Deloitte.

Deloitte Data Science Academy

Lesson 3:

Classification and Regression

Propensity models, binary classifiers, regression, model evaluation, model lifecycle

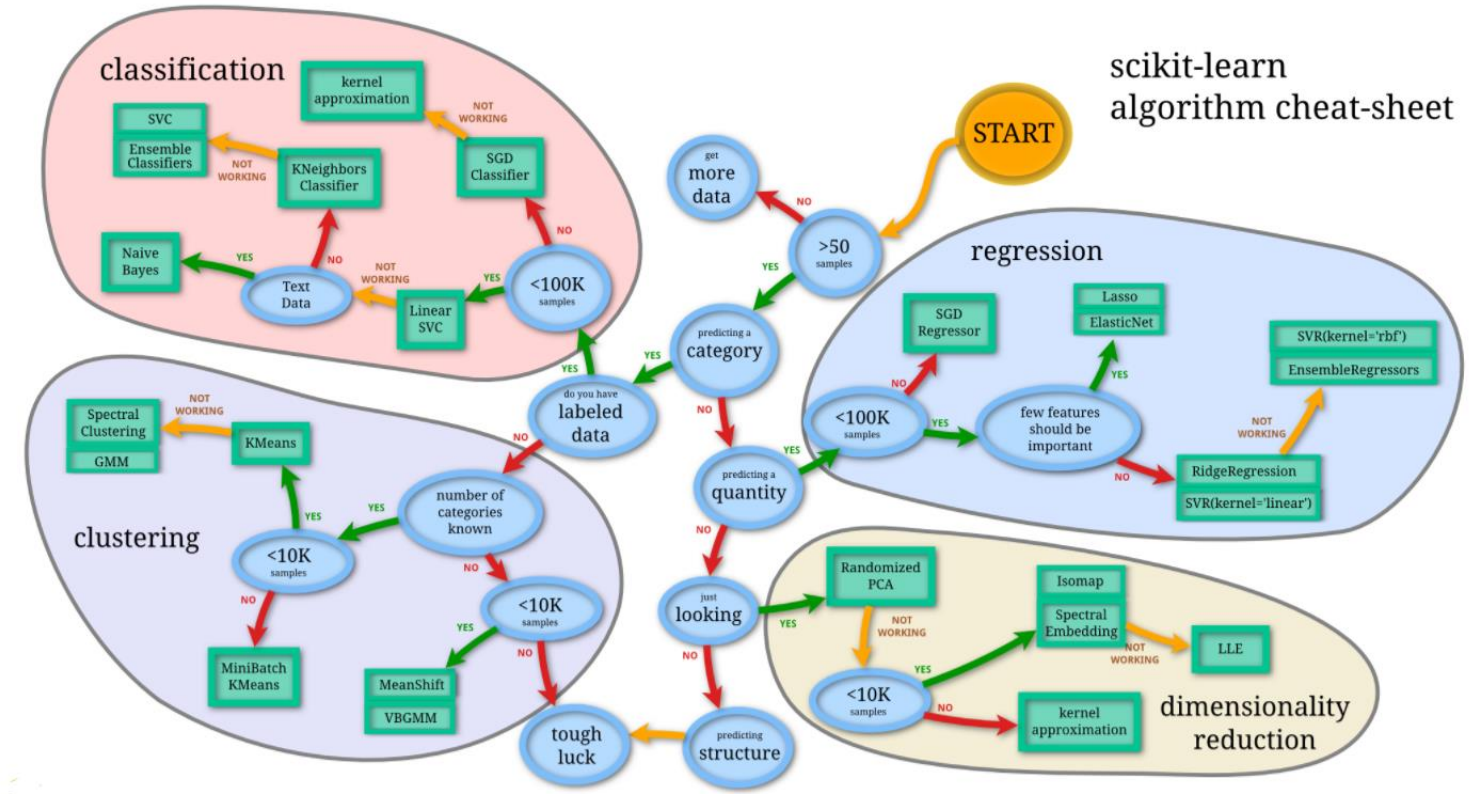
Hands on training for classification and regression

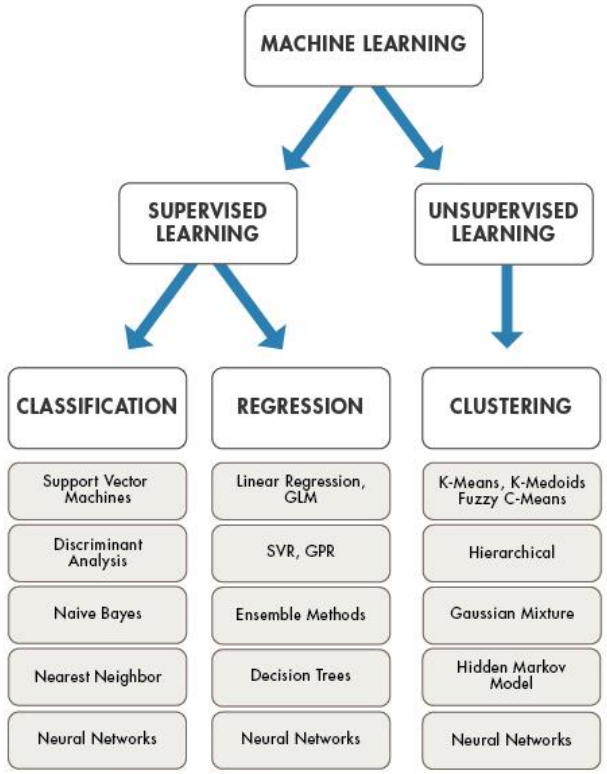
19/11/2019

Today's program

1	Introduction to Data Science, Data Science Tools, Python basics, Git – Code collaboration, Pure Python data exploration	HA1
2	Features and dataset preparation, Clustering, Behavioral segmentation, Hands on training for data exploration and clustering with Python, Anomaly Detection	HA2
3	**Regression and Classification, Hands on training for classification and regression**	**HA3**
4	Introduction to NLP and Computer Vision, AutoML	HA4

Recap – Machine Learning Overview





* **Reinforcement** (learning from mistakes/rewards) is often mentioned as a third type of ML.

Today

- Building a model
- Binary Classification
  - Motivation
  - Algorithms
  - Evaluation
- Task
- Regression
  - Motivation
  - Algorithms
  - Evaluation
- Model Interpretability
- Unbalanced Datasets
- Dimensionality reduction
- Model Lifecycle

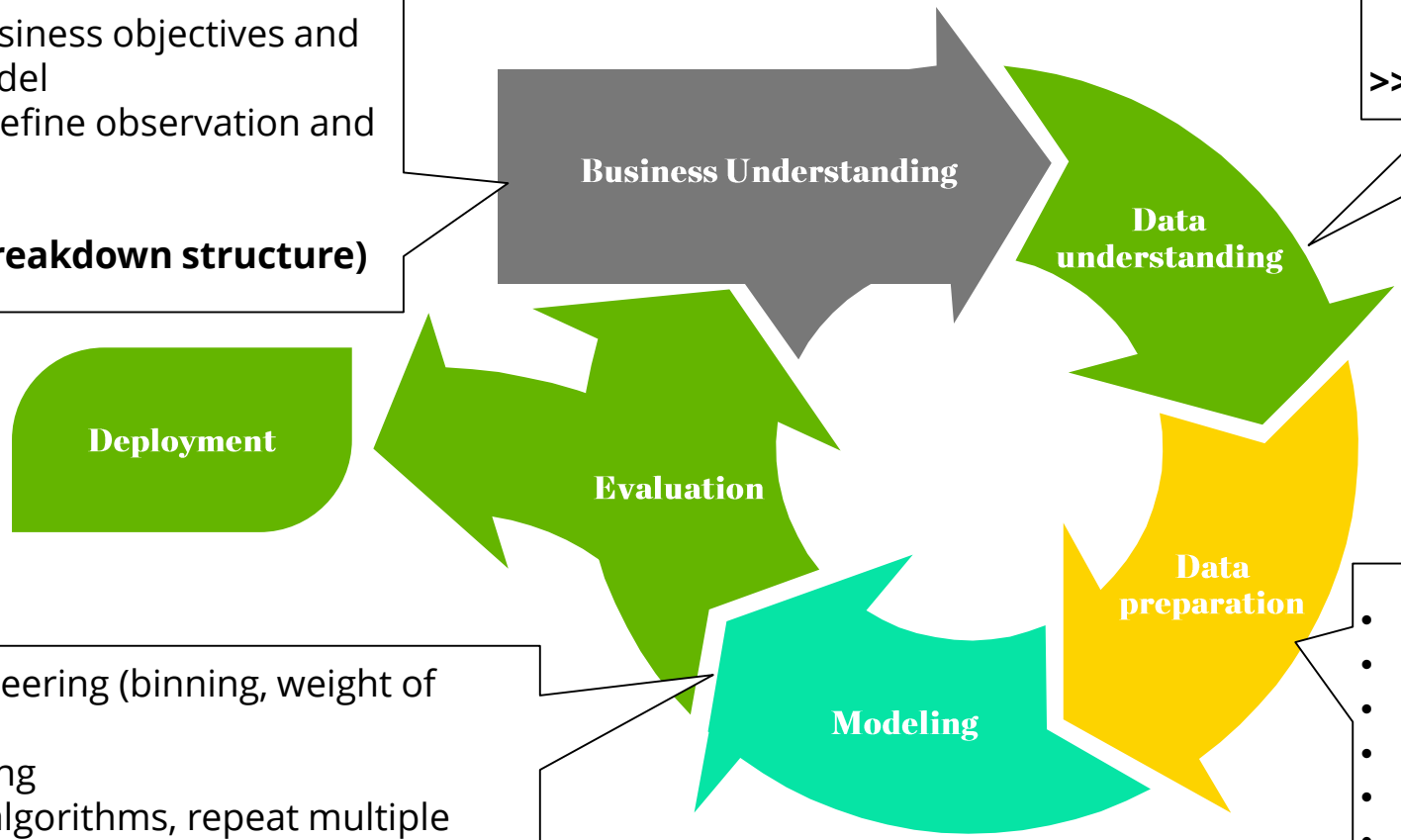
Building a model

CRISP-DM Framework

Applicable to all Data Science jobs

- Define the business objectives and operating model
- Define KPIs, define observation and target
- Planning

>> WBS (Work breakdown structure)



- Discuss data availability, volumes

>> Data request

- Data ingestion
- Data quality check
- Data acceptation
- Calculate target table
- Predictors design
- Predictors calculation

>> Modeling dataset

- Feature engineering (binning, weight of evidence, ...)
- Training, testing
- Try different algorithms, repeat multiple times

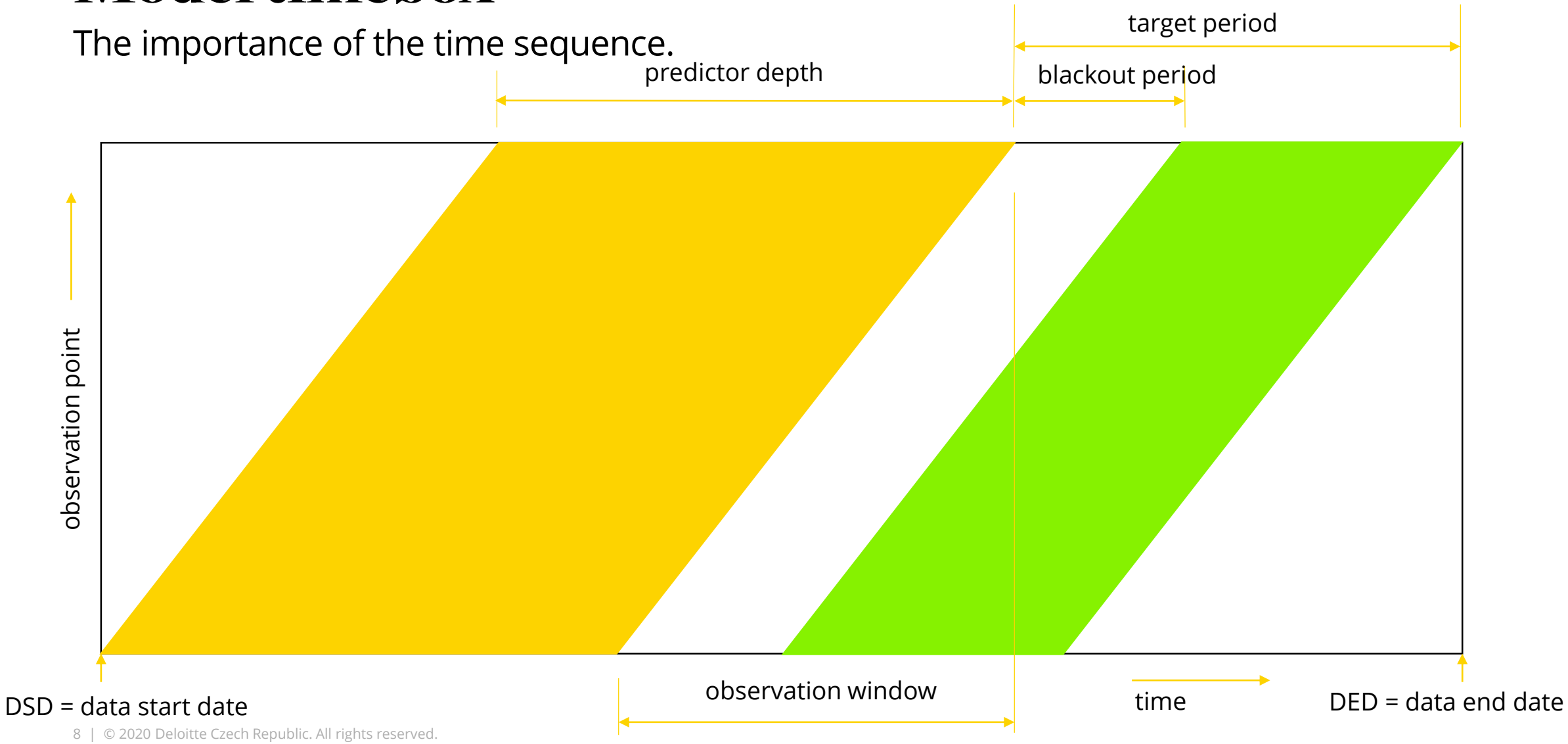
>> Model development report

Target calculation

1. Interpreting business requirements (various “KO” criteria) into SQL language.
2. Target is the quantity we want to predict.
3. Time sequence can be important – see the very next slide.

Model timebox

The importance of the time sequence.



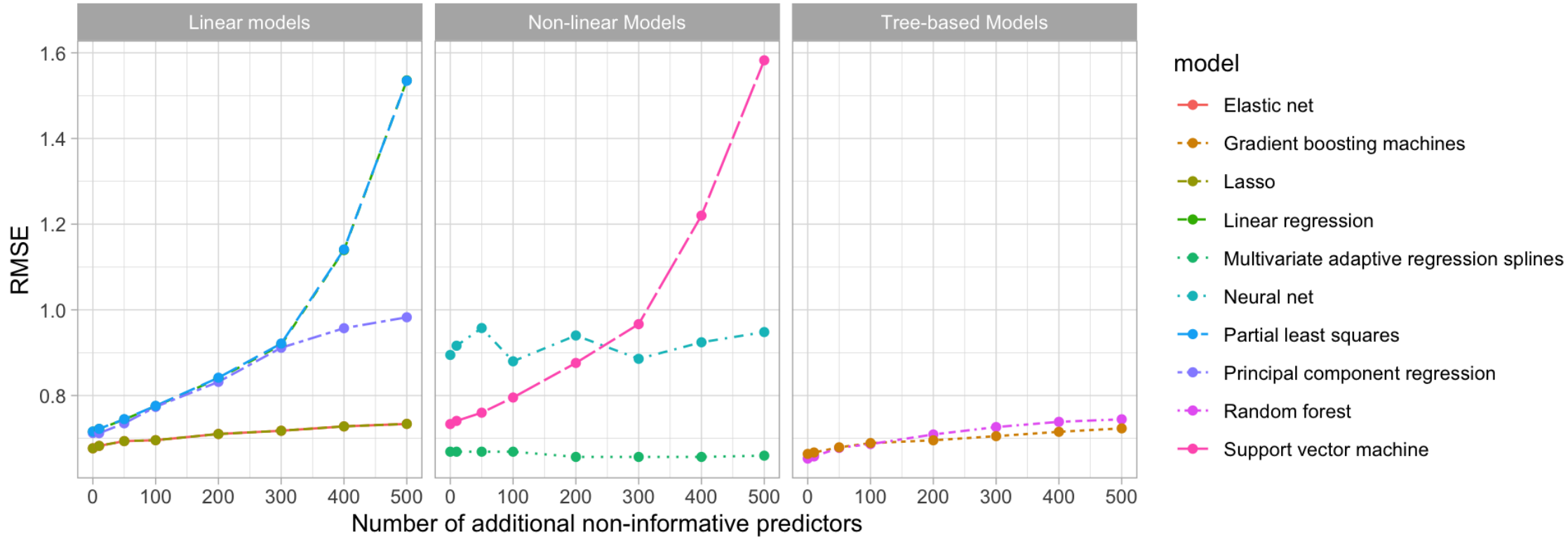
Feature engineering

Data preparation

- **Computation of relevant predictors**
  - Each predictor represents a more or less complex calculation that transforms data from source tables into a resulting continuous or categorical variable that describes reality at the level of observation.
  - Ratios, trends, differences, aggregated attributes, etc.
- **Feature selection & filtering** (near/zero variance variables, statistical significance (e.g., correlation test, chi-squared test, Kruskal-Wallis test, etc.), permutation-based variable importance, strong inter-correlation between predictors, business/ethical reasons, etc.)
- **Data transformations** (lumping, imputation, normalization, standardization, dimensionality reduction, dummy encoding, target encoding, etc.)

Feature selection & filtering

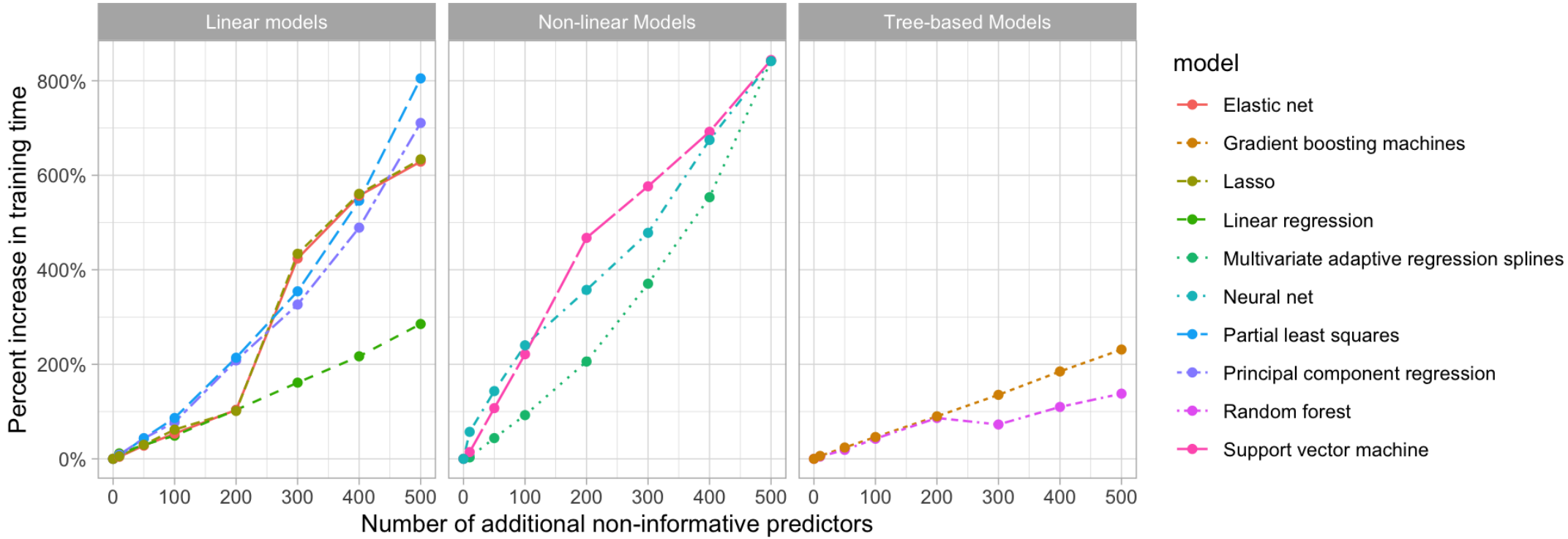
Negative impact of non-informative features on models' performance (and interpretability).



Source: Boehmke & Greenwell (2019)

Feature selection & filtering (cont.)

The time to train the models can be negatively impacted as more features are added.



Source: Boehmke & Greenwell (2019)

Feature selection & filtering (cont.)

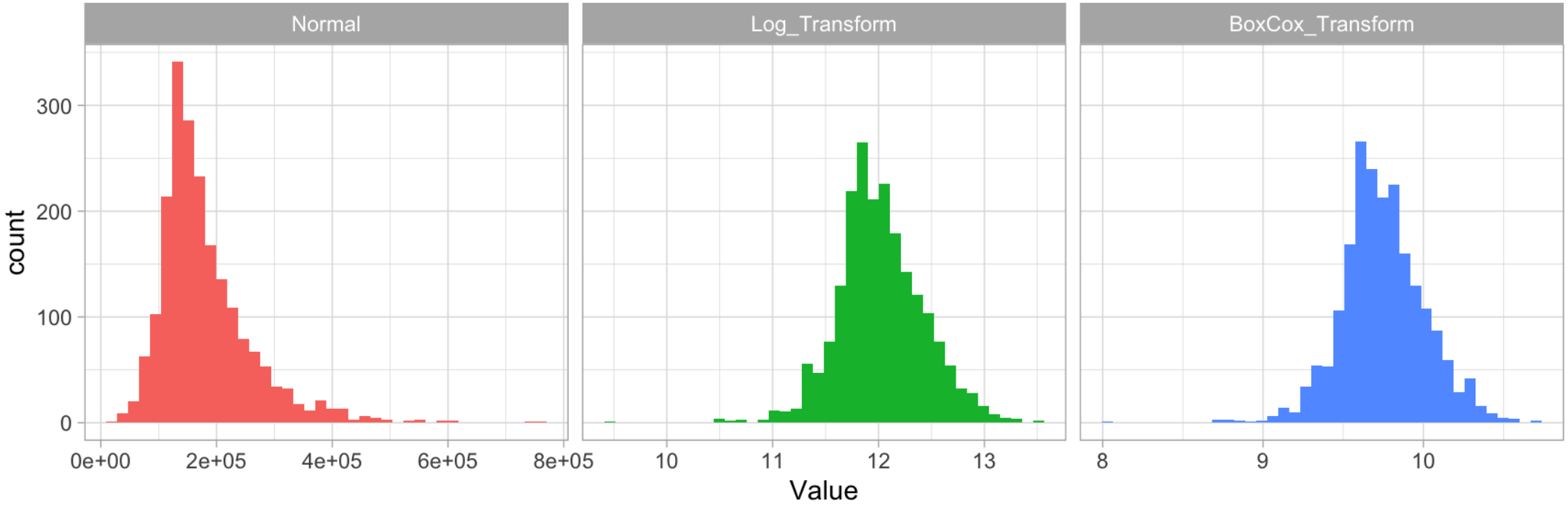
Two main ways to select relevant features (+ near/zero variance filtering and criteria based on business).

	Procedure	Advantages	Disadvantages
Filter Methods	Applying bivariate statistical metrics	• Computational cheap • Practical no overfitting • Independent of model algorithms	• Selection of redundant features • No wholesome consideration of the feature space
Wrapper Methods	Iteratively searching for the best feature set	• Granular selection of features • Wholesome consideration of the feature space • Selection of relevant features	• Computationally very, very, very expensive (NP-hard) • Potential overfitting

Source: STATWORK (2018)

Target transformation

Skewed target normalization using log, Box-Cox or Yeo-Johnson transformation.



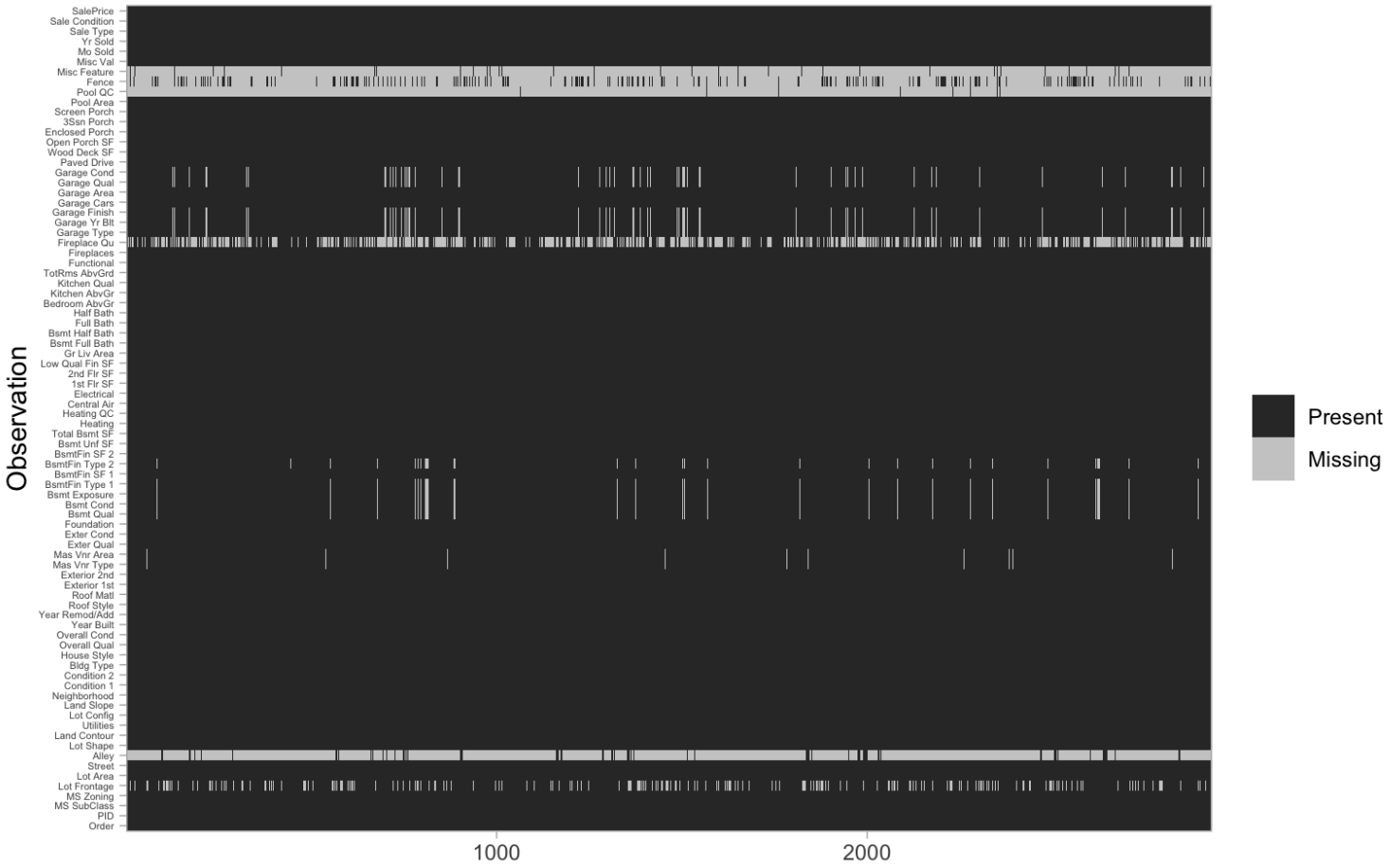
Source: Boehmke & Greenwell (2019)

Dealing with missing values

Variables with too many missing values (e.g., > 40%) are candidates for being excluded from the analysis.

Informative missingness implies a structural cause for the missing value that can provide insight in its own right. Missing values then may be given own category (e.g., “None”)

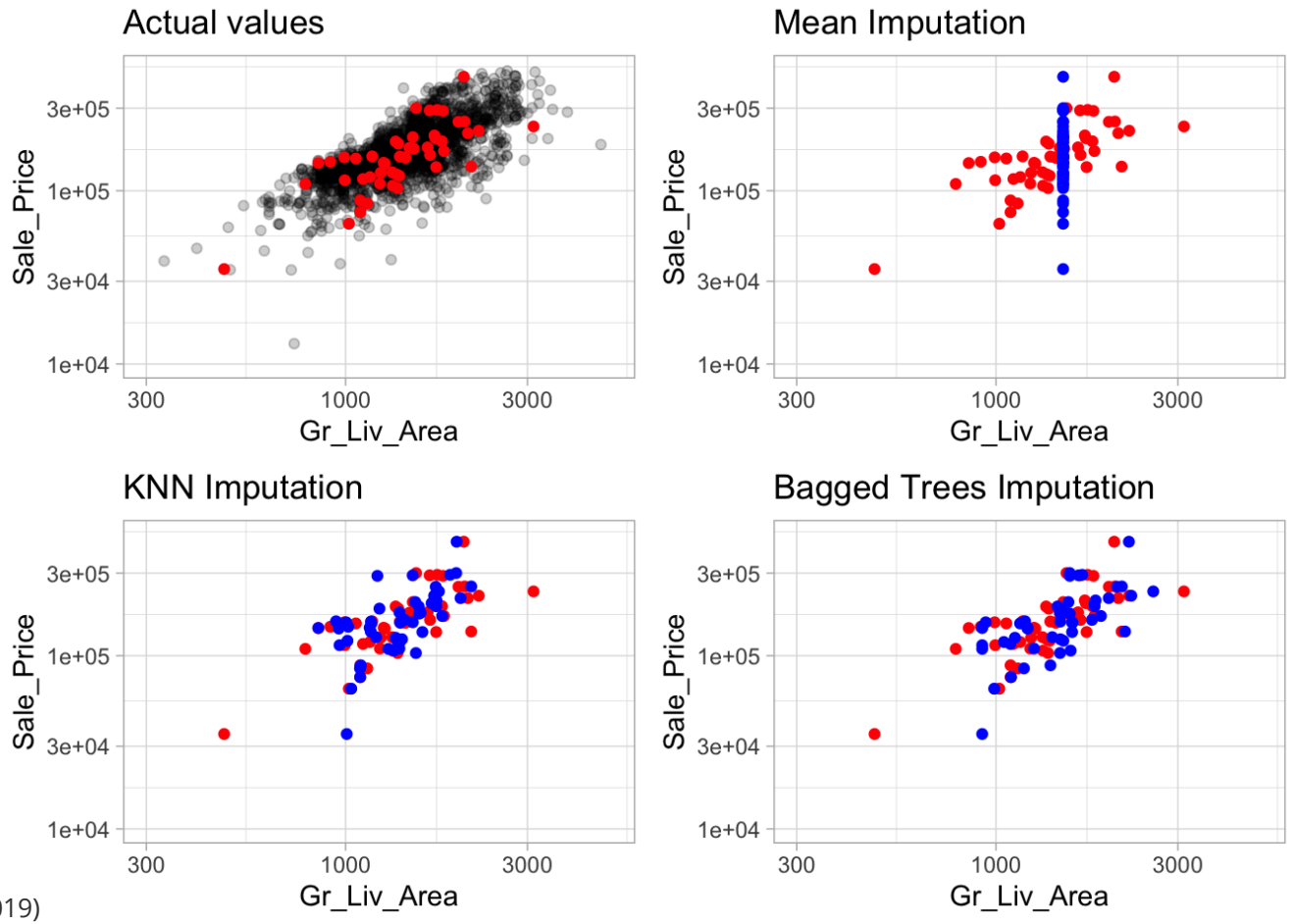
Missingness at random implies that missing values occur independent of the data collection process. Such missing values could be replaced by **imputation.**



Source: Boehmke & Greenwell (2019)

Imputation

Replacing missing values with substituted, “best guess” values.



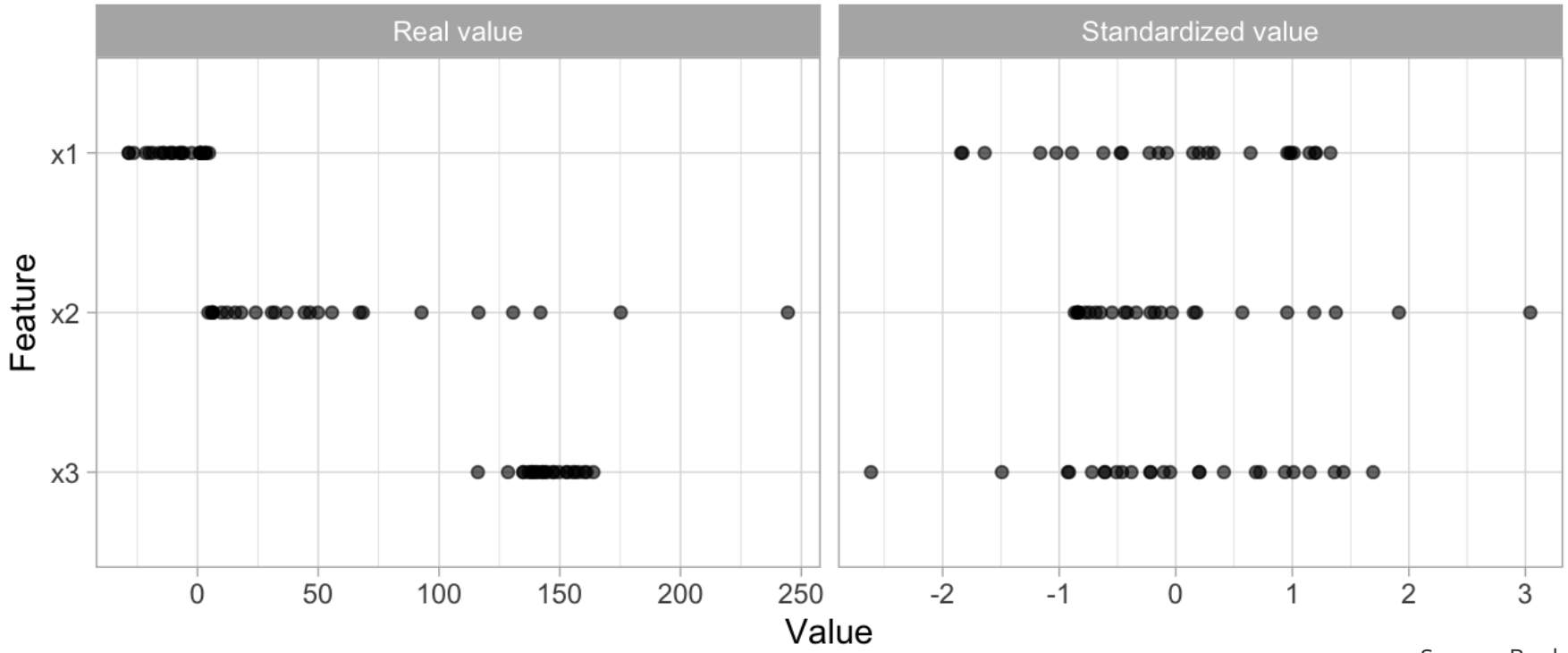
Source: Boehmke & Greenwell (2019)

Feature normalization & standardization

Parametric models that have distributional assumptions (e.g., GLMs, and regularized models) can benefit from **minimizing the skewness of numeric features** (using Box-Cox or Yeo-Johnson transformation).

Models incorporating smooth functions of input features (e.g., GLMs, regularized regression, neural networks, and support vector machines) are **sensitive to the scale of the inputs** and thus require **all features to be compared on a common value scale** regardless of their real value differences.

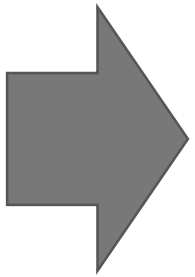
.



Source: Boehmke & Greenwell (2019)

Categorical – Lumping

```
count(ames_train, Neighborhood) %>% arrange(n)
## # A tibble: 28 x 2
##    Neighborhood                          n
##    <fct>                             <int>
##  1 Landmark                              1
##  2 Green_Hills                           2
##  3 Greens                                7
##  4 Blueste                               9
##  5 Northpark_Villa                      17
##  6 Briardale                            18
##  7 Veenker                              20
##  8 Bloomington_Heights                  21
##  9 South_and_West_of_Iowa_State_University    30
## 10 Meadow_Village                       30
## # … with 18 more rows
```



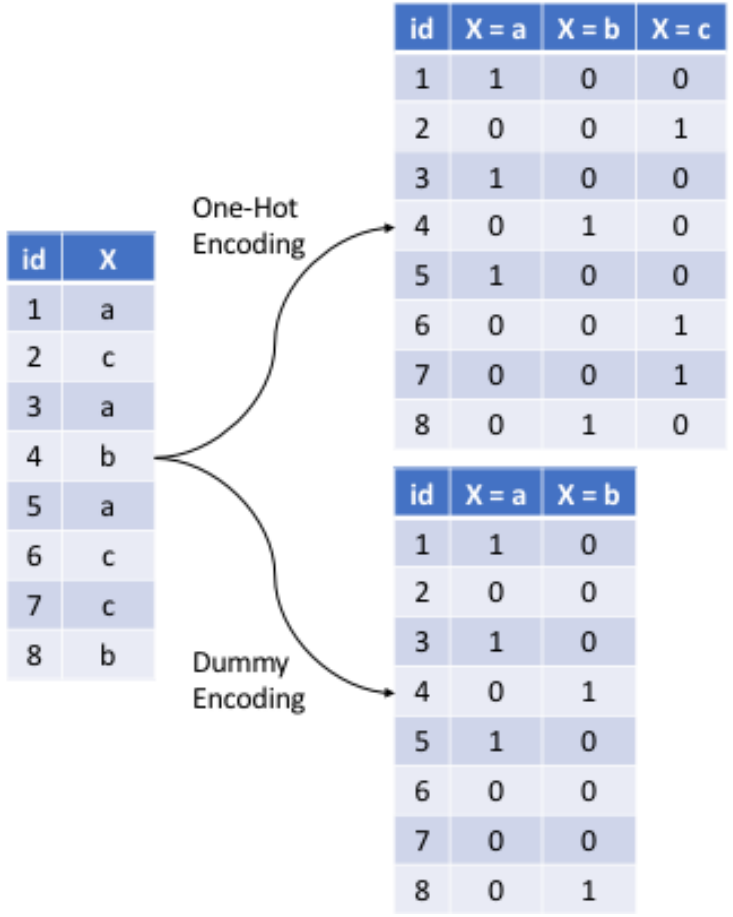
```
# New distribution of Neighborhood
count(apply_2_training, Neighborhood) %>% arrange(n)
## # A tibble: 22 x 2
##    Neighborhood                          n
##    <fct>                             <int>
##  1 Bloomington_Heights                  21
##  2 South_and_West_of_Iowa_State_University    30
##  3 Meadow_Village                       30
##  4 Clear_Creek                          31
##  5 Stone_Brook                          34
##  6 Northridge                           48
##  7 Timberland                           55
##  8 Iowa_DOT_and_Rail_Road               62
##  9 Crawford                             72
## 10 Mitchell                             74
## # … with 12 more rows
```

Source: Boehmke & Greenwell (2019)

Categorical– One-hot & dummy encoding

Due to **perfect collinearity** which causes problems with some predictive modeling algorithms (e.g., ordinary linear regression and neural networks) **dummy encoding is a better option.**

One-hot/dummy encoding can **significantly increase the dimensionality** of our data. If you have a data set with many categorical variables with many unique levels, you may want to utilize label encoding or some other alternative.

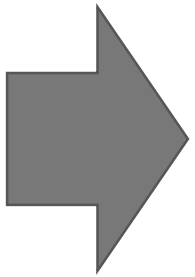


Source: Boehmke & Greenwell (2019)

Categorical – Label encoding

Pure numeric conversion of the levels of a categorical variable that makes sense when dealing with naturally ordered factors, e.g., related to quality assessment.

```
# Original categories
count(ames_train, Overall_Qual)
## # A tibble: 10 x 2
##    Overall_Qual       n
##    <fct>          <int>
##  1 Very_Poor          4
##  2 Poor               9
##  3 Fair              27
##  4 Below_Average    166
##  5 Average          565
##  6 Above_Average    513
##  7 Good             438
##  8 Very_Good        231
##  9 Excellent         77
## 10 Very_Excellent    23
```



```
## # A tibble: 10 x 2
##    Overall_Qual     n
##           <dbl> <int>
##  1            1     4
##  2            2     9
##  3            3    27
##  4            4   166
##  5            5   565
##  6            6   513
##  7            7   438
##  8            8   231
##  9            9    77
## 10           10    23
```

Source: Boehmke & Greenwell (2019)

Categorical – Target encoding

Replacing a categorical value with the mean (regression) or proportion (classification) of the target variable.

Neighborhood	Avg Sale_Price
North_Ames	144792.9
College_Creek	199591.6
Old_Town	123138.4
Edwards	131109.4
Somerset	227379.6
Northridge_Heights	323289.5
Gilbert	192162.9
Sawyer	136320.4
Northwest_Ames	187328.2
Sawyer_West	188644.6

Source: Boehmke & Greenwell (2019)

Categorical – Proportion encoding

An alternative is to change the feature value to represent the proportion a particular level represents for a given feature.

Neighborhood	Proportion
North_Ames	0.1441792
College_Creek	0.0910862
Old_Town	0.0832927
Edwards	0.0686800
Somerset	0.0623478
Northridge_Heights	0.0560156
Gilbert	0.0565027
Sawyer	0.0496834
Northwest_Ames	0.0467608
Sawyer_West	0.0414028

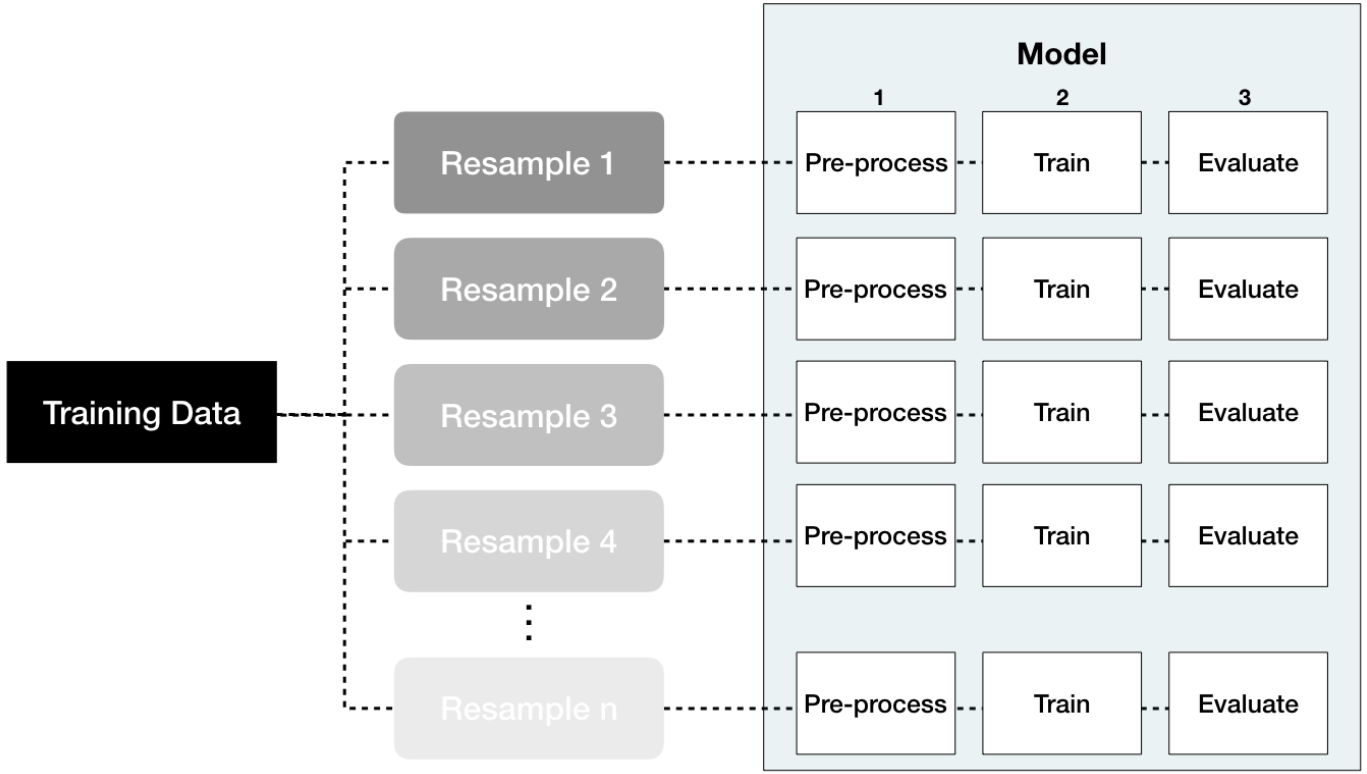
Source: Boehmke & Greenwell (2019)

Feature engineering implementation

We should think of feature engineering as **creating a blueprint** rather than manually performing each task individually. Such blueprint can be then applied appropriately within the resampling process and **prevent data leakage**.

Typical feature engineering worflow:

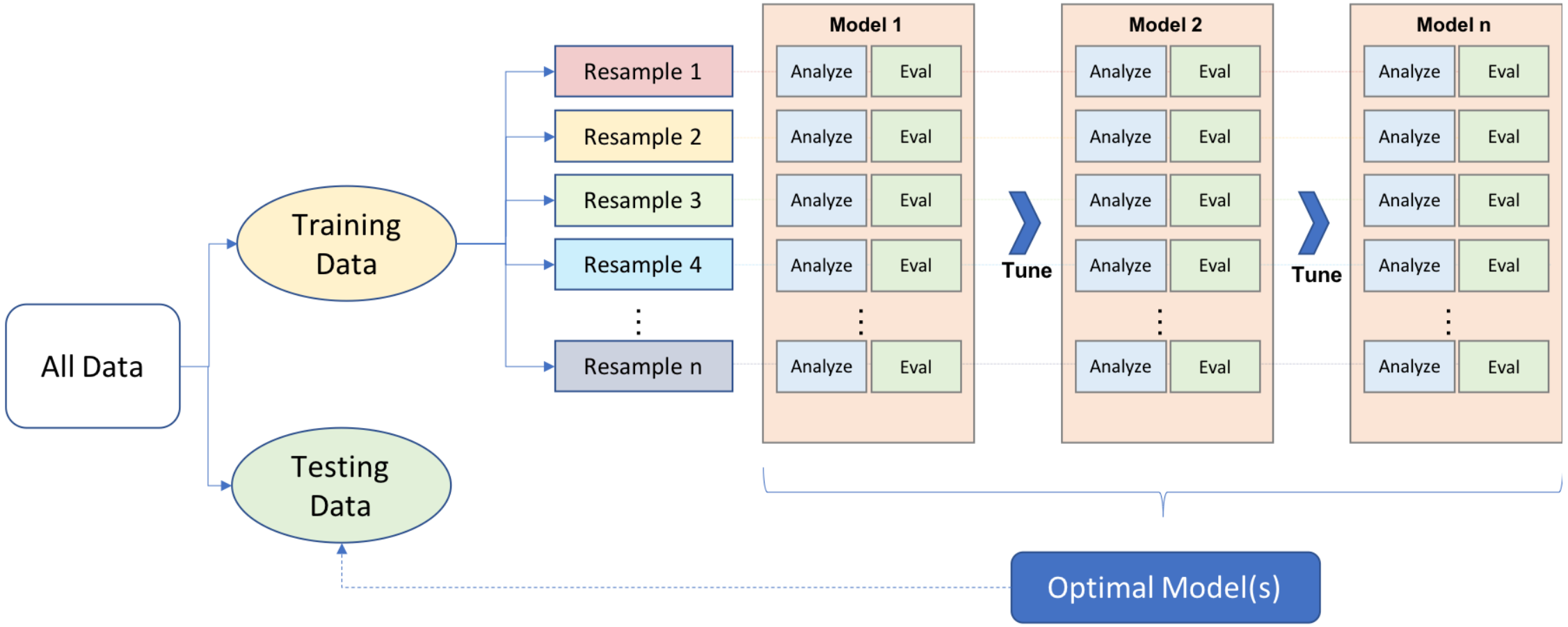
1. Filter out zero or near-zero variance features.
2. Perform imputation if required.
3. Normalize to resolve numeric feature skewness.
4. Standardize (center and scale) numeric features.
5. Perform dimension reduction (e.g., PCA) on numeric features.
6. One-hot or dummy encode categorical features.



Source: Boehmke & Greenwell (2019)

Sampling + resampling

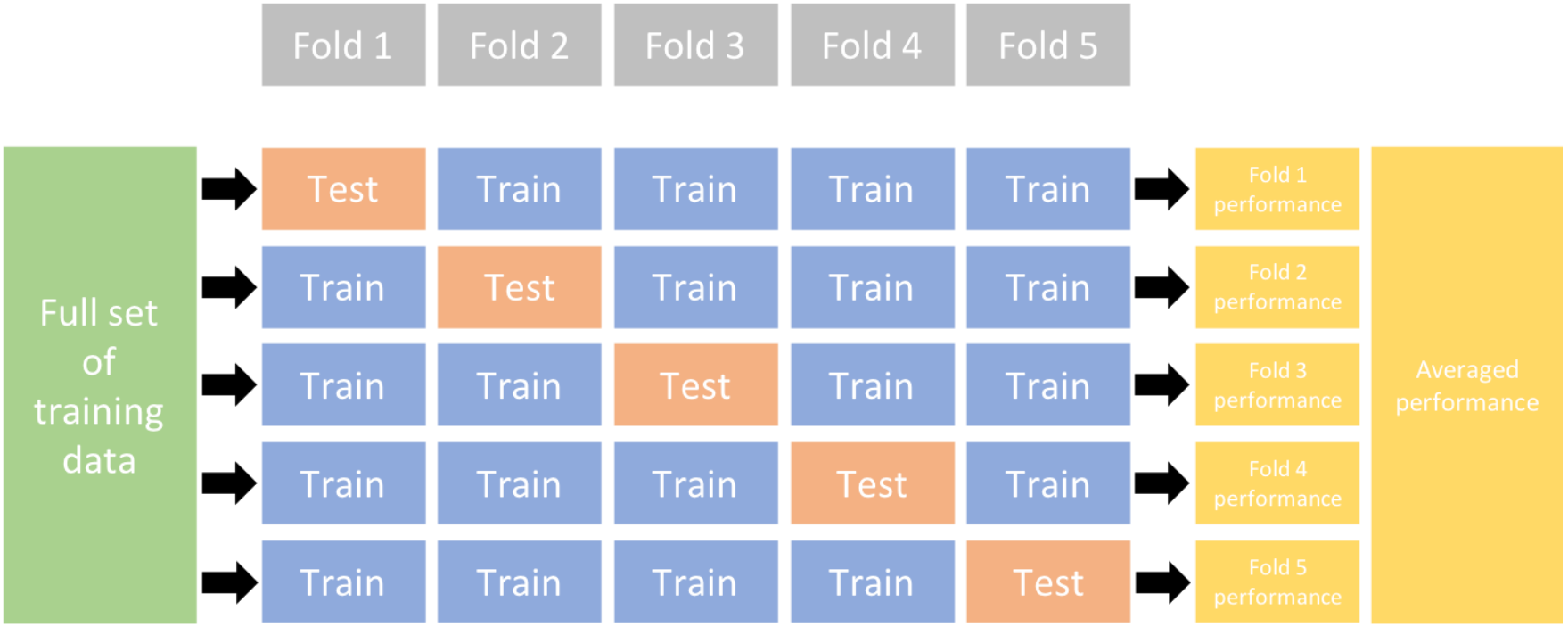
Strategically spending our data wisely on learning and validation procedures.



Source: Boehmke & Greenwell (2019)

Resampling – cross-validation

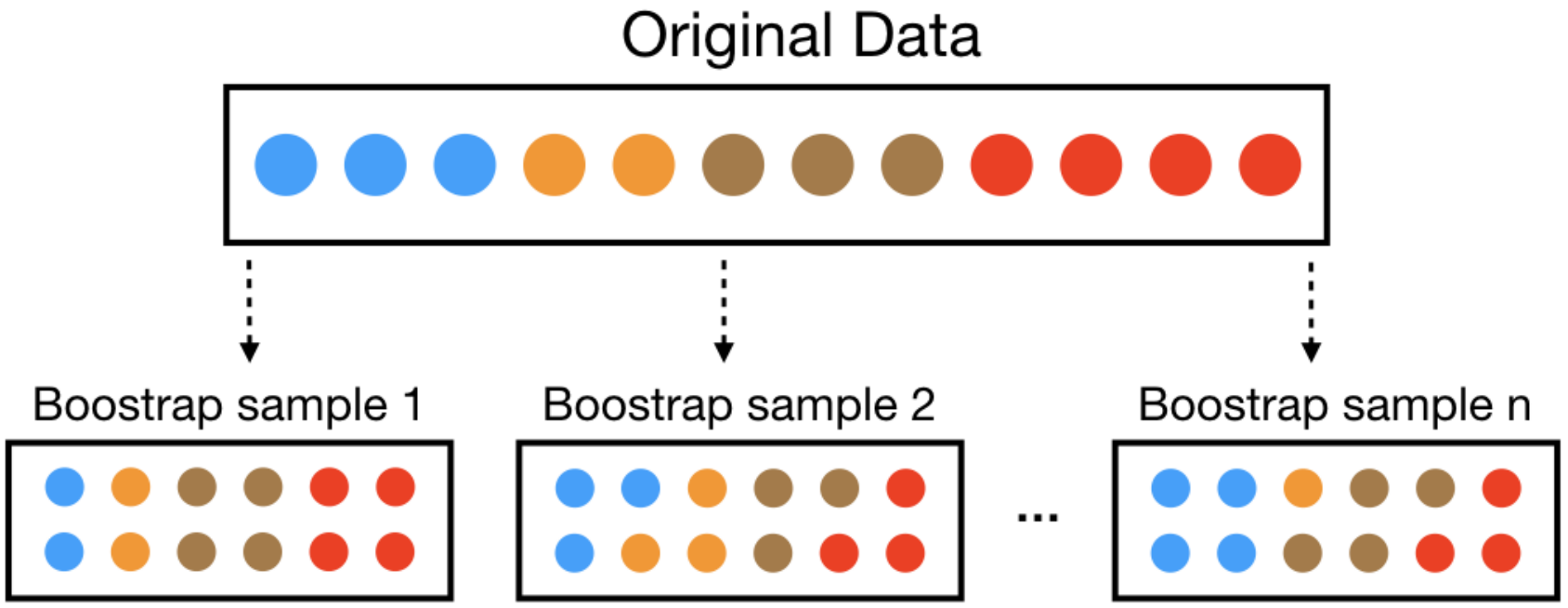
- Provides more accurate measurement of model performance and opportunity to tune model's hyperparameters to prevent overfitting/underfitting.
- Embedded into some machine learning packages.



Source: Boehmke & Greenwell (2019)

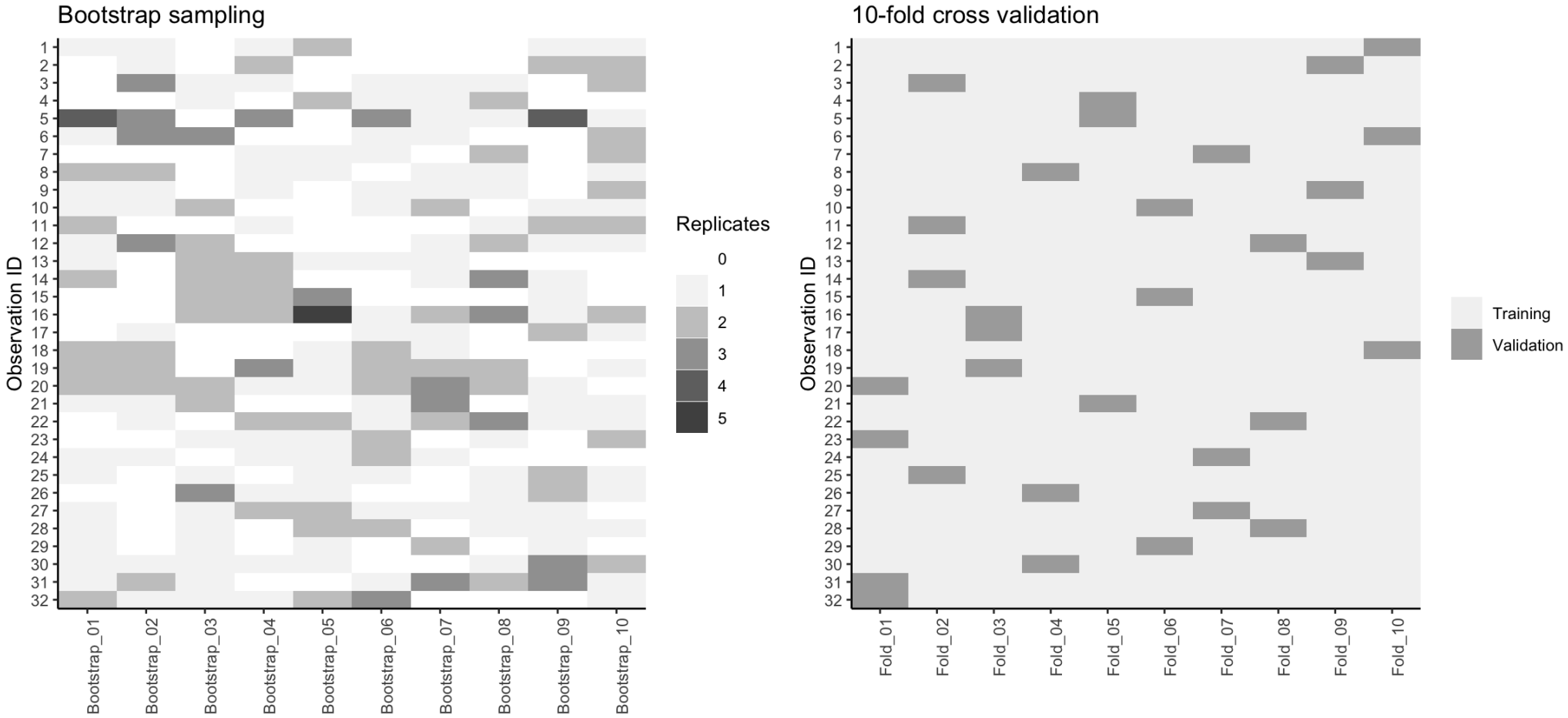
Resampling – bootstrapping

- Provides more accurate measurement of model performance and opportunity to tune model's hyperparameters to provent overfitting/underfitting.
- Embedded into some machine learning packages.



Source: Boehmke & Greenwell (2019)

Resampling – bootstrapping vs. cross-validation



Source: Boehmke & Greenwell (2019)

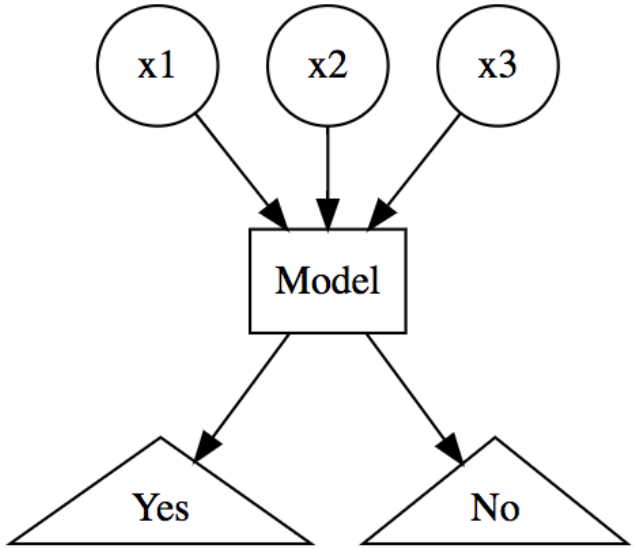
Classification

Motivation

Motivation

Propensity modelling

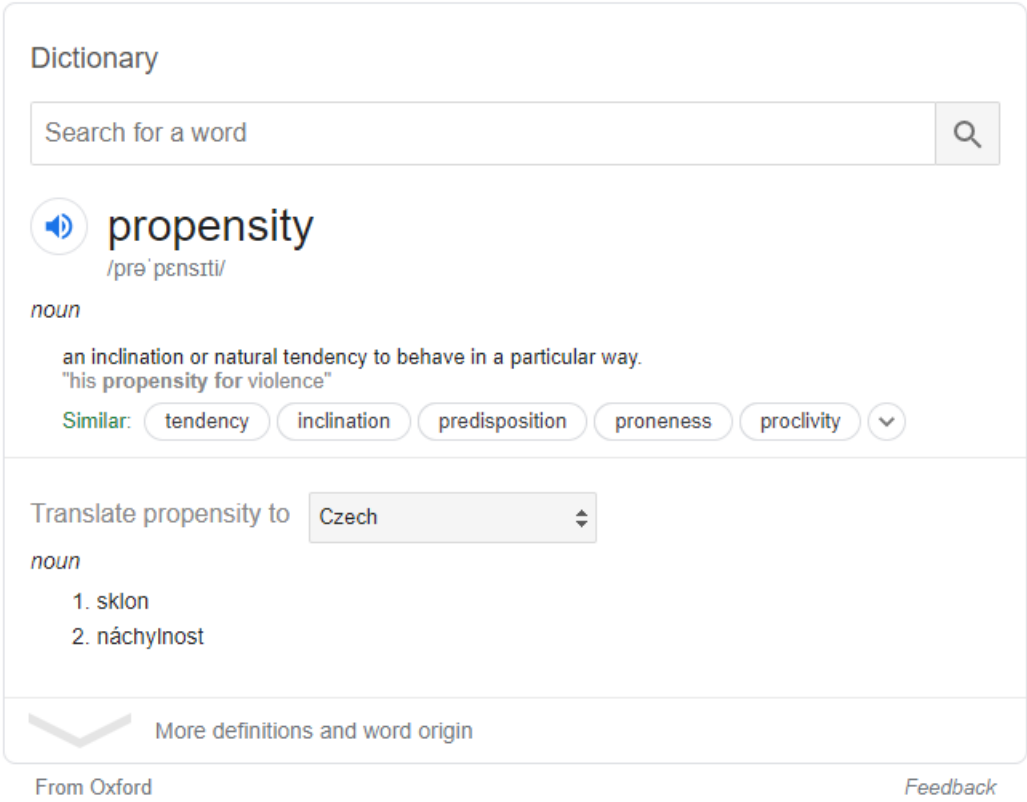
- Propensity modeling is **binary classification** task predicting and event.
- For customers / patients / clients / employees it answers question whether or not some event will happen in the future.



Motivation

Propensity modelling

- How likely is the customer going to buy a product?
- Is he about to leave in the near future?
- What's the chance I can convince him not to leave?
- What is the chance that he stops paying off debts?
- Will he take up the loan?
- How likely he is to have a disease?
- Is it worth to hire him?



Motivation

OCR – Optical character recognition

- Recognition of handwritten or printed text and transformation to text format.
- Example of **multiclass classification**
- Tesseract OCR

OCR Receipt Example

Output
Store #056663515
DEL MAR HTS,RD
SAN DIEGO, CA 92130
(858) 792-7040Register #4 Transaction #571140
Cashier #56661020 8/20/17 5:45PMwellnesst+ with Plenti
Plenti Card#: 31XXXXXXXXXX4553
1 G2 RETRACT BOLD BLK 2PK 1.99 T
SALE 1/1.99, Reg 1/4.69
Discount 2.70-

1 Items Subtotal 1.99
Tax .15

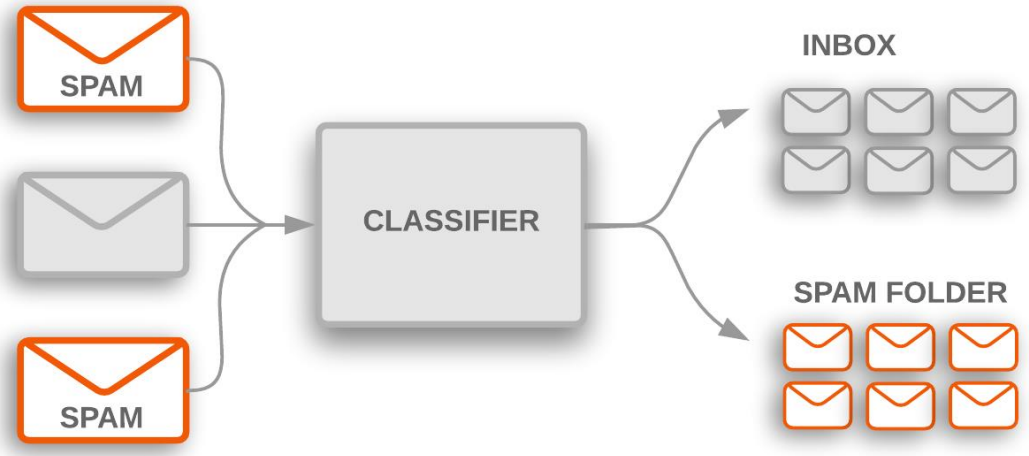
Total 2.14
xMASTER 2.14
MASTER card * #XXXXXXXXXXXX548S
Apo #AA APPROVAL AUTO
Ref # 05639E
Entry Method: Chip

https://www.learnopencv.com/deep-learning-based-text-recognition-ocr-using-tesseract-and-opencv/

Motivation

Spam filter

- Binary classification task
- Can employ NLP (Natural Language Processing) techniques (getting features from text and language)
- **Discussion:** What would you use as features for spam classification? Compare it with https://archive.ics.uci.edu/ml/datasets/spambase

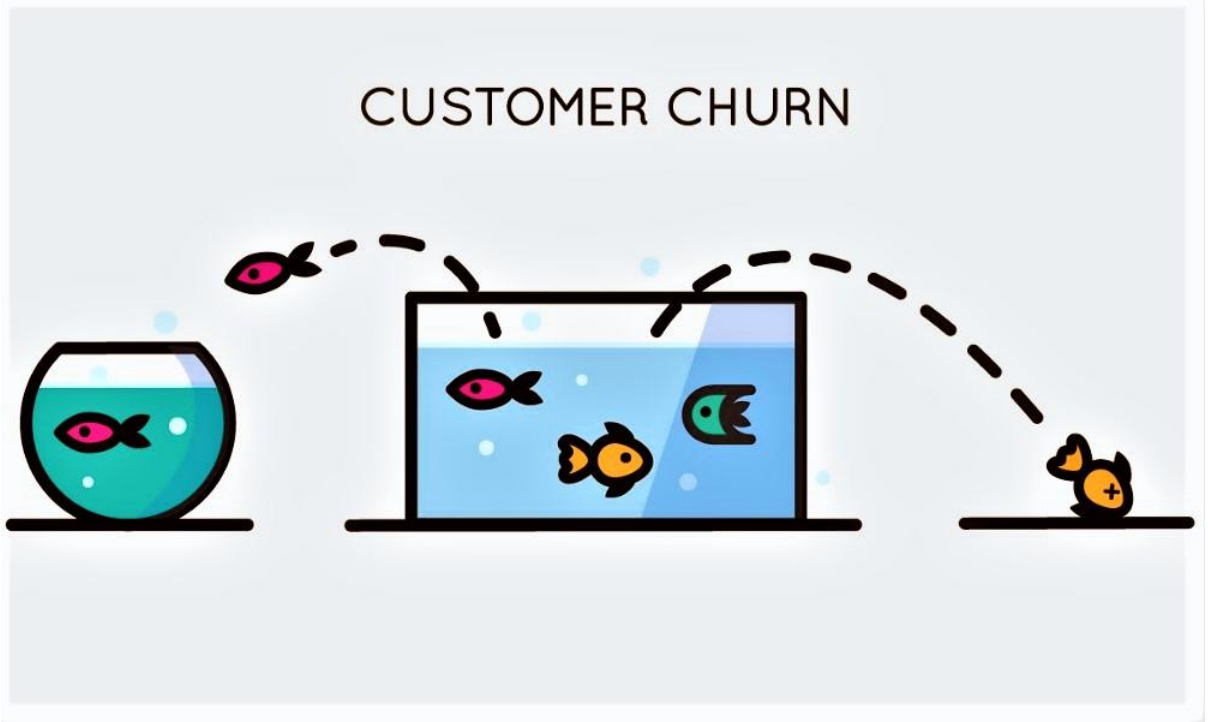


https://developers.google.com/machine-learning/guides/text-classification/images/TextClassificationExample.png

Motivation

Churn

- Based on customer behaviour and attributes, determine whether the customer is probable to leave
- Marketing activities aimed at people classified as churners can then be made
- Discovery of churn drivers



https://www.displayr.com/predict-customer-churn-gradient-boosting/

Motivation

Discussion

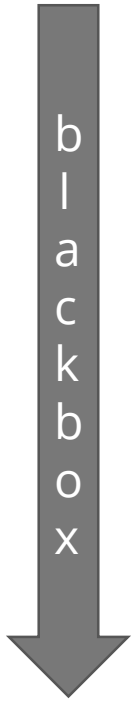
Try to come up with possible uses of classification in your company. Think about what data could be used for the classification. What benefits would it have for your business?

Binary Classification

Algorithms

Binary classification

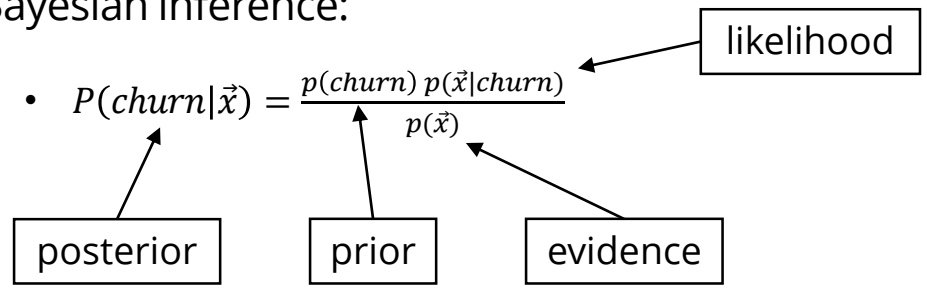
- Supervised learning task
  - The training set is labeled with target value (0/1)
- Algorithms
  - Naïve Bayes
  - K Nearest neighbor (KNN)
  - Logistic regression
  - Decision tree
  - Bagged trees & Random forests
  - Perceptron
  - Support Vector Machines (SVM)
  - Neural Networks
  - Gradient boosting machines & XGBoost
  - Stacked ensembles



> *Everything should be made as simple as possible, but not simpler.*

Naive Bayes

- Naive based algorithm is based on Bayesian probability.
- Probability, conditional probability:
  - $P(churn) = 0.1$ (prior)
  - conditional probability $P(churn \mid x_1) = \frac{P(churn)}{P(x_1)}$, where x_1 is e.g. gender
    - $P(churn|x_1 = male) = 0.2$
    - $P(churn|x_1 = female) = 0.05$
  - conditional probability $P(churn \mid \vec{x})$ - $\vec{x}$ is a vector of feature values
    - When number of feature is high, we would need a very large dataset spanning over all possible combinations (denominator in the conditional probability)
  - Bayesian inference:
    - $P(churn|\vec{x}) = \frac{p(churn)\, p(\vec{x}|churn)}{p(\vec{x})}$



Naive Bayes

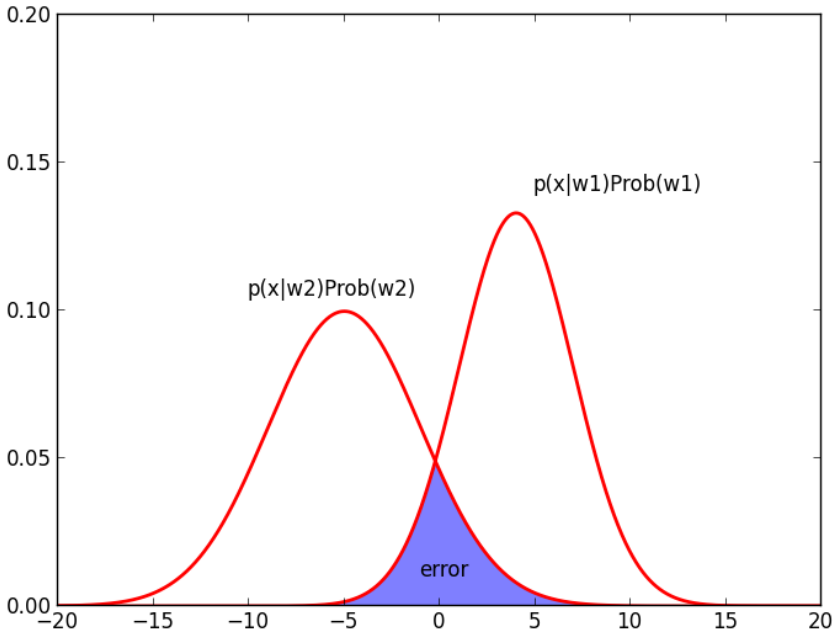
- $P(churn|\vec{x}) = \frac{p(churn)\, p(\vec{x}|churn)}{p(\vec{x})}$
  - The results do not depends on p(\vec)
  - Naivity -> assume conditional mutual **independence** of the features:

$$
\begin{aligned}
p(C_k \mid x_1, \ldots, x_n) &\propto p(C_k, x_1, \ldots, x_n) \\
&\propto p(C_k)\, p(x_1 \mid C_k)\, p(x_2 \mid C_k)\, p(x_3 \mid C_k) \cdots \\
&\propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k),
\end{aligned}
$$

- Building a classifier:

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

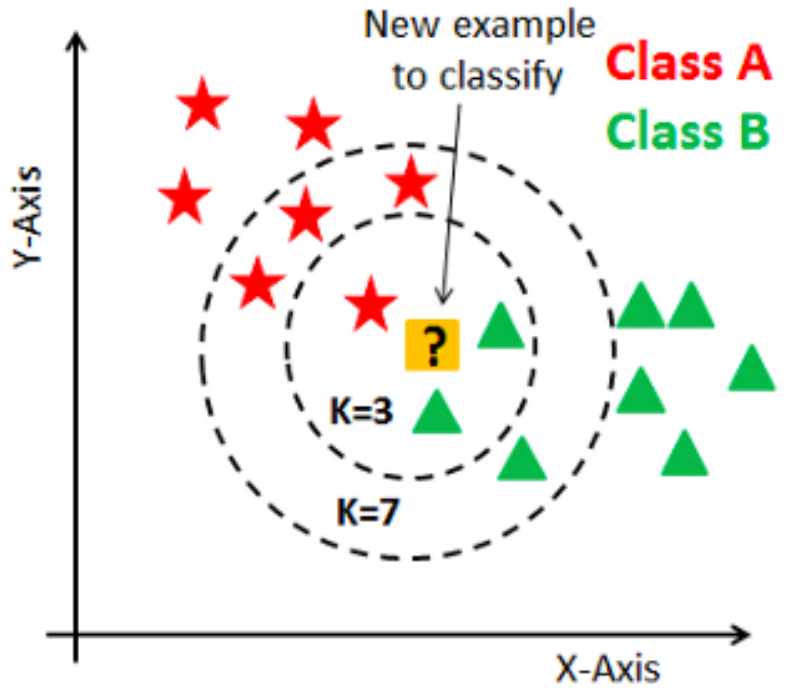
- Can be also used for multi-class classification



https://www.projectrhea.org/rhea/images/4/49/Error.png

K-Nearest Neighbour

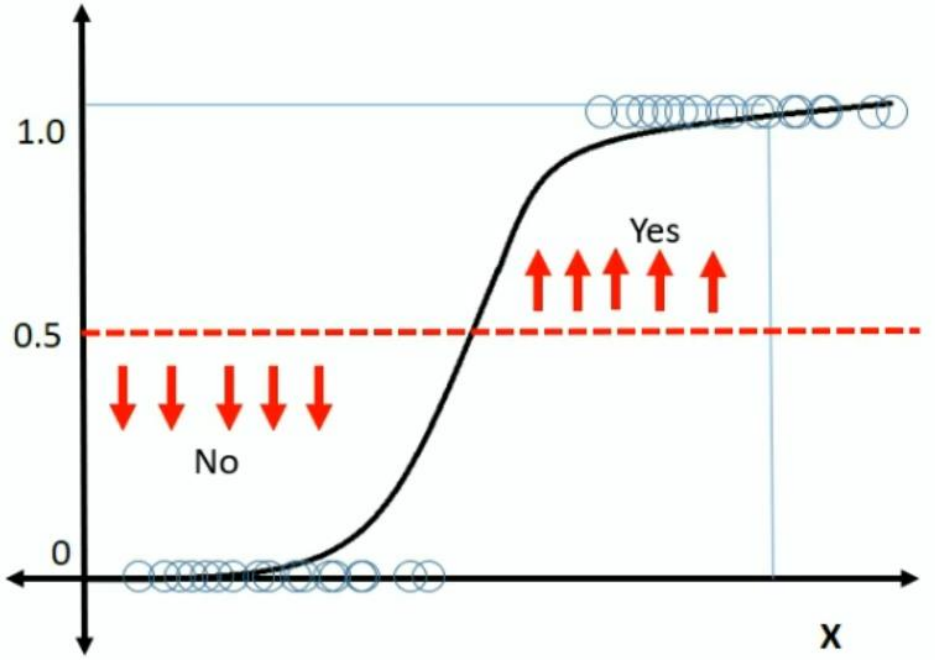
- Determined the class of new data point based on labels (classes) of k nearest neighbours.
- Determining k is highly data-dependent
  - Choosing high k results in better noise handling, but makes the boundaries less distinct
  - Low k makes the results more affected by noise
- Distance can be arbitrary, from standard one like Euclidean to user custom defined
- Usable also for multi-class classification



https://miro.medium.com/max/338/1*rLz1WJep3X8HMJZbXsvrXg.png

Logistic regression

- "Linear model" for classification:
  - $y = b_0 + b_1 x_1 + \ ... + b_n x_n$, where b_0 is called intercept, b_i are coefficients and x_i are features
- Seeks maximum likelihood estimate of **logistic function** (range 0-1):
  - $p = \frac{1}{1+e^{-(b_0+b_1 x_1+\cdots+b_n x_n)}}$
- Mathematical foundation and derivation e.g. on https://dorianbrown.dev/logistic-regression/



https://dorianbrown.dev/assets/images/logreg/prob_threshold.png

Logistic regression

Interpretation

- Results are on logarithmic scale, tramsforation is neccessary to get interpretable output
- Using exp (inverse function to natural logarithm) to get % change in odds in favor of the event and converting odds to probabilities (odds/(1+odds)).

$$\log\left(\frac{p_i}{1-pi}\right) = b_0 + \boldsymbol{b_1}x1 + \cdots b_n x_n + \varepsilon_i$$

- **Regularization** for avoiding overfitting (default in scikit-implementation)

```
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = "binomial",
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6268  -0.8662  -0.6388   1.1490   2.0790
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.989979   1.139951  -3.500 0.000465 ***
## gre          0.002264   0.001094   2.070 0.038465 *
## gpa          0.804038   0.331819   2.423 0.015388 *
## rank2       -0.675443   0.316490  -2.134 0.032829 *
## rank3       -1.340204   0.345306  -3.881 0.000104 ***
## rank4       -1.551464   0.417832  -3.713 0.000205 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

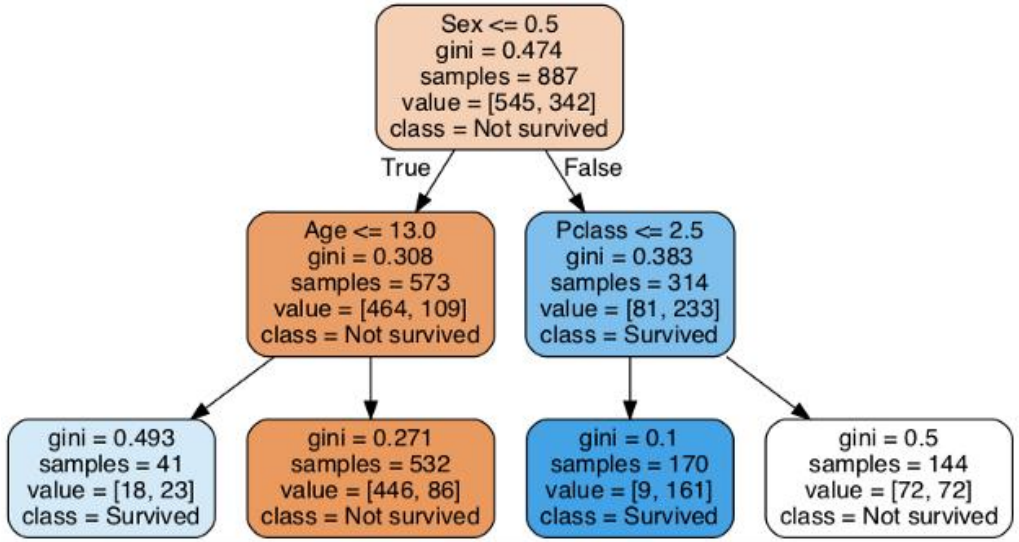

Logistic regression

Predictor selection

- There are three basic types of predictor selection:
  - **Forward selection** begins with a model without predictors and tests whether adding a particular predictor will improve the model. At each step, the predictor is selected to make the most of the model. This continues as long as there are predictors that add improvements to the model.
  - **Backward selection** begins with a model that contains all the predictors, and tests to see if excluding a particular predictor will improve the model.
  - **Stepwise selection** combines forward-backward selection and performs both steps at a time - it first adds the best predictor, and then tests all the predictors in the model to see whether excluding any of them would improve the model. This continues until we reach the stage when the addition or removal of any predictor does not improve the model.

Decision Tree

- Decision about classification is based on traversing a tree structure, where:
  - **internal** node test an attribute
  - each **branch** corresponds to an attribute value
  - Each **leaf** node determines probability of given class



Decision Tree

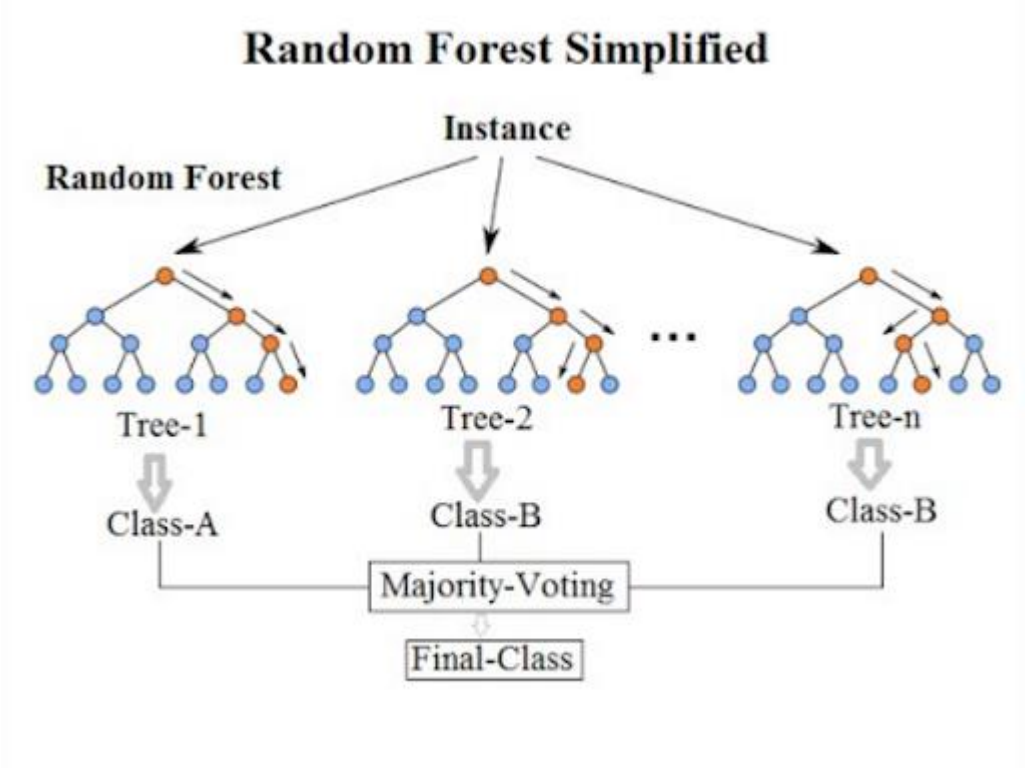
Pros

- Great business interpretability and visualization possibilities
- Requires little data preparation, able to handle both numerical and categorical data
- Cost is logarithmic to the number of training points
- White box model

Cons

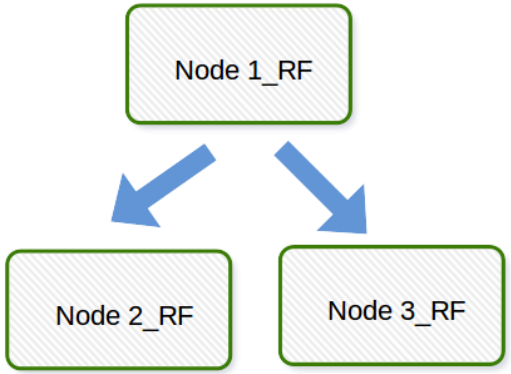
- Overfitting, can be fixed by pruning, limiting tree depth or setting minimal number of datapoints at leaf nodes
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. Ensamble models can be used to fix it.
- Decision tree learners create biased trees if some classes dominate

Random Forests



Random forests--

Only m<M features considered for each node for split



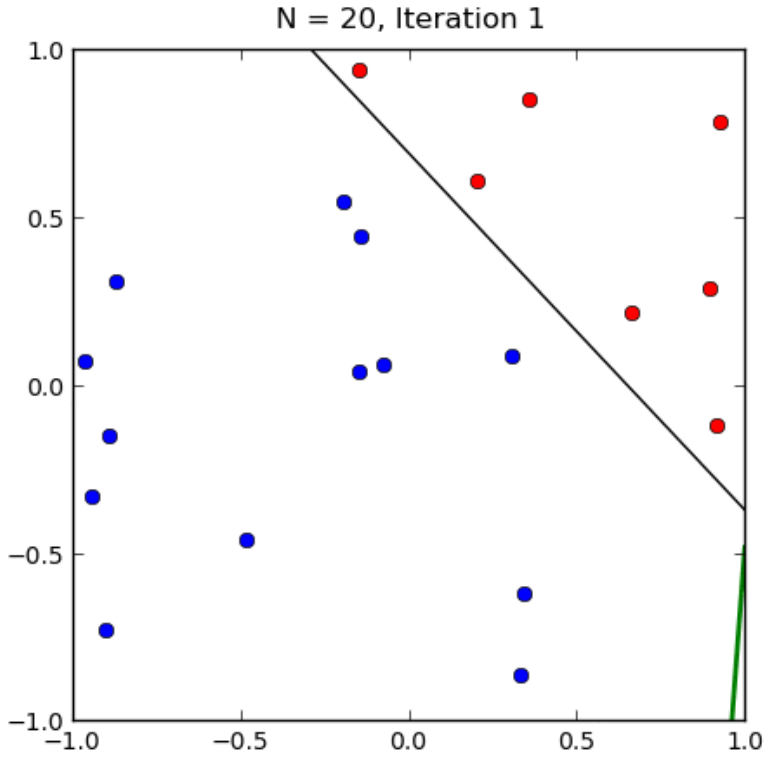
m can be selected via out-of-bag error, but ***m*** = sqrt(***M***) is a good value to start with

Perceptron

- Linear model only for **separable** datasets
- Decision is based whether the data point is "under" or "above" the linear decision boundary

$$\sum_{i=1}^{d} w_i x_i < b \qquad \sum_{i=1}^{d} w_i x_i > b$$

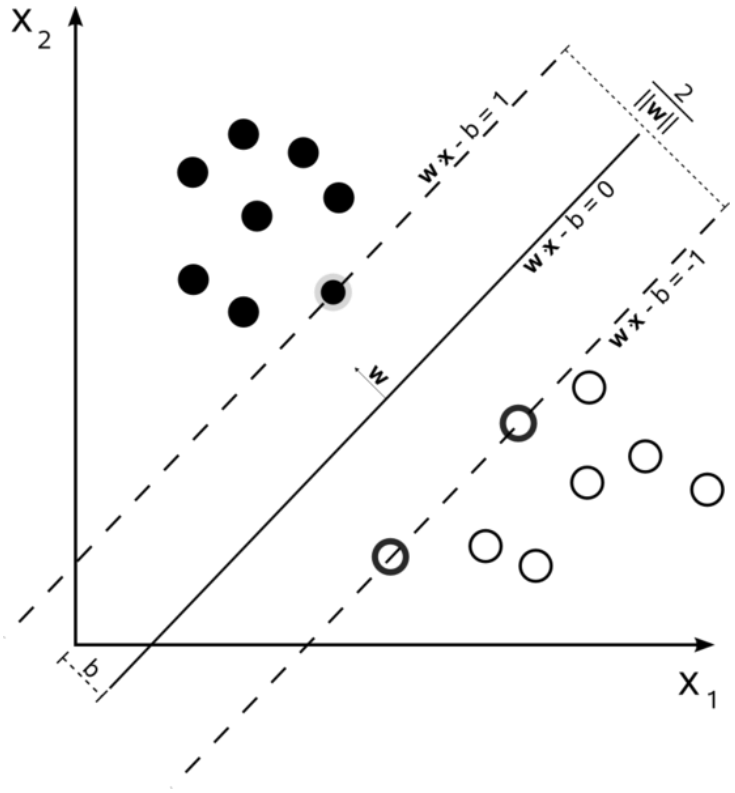
- The goal is to find a set of weights (parameters of the decision boundary line (hyperplane)
- Algorithm (converges when dataset is separable):
    1. Initialize weights randomly
    2. Classify datapoints, for each error adjust the weight
    3. Return final weight vector



https://datasciencelab.wordpress.com/2014/01/10/machine-learning-classics-the-perceptron/

Support Vector Machines

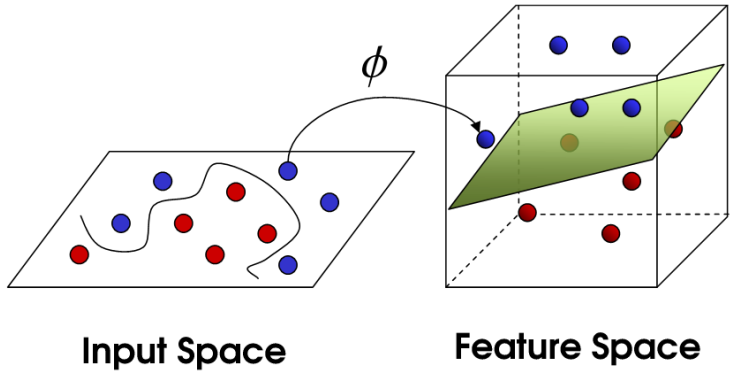
- Improves the perceptron algorithm by **maximizing the margin** between the classes – algorithm also constructs a separating **hyperplane**
- Introduces so called **slack variables** allowing the algorithm to also work on inseparable datasets
- Cheap computation
- It only needs **support vectors** – data points closest to the boundary from each classes to make a decision about an incoming data point

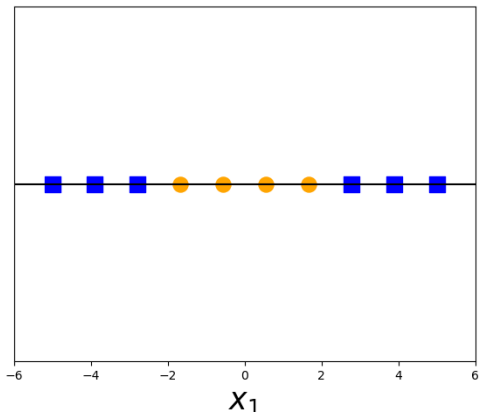


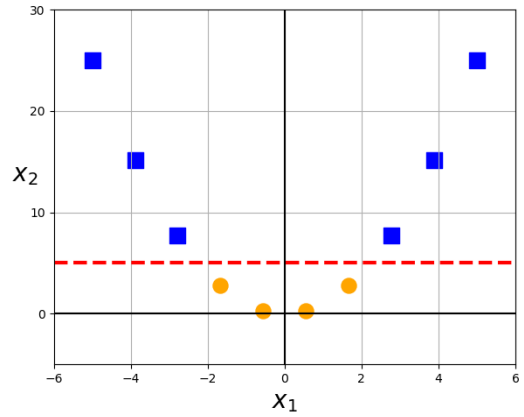
Support Vector Machines

Kernels

- Support Vector Machines (SVNs) can also be used for non-linear decision boundaries by using kernels
- Kernel allows SVNs **non-linear classification** by applying transformation ϕ to the data
  - Polynomial kernels
  - Gaussian
  - ...
- To learn more, see e.g. https://www.jeremyjordan.me/support-vector-machines/



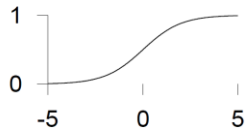




Neural networks

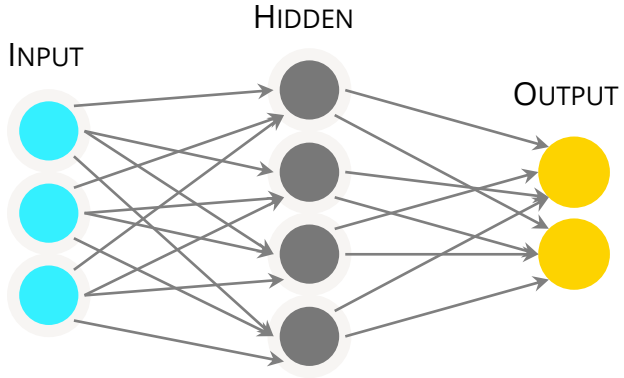
Neuron (Perceptron wrapped in activation function)

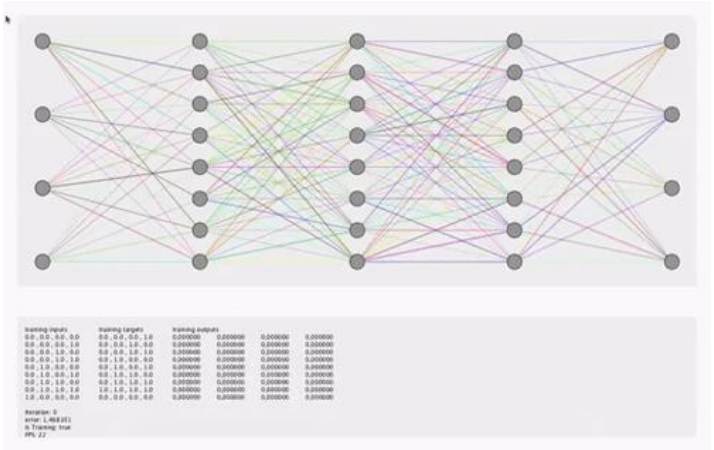
$$answer = f\left(\sum_{i=1}^{no.inputs} w_i x_i + w_0\right)$$



Neural Net

- Neurons at layer i connect to neuron at layer $i + 1$
- Core concept of neural network is to find out the best weights of the neurons with regard to a **loss function**
- Learning is based on feeding data forward and then **backpropagation** (adjusting the weights) after presenting datapoints
- Data hungry – each neuron brings $n + 1$ new parameters
- Can be used for both classification and regression (depends on output layers and loss function



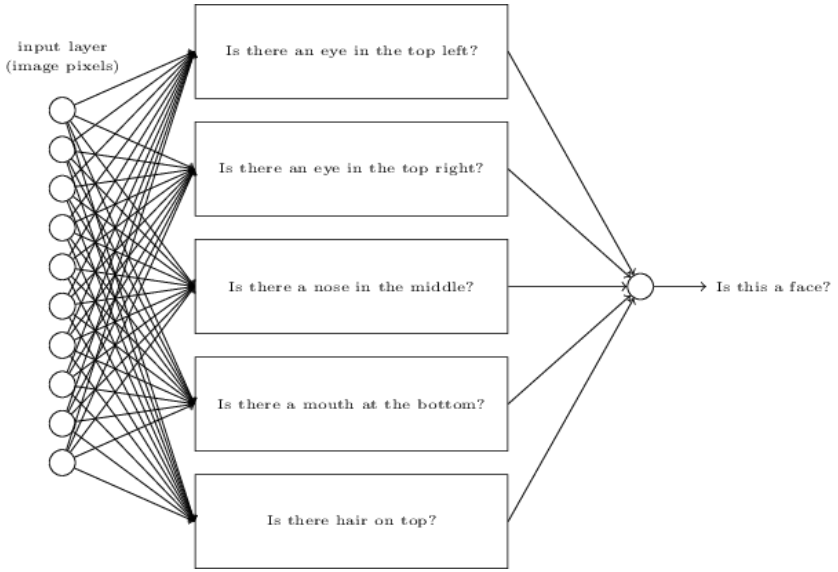


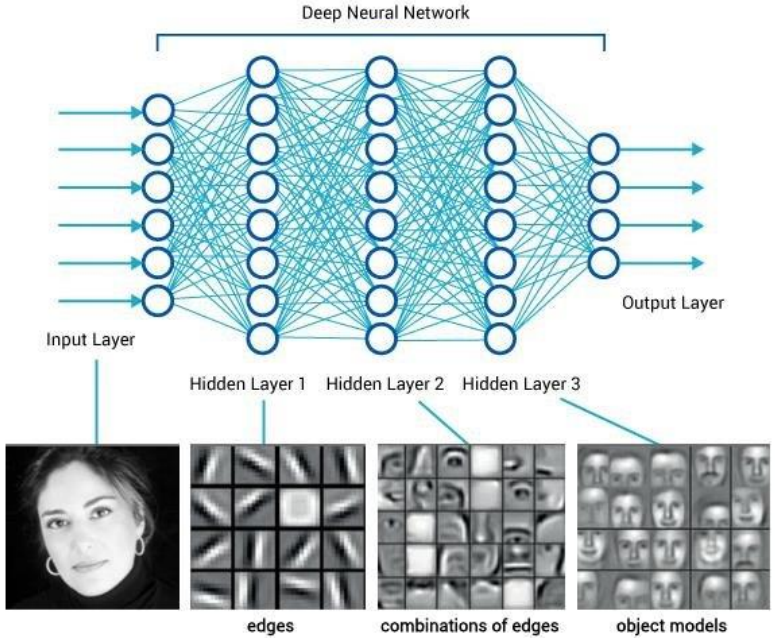
https://www.youtube.com/watch?v=7X1QjM28gf4&feature=youtu.be

Neural networks

Deep learning

- One of the most appealing thing about neural network learning is that the algorithm is capable of **abstraction** – each layers brings a new layer of abstraction
- Black box – it's not easy to interpret the output
- Libraries / packages (Tensorflow, Kears, Theano, ...)

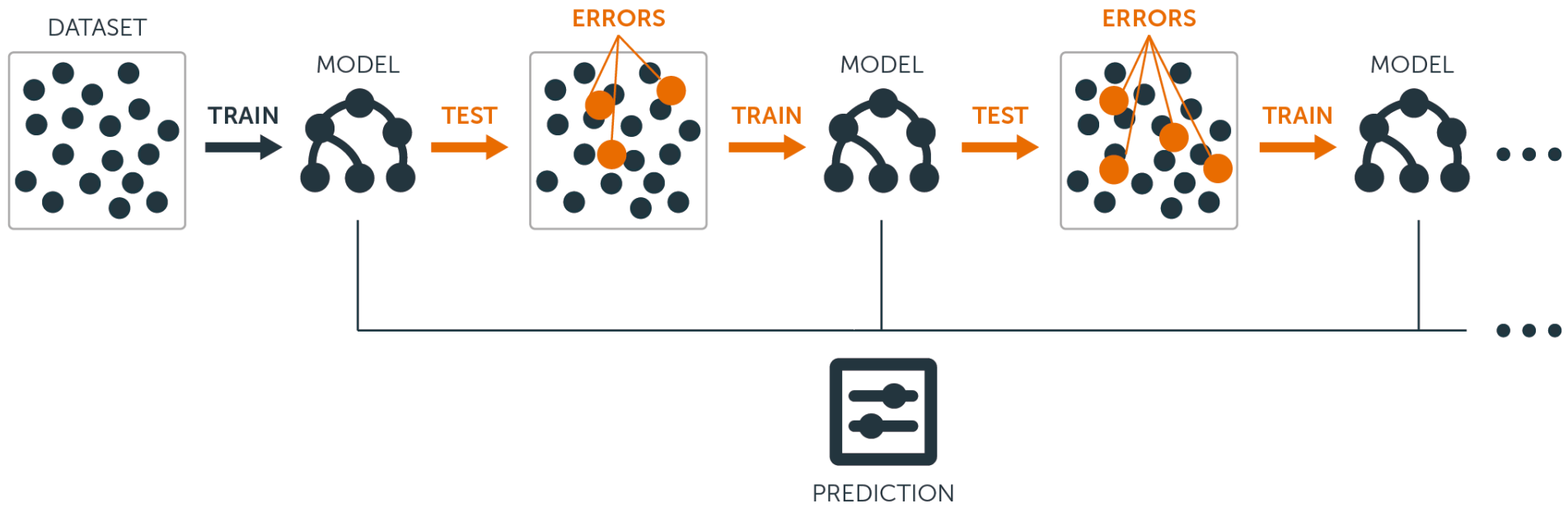




https://www.researchgate.net/figure/Layers-and-their-abstraction-in-deep-learning-Image-recognition-as-measured-by-ImageNet_fig17_326531654

Gradient boosting machines & XGBoost

- Unlike bagging that had each model run independently and then aggregate their outputs.
- Boosting is all about “teamwork” - each model that runs dictates (by its specific errors) what observations and thus also features the next model will focus on.



Stacked ensembles

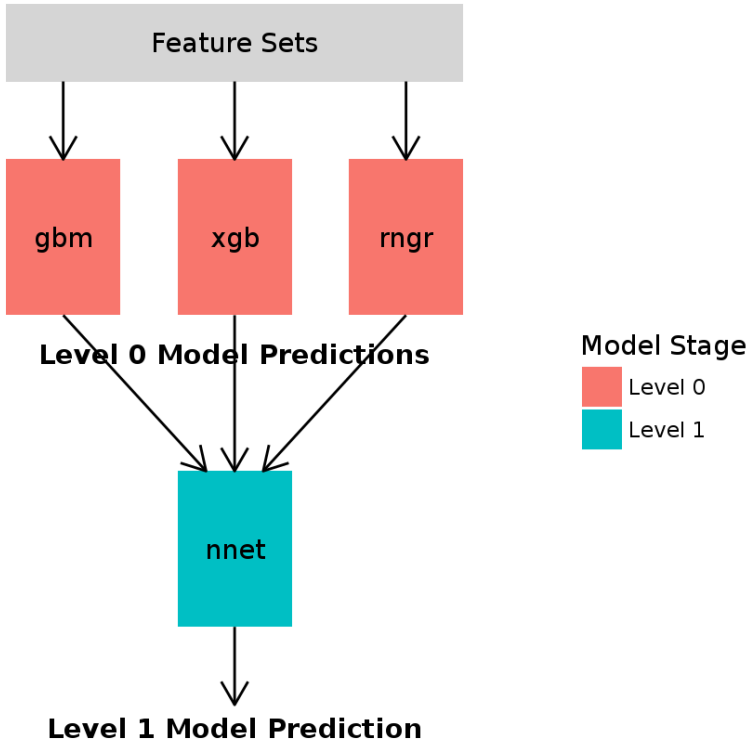
Wisdom of crowds & combination of strong learners.

Examples of the wisdom of crowds

At a 1906 country fair in Plymouth, 800 people participated in a contest to estimate the weight of a slaughtered and dressed ox.

Francis Galton observed that the median guess, 1207 pounds, was accurate to within 1% of the true weight of 1,198 pounds.

Just over a century later 40 guesses at Oxford's Venturefest came within 11.4% of 2,173 jelly beans!



Classification Evaluation

Confusion matrix, precision and recall

Confusion matrix is a table that is used to describe the performance of a classifier on a test data for which the true values are known.

Precision is the fraction of success events in predicted positive.

		True condition			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$	

F_1 score = $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$

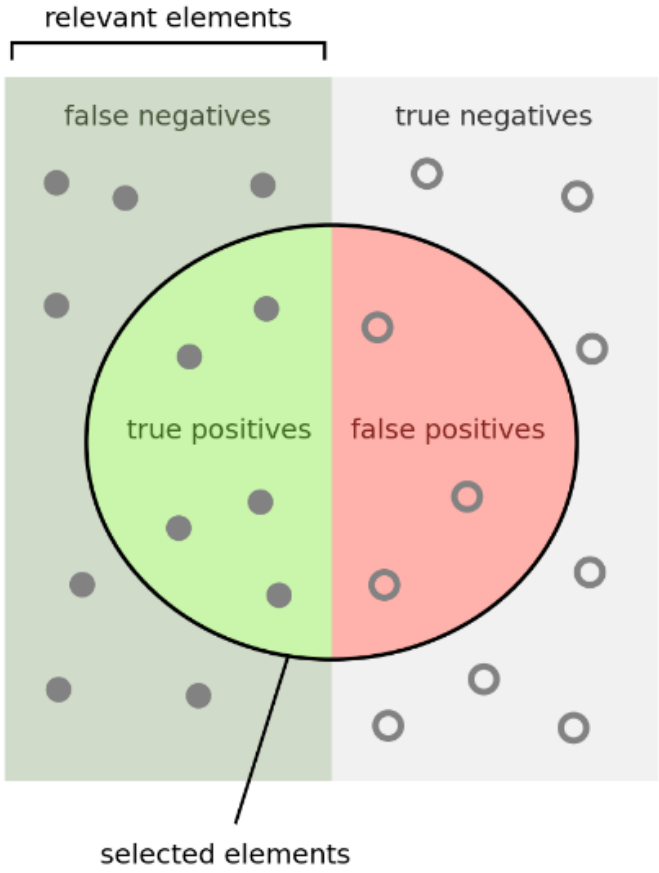
Recall (sensitivity) is the fraction of the predicted positive in success events.

Accuracy is number of correct predictions divided by whole population

Accuracy, Precision, Recall

- **Accuracy** = Number of correct predictions
  - (tp + tn) / (tp + fp + tn + fn)
- **Precision** = How many selected items are relevant?
  - tp / (tp + fp)
- **Recall** = How many relevant items are selected?
  - tp / (tp + fn)

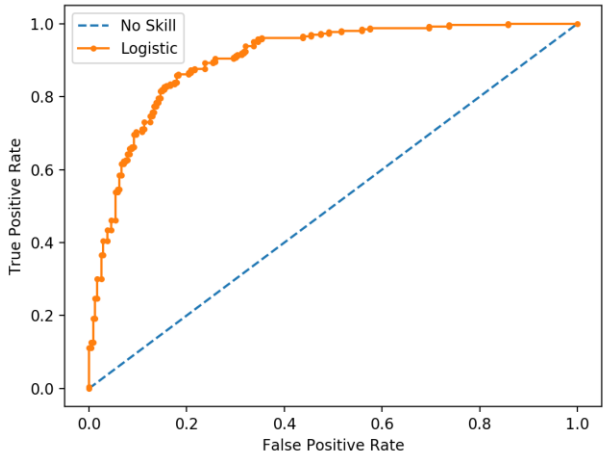
- **Is one number enough?**

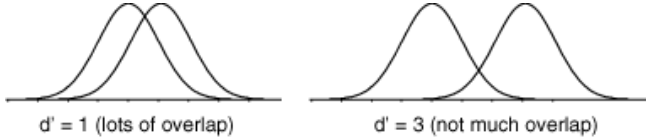


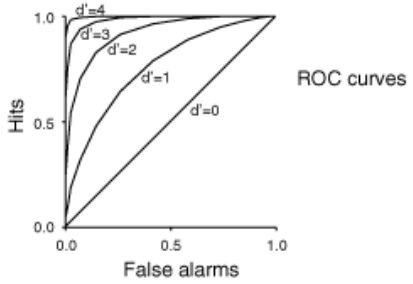
ROC curve & AUC

- Plots how good the model for different decison thresholds.
- Area under the ROC curve (**AUC**) can be used to score how good the model is (e.g. to compare approaches, models, etc.)
- **Balanced datasets!**

AUC	Quality of the model
0.50-0.65	Not usable
0.65-0.70	weak
0.70-0.75	below average
0.75-0.80	average
0.80-0.90	excellent
0.90-1.00	too good to be true





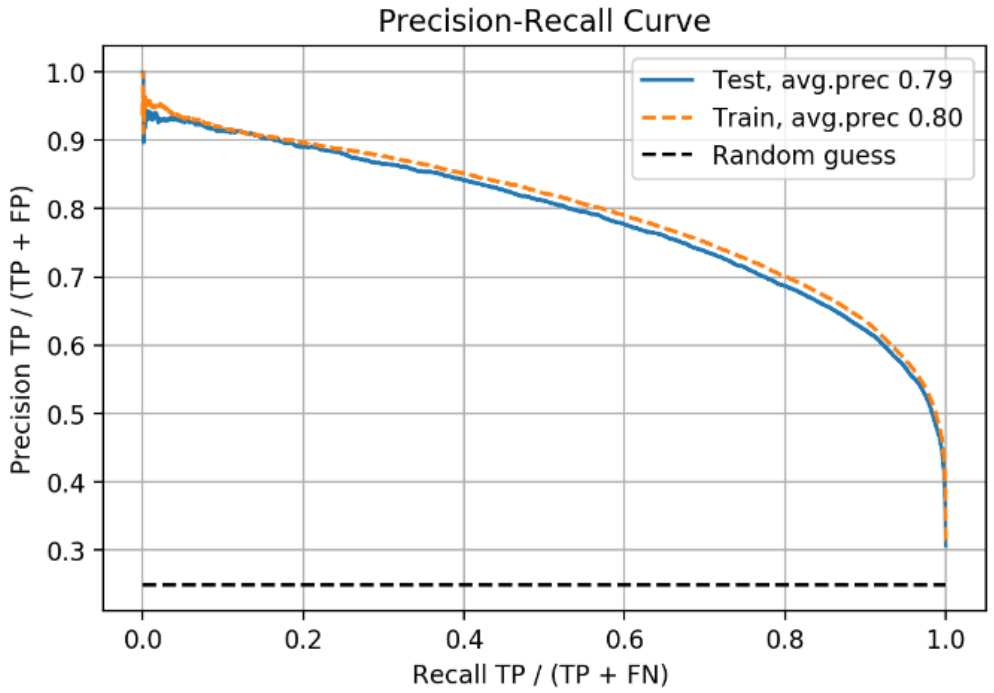


Precision-Recall Curve

- Plots precision vs recall
- Can be summarized with **average precision** (same as AUC in ROC curve) – weighted by recall difference at each threshold

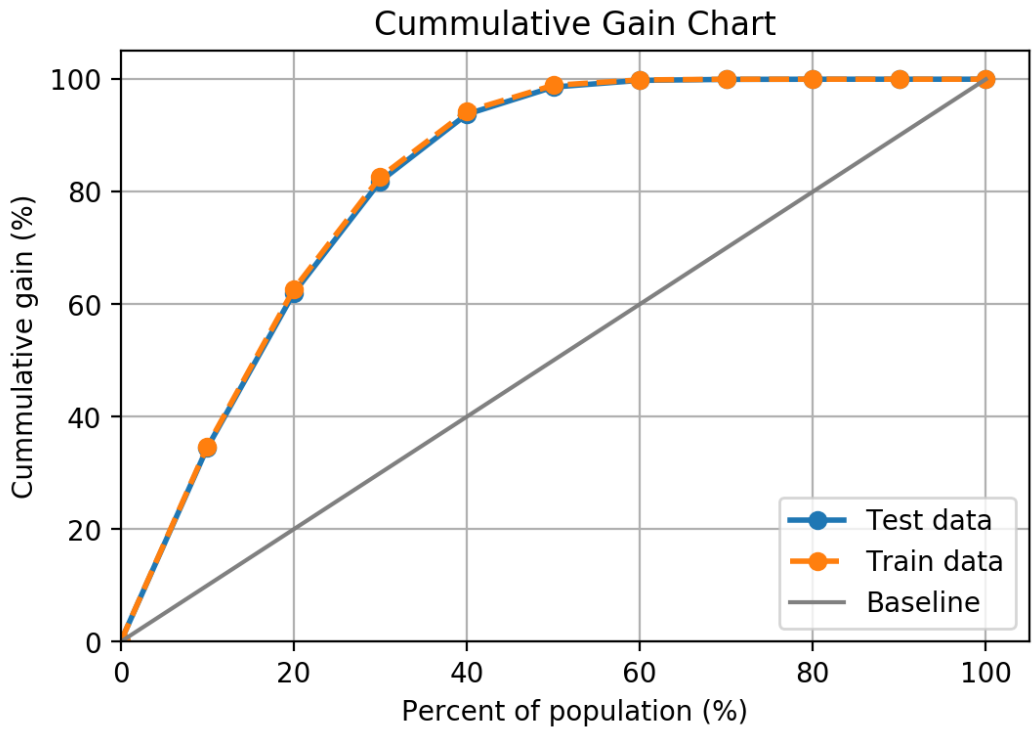
$$\mathrm{AP} = \sum_n (R_n - R_{n-1}) P_n$$

- **Suitable even for imbalanced datasets.**



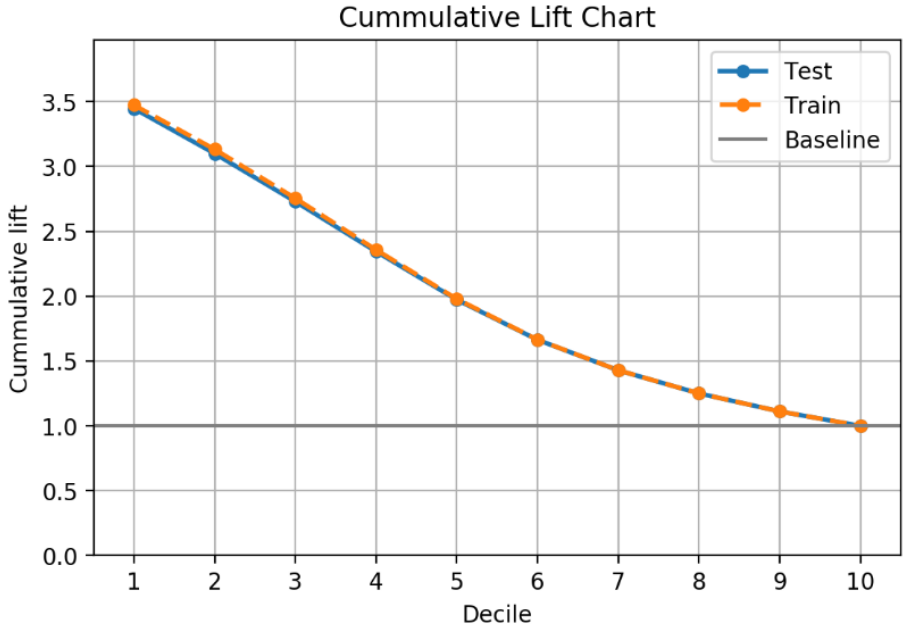
Cummulative Gain

- Great for visualizing response rates and presenting results to business.
- Classifiers output probabilities of classses:
  - Sort data according to probability of class 1
  - Split data into equal bins (deciles in presented plot)
  - Compute the gain as sum of class 1 / number of datapoints in decile for each decile.
- The output is read as follows:
  - By contacting 30% of the people selected by this model, we will get 80% of all the positive responses.



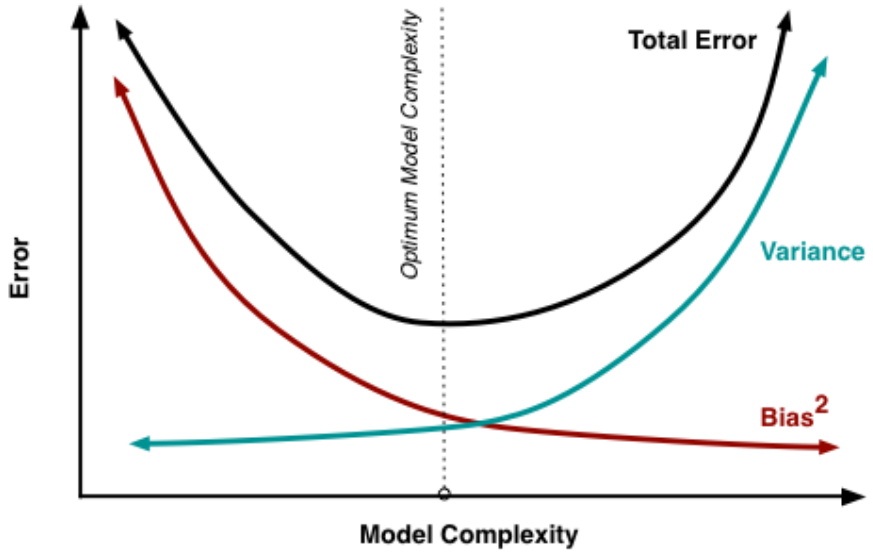
Model evaluation – Lift

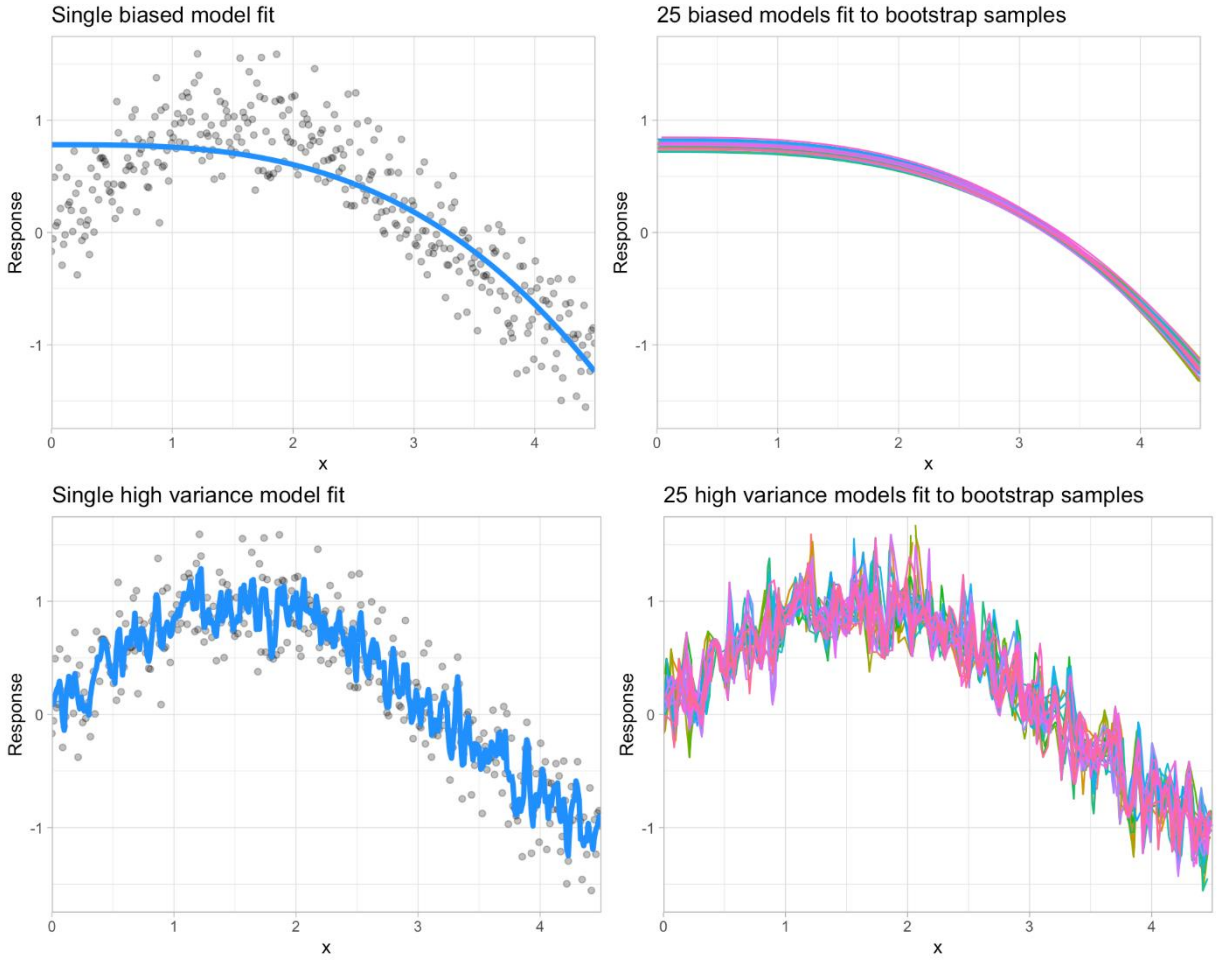
- Lift is a measure of the effectiveness of a predictive model calculated as the **ratio between the results obtained with and without the predictive model.**
- We sort items by descending score and split into deciles.
- Values $\frac{TP\prime}{TP}$ are plotted for each decile.
- Alternatively – we can obtain the datapoints from cummulative gain by dividing the y/x of the datapoints for each decile.
- The output is read as follows:
  At second decile, the model performs three times better than random guess.



Bias variance trade-off

- Prediction errors have two major sources: **bias** and **variance**.
- There is often a **tradeoff between a model's ability to minimize bias and variance**.

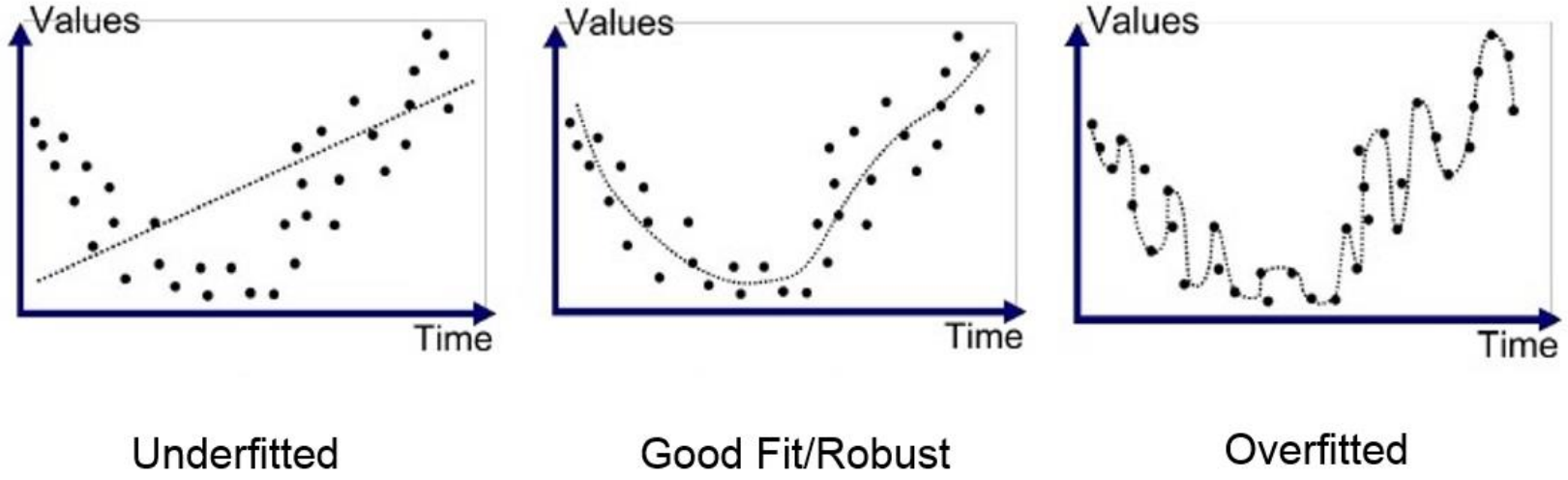




Source: Boehmke & Greenwell (2019)

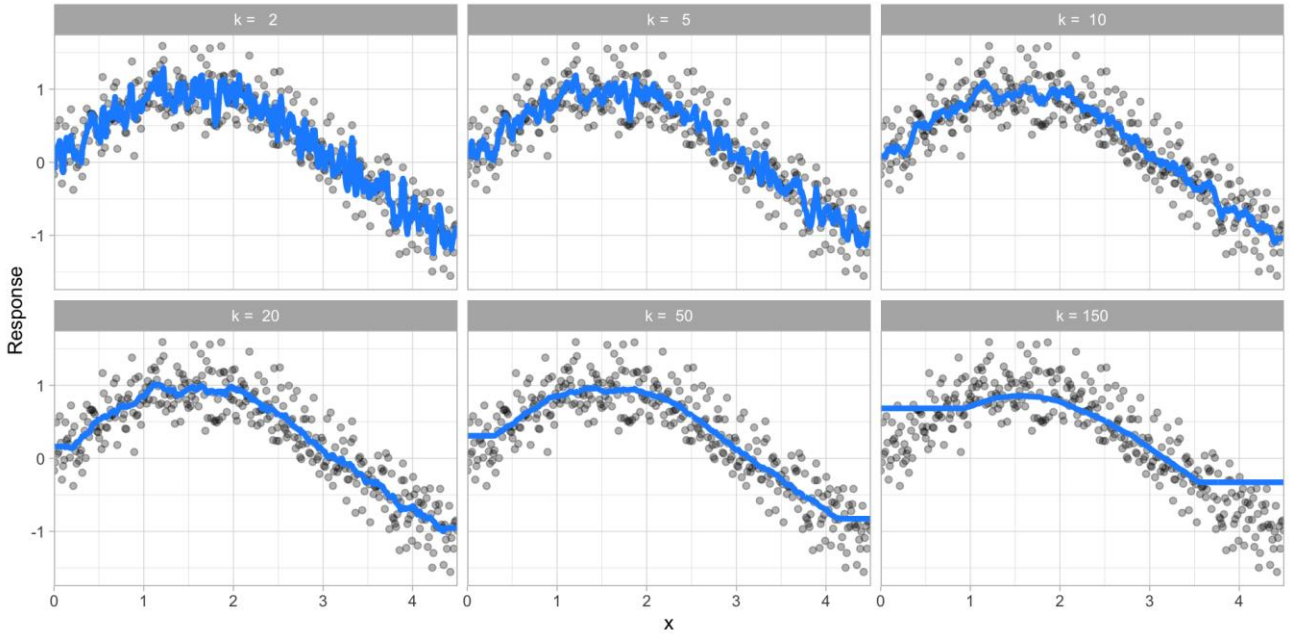
Overfitting vs. underfitting

- **Overfitting** is when model fits too closely to the training data (including the noise) and fails to fit future observations reliably.
- Overfitting happens when model is **too complex/flexible** or **training dataset is too small**.
- **Underfitting** is when model is too simple/inflexible to capture the patterns hidden in the data.



Dealing with overfitting/underfitting

We handle the bias-variance trade-off by **controlling the complexity of the machine learning models** through **tuning of their hyperparameters**.

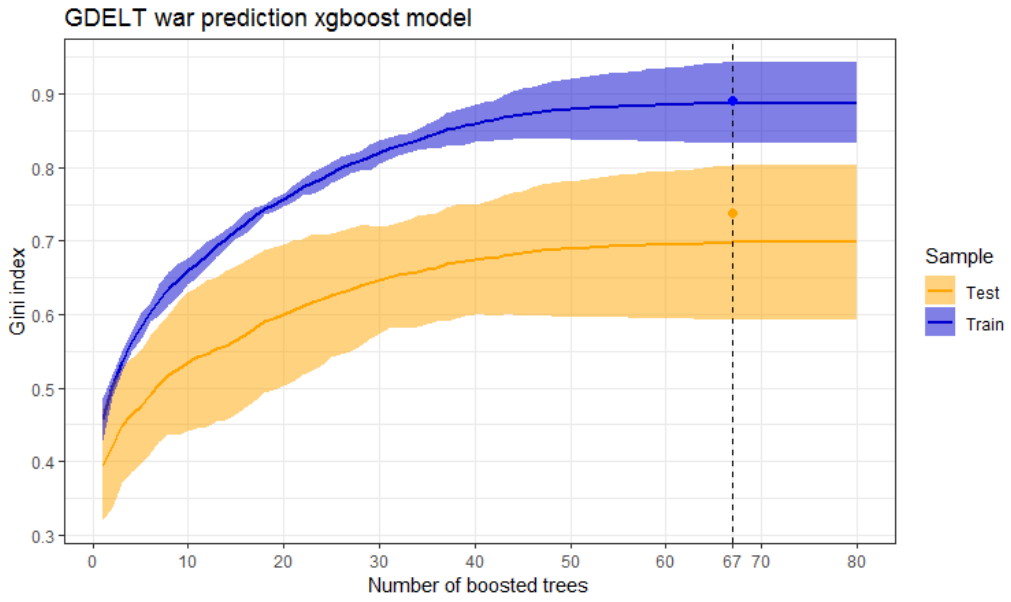


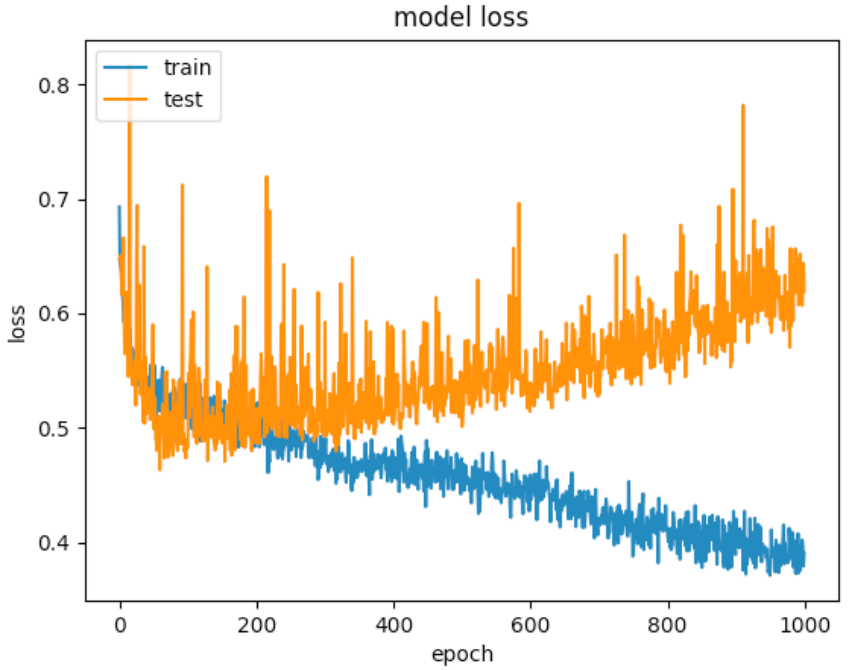
k-nearest neighbor model performance with differing values for k.

Source: Boehmke & Greenwell (2019)

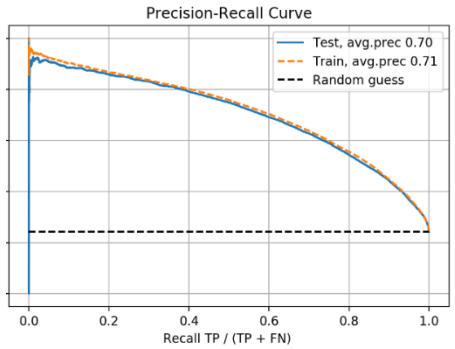
Dealing with overfitting/underfitting (cont.)

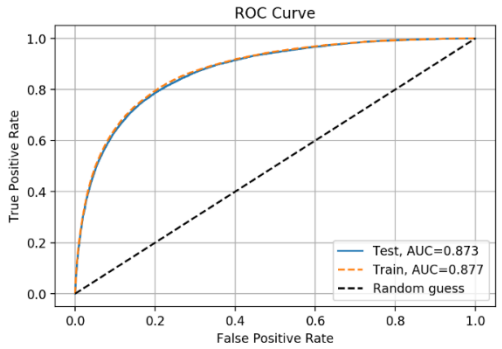
- We can identify overfitting by **comparing model performance on train and test dataset**, or by **cross validation**.
- Splitting data between train and test (80/20 – 70/30). The result should be similar to show that the model is not overfitting the problem

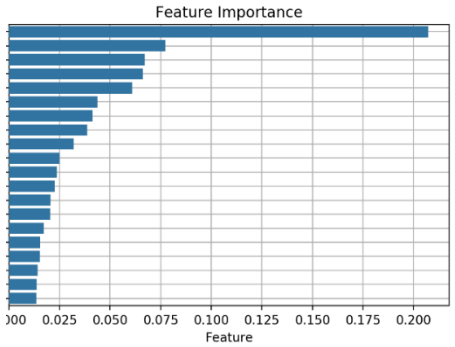


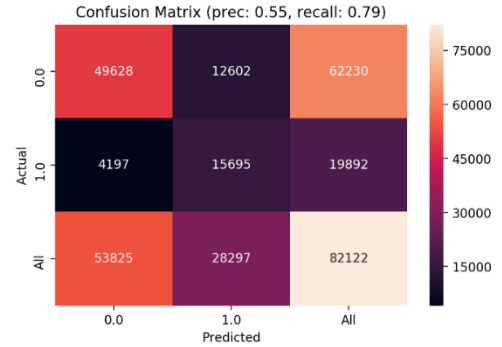


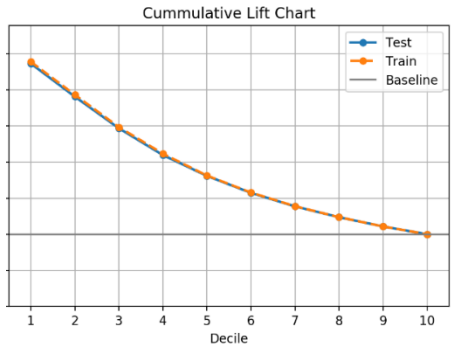
Comparing models together

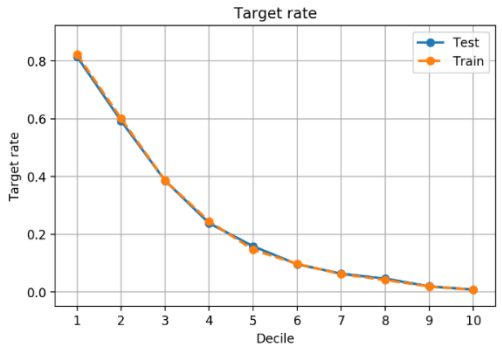


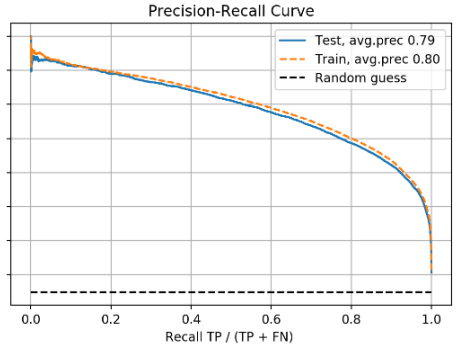


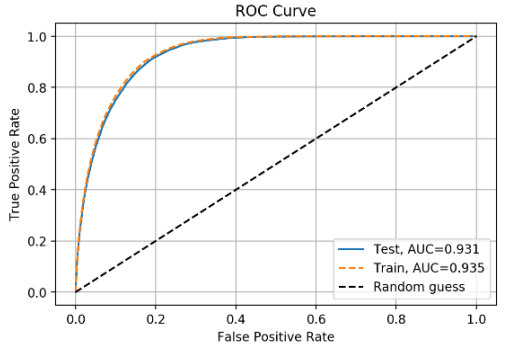


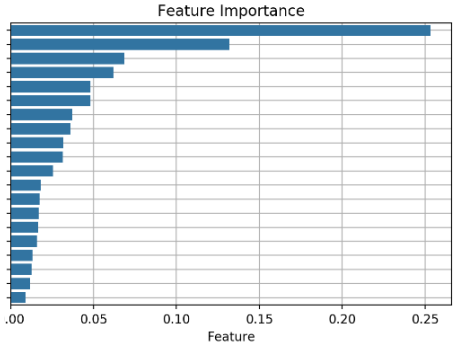


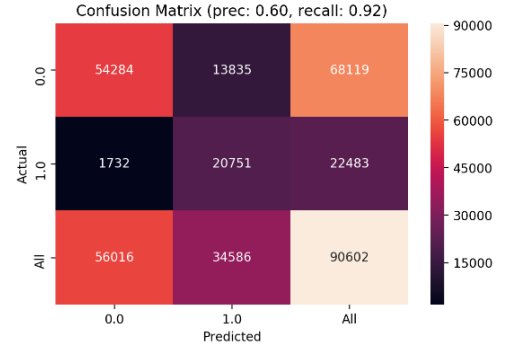


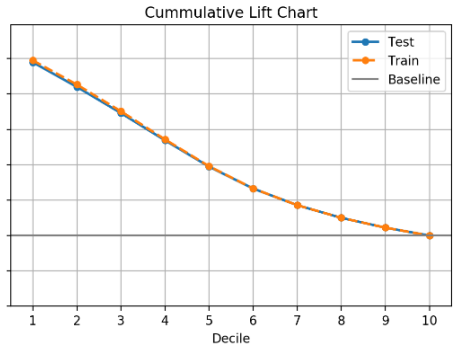


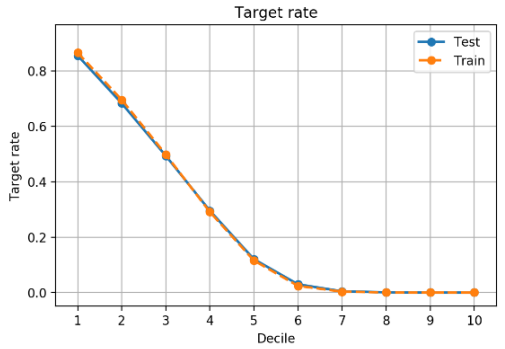


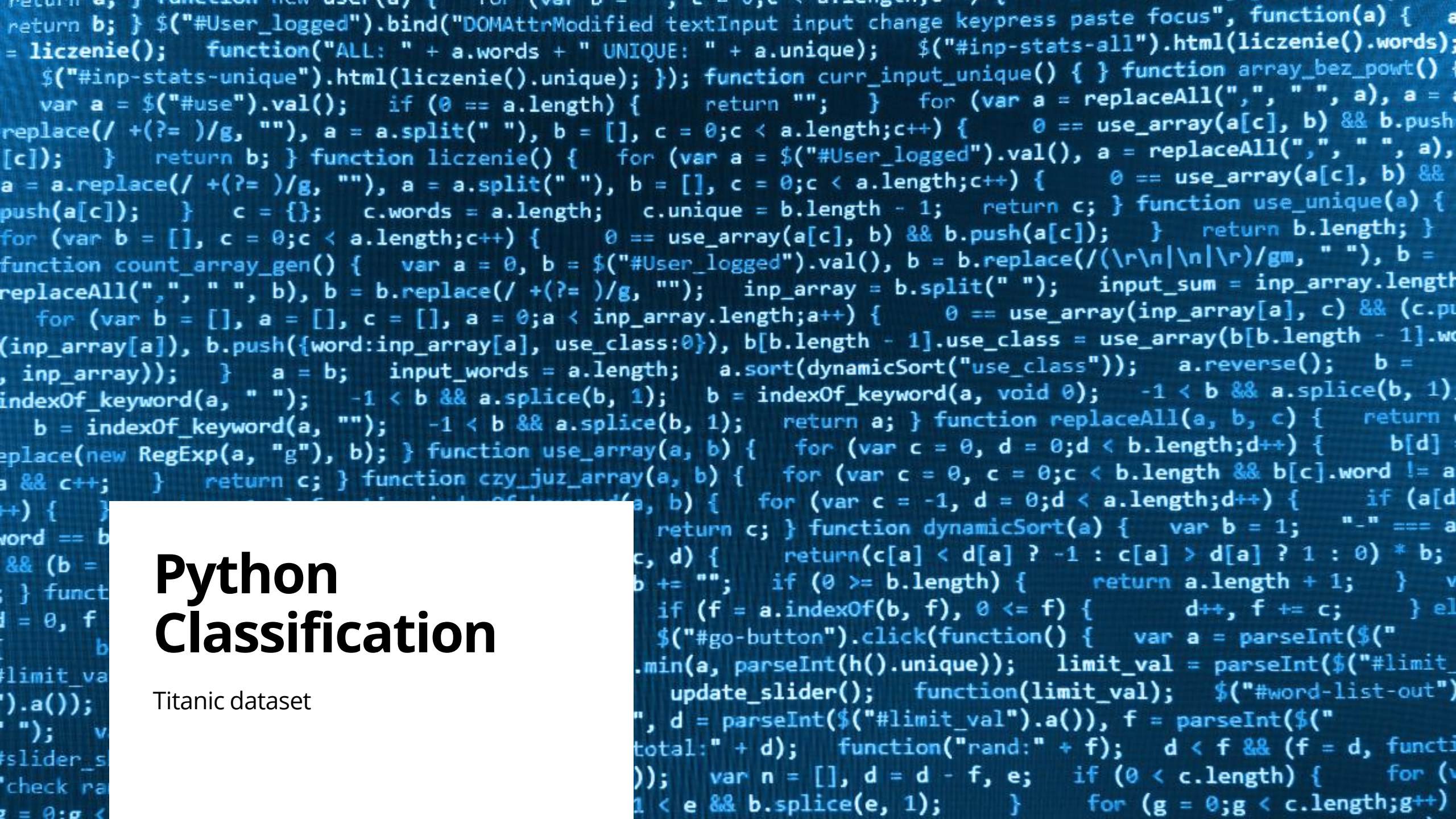












Python Classification

Titanic dataset

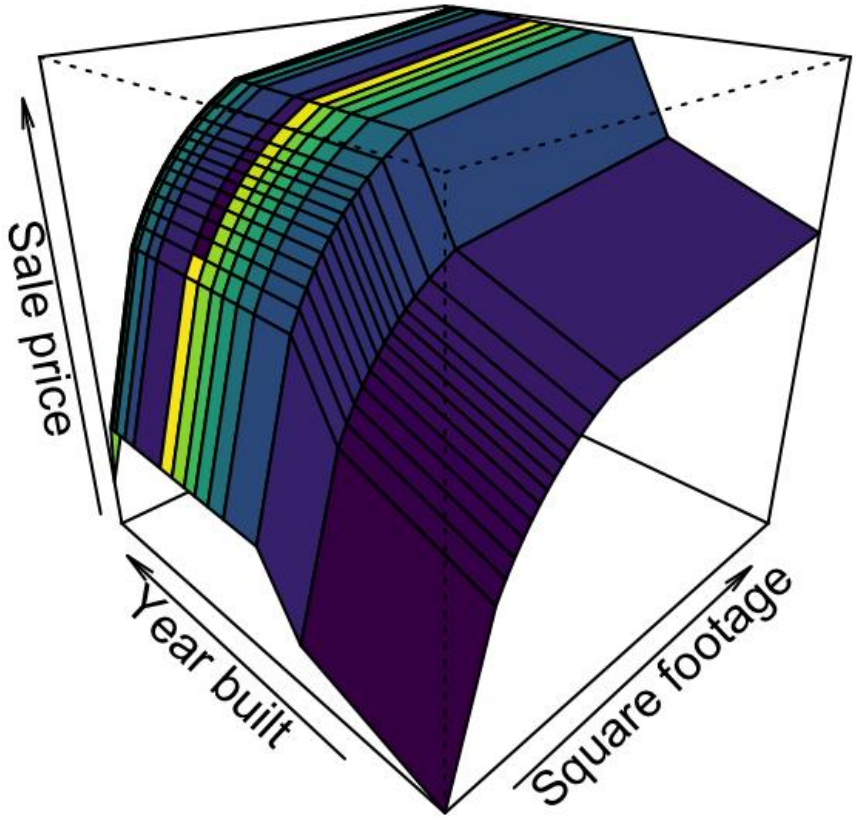
Regression

The motivation of regression modeling

Regression problem = when the objective of our supervised learning is to predict a **numeric outcome**.

Regression predictive modeling is the task of **approximating a mapping function (*f*) from input variables (*X*) to a continuous output variable (*y*).**

A continuous **output variable is a real-value, such as an integer or floating point value**. These are often quantities, such as **amounts and sizes**., e.g., home sales prices, insurance costs, time to market etc.



Average home sales price as a function of year built and total square footage.

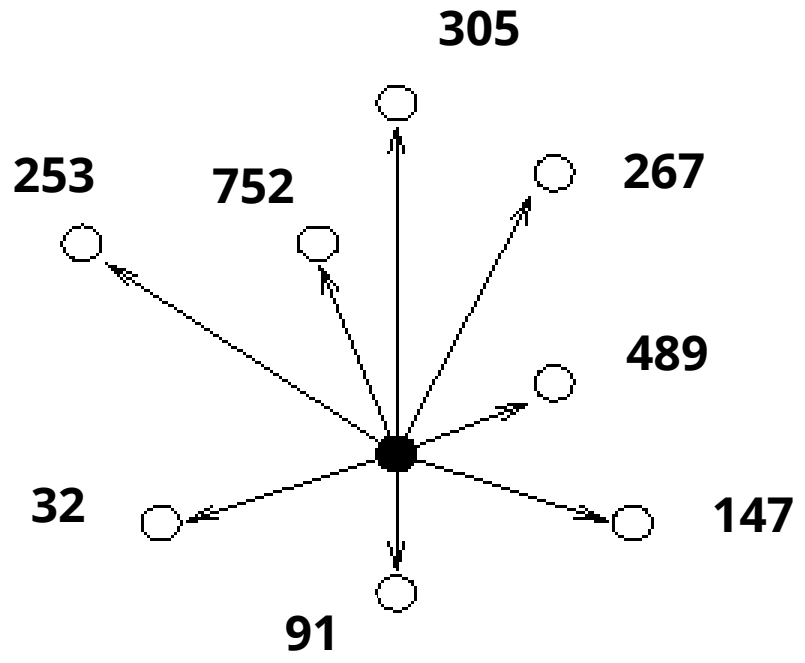
Typical issues

- What insurance costs can be expected for a specific client given his/her attributes?
- What is the expected price of specific house with given attributes?
- How much money will specific client spent next quarter given his RFM attributes?
- What is expected time to market for specific product given its production attributes?
- What factors are related to salary of employees within given company?
- What sales can we expect given our budget spent for various marketing channels?
- How much more points than our competitors we need to win to achieve 95 wins in the regular season and thus to get to play-off?

K Nearest neighbour

k-nearest neighbour is non parametric regressor. The predicted value of test sample is set equal to the mean value among nearest k training samples.

The “nearest” is assessed through Euclidean (L2), Manhattan (L1) or Chebychev (L∞) distance.

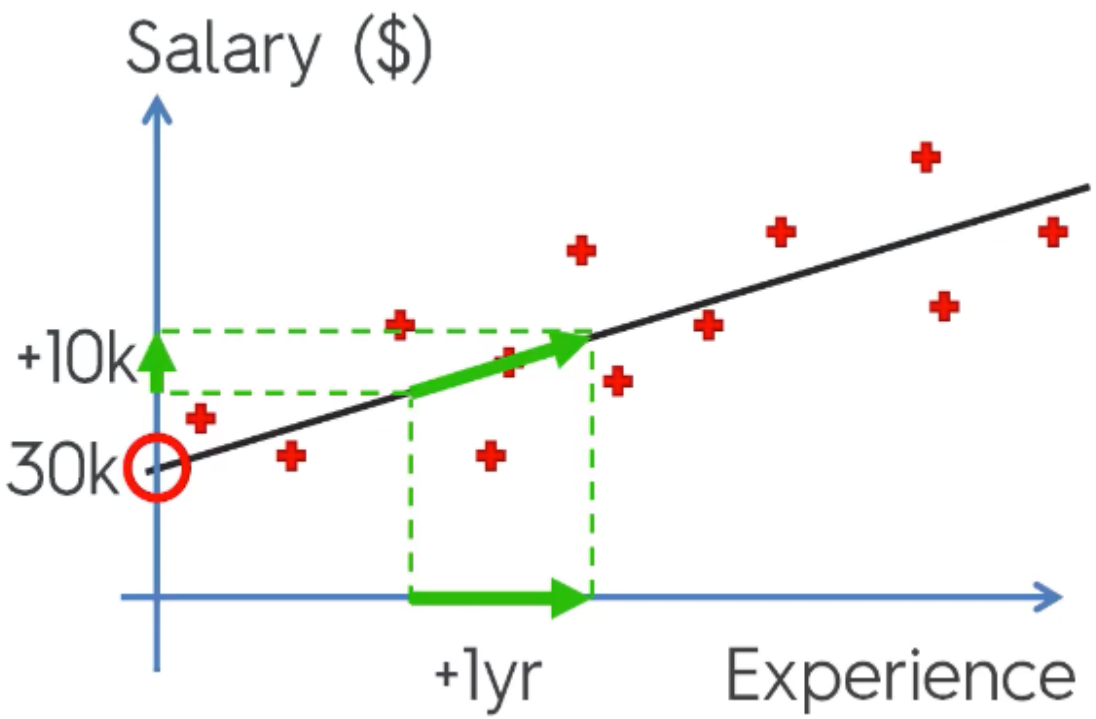


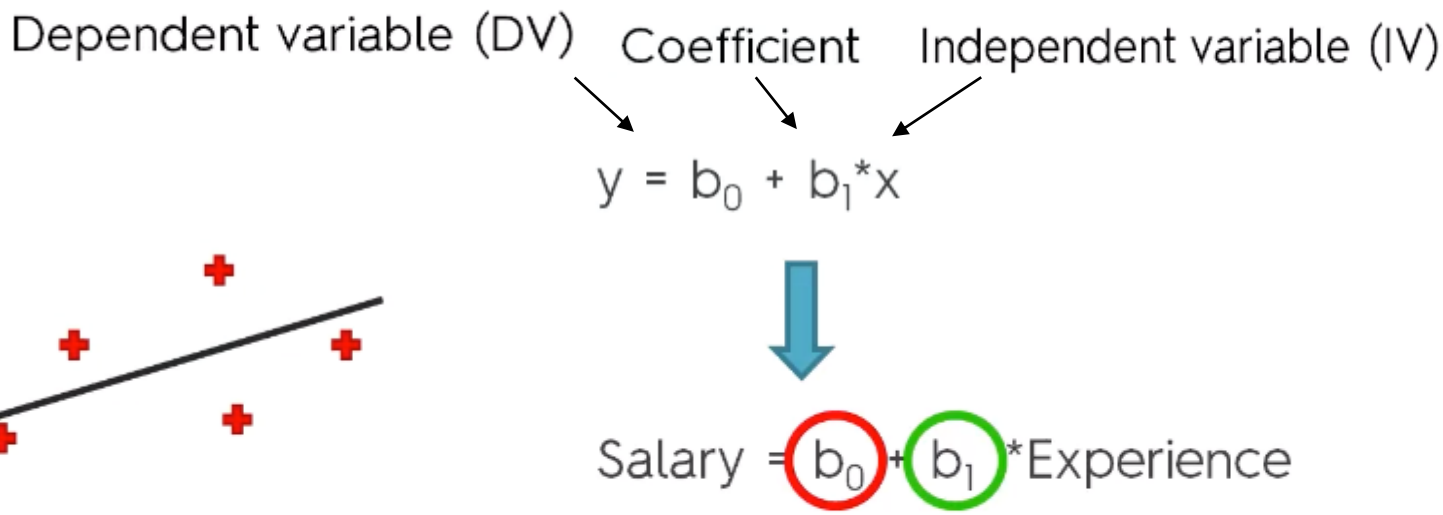
Straightforward, but CPU intensive algorithm.

Be aware of **overfitting**.

No business interpretation.

Simple linear regression

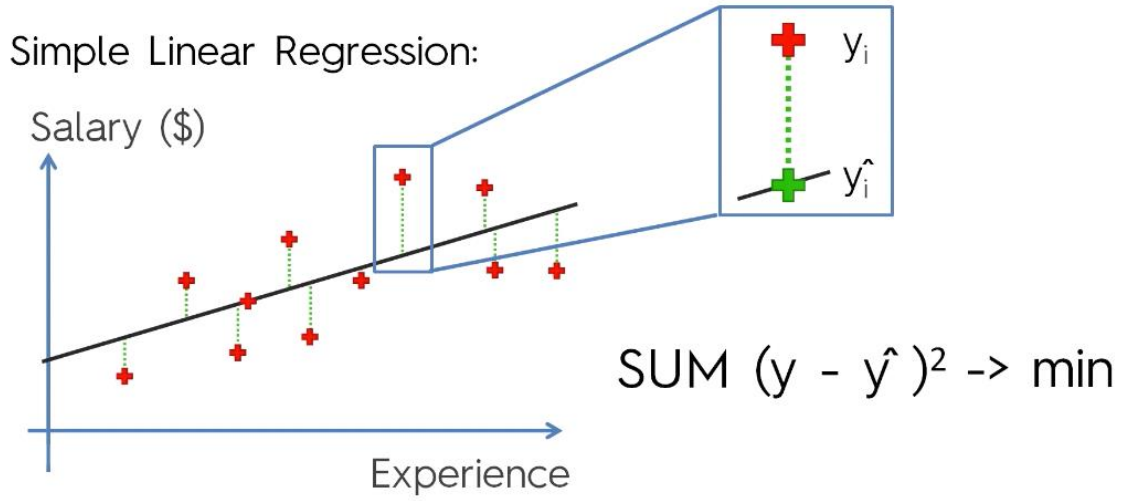




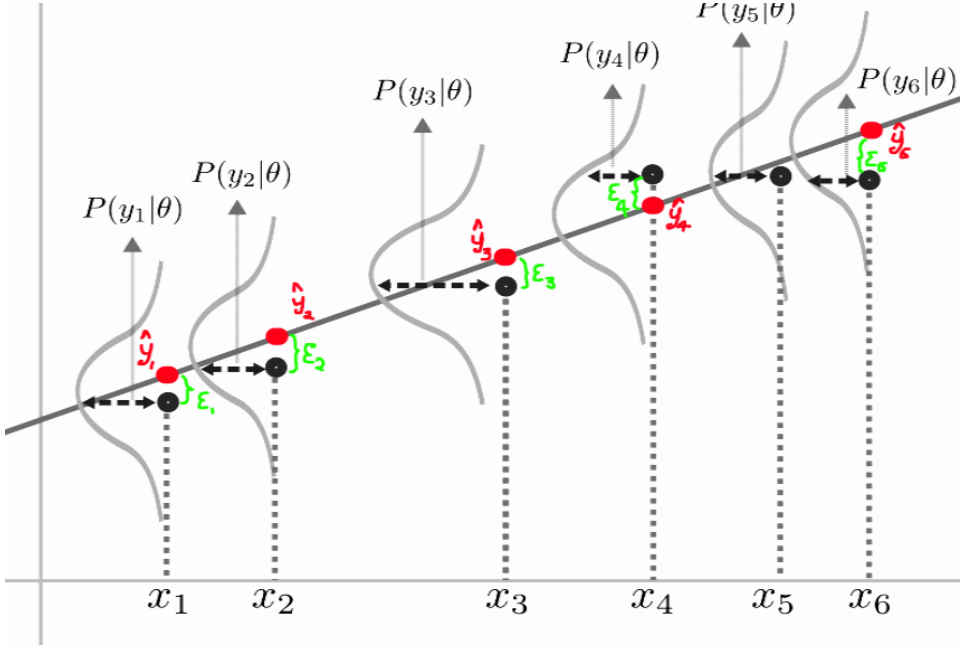
Source: Eremenko & de Ponteves (2017)

Linear regression – parameters estimation

Ordinary least squared

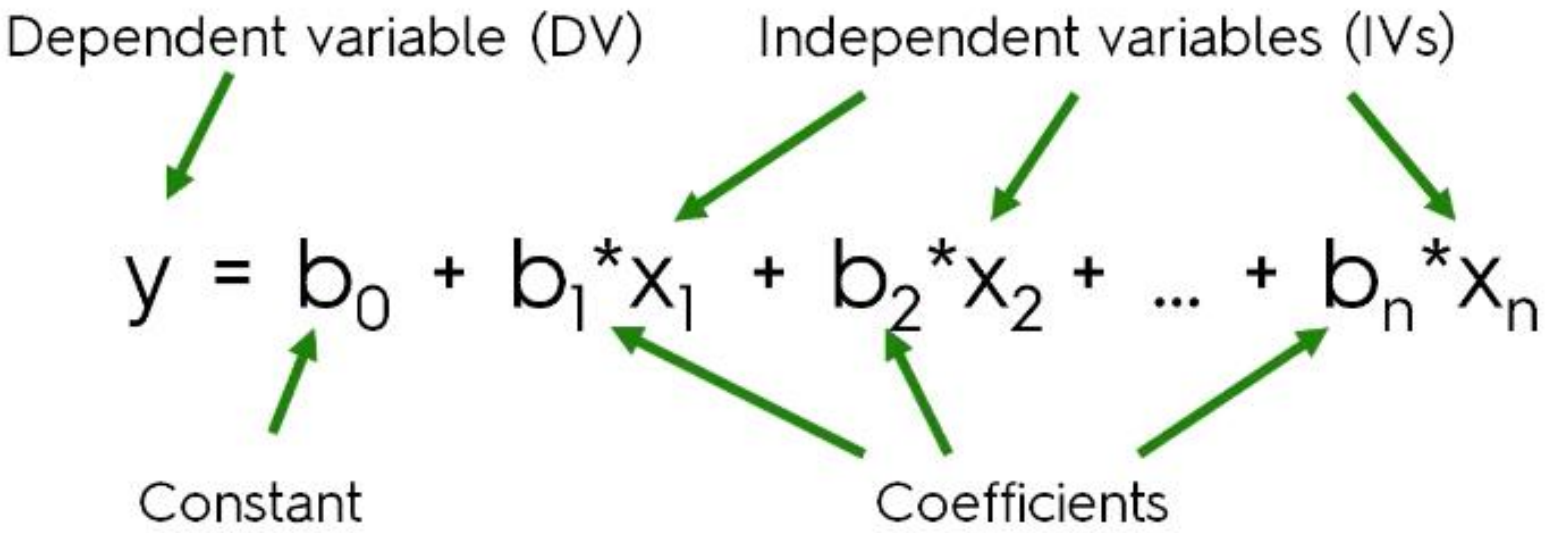


Maximum likelihood



Source: Eremenko & de Ponteves (2017)

Multiple linear regression

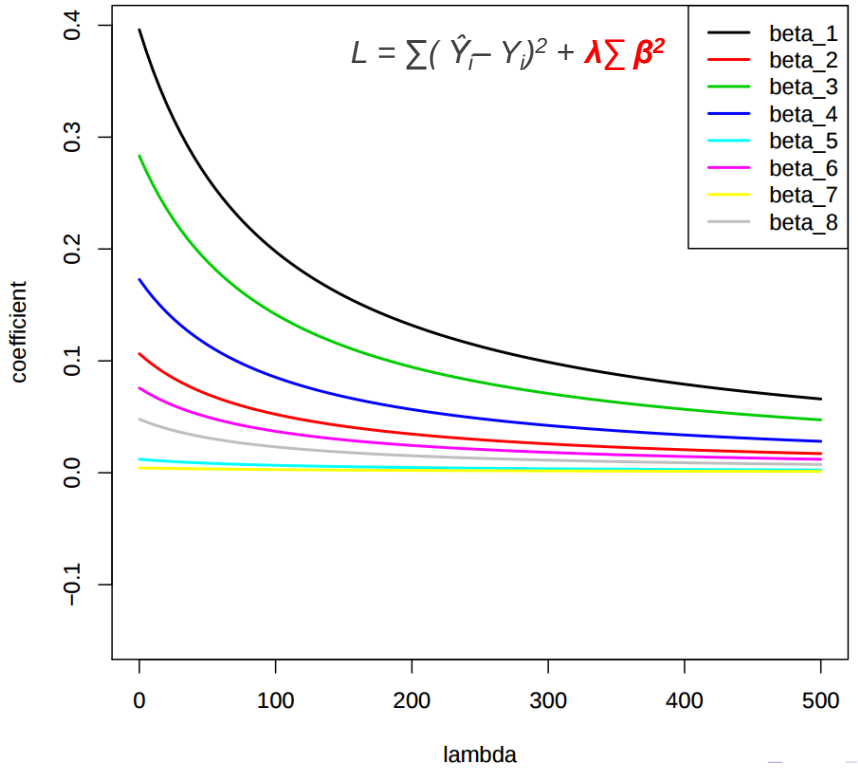


Source: Eremenko & de Ponteves (2017)

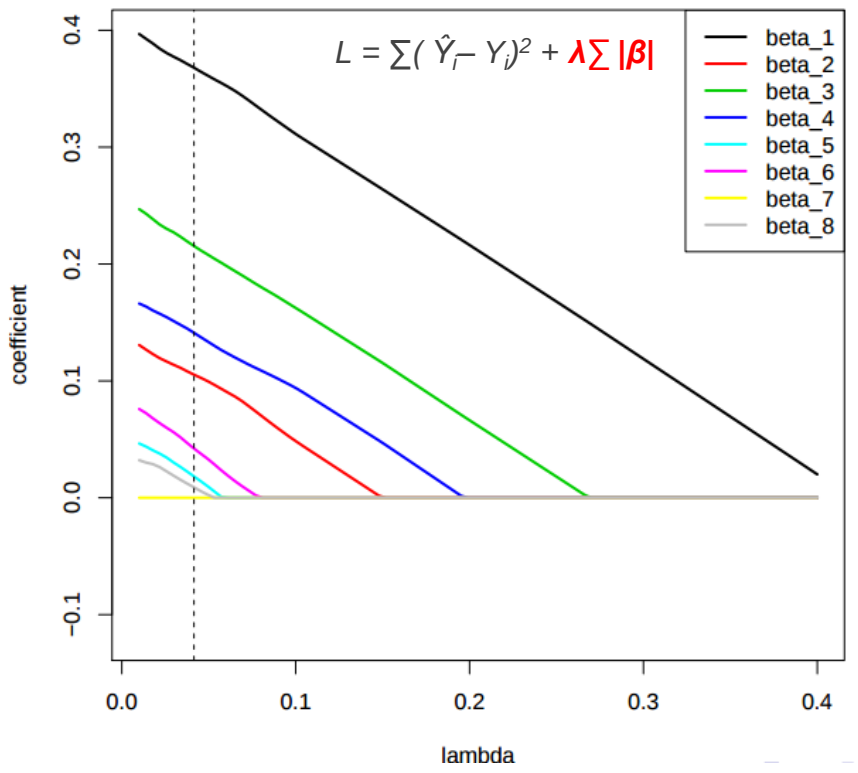
Linear regression – regularization

Two types of **regularized regression** that both help to deal with **overfitting**

Ridge



Lasso



Polynomial regression

Simple linear regression

$$y = b_0 + b_1 x_1$$

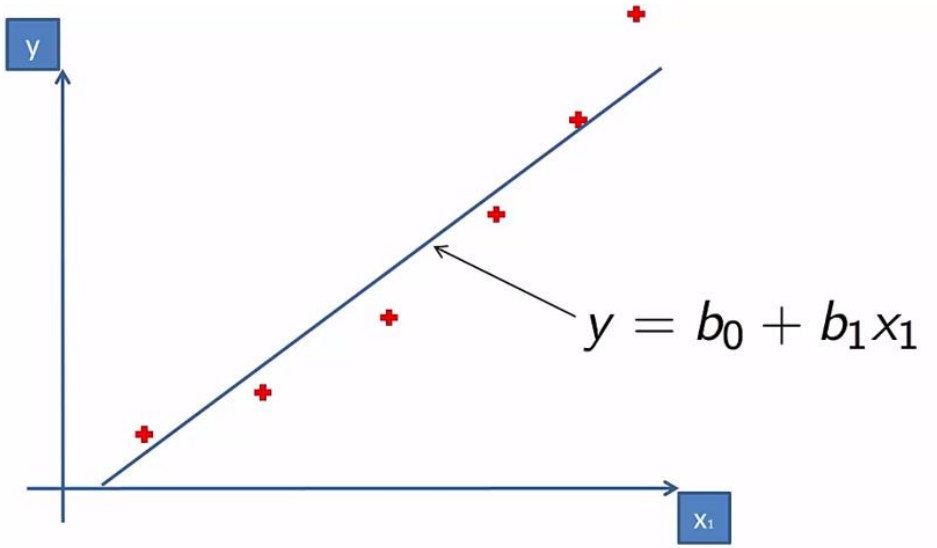
Multiple linear regression

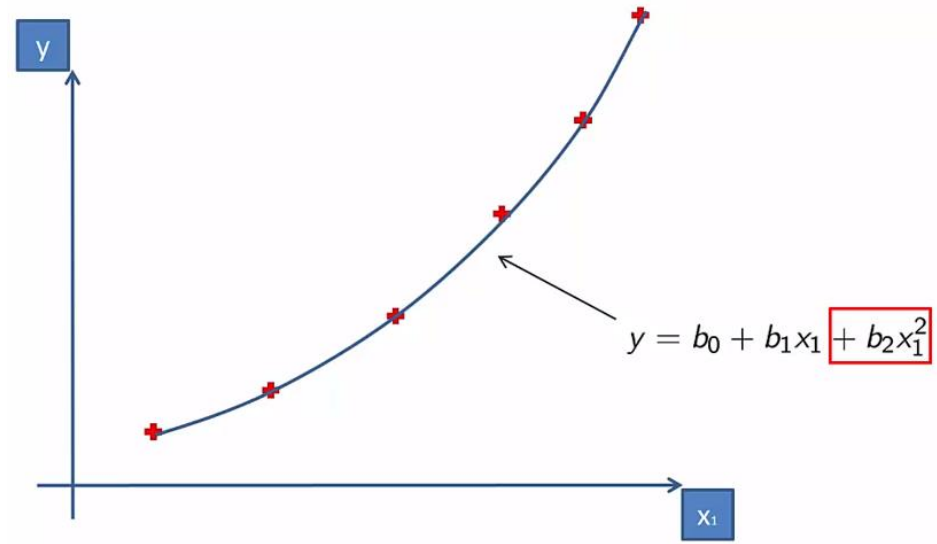
$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

Polynomial regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

and others (binomial, Gaussian, ...)

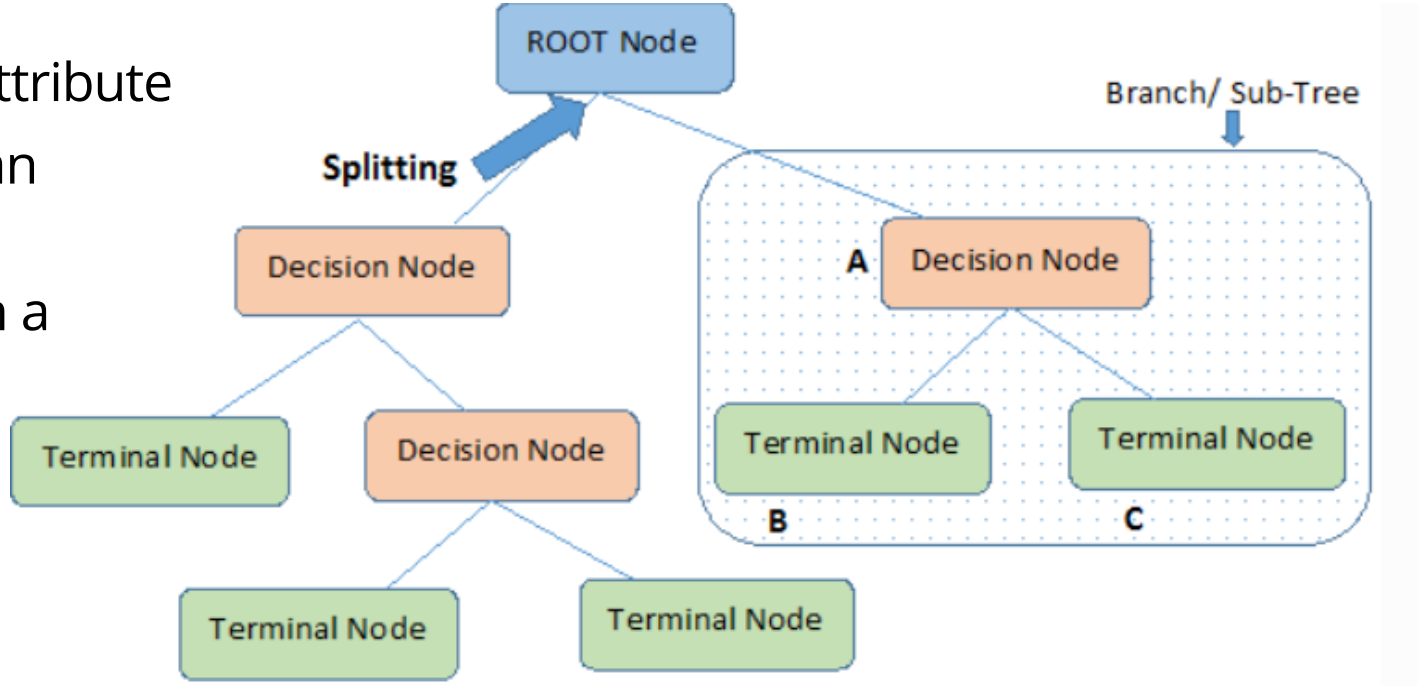




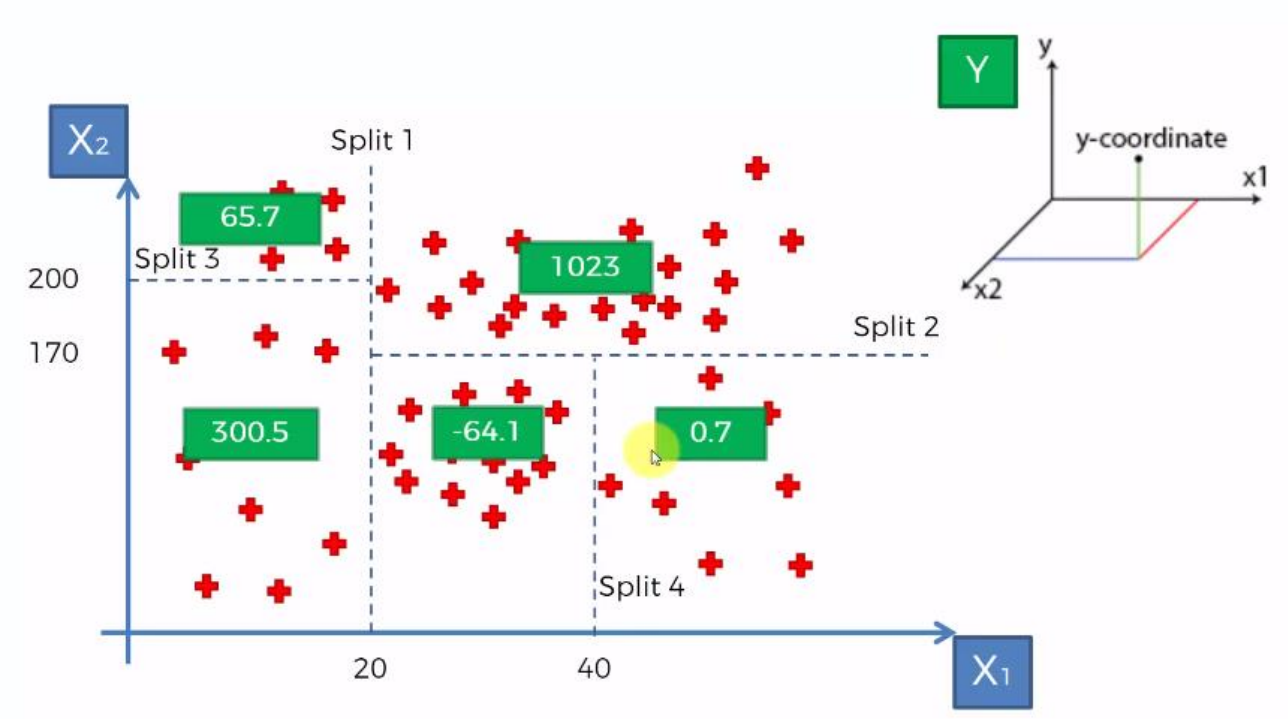
Source: Eremenko & de Ponteves (2017)

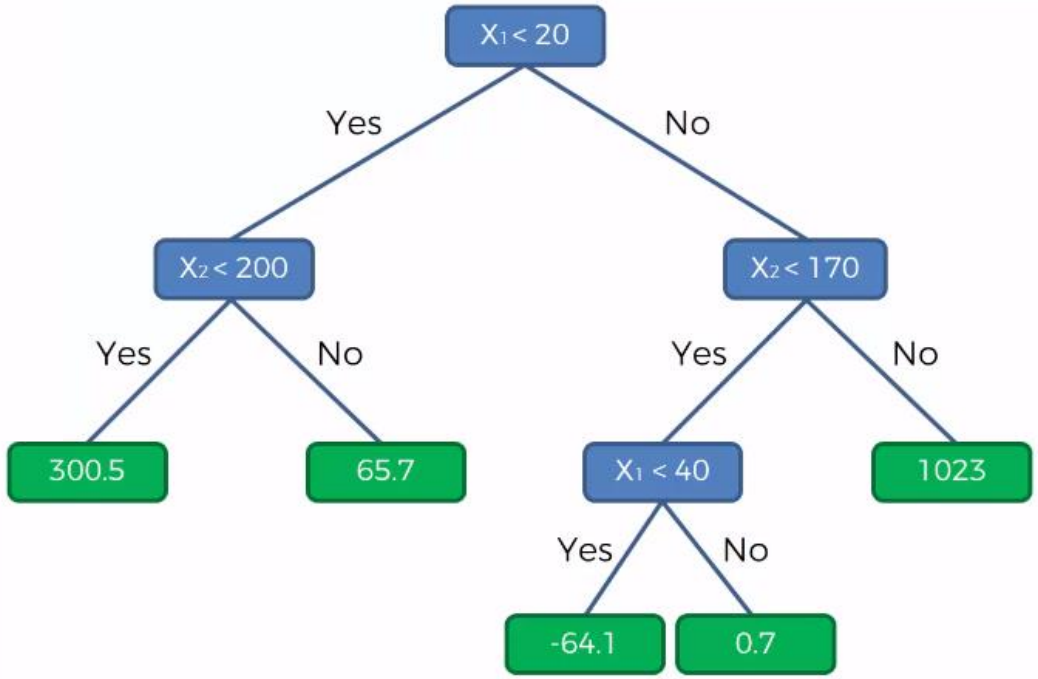
Decision tree regression

- **Classification and regression trees** (*CART*) handle both classification and regression problems.
- A decision tree is a tree where:
  - Each **internal** node tests an attribute
  - Each **branch** corresponds to an attribute value
  - Each **leaf** node is labelled with a average numerical output



Decision tree regression





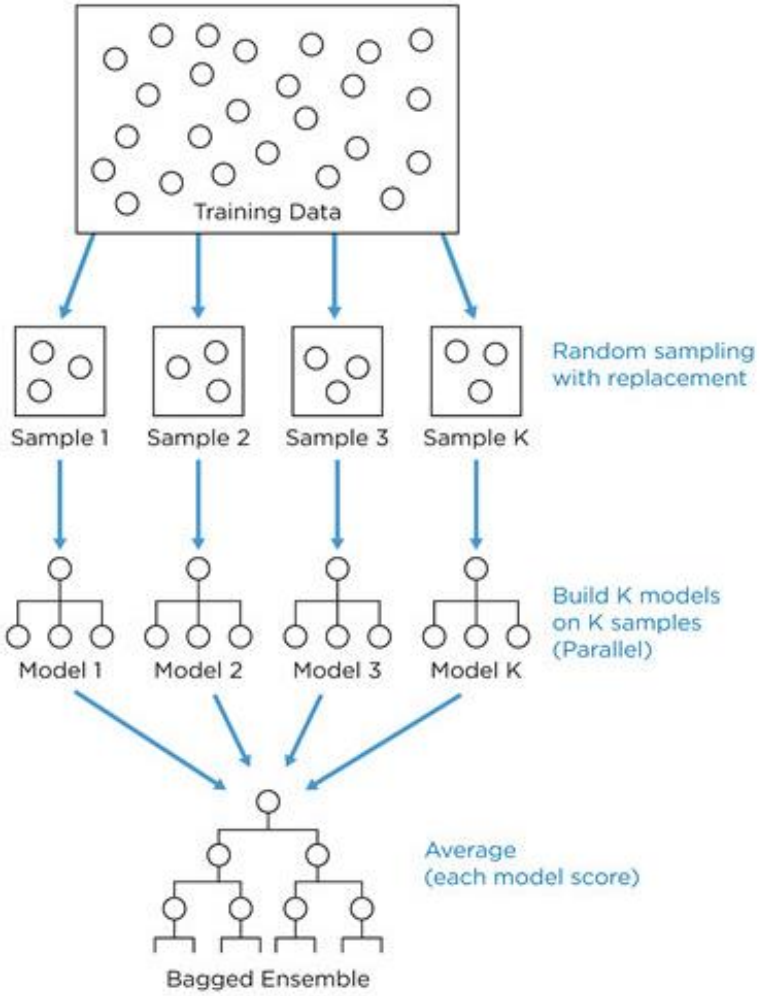
Source: Eremenko & de Ponteves (2017)

Decision tree regression

- CART handles both continuous and discrete inputs and NAs.
- Good **interpretability**.
- Be aware of **overfitting** (can be handled by using **bagged trees**).
- Regression tree depends on:
  - **Splitting rule** – how to choose best subsets (information entropy).
  - **Stopping rule** – when to decide this is a terminal node (e.g. min 200 observations).

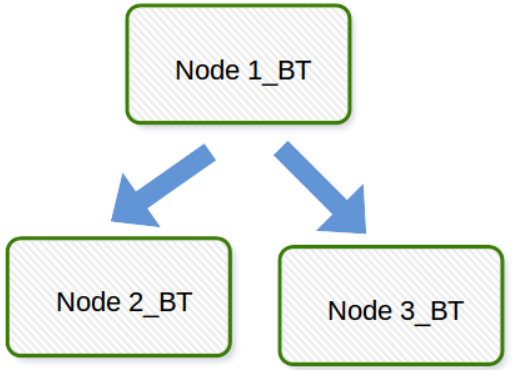
Bagged trees & Random forests

Bagging (Bootstraped Aggregation)



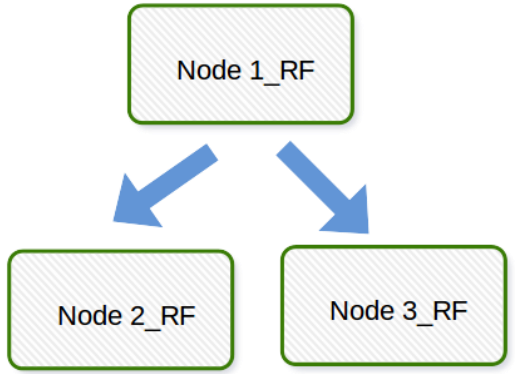
Bagging Trees--

All of M features considered for each node for a split



Random forests--

Only m<M features considered for each node for split



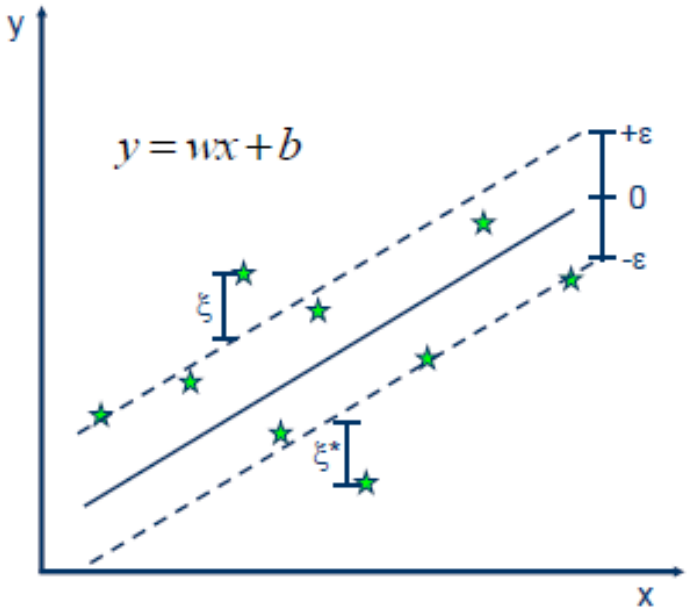
m can be selected via out-of-bag error, but ***m*** = sqrt(***M***) is a good value to start with

Support vector regression

SVR support linear and non-linear regression using the same principles as the SVM for classification, with only a few minor differences.

Instead of trying to fit the largest possible “street” between two classes while limiting margins violations, SVR tries to fit as many instances as possible on the “street” while limiting margin violations.

The width of the street is controlled by a hyper parameter Epsilon.



- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

- Constraints:

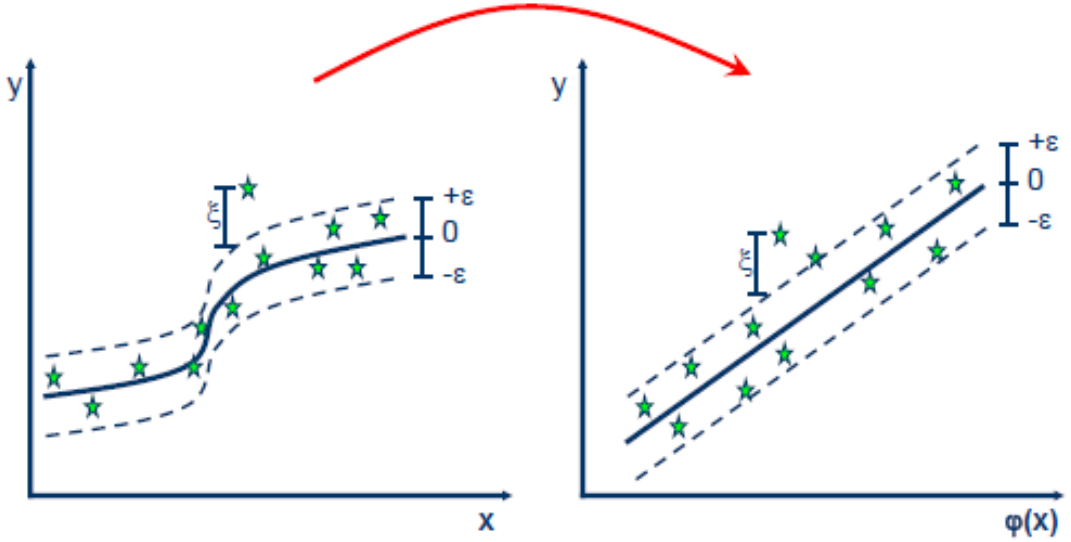
$$y_i - wx_i - b \le \varepsilon + \xi_i$$

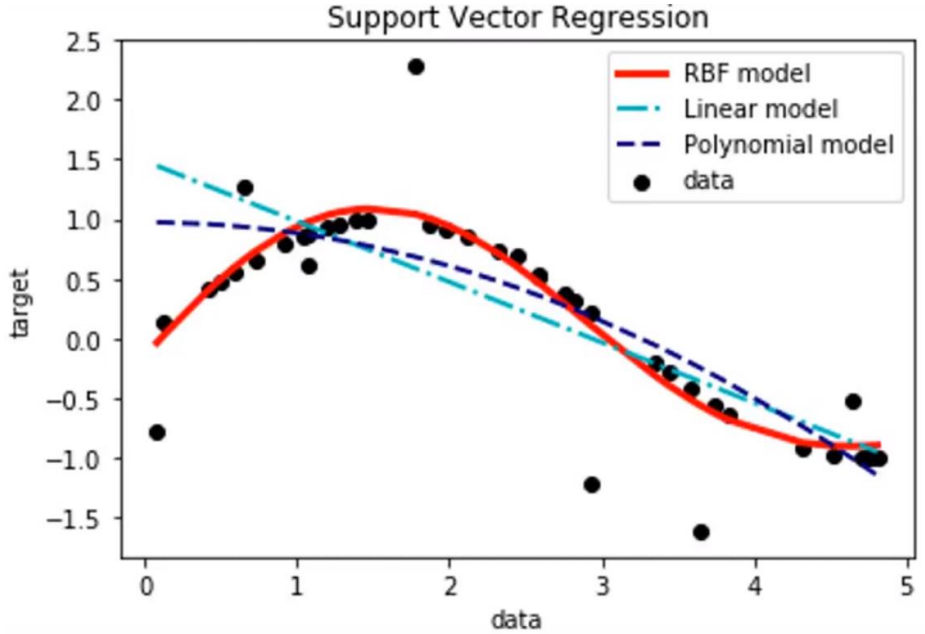
$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$

Non-linear support vector regression

The kernel functions transform the data into a higher dimensional feature space to make it possible to perform the linear separation.





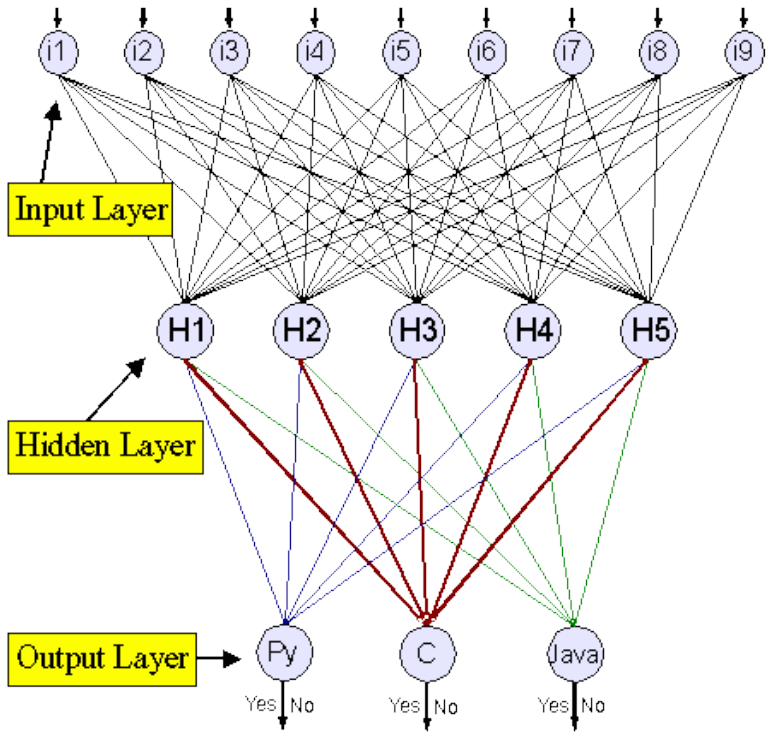
Neural networks and deep learning

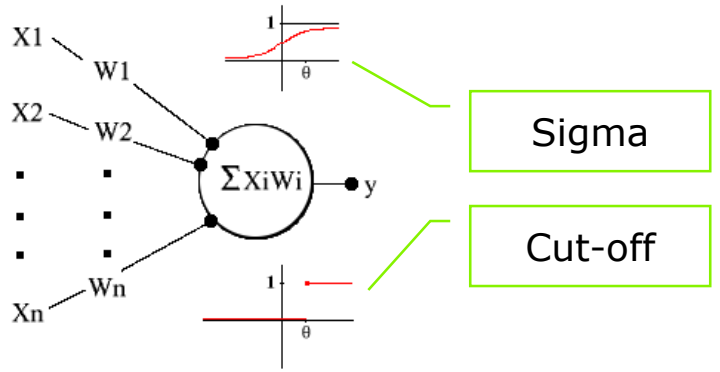
NN is a complex structure consisting of **neurons** and **connections** (synapses) organized in multiple layers, inspired by brain.

Neurons are real valued functions (**activation function**) with multiple inputs and one output. Activation functions used are:

- hyperbolic tangent f(x) = tanh(x)
- sigmoid f(x) = 1/(1 + exp(-x))
- softmax f(x) = exp(x)/sum(exp(x))

NNs are trained using MSE and other criteria by **back-propagation** learning algorithm.





Regression Evaluation

Loss function

Assessing model performance is **to assess the predictive accuracy via loss functions**, i.e., via metrics that **compare the predicted values to the actual value** (*error/pseudo residual*).

The way a loss function is computed will tend to **emphasize certain types of errors over others** and can lead to significant differences in how we interpret the “optimal model”.

When **comparing multiple models**, we need to compare them across **the same metric.**

Major loss functions for regressions models

MSE: Mean squared error is the average of the squared error. The squared component results in larger errors having larger penalties. Objective: minimize.

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

RMSE: Root mean squared error. This takes the square root of the MSE metric, so that error is in the same units as response variable. Objective: minimize.

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

MAE: Mean absolute error. Similar to MSE but rather than squaring, it just takes the mean absolute difference between the actual and predicted values. This results in less emphasis on larger errors than MSE. Objective: minimize.

$$(MAE = \frac{1}{n}\sum_{i=1}^{n}(|y_i - \hat{y}_i|))$$

Major loss functions for regressions models (cont.)

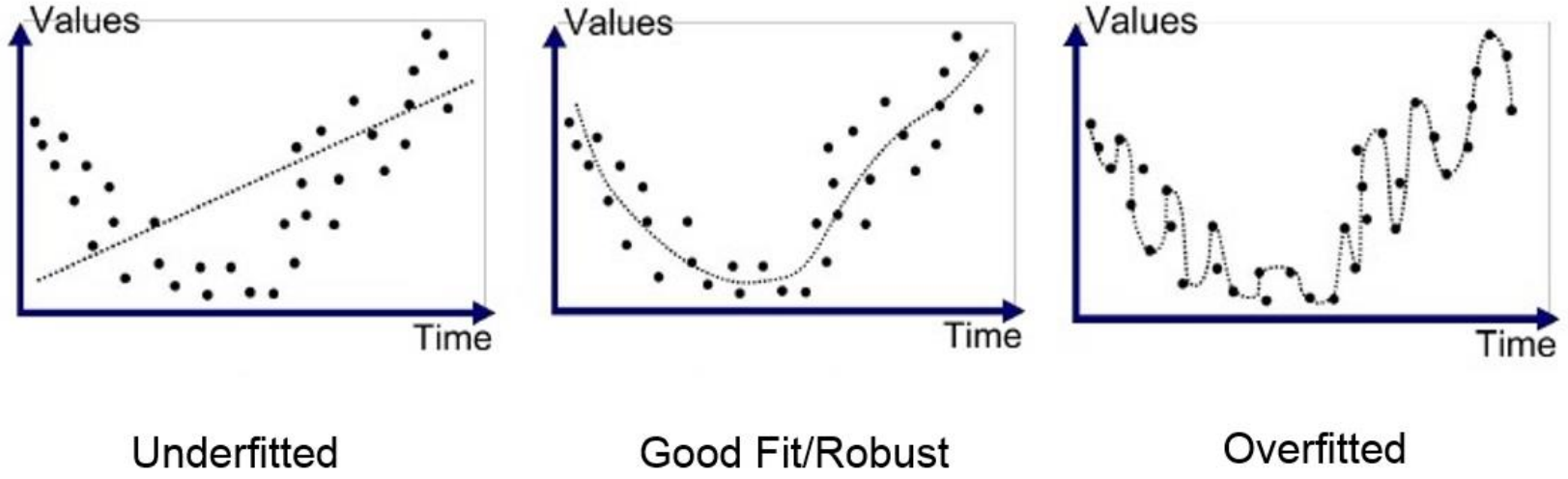
RMSLE: Root mean squared logarithmic error. Similar to RMSE but it performs a log() on the actual and predicted values prior to computing the difference. When response variable has a wide range of values, large response values with large errors can dominate the MSE/RMSE metric. RMSLE minimizes this impact so that small response values with large errors can have just as meaningful of an impact as large response values with large errors. Objective: minimize.

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(y_i + 1) - log(\hat{y}_i + 1))^2)}.$$

***R²*:** The proportion of the variance in the dependent variable that is predictable from the independent variable(s). It has several limitations. For example, two models built from two different data sets could have the exact same RMSE but if one has less variability in the response variable then it would have a lower R^2 than the other. You should not place too much emphasis on this metric. Objective: maximize.

Overfitting vs. underfitting

- **Overfitting** is when model fits too closely to the training data (including the noise) and fails to fit future observations reliably.
- Overfitting happens when model is **too complex/flexible** or **training dataset is too small**.
- **Underfitting** is when model is too simple/inflexible to capture the patterns hidden in the data.

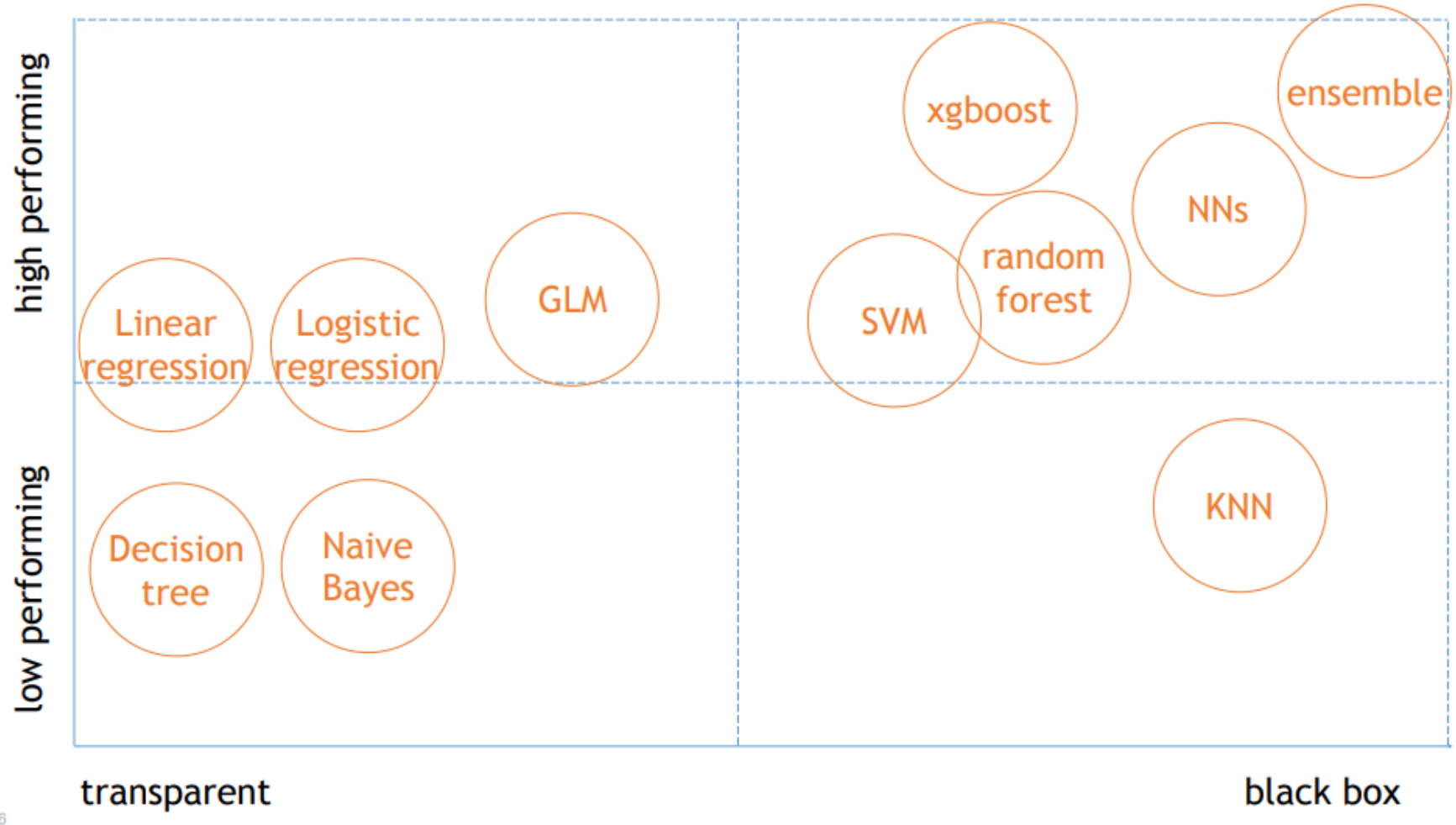


Python Regression

Diamons dataset

Model Interpretability

Performance vs. interpretability trade-off



Why should we improve our understanding of ML models?

Improving our models

Generalisability

“Sanity Check”

Prevent wrong conclusions &

potentially adversarial attacks

Trust and transparency

Can I trust my model's decisions?

Why does my model make the predictions it makes?

Prevent Bias

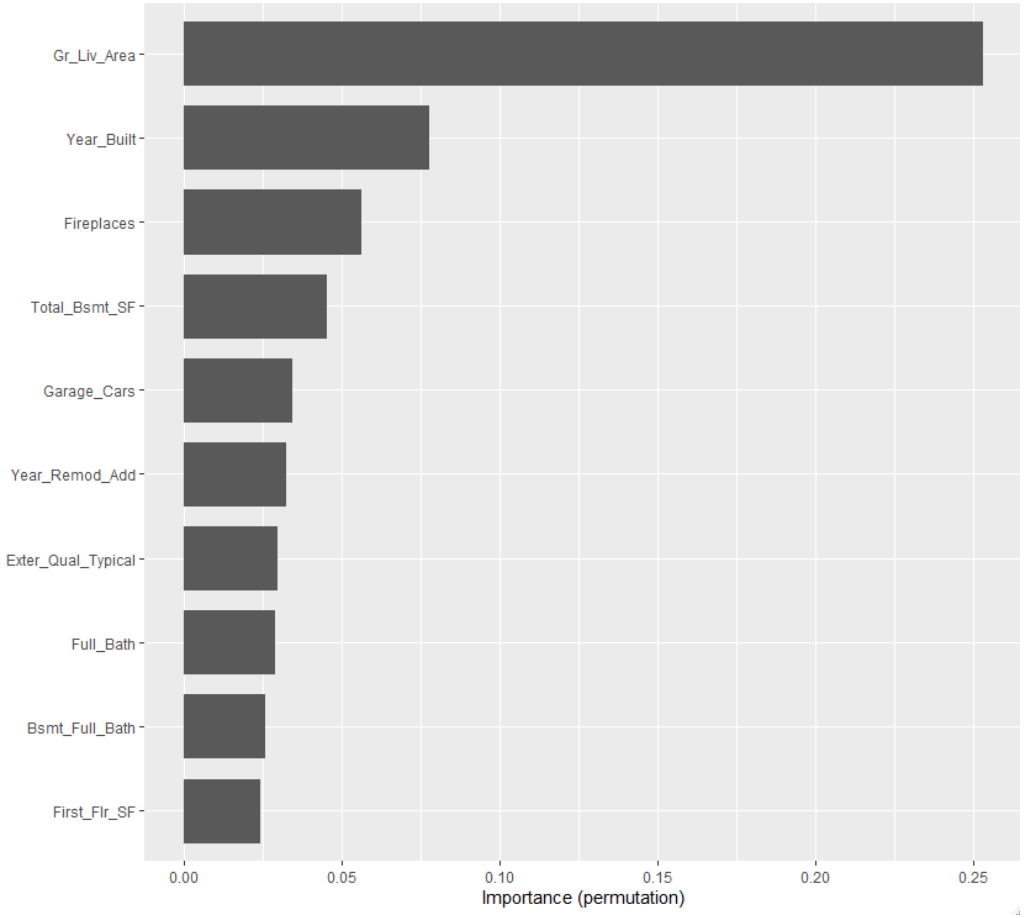
Fairness

Identify and prevent bias

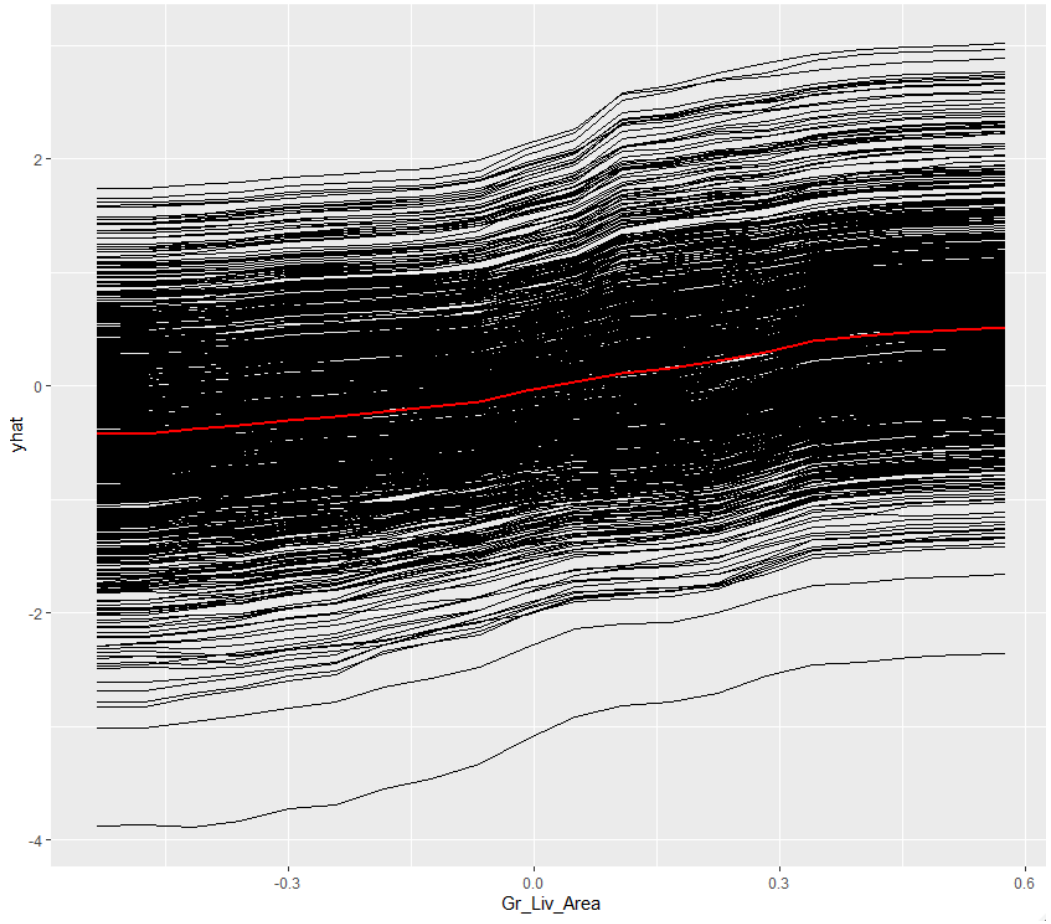
Source: Glander (2018)

Looking under the models' hood

Permutation-based variable importance

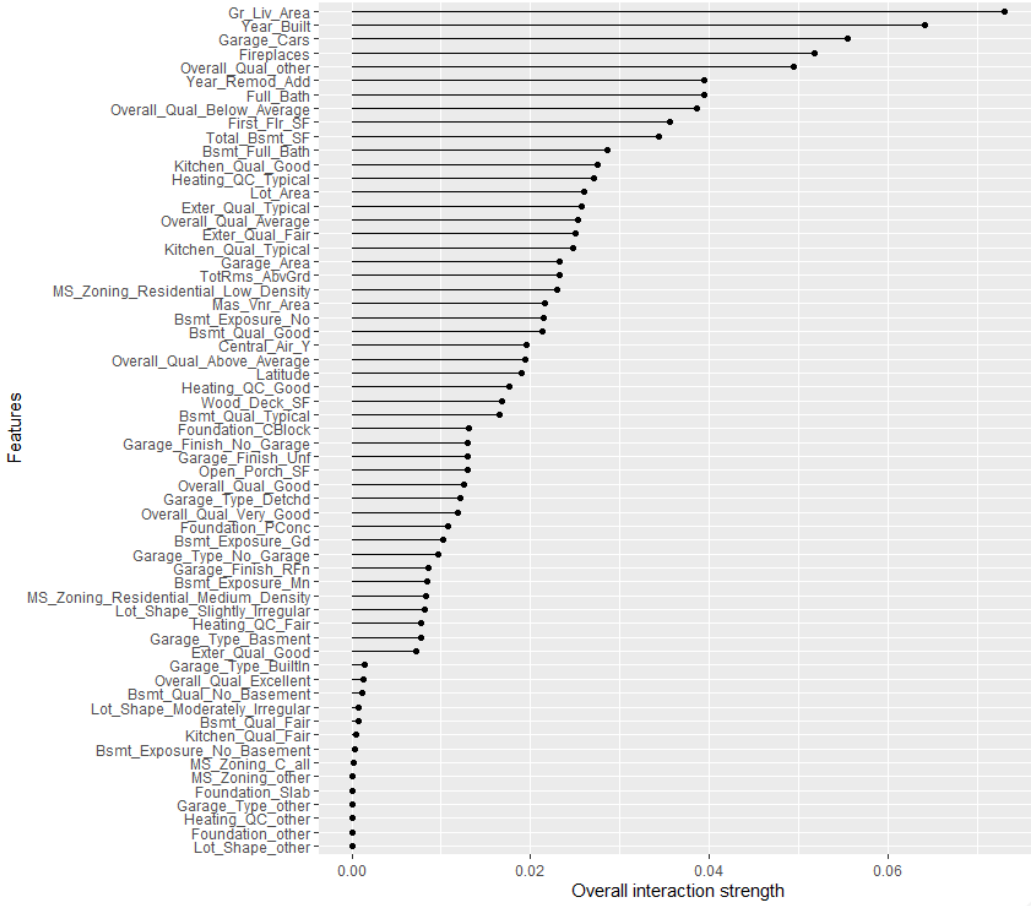


Partial dependence/individual conditional expectation curves

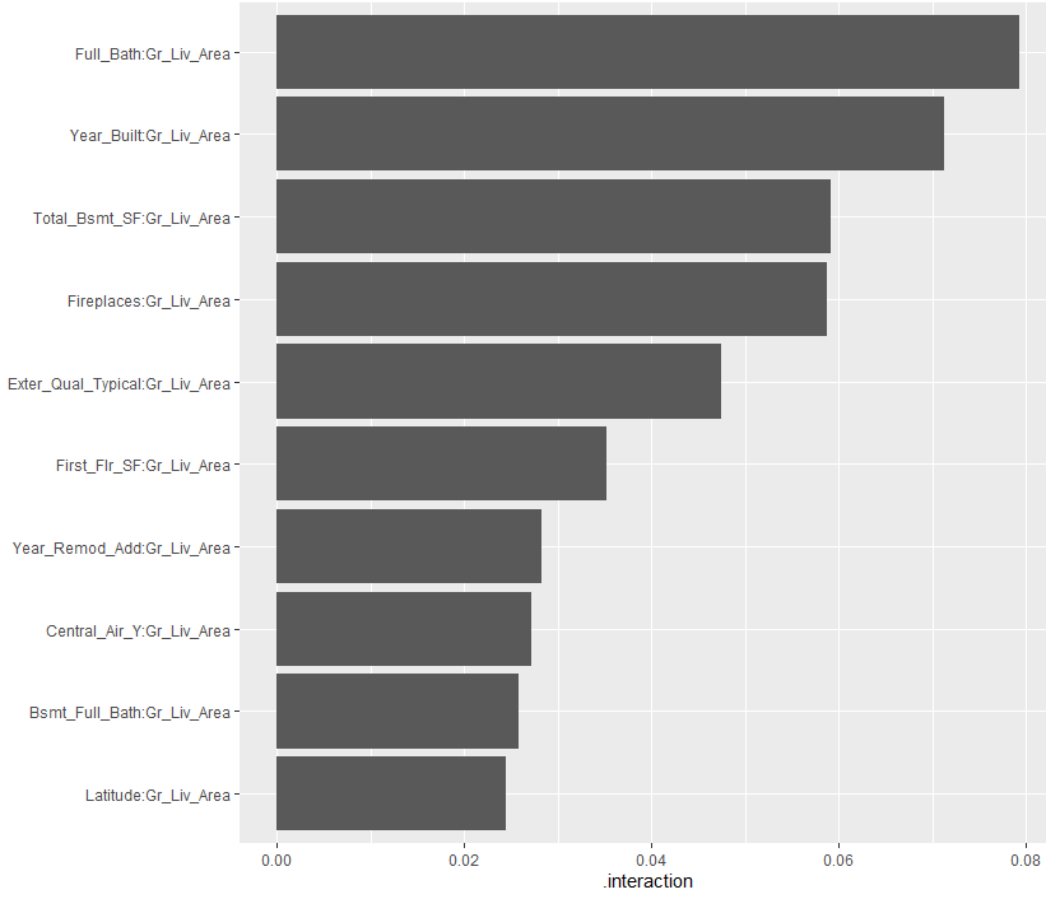


Looking under the models' hood (cont.)

Detection of interactions

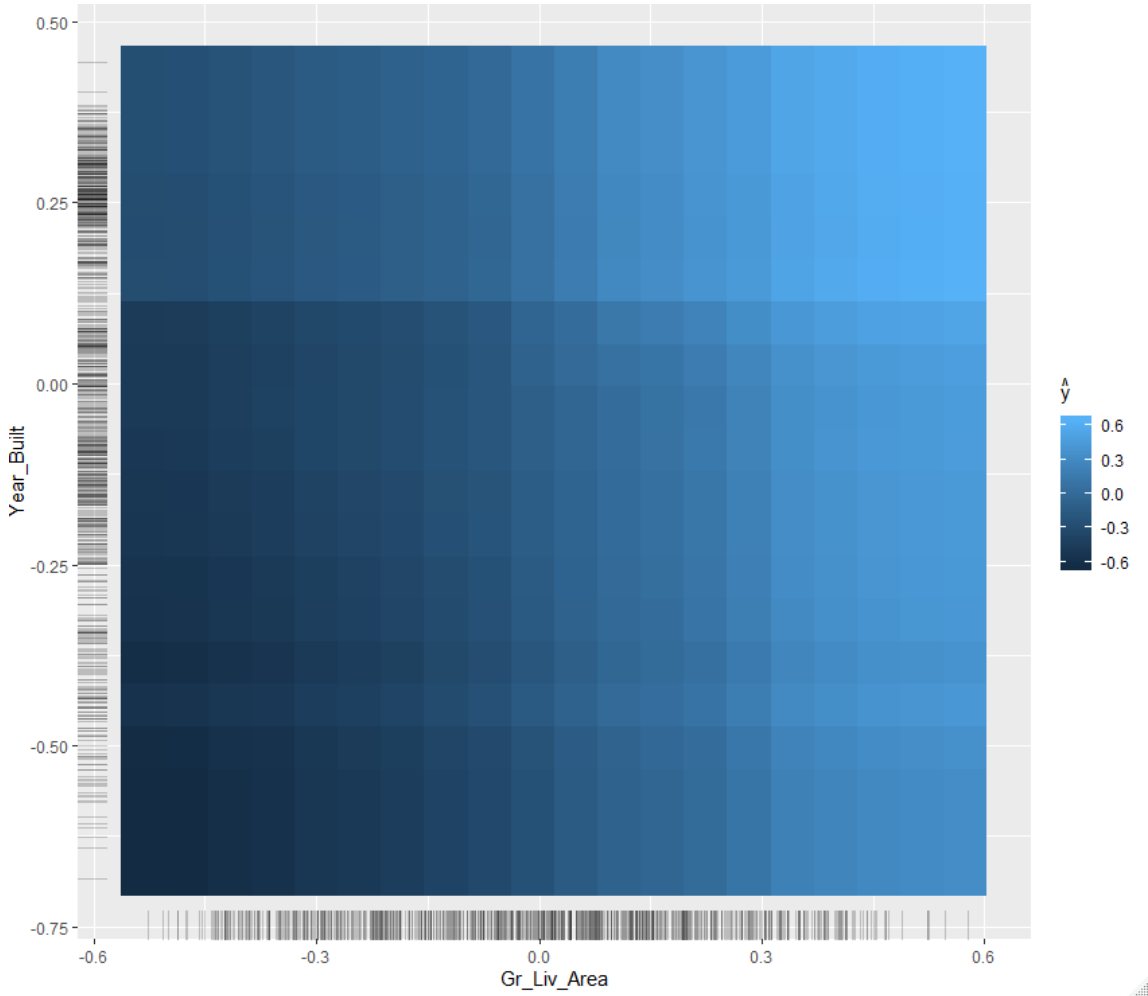


The Strongest two-way interactions



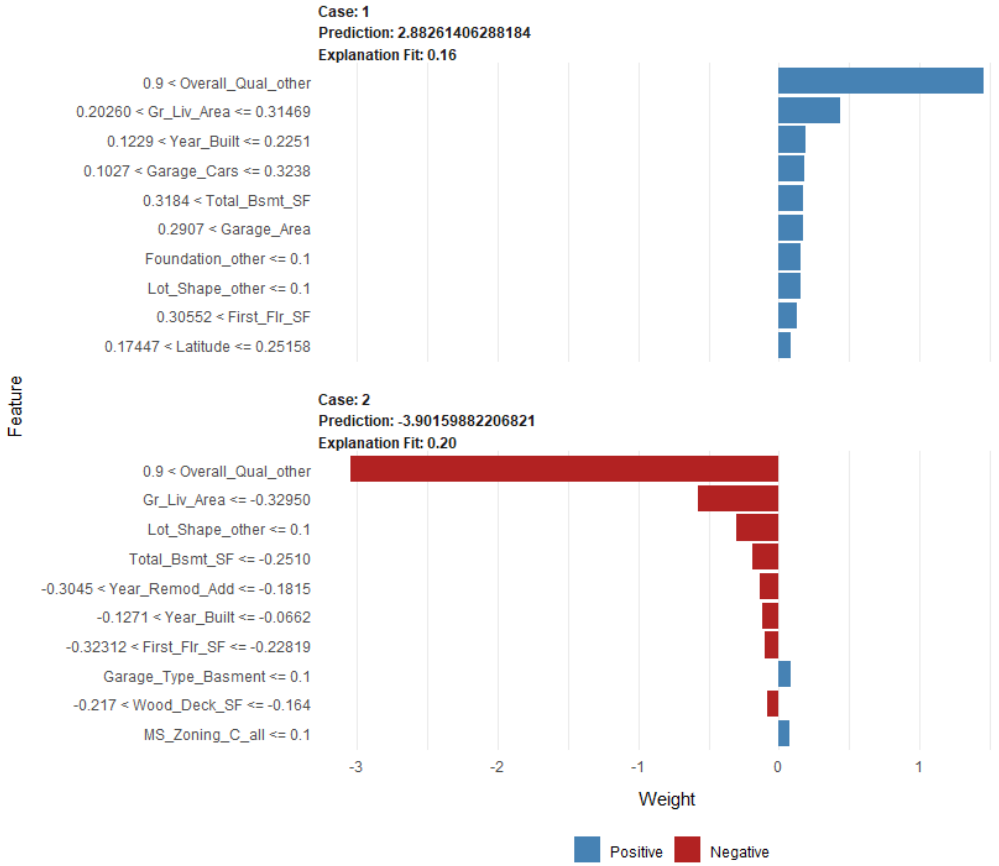
Looking under the models' hood (cont.)

Visualizing interactions

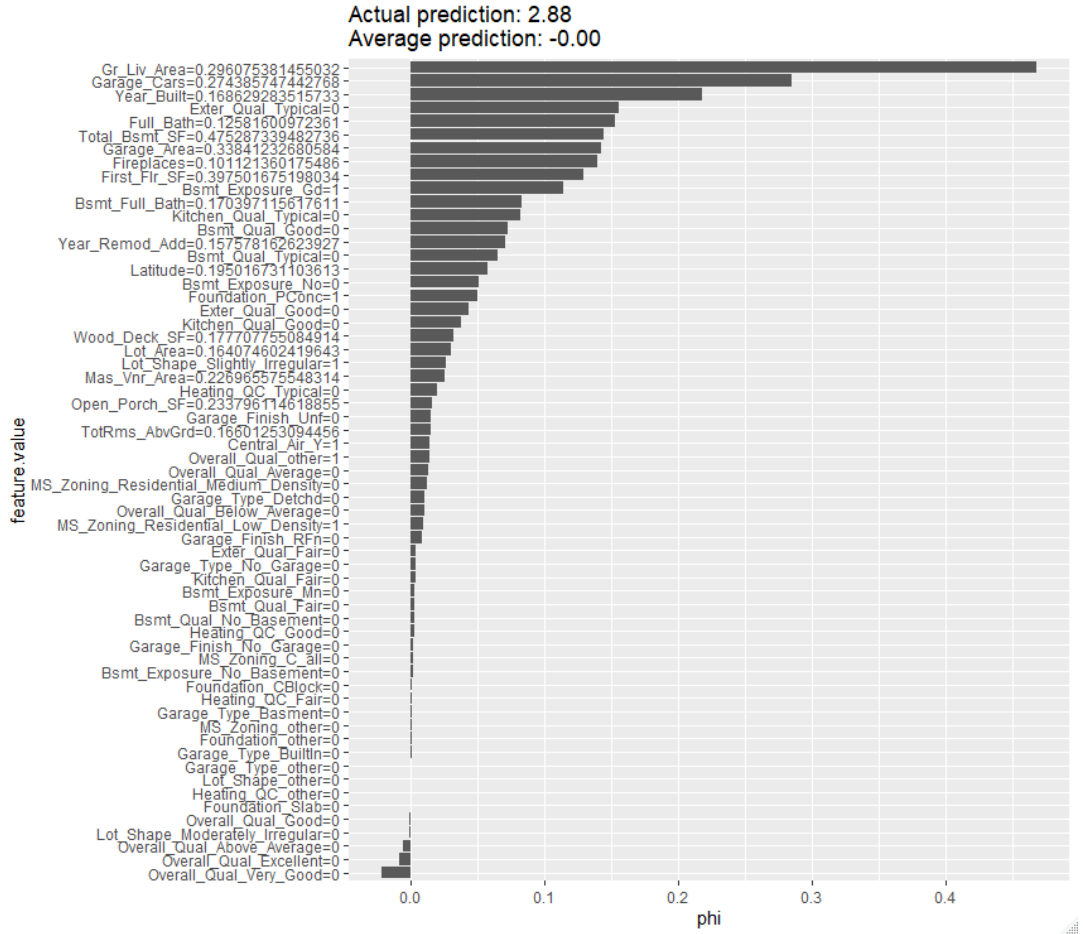


Looking under the models' hood (cont.)

Local interpretation of individual predictions (LIME)



Local interpretation of individual predictions (Shapley)



Balancing unbalanced datasets

Unbalanced data

Unbalanced data refers to classification problems where we have very **unequal instances for different classes** (≤ 10% vs. ≥ 90%).

Having unbalanced data is quite common in general, but it is especially **prevalent when working with disease or fraud data**, where most cases are okay and only very few will be sick/fraudulent.

Why is unbalanced data a problem in ML?

Most machine learning classification algorithms are **sensitive to unbalance in the predictor classes.**

An unbalanced dataset will **bias the prediction model towards the more common class.**

Example: Let's assume we had 5% fraudulent vs. 95% genuine samples. A machine learning model that has been trained and tested on such a dataset could now predict "fraudulent" for all samples and still gain a very high accuracy.

Options for dealing with unbalanced datasets

Class weights

- Imposing a heavier cost when errors are made in the minority class.

Under-sampling

- Random selection of a subset of samples from the class with more instances to match the number of samples coming from each class.
- Warning: We lose potentially relevant information from the left-out samples.

Over-sampling

- Random duplication of samples from the class with fewer instances.
- Warning: We run the risk of overfitting our model as we are more likely to get the same samples in the training and in the test data. This would lead to an overestimation of our model's performance and generalizability.

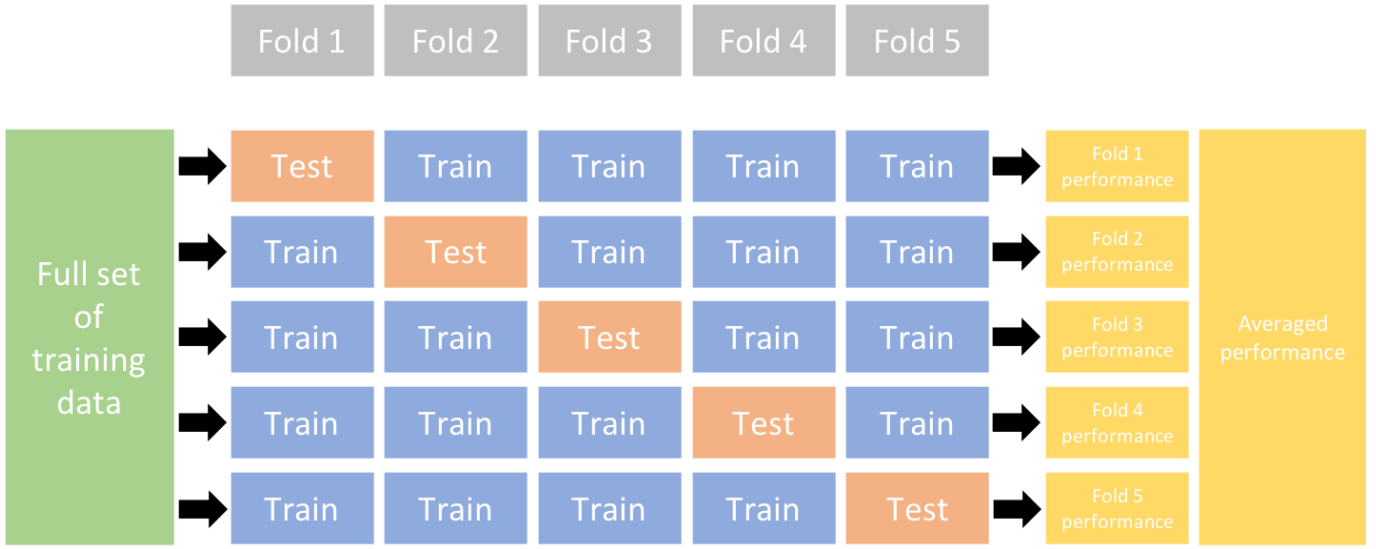
Synthetic minority sampling techniques (e.g., ROSE or SMOTE)

- Hybrid method that combine under-sampling with the generation of additional data based on the data we have.
- ROSE uses smoothed bootstrapping to draw artificial samples from the feature space neighbourhood around the minority class.
- SMOTE draws artificial samples by choosing points that lie on the line connecting the rare observation to one of its nearest neighbors in the feature space.

How to do over/under-sampling in practice?

We should not simply perform over/under-sampling on our training data and then run the model.

We need to account for cross-validation and perform over/under-sampling on each fold independently to get an honest estimate of model performance.



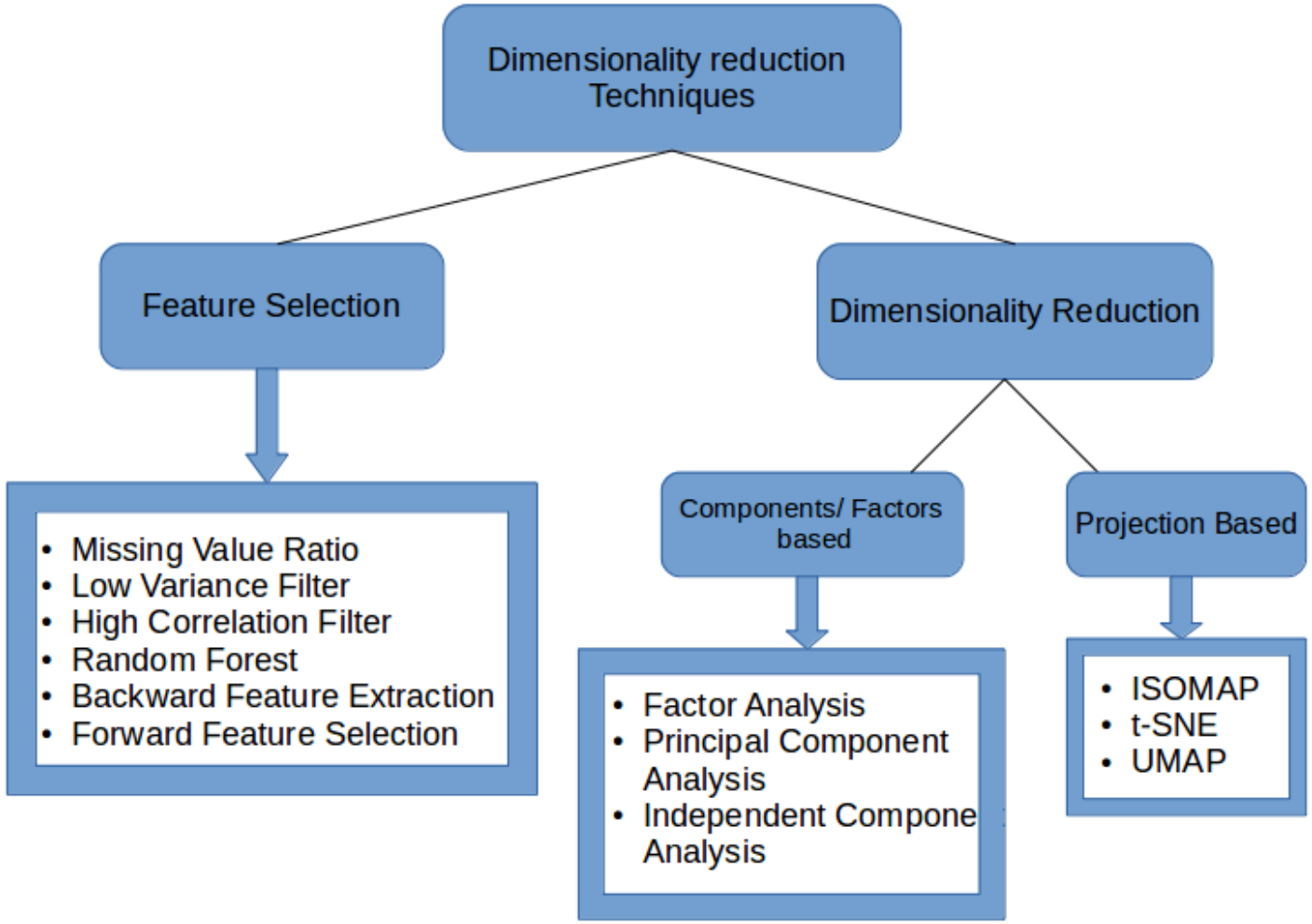
Dimensionality reduction

Algorithms

Motivation

1. It avoids the **curse of dimensionality**. (Neural network example)
2. Removal of multi-collinearity improves the interpretation of the parameters of the machine learning model.
3. It reduces the time and storage space required.
4. It becomes easier to visualize the data when reduced to very low dimensions such as 2D or 3D.

Tools for dimensionality reduction

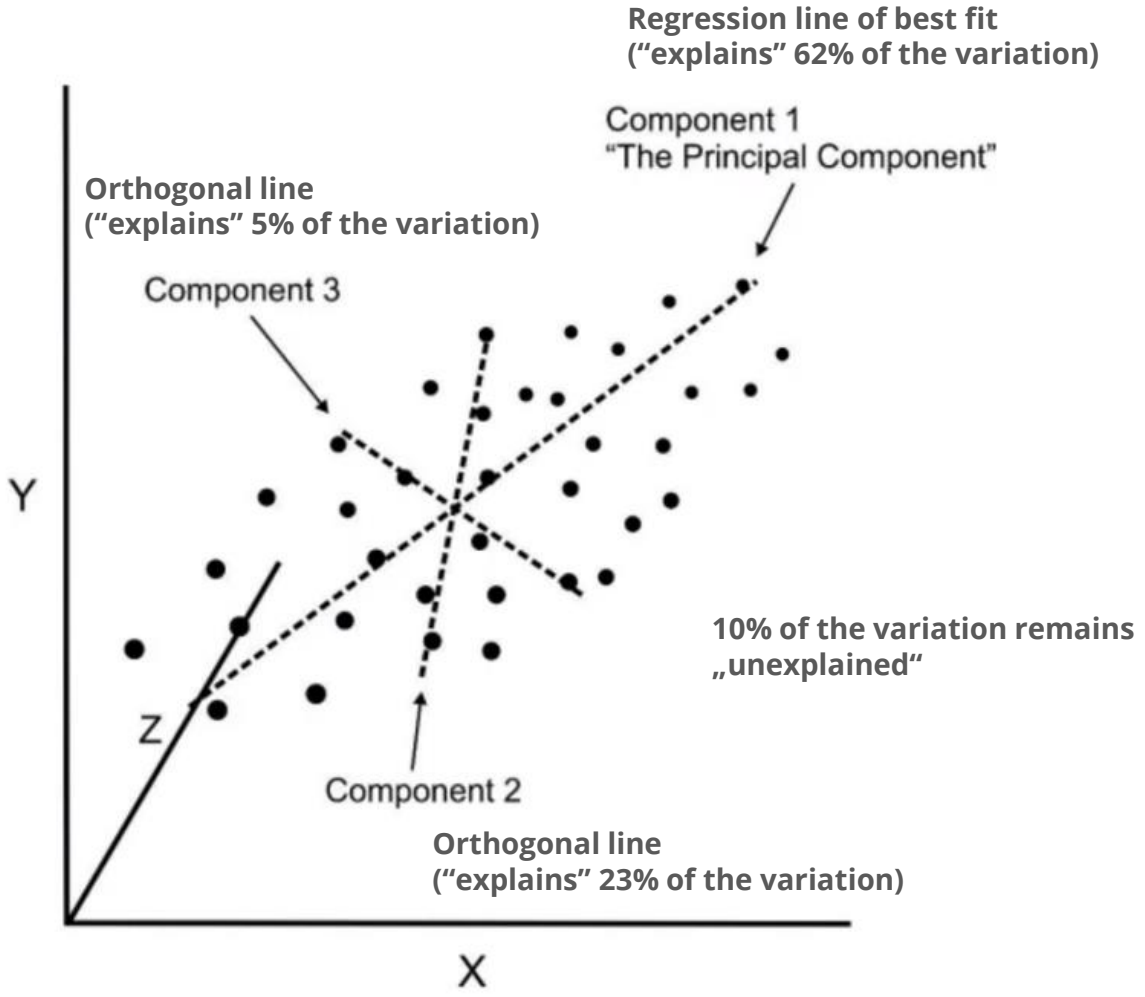


Principal component analysis (PCA)

Where regression determines a line of best fit to a dat set, **PCA determines several orthogonal lines of best fit to the data set.**

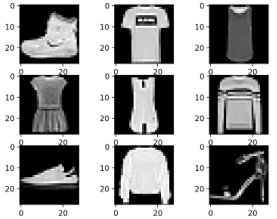
Components are a linear transformation that chooses a variable system for the data set such that **the greatest variance of the data set comes to lie on the first axis, the second greatest variance on the second axis, etc.**

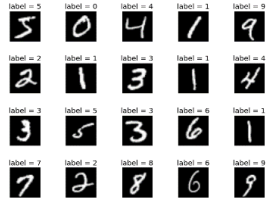
This process allows reduction of the number of variables used in an analysis.

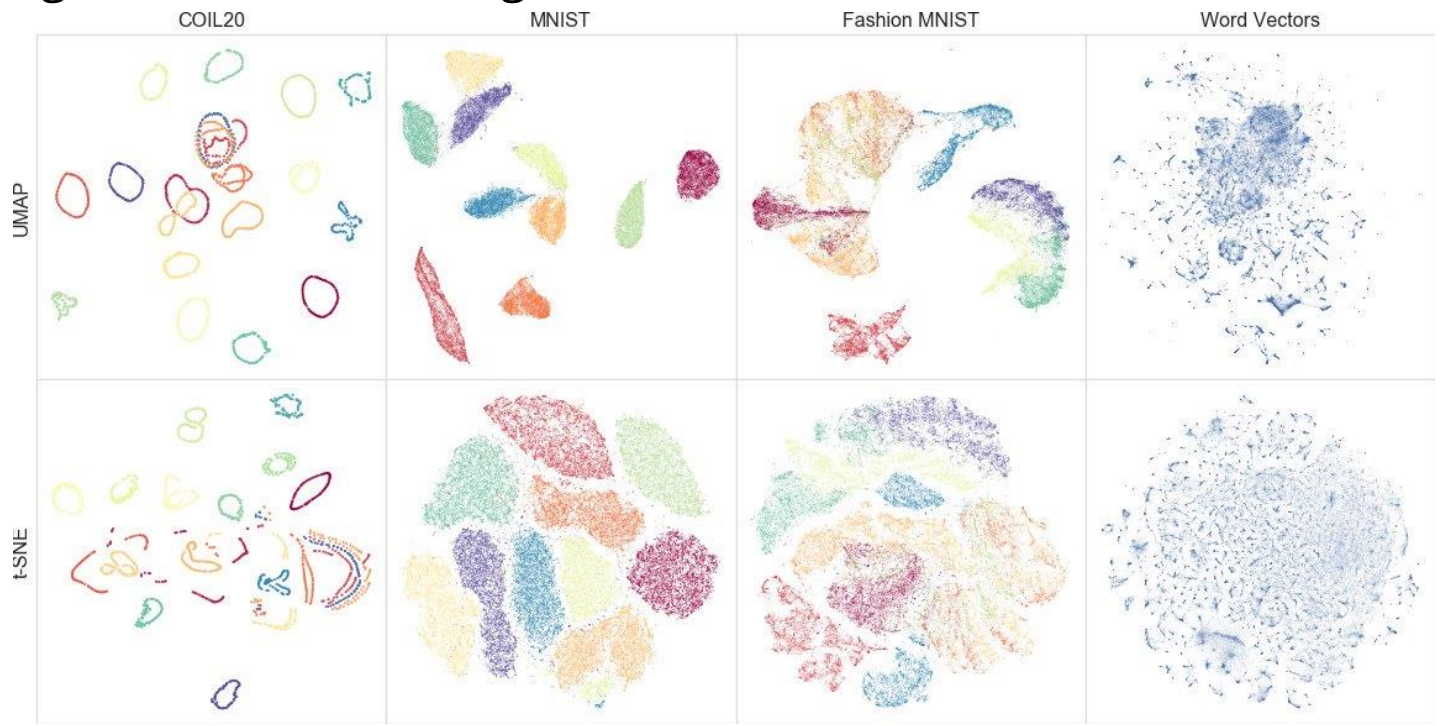


UMAP (Uniform Manifold Approx. and Proj.)

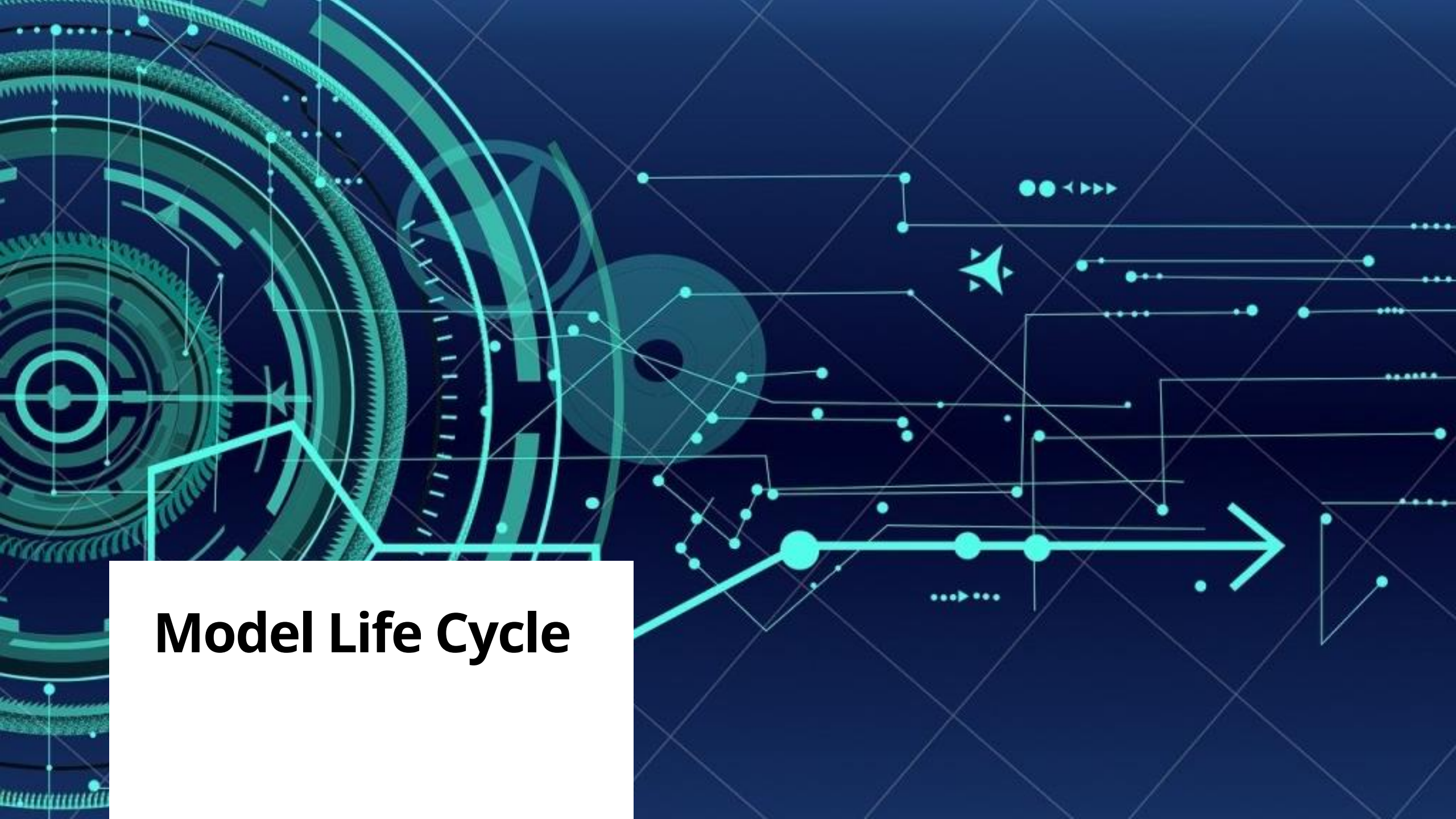
UMAP is like t-SNE, but faster and more general-purpose – while t-SNE doesn't have much use outside of visualization, UMAP is a general-purpose dimensionality reduction technique that can be used as preprocessing for machine learning.





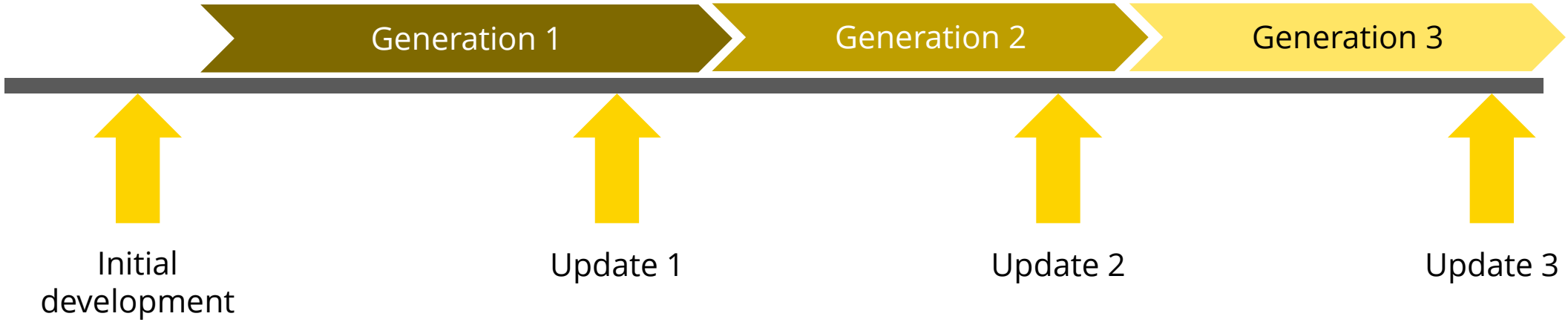


https://umap-learn.readthedocs.io/en/latest/how_umap_works.html



Model Life Cycle

Model life cycle



Initial development

Specification	1 week
Data preparation	2-12 weeks
Transformations	1 week
Modeling	2 weeks
Evaluation	1 week

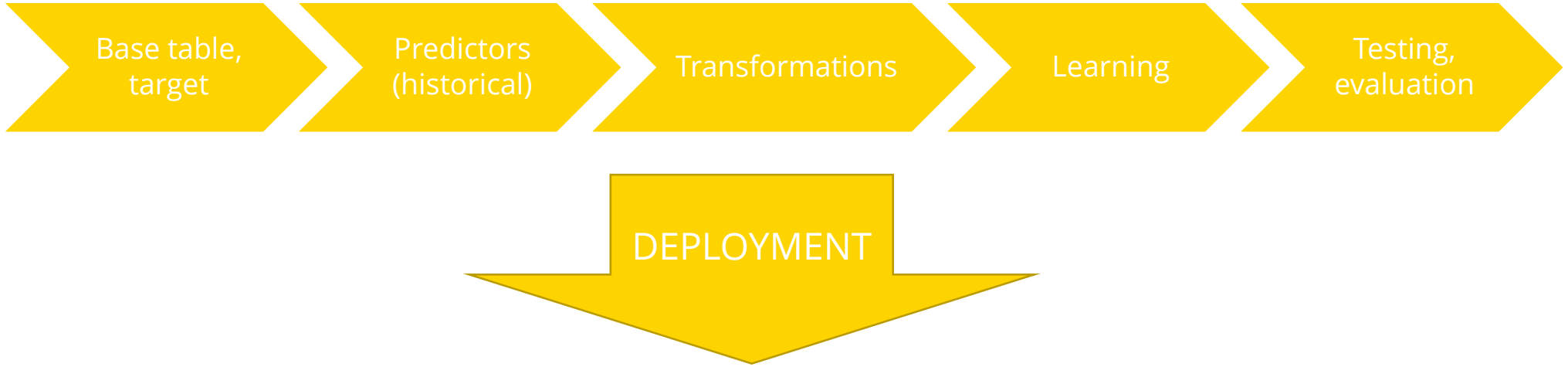
Model update

Not easier than initial development. Why?

Reuse old stuff, but

- different people
- old stuff not very reusable
- comparison with old model required
- changes in data and processes

Development pipeline – SAS, R, SPSS, Matlab, Spark, Python, …



Production pipeline – SQL, Java, C++, C#, SAS, Python, R, …

Home Assignment HA3

Create your own training/testing datasets from Airbnb dataset and prepare your own predictors-target table you will use for training of your model(s). You are free to use SQL or Python for this part of the task.

Your task is to predict whether a listing will be successful as you need to figure out which house to buy so you could rent it successfuly. The success of a listing is determined by number of booked days / total days in calendar being bigger than 75%.

Prepare predictors (be creative) and reason for the selection of each predictor (variance, event rate, ...). Prepare the dataset for the task and train a model of your choosing (reason the selection of the model). Evaluate the model so it can be seen the model is not overtrained.

Describe also predictors used in your final model using appropriate plots from which it will be apparent what is their respective relationship to the target.

Save your SQL and/or script and send it to *pmilicka@deloitteCE.com* with subject **DSI_HA01_surname** for review.

Deloitte.