambda_1 \geq 0, \lambda_2 \geq 0$),
  - computes out-of-sample MSE for all pairs (λ_1, λ_2) ,
  - selects the MSE-minimizing model.

Evaluating Regression Models: R^2

MSE is good for comparing regression models, but the units depend on the outcome variable and therefore are not interpretable

Better to use R^2 in the test set, which has same ranking as MSE but it **more interpretable**.

Classifications

Reference:

- JWHT: chap 2.2.3, 4

Classification Framework

- Response/target variable y is **qualitative** (or **categorical**):
  - 2 categories $\rightarrow$ binary classification
  - More than 2 categories $\rightarrow$ multi-class classification
- Features X:
  - can be high-dimensional
- We want to assign a class to a **quantitative response**

  $\rightarrow$ probability to belong to the class
- **Classifier**: An algorithm that maps the input data to a specific category.
- Performance measures specific to classification

Application examples

- In business:
  - Loan default prediction
  - Type of costumer
- In public economics:
  - Tax evasion prediction
- In political sciences:
  - political affiliation of author of texts
- In medical sciences:
  - Diagnostic diseases, drug choice
- Other:
  - email filtering, speech recognition...

Why not fitting a linear regression?

- **Technically possible** to fit a linear model using a categorical response variable but it implies
  - an **ordering** on the outcome
  - a **scale** in the class difference

$\rightarrow$ If the response variable was coded differently, the results could be completely different

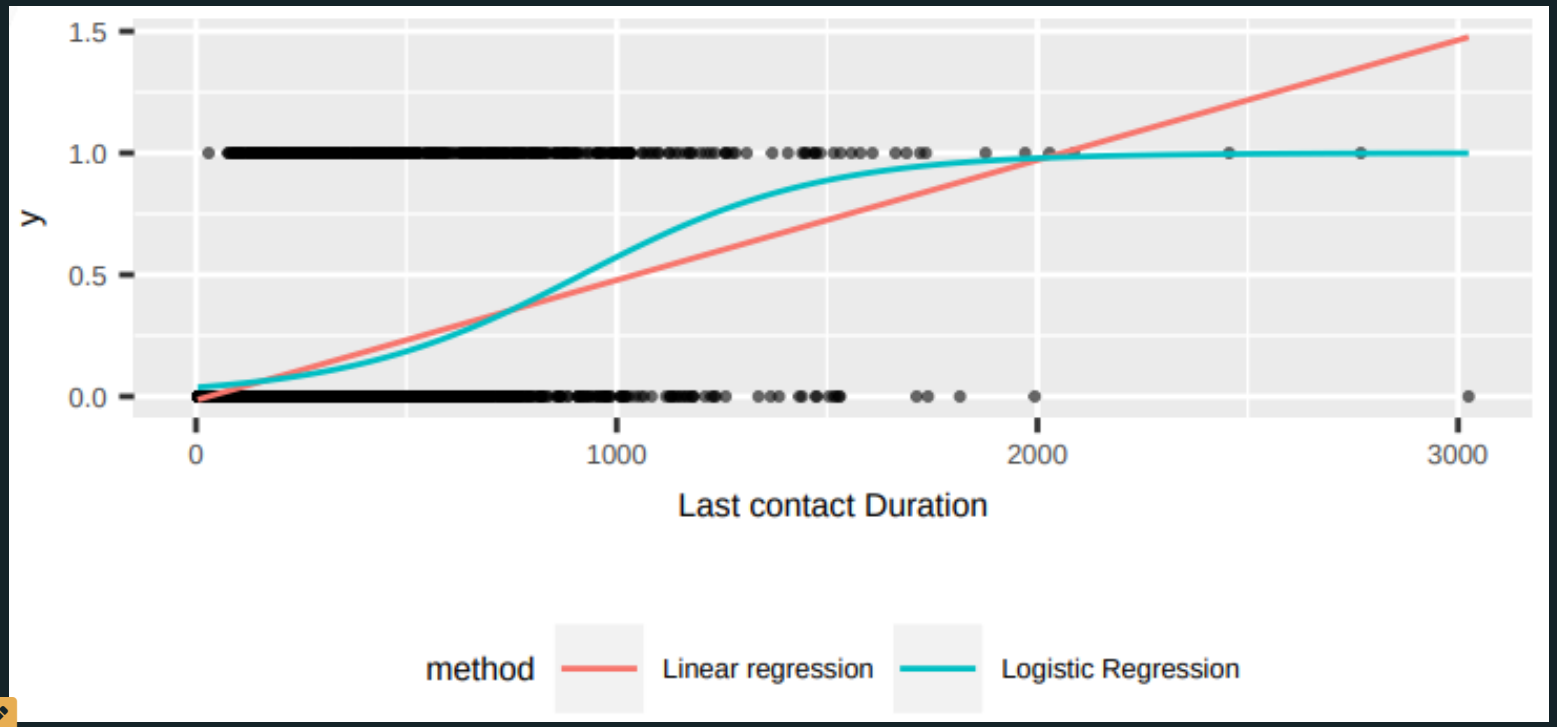
- Less problematic if the response variable is **binary**
  - The result of the model would be stable
  - But prediction may lie outside of $[0, 1]$: hard to interpret them in terms of probabilities

Example

- We predict y, the **occupation of individuals**:

$$y = \begin{cases} 0 & \text{if blue-collar} \\ 1 & \text{if white-collar} \end{cases}$$

- based on their characteristics X (gender, wage, contract duration, experience, age...)



Linear Regression vs Binary Classifier

- We model the probability of belonging to a category

$$P(y = 1 \mid X)$$

- We can rely on this probability to assign a class to the observation.
  - For example, we can assign the class yes for all observations where $P(y = 1|x) > 0.5$
  - But we can also select a different **threshold**.

Performance measures

Confusion Matrix

- For comparing the predictions of the fitted model to the actual classes.
- After applying a classifier to a data set with known labels *Yes* and *No*:

		Predicted class	
		no	yes
True class	no	TN	FP
	yes	FN	TP

Precision and Recall

- **Precision**
  - accuracy of positive predictions.
  - $\frac{\text{True Positives}}{\text{True Positives}+\text{False Positives}}$
  - decreases with false positives.
- **Recall**
  - true positive rate.
  - $\frac{\text{True Positives}}{\text{True Positives}+\text{False Negatives}}$
  - decreases with false negatives.

F1 Score

- The F_1 score provides a single combined metric it is the **harmonic mean** of precision and recall

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

$$= \frac{\text{Total Positives}}{\text{Total Positives} + \frac{1}{2}(\text{False Negatives} + \text{False Positives})}$$

- The harmonic mean gives **more weight to low values.**
- The F1 score values precision and recall **symmetrically.**

The Precision/Recall Trade-off

- F_1 favors classifiers with similar precision and recall,
- but sometimes you want **asymmetry**:

1. **low recall + high precision is better**
    - e.g. **deciding "guilty" in court**, you might prefer a model that
    - lets many actual-guilty go free (high false negatives $\leftrightarrow$ low recall)...
    - ... but has very few actual-innocent put in jail (low false positives $\leftrightarrow$ high precision
2. **high recall + low precision is better**

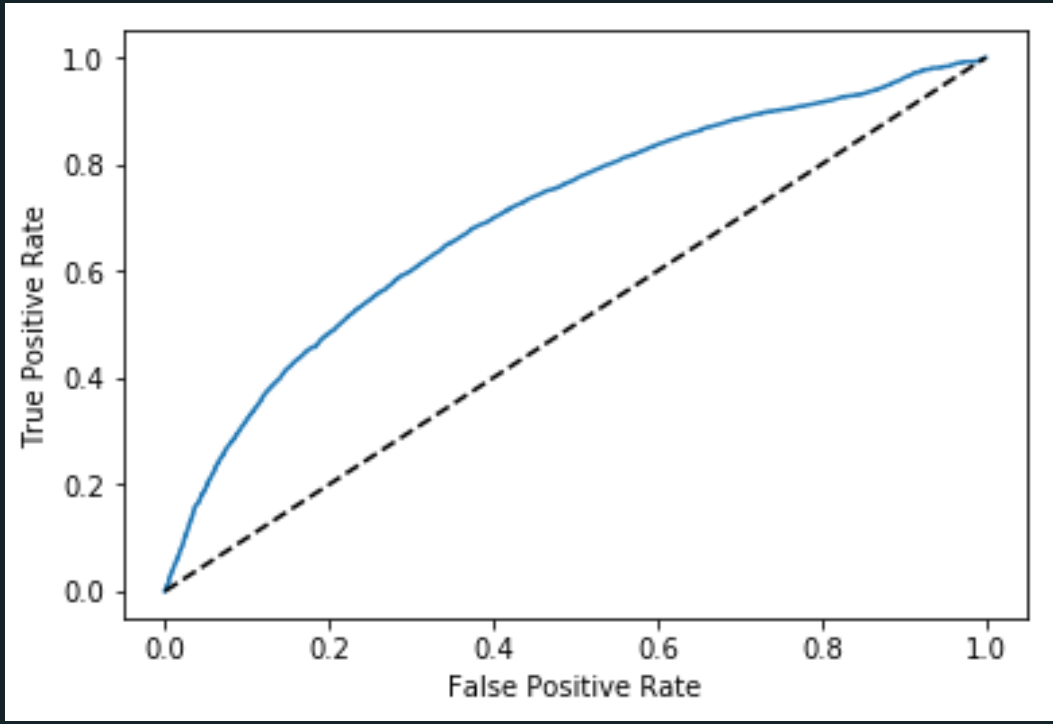
The Precision/Recall Trade-off

- F_1 favors classifiers with similar precision and recall,
- but sometimes you want **asymmetry**:

1. **low recall + high precision is better**
2. **high recall + low precision is better**
    - e.g classifier to **detect bombs during flight screening**, you might prefer a model that:
    - has many false alarms (low precision)...
    - ... to minimize the number of misses (high recall).

ROC Curve and AUC

- Plots *true positive rate* (recall) against the *false positive rate* ($\frac{FP}{FP+TN}$):



ROC Curve and AUC

- The area under the ROC curve (AUC) is a popular metric ranging between:
  - 0.5
    - **random classification**
    - ROC curve = first diagonal
  - and 1
    - **perfect classification**
    - = area of the square
  - better classifier → ROC curve toward the top-left corner
- Good measure for model comparison

Binary Classifier

- Logistic Regressions
- K-Nearest Neighbors
- Support Vector Machine

Logistic Regression

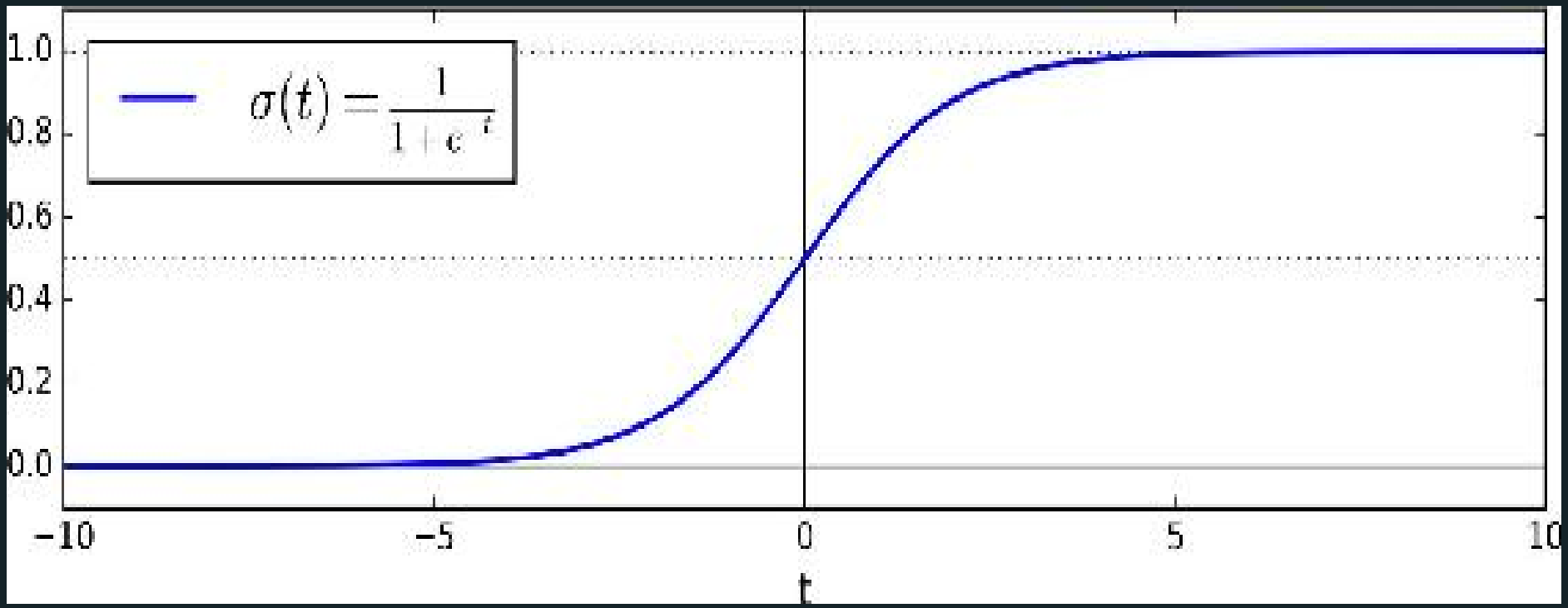
- Like OLS, logistic "regression" computes a weighted sum of the input features to predict the output.
  - But it transforms the sum using the **logistic function**.

$$\hat{p} = \Pr(Y_i = 1) = \sigma(\theta' x)$$

where $\sigma(\cdot)$ is the sigmoid function

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

Logistic Regression



- Prediction:

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < .5 \\ 1 & \text{if } \hat{p} \geq .5 \end{cases}$$

Logistic Regression Cost Function

- The cost function to minimize is

$$J(\theta) = \underbrace{-\frac{1}{m}}_{\text{negative}} \sum_{i=1}^{m} [\underbrace{y_i}_{y_i=1} \underbrace{\log(\hat{p}_i)}_{\text{log prob}y_i=1} + \underbrace{(1-y_i)}_{y_i=0} \underbrace{\log(1-\hat{p}_i)}_{\text{log prob}y_i=0}]$$

  - this does not have a closed form solution
  - but it is convex, so gradient descent will find the global minimum.

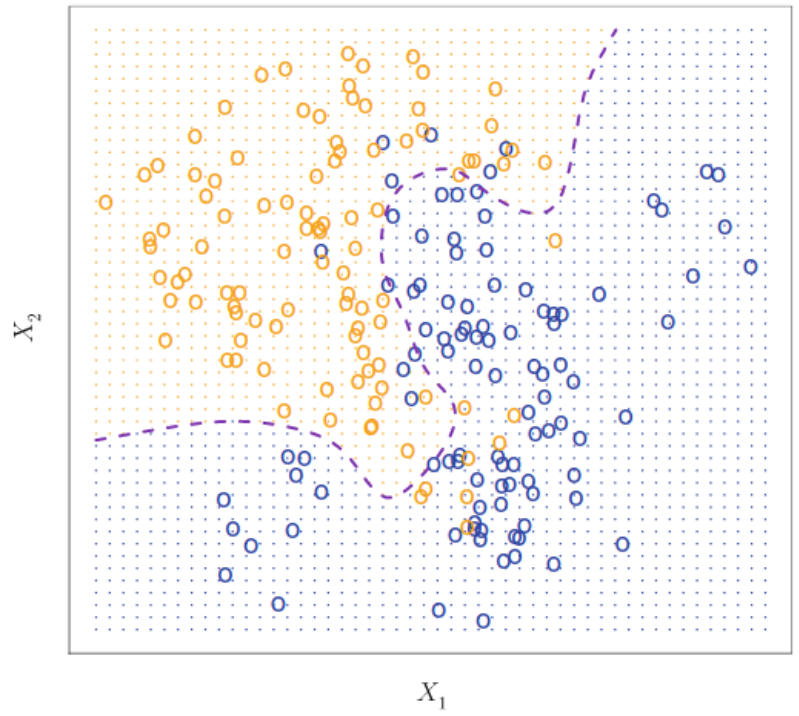
- Just like linear models, logistic can be regulared with L1 or L2 penalties, e.g.:

$$J_2(\theta) = J(\theta) + \alpha_2 \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

Naive Bayes Classifier

- Relies on the observed conditional probabilities (and the Bayes theorem)
- For a 2-class problem for a given observation $X = x_0$:
  - Predict class 1 if $P(Y = 1|X = x_0) \geq 0.5$
  - Predict class 0 if $P(Y = 1|X = x_0) < 0.5$
- Relies on the independence assumption

Naive Bayes Classifier



K-Nearest Neighbors

- With real data, we do not know the conditional distribution of Y given X.
- computing the Bayes classifier is not possible.
- The K-nearest neighbors (KNN) classifier estimates the conditional distribution of Y given X.
- Approximate Bayes decision rule in a subset of data around the testing point
- **Non-parametric method** often successful in classification situations where the **decision boundary is very irregular**

K-Nearest Neighbors

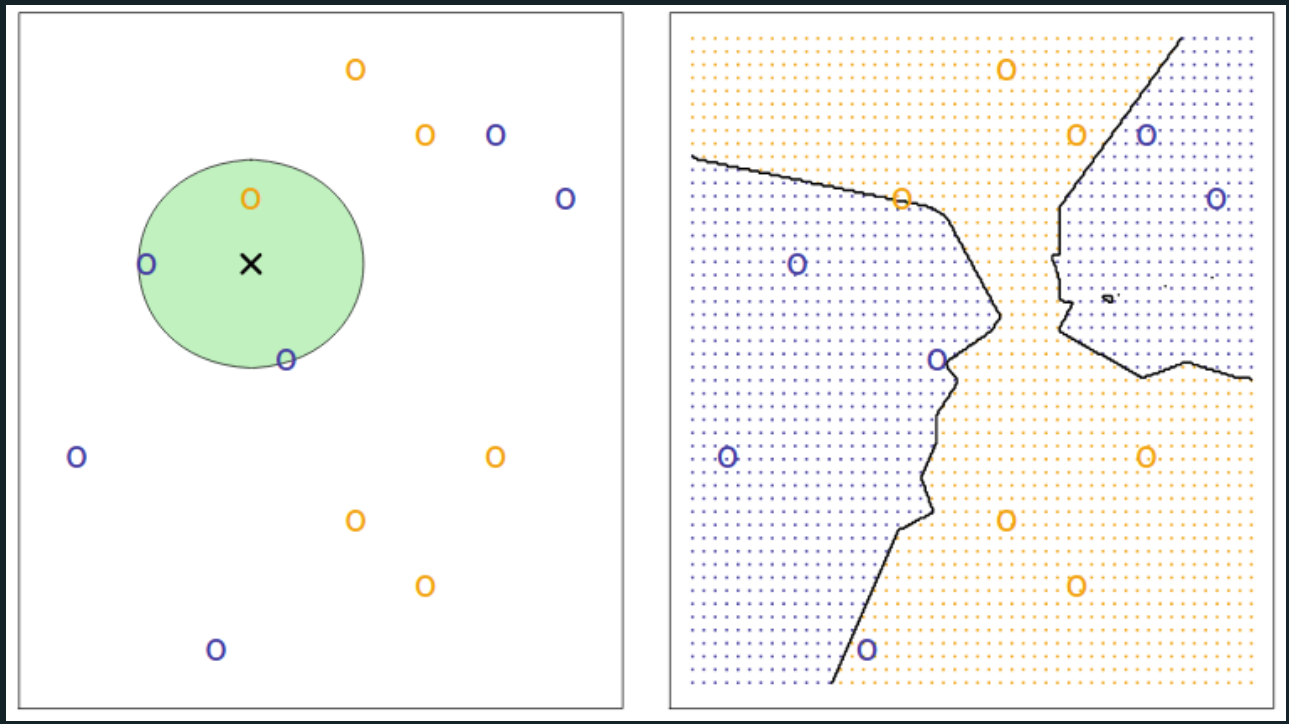
For K and a test observation x_0

1. KNN classifier first identifies the K points in the training data that are closest to x_0 (i.e N_0)
2. estimates the conditional probability for class j as the fraction of points in N_0 whose response values equal j:

$$P(Y = j | X = x_O) = \frac{1}{K} \sum_{i \in N_O} I(y_i = j)$$

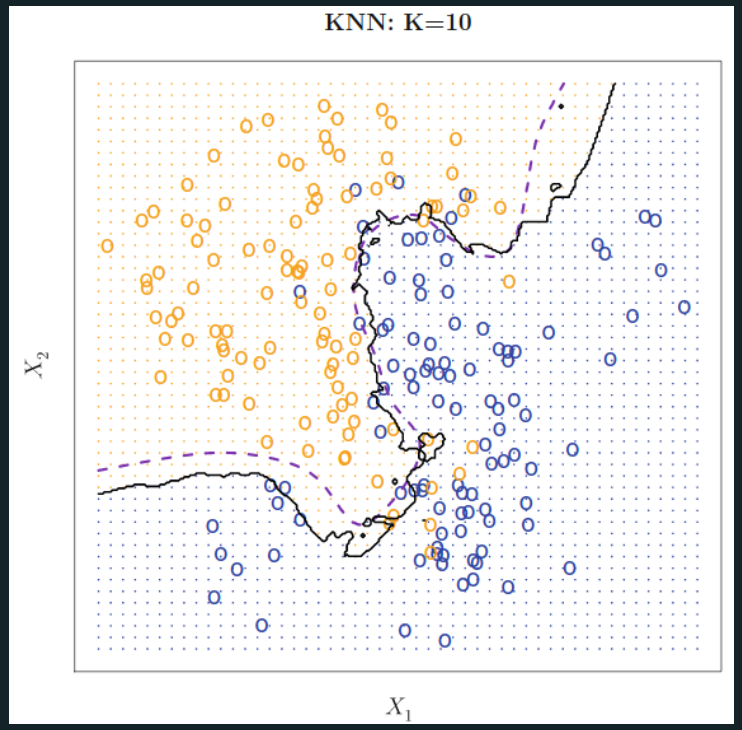
3. applies Bayes rule and classifies the test observationx0tothe class with the largest probability

KNN: illustration



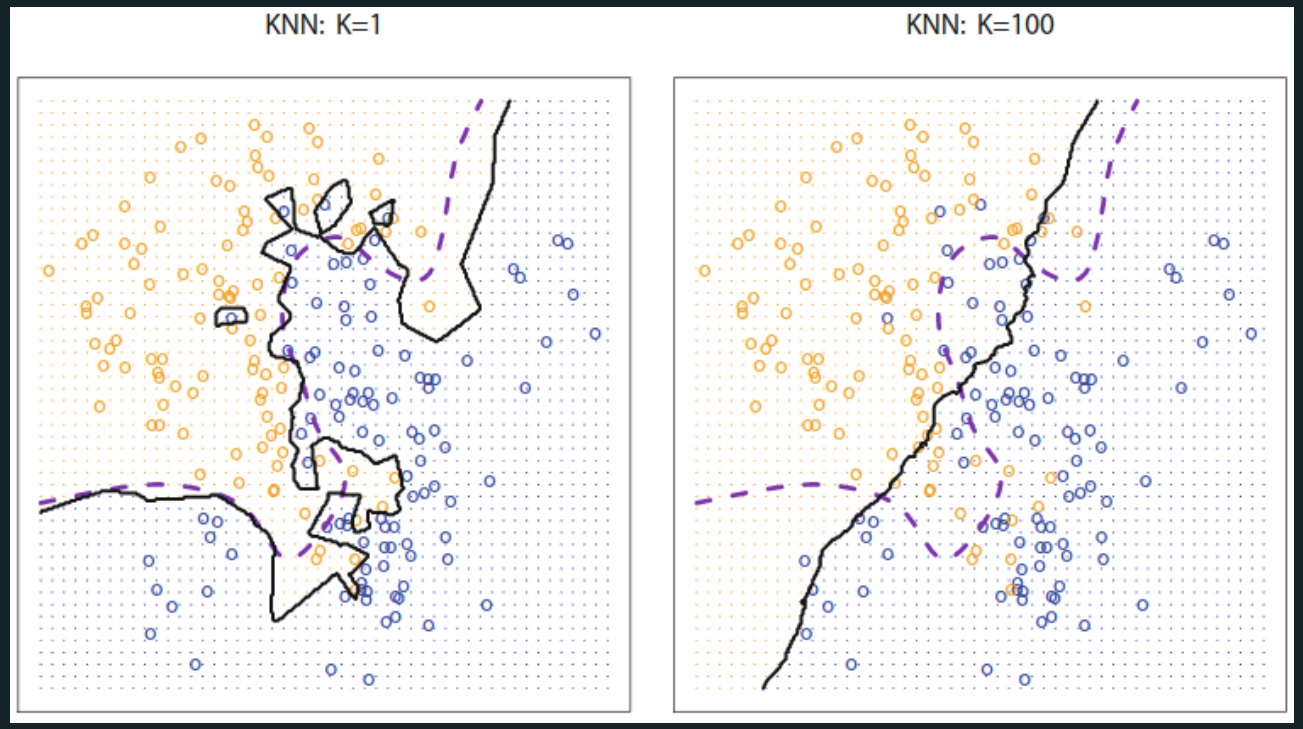
- Assume $K = 3$
- Left: small training data set consisting of 6 blue and 6 orange observations
- Right: KNN approach at of the possible values for X_1 and X_2, and corresponding KNN decision boundary

KNN: illustration



- black curve: KNN decision boundary
- dashed line: Bayes decision boundary

KNN: choice of K



- $K = 1$,the KNN training error rate is 0, but the test error rate may be quite high

Support Vector Machine

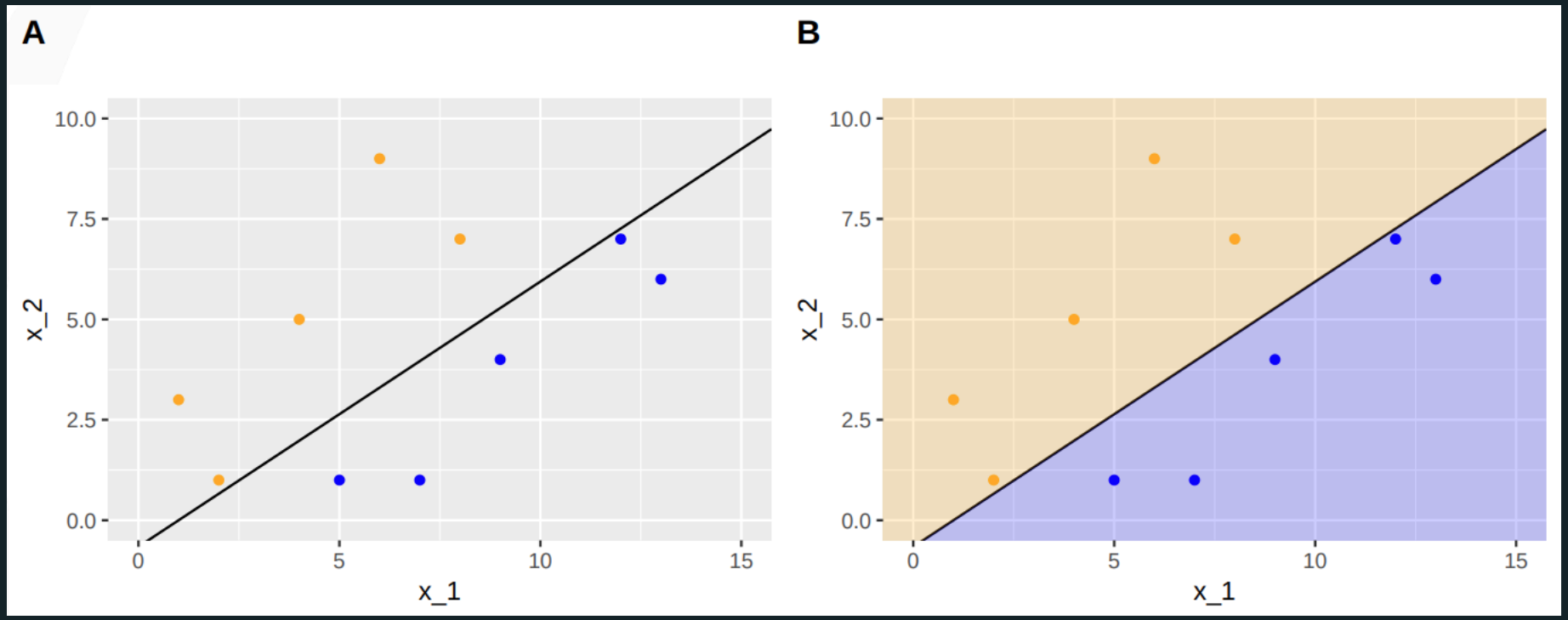
- Context: developed in the mid-1990s
- A generalization of the early logistic regression (1930s)
- One of the best “out of the box” classifiers
- Core idea: hyperplane that separates the data as well as possible, while allowing some violations to this separation

Support Vector Machine: context and concepts

- Pieces of the puzzle:
  1. A **maximal margin classifier**: requires that classes be separable by a linear boundary.
  2. A **support vector classifier**: extension of the maximal margin classifier.
  3. **Support vector machine**: further extension to accommodate non-linear class boundaries.
- For binary classification, can be extended to multiple classes

Classification and Hyperplane

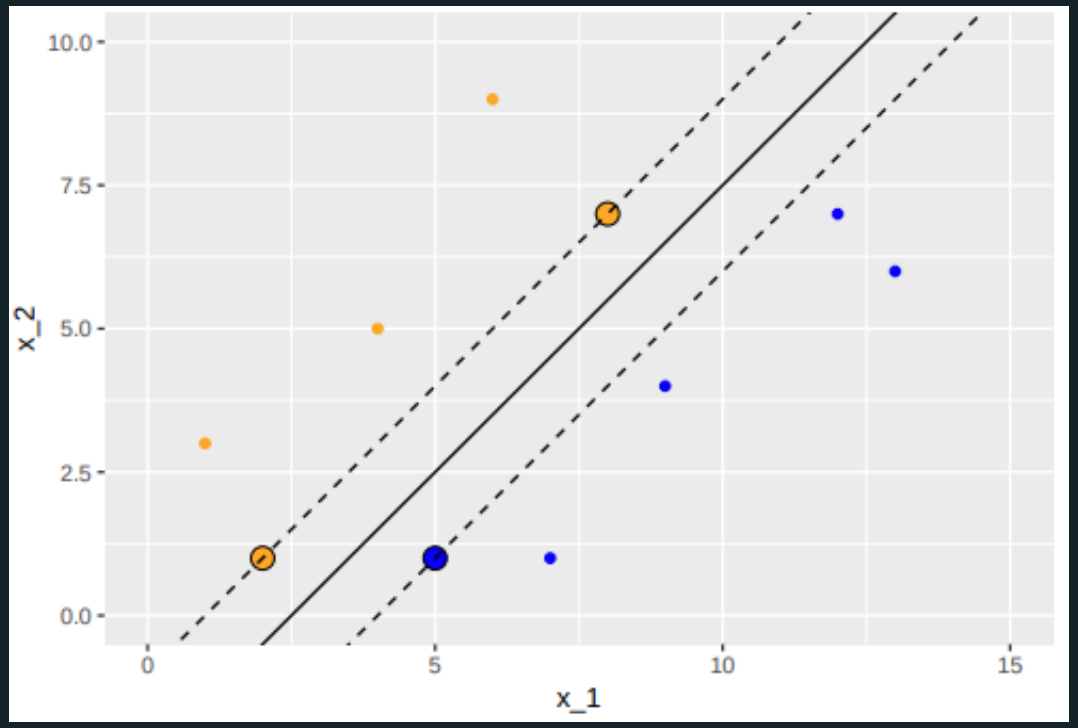
A perfectly separating linear hyperplan for a binary outcome



There are an infinity of such separating hyperplan
→ we need to choose one

Maximum Margin

Maximum margin classifier for a perfectly separable binary outcome variable



Criterium for optimal choice: the separating hyperplane for which the margin is the farthest from the observations
i.e., to select the **maximal margin hyperplane**

Support Vector

Support vector = the 3 observations from the training set that are equidistant from the maximal margin hyperplane

→ they "support" the maximal margin hyperplane (if they move, the the maximal margin hyperplane also moves)

Overcoming the perfectly separable hyperplan assumption

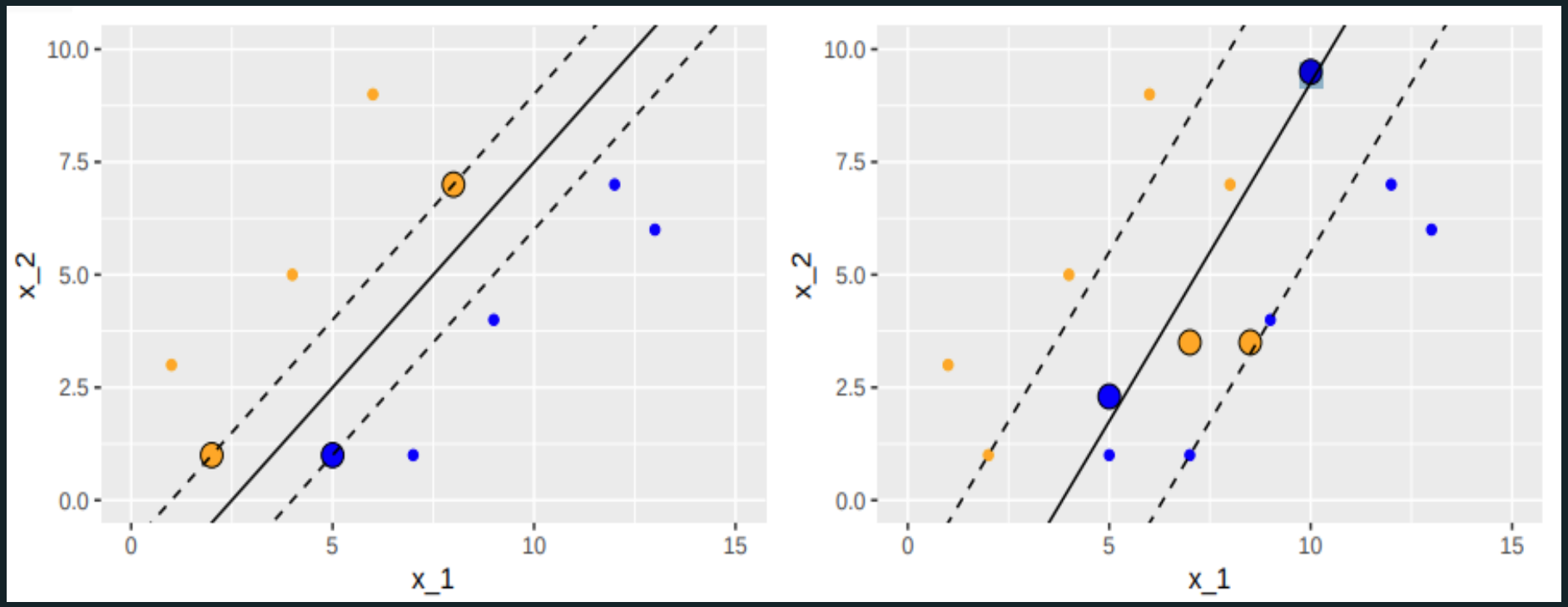
We allow some number of observations to violate the rules so that they can lie on the wrong side of the margin boundaries.

$\rightarrow$ find a hyperplane that almost separates the classes

The **support vector classifier** generalizes the maximum margin classifier to the non-separable case.

Support Vector Classifiers

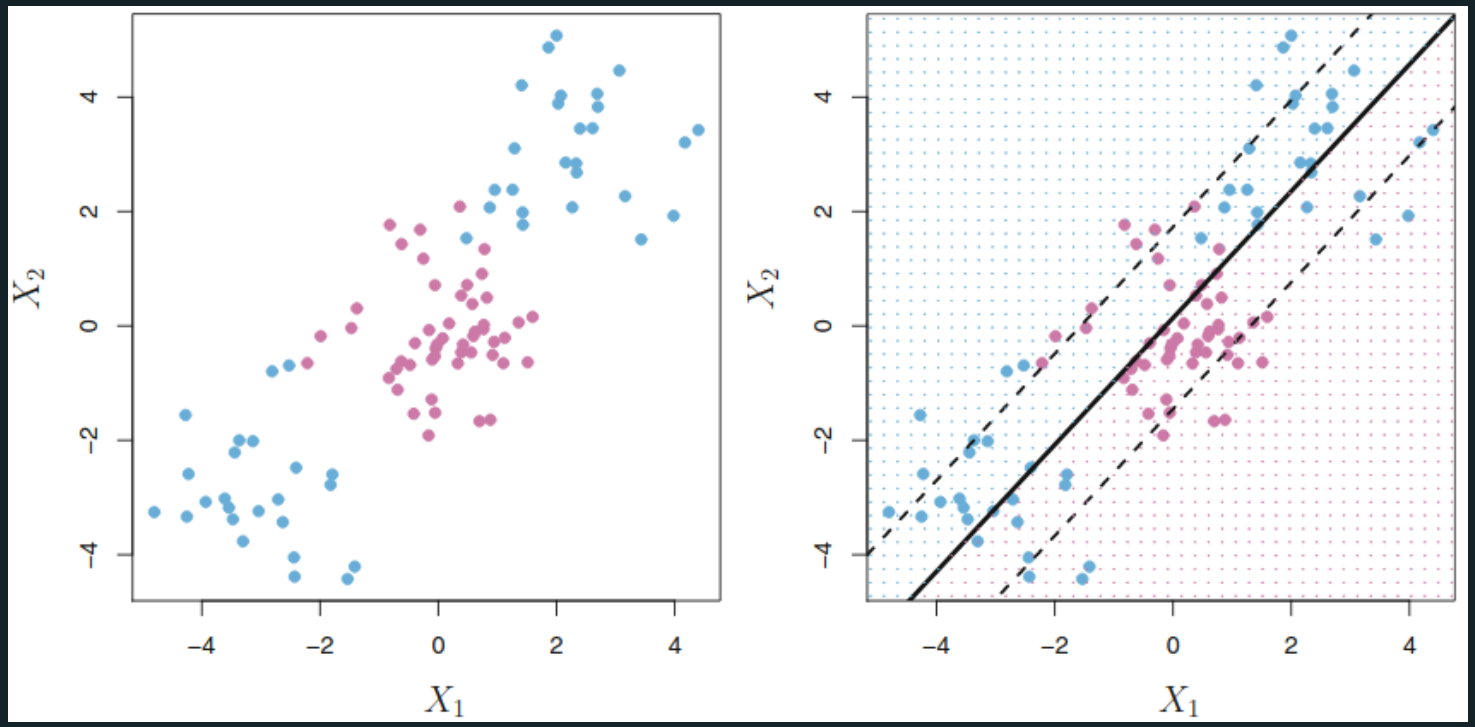
Maximal margin classifier (left) and support vector classifier (right)



Support Vector Classifiers: Details

- A **tuning parameter** C determines the severity of the violation ot the margin that the model tolerates
  - chosen by cross Validation
  - controls the bias-variance trade-off
- C small $\rightarrow$ narrow margins, rarely violated
- C large $\rightarrow$ wide margins, allow more violation
  - More bias classifier, but lower variance

Shortcomings of the linearity assumption:

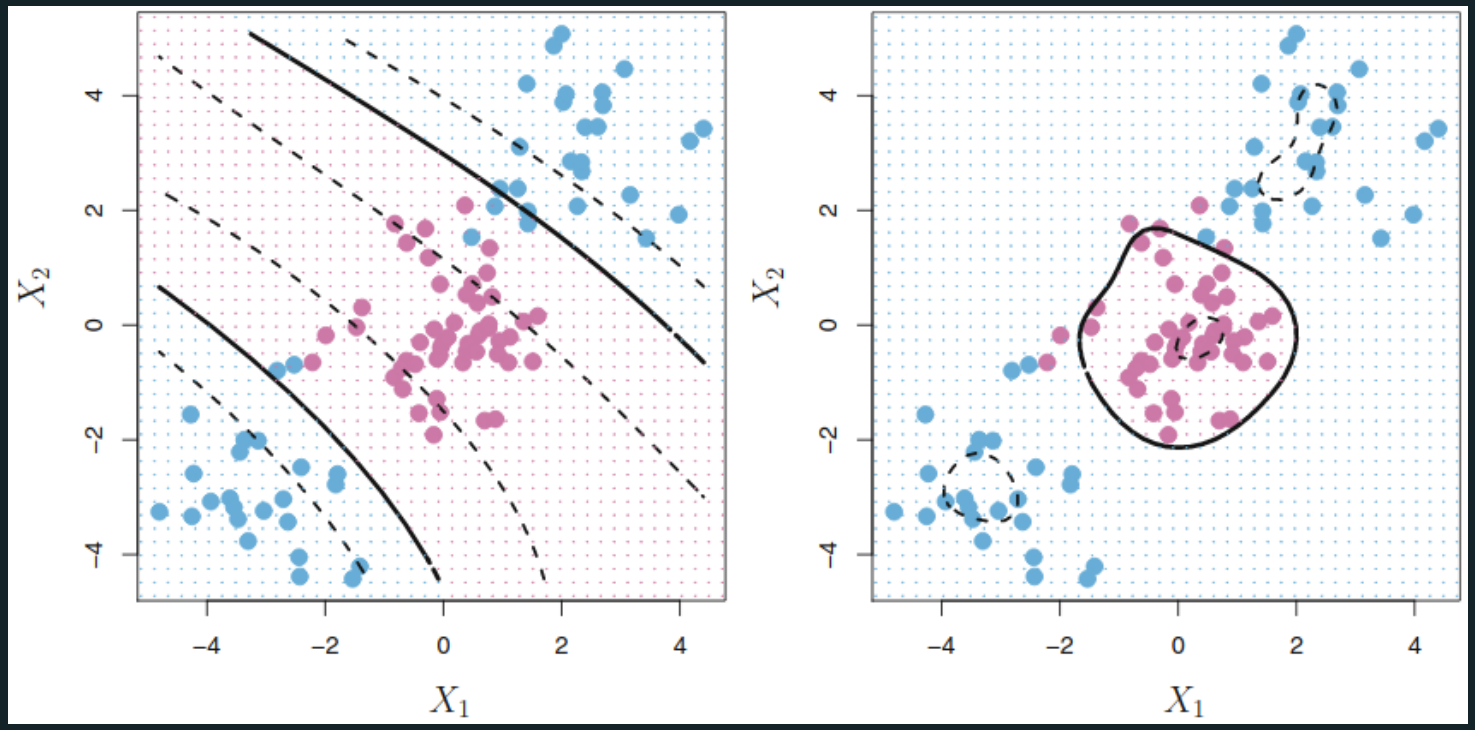


Overcoming the linearity assumption:

Support vector machines

- *Idea 1*: (polynomial) transformation of the features + StandardScaler + LinearSVC.
- *Idea 2*: convert a linear classifier into a classifier that produced **non-linear decision boundaries**. → using a **Kernel** such as:
  - Gaussian RBF kernel
  - Polynomial kernel
- **We do not open the kernel box.**
  - Just think as them as a way to construct non-linear hyperplans
  - Try out different kernel and distance specification

Support vector machines



- *Left*: polynomial kernel of degree 3;
- *Right*: radial kernel

Wrap-up

Selecting the Tuning Parameter By Cross-Validation

1. Choose a **grid** of λ values
2. Compute the **CV error** for each lambda
3. Select the tuning parameter value for which the CV error is smallest
4. **Re-fit** the model using all available observation and the best λ

Data Prep for Machine Learning

- See Geron Chapter 2 for pandas and sklearn syntax:
  - imputing missing values.
  - feature **scaling** (coefficient size depends on the scaling)
  - **encoding** categorical variables.
- Best practice
  - **reproducible** data pipeline
  - **standardize** coefficients

Other Supervised Machine Learning Methods

- Forward Selection,
- Backward Selection
- Trees and Forests
- Neural Networks
- Boosting
- Ensemble Methods

Types of Classification Algorithms

- Linear Classifiers
  - Logistic regression
  - Naive Bayes classifier
- Support vector machines
- Kernel estimation
  - k-nearest neighbor
- Decision trees
  - Random forests

“Essentially, all models are wrong, but some are useful” -- George Box