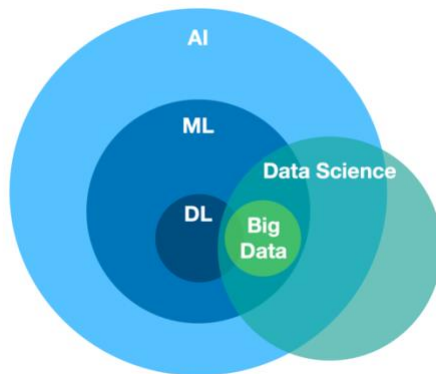


Introduction to Data Science Summary



Artificial Intelligence: describes the ability of machines to perform tasks that would normally require human intelligence, such as: visual perception, speech recognition, decision-making, language translation, image generation. AI involves the development of computer programs that can process and learn from data and make predictions or propose actions based on that learning.

Machine Learning: is a subset of AI focusing on the development of algorithms and statistical models that enable computers to perform specific tasks without using explicit instructions.

ML is about making predictions or inferences from data, and it is a core tool within data science for building models from input data.

Categories of Machine Learning:

- Classical Machine Learning: human intervention required (e.g., for feature extraction)
- Deep Learning: class of ML methods that uses deep neural networks, can perform many tasks without human intervention
- Reinforcement Learning: learning via feedback (rewards/penalties) from actions (e.g, by interacting with the environment)

Data Science: is an interdisciplinary field, combining aspects of statistics, mathematics, computer science, and domain expertise, that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from structured and unstructured data.

Data science encompasses a broad range of techniques for data analysis, including both traditional statistical methods and modern machine learning techniques. It is the science of learning from data.

Four V's of Big Data: Volume (scale of data), velocity (analysis of streaming data), variety (different forms of data), veracity (uncertainty of data). "Big data is data that does not fit into your computer's memory".

Why is it a thing now?

- 1) Increase in computing power: multicore CPUs, GPUs (can train larger models: e.g. deeper neural networks). Cheaper memory (can store data, models).
- 2) Availability of larger data sets (ImageNet, COCO, GPT-3)

- 3) Cloud Computing (AWS, Azure): access to computer power & scaling
- 4) Public / Open Source Software (tensorflow, pytorch): simplifies development & deployment of ML models
- 5) Algorithmic development: significant advances in deep learning (e.g, CNN, transformers). This leads to more efficient, easier to train models.

Data Science Components:

- Data Exploration and Preparation: preparation / data cleaning involves reformatting, grouping, smoothing etc. EDA is a check on most basic properties and is used to expose unexpected features.
- Data Representation and Transformation: transform data type of variables (either numerical or categorical)
- Computing with Data: knowledge of ML methods and their APIs
- Data Modeling: two types:
 - Inferential modeling: develop stochastic model that describes the data, infer properties of the model. Interested in model parameters.
 - Predictive modeling: train models based on data, apply models to predict outcome over some data universe. Interested in the accuracy of the model predictions.
- Data Visualization and Preparation

Overfitting: fit model with higher level of complexity than data, noise and artifacts are fitted along true trends. Overfitted models often fail when applied to future data.

Learning to use the various AI tools is a valuable skill, and saves time, but does not replace learning how to solve problems on your own.

Descriptive Statistics: analysis of data that helps describe, show or summarize data in a meaningful way. Descriptive statistics only concern with the data at hand. They usually do not allow us to make conclusions beyond the data we have analyzed or reach conclusions regarding any hypotheses we might have made.

Inferential Statistics: infer the properties of a large data set (population) from that of a subset. Inferential statistics are techniques that allow us to use these samples to make generalizations about the population from which the samples were drawn. The main methods of inferential statistics are estimation of parameters and hypothesis testing.

Prediction: Predict outcome properties of data points from their feature properties.

$$Y = f(X) + \epsilon \quad \epsilon \text{ is random error (zero mean)}$$

- given set of data points $\{(X_i, Y_i)\}_i$
- want F (estimation of f) such that $F(X)=\hat{Y} \approx Y$ for new data point (X,Y)
- often F treated as black box

The residual $Y - \hat{Y}$ measures how good the approximation F is. How do we get F ? take training sample from population, train algorithm and get F , validate F on other samples.

Causal Inference: we want to know whether feature X causes outcome Y . Complicated because: correlation does not imply causation.

- Randomized Control Trial: two groups (one treatment, one placebo). Compare outcome for the two groups and assess differences statistically. It eliminates selection bias and confounding in treatment assignment.

Mechanistic Analysis: we want to show that changing on parameter always leads to a specific, deterministic behavior in another. Requires perfectly controlled experiments, almost never possible in social sciences, health science.

Common mistakes: interpreting an inferential analysis as casual. Avoid words "causing" and "affects" unless it is a causal inference. Can lead to causation creep: interpretation of results that suggest causation.

Do not use the same data for model building and testing.

Cook-book for a data scientist:

1) Define your analytical question first. The question has to match the data (or viceversa).

2) Tidy the data: each variable you measure is in one column, each different observation in one row, one table for each kind of variable, use descriptive names (e.g., AgeAtDiagnosis, not ADx). Common mistakes: combining multiple variables into a single column, merging unrelated data into a huge file.

3) Checking the data:

- Data wrangling: the process of transforming and mapping data from one "raw" data form into another format with the intent of making it more appropriate and valuable for a variety of downstream purposes such as analytics. The goal is to assure quality and useful data. Data analysts typically spend the majority of their time in the process of data wrangling compared to the actual analysis of the data. Basically always required. Common mistakes: moving to statistical modelling without proper data checking, wrong data coding (e.g., treat categorical variables as numerical), just looking at summaries, not sufficient plots, failed to look for outliers and missing values.

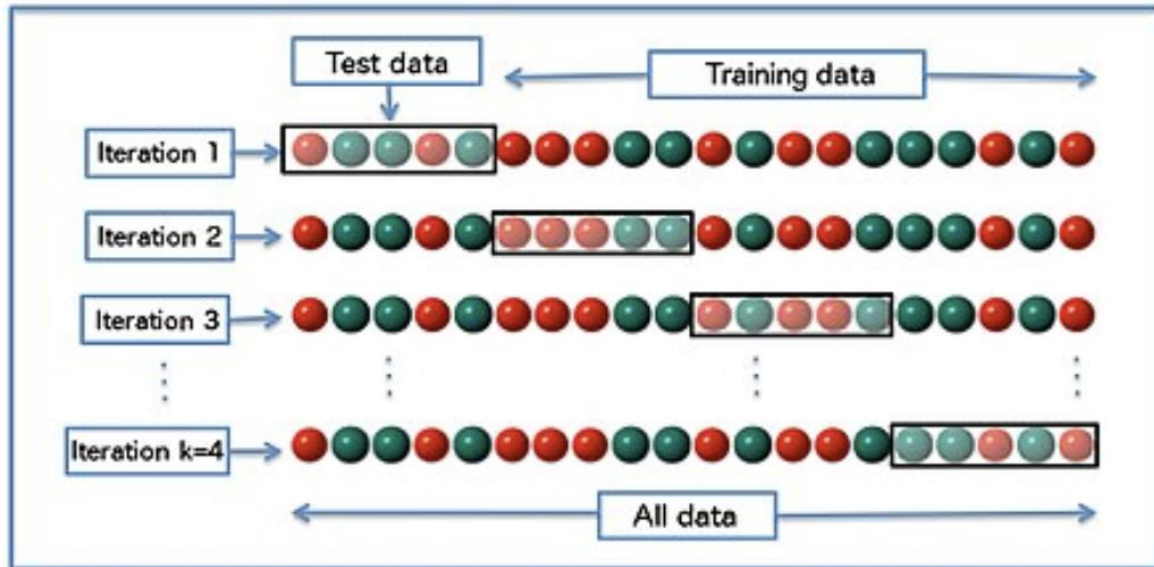
4) EDA: Summarizing and visualizing data before performing formal modelling

- to understand properties of the data
- to inspect qualitative features
- to discover new patterns or associations

Make plots, tables etc quickly (pretty plots for published papers, not for EDA).

Common mistakes: not spending enough time and jumping right to statistical analysis, wasting time on making plots pretty (speed is priority), finding patterns but not exploring their origin (e.g., confounders), failing to look at patterns of missing values and their impact on conclusions (leading to bias).

Prediction: first split sample randomly into training set (70%) and test set (30%). Put test set aside and use it at the very end once and only once to estimate true error rate of your algorithm. To estimate true error of algorithm on smaller data sets: use cross-validation



(4-fold cross validation)



Summary

Descriptive: Example US census

- collect resident type, age, location, etc.
- interpretation/use left to politics

Exploratory:

- search for trends, correlations to generate ideas or hypothesis
- to be tested rigorously later

Predictive:

- build (close) approximation of relationship
- predict individual outcomes on new data set

Inferential:

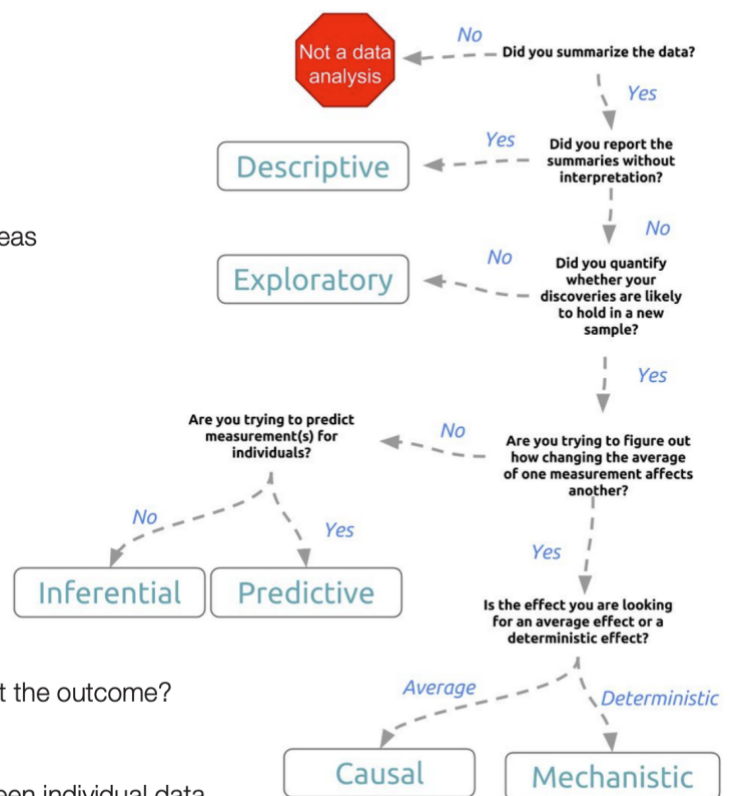
- Make statistical inferences about the data
- Is pattern real or by chance (stat. tests)?
- pattern holds in other data sets?

Causal:

- Learn about causal relationships in the data
- How does changing one measurement affect the outcome?

Mechanistic:

- Learn about deterministic relationships between individual data
- Goal: Explain what happens based on underlying (often deterministic) processes



Supervised Learning: data includes outcomes, goal is to describe or predict outcome as function of input. Examples: regression, classification, random forest.



f .. systematic information
that X provides on Y

Y .. outcome
(or vector of outcomes)

$$Y = f(X) + \epsilon$$

ϵ .. random term with zero mean,
independent of X

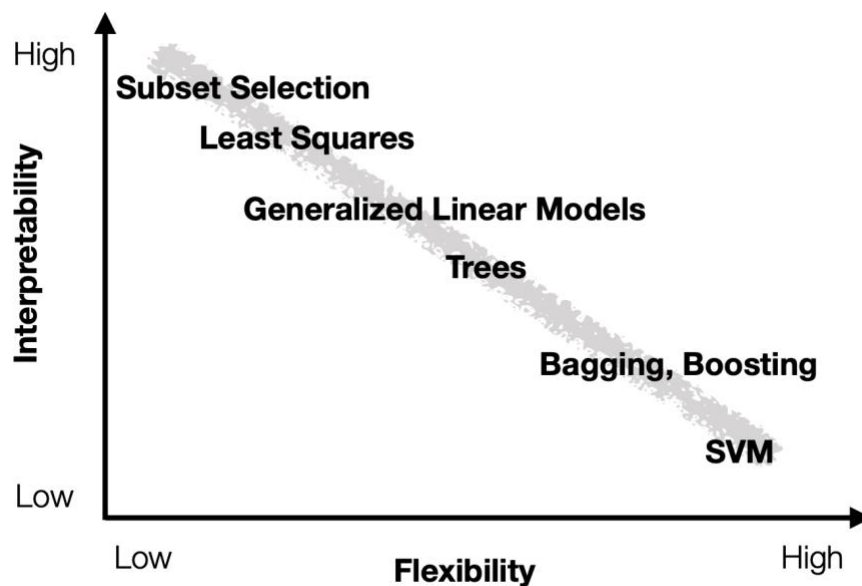
X .. predictor
(or vector of predictors)

f allows to predict new data points and learn about the statistical relationship between Y and X . Rule of thumb: linear models are good for inference; flexible and non-linear models are good for prediction.

Estimation of f :

- Parametric approach: model described by a function of parameters (example: linear model). Procedure to train model with a training data set (LS, MLE).
- Nonparametric approach: no strong assumptions about the form of f . Aim to get some function that is close to the data points

Unsupervised Learning: there are no outcomes, goal is to describe associations and patterns in the data. Examples: PCA, Clustering.



Why choose a more restrictive method to a very flexible one? easier to interpret, have less variance.

How to measure model accuracy?

Mean squared error $MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}(x_i))^2$

MSE computed on training data set:
(square of) 'training error'

MSE computed on a test data set:
(square of) 'test error'

Accuracy of the model: MSE of test error!!

Error: difference between true outcome and expected (average) outcome.

$$\epsilon_i = y_i - f(x_i) = y_i - E[Y]$$

Residual: difference between true outcome and predicted outcome.

$$r_i = y_i - \hat{f}(x_i) = y_i - \hat{y}_i$$

With overfitting there is high test error (low training error), method too flexible.

With underfitting there is high test error, method not flexible enough.

Choose the method with the right amount of flexibility to minimize the test error (via cross validation).

Regression Problems vs Classification Problems:

Problems with numerical/quantitative response: regression problems

Problems with categorical/qualitative response: classification problems

Not always a clear cut: Logistic regression, dichotomous response => classification problem, but also understood as regression on a numerical response variable (log odds).

Parameter Estimation in Simple Linear Regression (only one predictor variable, but similar for multiple linear regression):

- 1) Least Squares: minimize the residual sum of squares
- 2) Maximum Likelihood.
- 3) Gradient Descent

Problems of Linear Regression.

- Non-linearity of the predictor-response relationship: linear regression is linear in the parameters, does not need to have linear relationship ($\log(x)$ or x^{**2} is possible in linear regression). Remedies are polynomial regression or transformation of predictor/response.
- Correlation of error terms: remedies are transformation of response or weighted regression.
- Non-constant variance of error term

- Outliers: a point with response far from the value predicted by the model (far from other response values). Do not blindly remove them, try to understand why it's there, if it is due to an error with data acquisition, remove it.
- High-leverage points: a point with extreme/unusual predictor values (far from other values of predictor)
- Collinearity: predictor variables are highly correlated. How to deal with it: ignore it (collinearity is not a concern if all you want is to predict values), remove highly correlated predictors (variable selection), use a different regression approach that is less sensitive to collinearity (ridge regression or partial least squares (PLS) regression).

Influential point: a point that influences any part of a regression analysis. If excluded, regression results should not change by much.

Outliers and high-leverage points have the potential to be influential, but usually they have to be investigated further to find out whether they are actually influential.

Logistic Regression:

- classification approach for dichotomous variables. Does not model Y directly, but the probability of Y for a given X.

$$\ln \left(\frac{Pr(Y = 1|X)}{1 - Pr(Y = 1|X)} \right) \sim \beta_0 + \beta_1 X$$

- β_0 is the log-odds of $Y=1$ for $X=0$
- β_1 is the increase in log-odds if X increases by one unit
- non-linear (but monotonic) mapping from log-odds to probability

Parameters are typically estimated via MLE.

It does not deal with > 2 classes (extension: multinomial logistic regression).

Classification: has the goal to minimize the fraction of incorrect classifications.

$$\frac{1}{n} \sum_{i=1}^n I(y_i \neq \hat{y}_i)$$

$I(\text{False}) = 0$
 $I(\text{True}) = 1$

actual class of x_i predicted class for x_i

If 2 classes (e.g., logistic regression):

- predict observation x in class 1 if $P(Y = 1 | X = x) > P(Y = 0 | X = x)$, else in class 0.

Bayes Classifier (decision rule):

- assigns each observation its most probable class. It assigns observation x_0 to the class j for which $P(Y = j | X = x_0)$ is largest-
- it has lowest possible test error rate

Discriminant Analysis: makes specific assumptions how predictors for a given class are distributed (for the density functions).

Linear Discriminant Analysis (LDA): find class k for which posterior is largest.

$$\delta_k(x) \equiv x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \ln \pi_k$$

**discriminant
function**

is a linear function of x (→ linear decision boundaries)

Estimate the discriminant function by replacing μ_k , Σ , π_k by their estimators

→ For given x , select class which maximizes the estimated discriminant function

Quadratic Discriminant Analysis (QDA): discriminant function is now a quadratic function of x .

$$\delta_k(x) \equiv x^T \Sigma_k^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma_k^{-1} \mu_k + \ln \pi_k - \frac{1}{2} x^T \Sigma_k^{-1} x - \frac{1}{2} \ln |\Sigma_k|$$

extra terms

discriminant function is now a quadratic function of x

General approach same as in LDA

- estimate μ_k , Σ_k , π_k from training data and insert into (*)
- then, choose class for which discriminant function is largest

QDA allows class specific Σ_k hence it is more flexible than LDA.

QDA has more variance, LDA has more bias.

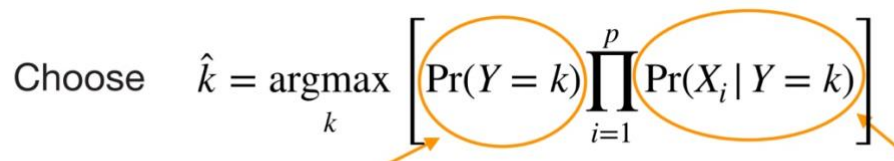
LDA is better if there is a small training data set or if linear boundaries provide a good approximation.

QDA is better if there is a large training data set or if the covariance matrices clearly differ among the classes.

Logistic regression vs LDA: in logistic regression the parameters are estimated from maximum likelihood, in LDA the parameters are estimated from the sample mean and covariance for a normal distribution with common variance. LDA/QDA should be preferred over logistic regression if the gaussian assumptions are met and /or if there are multiple classes.

Naive Bayes Classifier: family of methods that make use of the assumption that features are independent from each other at fixed class. Example: assumes that the probability of word lottery is independent of the word donation in spam email.

Choose $\hat{k} = \operatorname{argmax}_k \left[\Pr(Y = k) \prod_{i=1}^p \Pr(X_i | Y = k) \right]$



Prior (π_k): Usually estimated from the relative frequency of class k in training data set

Different Naive Bayes classifiers differ on how to calculate $\Pr(X_i | Y = k)$

Gaussian Naive Bayes: for continuous features. $P(x_i | Y = k)$ is replaced by pdf $p(x_i | Y = k)$.

Parametric vs nonparametric classification methods:

parametric methods: logistic regression, lda, qda, gaussian naive bayes. They are all based on models with fixed number of parameters estimated from a training set.

nonparametric methods: number of parameters depends on dataset (potentially infinite), or not explicitly parametrized. Example: k nearest neighbors classification.

K nearest neighbor method:

- the trick is to estimate $P(Y = k | X = x)$ and then choose the class k which maximizes this posterior

- k nearest neighbor estimate:

$$\Pr(Y = k | X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = k)$$

\mathcal{N}_0 is set of K nearest neighbors (in training set) of observations x_0

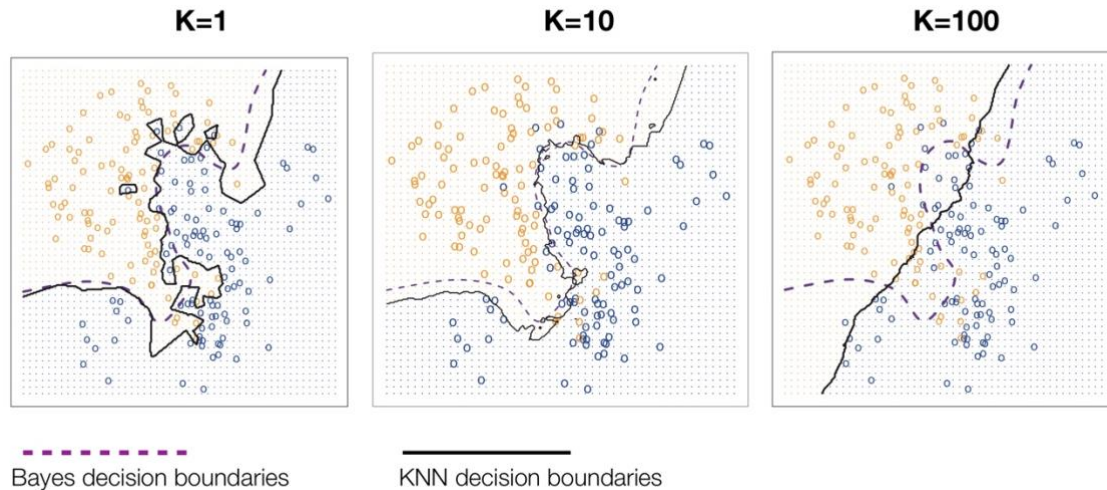
Assigns class that is the majority among the K nearest neighbors (in training set). Allows to model non-linear boundaries between classes. No assumptions about the distribution of the data points. Need to keep all training data when making predictions.

k is a freely choosable hyper-parameter of the method.

Higher K -> smoother decision boundaries (higher bias, lower variance)

Lower K -> irregular decision boundaries (lower bias, higher variance).

There is an optimal K for a given training data set, it can be found with resampling methods.



Support Vector Machines (SVMs): general class of algorithms useful for non-linear decision boundaries. Hyperplanes divide feature space. The objective is to find a hyperplane in an N-dimensional space (N: number of features) that distinctly classifies the data points. The plane we want to find is the one with the maximum margin, i.e the maximum distance between data points of both classes.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane.

Maximal Margin Classifier:

- estimate parameters of a separating hyper-plane from training set
- classify test data $\mathbf{x}^* = (x_1^*, \dots, x_p^*)$ based on sign of $\beta_0 + \beta_1 x_1^* + \dots + \beta_p x_p^*$
- magnitude provides orthogonal distance of point from hyper-plane
- does not provide class probabilities, only "scores"

The position of the maximal margin hyperplane depends only on the support vectors. The maximal margin classifier cannot be used if there is no separating hyper-plane.

Support Vector Classifier: wants to allow some misclassification if it leads to increased robustness and better classification of most training data. Support vectors are vectors that affect the training, they are on the margin or are those violating the margin. Non-support vectors do not affect the training result. It assumes linear boundaries.

larger C = more support vectors (lower variance, higher bias), wider margin
 lower C = fewer support vectors (higher variance, lower bias), smaller margin

Support Vector Machine = Support vector classifier with general form of kernel

$K(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$ linear kernel => support vector classifier

$K(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^d$ polynomial kernel of degree d

$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \sum_{l=1}^p (x_l - x'_l)^2)$ radial kernel (local, only nearby training observations affect classification)

Non-linear kernel can deal with non-linear decision boundaries.

Extension to $K > 2$ classes:

- One-vs-One classification:
 - construct $K(K-1)/2$ SVMs, one for each pair of classes
 - apply each of the SVMs to test observation and take note of predicted class
 - assign the test observation to the most often predicted class
- One-vs-All classification:
 - construct K SVMs each separating one class from the remaining $K-1$ classes
 - compare the scores $f(x)$ of each trained SVM (positive f for SVM k , negative f for rest)
 - assign test observation x^* to class for which $f(x^*)$ is largest

One-vs-One Example

SVM	1 vs 2	1 vs 3	2 vs 3
pred. class	1	1	3

=> pick class 1

One-vs-All Example

SVM	1 vs rest	2 vs rest	3 vs rest
f(class)	5	-3	2

=> pick class 1

Resampling Methods:

Use cases:

- Model selection: estimate performance of different models in order to select the best one
- Model assessment: after having chosen a final model, estimate its prediction (test) error

- Hyper-parametrized optimization: finding optimal value(s) for the hyper-parameter(s) of a method

Cross-validation: repeated training set / validation set split -> test error

Bootstrapping: recreating "new" data sets from old ones -> quantify uncertainty

Hyperparameter: parameter, such as the learning rate or choice of the optimizer, whose values control the learning process and determine the values of the model parameters.

Validation:

We want to estimate the test error of our trained model (compared model predictions with the true values) but testing on the training data (training error) underestimates the error).

Basic strategy: test model on separate validation set not used in training.

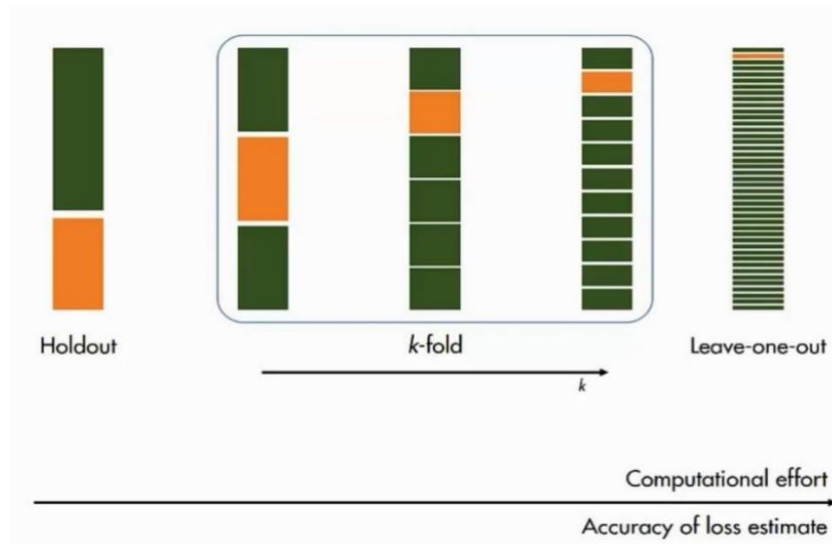
The validation set should be a random subset of the original data.

Simple approach, but:

- high variance, estimated test error depends on split
- inefficient because validation set is never used to inform the model
- test error overestimated as only fraction of full data set used in training the model

Leave-one-out cross-validation: We compute models n times and $n-1$ sets are in the training set and one is the validation set. One observation at a time is omitted, and the remaining observations are used to fit the model and compare the prediction against the omitted observation. LOOCV is form of k -fold where k is the size of the dataset.

K-fold cross validation: we fit the model k times. There are multiple subsets, one is the validation set and the rest $n-1$ is in the training set, we perform this operation for k different validation sets and at the end we compute k different models. $1/k$ is the percentage of data we use as the validation set (e.g. if we have a 3-fold cross validation each time we use 33.3% of the data as validation set and the remaining 66.6% as the training set). This method of cross validation is usually used in complex models (neural networks, etc.). After selecting the best model out of all the computed models, which is the one with the smallest mean squared prediction error, we take the test set and calculate the testing error of the model against it. If we have n data points, n -fold-cross validation is the same as leave-one-out cross validation. In k -fold cross validation, each observation is used $k-1$ times in a training set, and exactly one time in a validation set. Be careful not to test the model by re-using the data that has been used for fitting the model.



Bootstrapping: method that involves sampling repeatedly with replacement (one observation can be chosen multiple times in the same sample) from a sample of a population to estimate its parameters. It is a resampling technique to approximate distributions based on an empirical sample. It is a tool of model validation.

Some people might never be sampled (first sample doesn't contain them or never picked in second sample) others will be sampled multiple times. We sample from a sample. The randomly chosen sample looks quite the population it came from so we can make assumptions about the distribution of that population (e.g. normal, binomial).

In bootstrapping, the size of the bootstrapped sample is usually the same as the size of the original sample we picked from a population. The sample size is fix.

Tree: a cycle-free directed graph which contains one node r (root) such that there is exactly one path from r to each node.

Tree-based methods: idea is to segment the feature space into simple sub-regions and compute a prediction value (mean or median or mode etc) for each subregion. For new observations find which subregion it belongs to and assign the prediction value of the region.

Decision trees are used for regression (numerical outcomes) and classification (categorical outcome).

Regression Trees (Decision trees for Regression):

- divide predictor/feature space (X_1, \dots, X_p) into J distinct and nonoverlapping regions $R_1, \dots, R_J \rightarrow$ find a segmentation R_1, \dots, R_J that minimizes RSS.

- prediction for any observation falling into region R_j is the mean of responses of responses for the training observations in R_j .

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α .
3. Use K-fold cross-validation to choose α . That is, divide the training observations into K folds. For each $k = 1, \dots, K$:
 - (a) Repeat Steps 1 and 2 on all but the k th fold of the training data.
 - (b) Evaluate the mean squared prediction error on the data in the left-out k th fold, as a function of α .Average the results for each value of α , and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α .

How to build a tree:

- Recursive binary splitting:
 - top-down (begins at root)
 - greedy (best split is made for that particular step, no looking ahead)

Classification trees (Decision trees for Classifications): since the outcome is categorical, the leaf nodes are categories. We want to predict the most commonly occurring class (bayes decision rule). Building tree is similar to regression trees. The goal in classification is to minimize the classification error rate.

$$E_j = 1 - \max_k (\hat{p}_{jk})$$

Used during **pruning**, but not during the tree build!

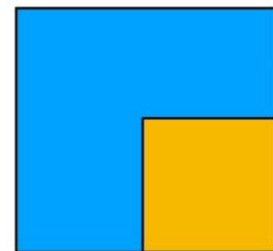
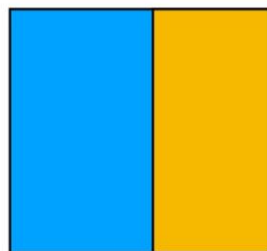
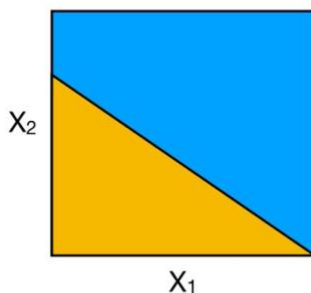
- E_j is fraction of training observations in region j that are not in most common class
- \hat{p}_{jk} is fraction of training observations in region j that are in class k
- \hat{p}_{jk} is estimate of prob. $\Pr(k|j)$ that an observation in region j is in class k

Gini Impurity: measures probability that a randomly chosen element would be labeled incorrently if it was labeled (randomly) according to the class distribution. If gini impurity of region j is 0 then it means no impurity: all elements in j have the same class.



Linear methods vs trees

When to use which?



Classification:



logistic regression, LDA

either one

classification trees

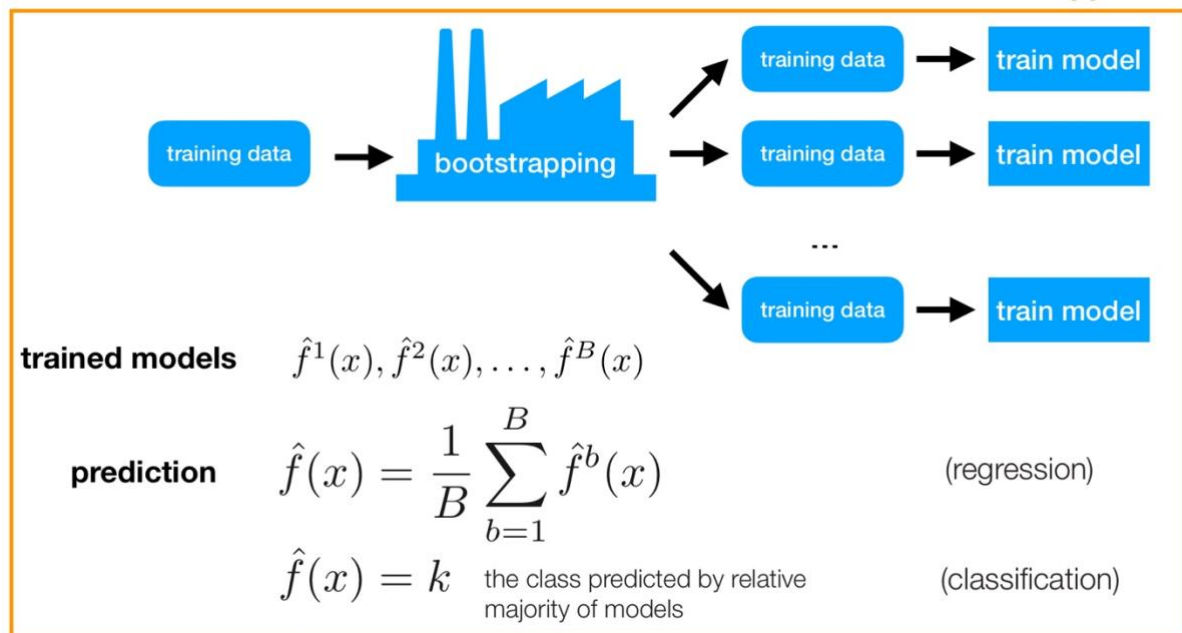
Regression:

linear regression

either one

regression trees

Bagging: abbreviation of Bootstrap aggregation. It tries to reduce the prediction variance by artificially increasing the training data. A bootstrap approach resamples from the original (training) dataset and creates a set of additional training sets of equal sizes. To each of the samples a simple classification algorithm is applied, and a separate prediction model using each training set is built. We then take the average of the resulting prediction for the classification. Deep decision trees.



Boosting: based on the idea that it is possible to construct a strong classifier from many weak classifiers. It calculates the output using several different models and then averages the result using a weighted average approach. The key is to wisely select the weak classifiers and to appropriately weight their result. A weak classifier is a classifier that performs slightly better than random guessing. The weak classifiers are trained on different subsets of the data and their predictions are combined in a way that gives more weight to the instances that are misclassified by the previous classifiers. Small trees, where first tree learns a bit about data, second a bit more, third even more and so on -> incremental approach. Final prediction is the sum of the individual predictions (weighted sum of the different trees).

Random forest: most often used method out of the tree methods. It is similar to bagging and decorrelates the trees: before fitting each new tree or split a random subset of predictors/features is chosen. For each tree we choose approximately predictors. At each split in the tree, the algorithm is not allowed to consider a majority of the available predictors. Individual trees are built based on a subset of features and observations. Random forests overcome the problem of highly correlated trees (problem of bagging) by forcing each split to consider only a subset of the predictors. This approach prevents strongly correlated trees that may occur if there are very prominent predictors. Once a forest consists of a (determined) number of trees, the classification for a new one is given by the majority vote based on the classification of each tree.

K-means algorithm: clustering method which has an a priori fixed number of clusters. The clusters are here represented by a mean vector (representing the “center” of the cluster).

Step-by-step algorithm of the k-means clustering method:

- 1) Start with k cluster centers
- 2) Assign observations to the nearest cluster center
- 3) Recompute the k cluster centers (mean of observations assigned to each cluster)
- 4) If centers are the same as before then stop, otherwise go to step 2.

Assignment step (k means algorithm): Each observation is assigned to the cluster whose mean yields the smallest within-cluster sum of squares. Since the sum of squares is the squared Euclidean distance, this is intuitively the “nearest” mean.

Update step (k means algorithm): The centroids or means of the observations in the new clusters are calculated with the formula of the arithmetic mean. Since the arithmetic mean is a least-squares estimator, this also minimizes the within-cluster sum of squares.

PCA: One of the main motivations for PCA is to reduce the complexity of the data while retaining as much of the original information as possible. In many applications, the data has many features and dimensions, and it is often difficult to make sense of the data or to find relationships in it. By reducing the dimensionality of the data, PCA makes it possible to visualize and understand the data in a simpler and more interpretable form, while still capturing the essence of the data in a few principal components, which explain most of the variation in the data. Dimensionality reduction has multiple benefits: first of all we can better visualize the data, and with fewer dimensions we can make a better generalization about the data. PCA also helps minimizing the residuals (squared distance) and maximizing the variance (squared distance).

Principal Component Analysis (PCA) is a technique for dimensionality reduction. PCA expresses the data on a new coordinate basis. The new basis vectors, called Principal Components, are linear combinations of the original variables. They are chosen in such a way, that the first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. It is achieved by choosing as the PCs the eigenvectors of the empirical covariance matrix of the data. PC1 corresponds to the eigenvector with the largest eigenvalue, PC2 to the eigenvector with the second largest eigenvalue and so on. Because PCs are ordered by the amount of the variability explained, one can represent the data using only a subset of PCs without losing a lot of information, which allows to reduce the number of dimensions, either for further analysis, visualization, or data compression.

The method we will see in this lecture is called Feature extraction.

Feature extraction: we create “new” independent variables where each new independent variable is a combination of the old independent variables. We drop the least important/significant variables/dimensions. Because these new independent variables are based on the old ones, we are still keeping the most valuable parts of our old variables, even when we drop one or more of these new variables.

The principal components are orthogonal to each other. Because they are orthogonal to one another, they are statistically independent of one another. To obtain principal components perpendicular to each other data has to be centered (center = true).

We should not use categorical variables for a PCA analysis, only quantitative variables.

It is better to have a highly correlated set of variables to perform a PCA rather than a low correlated set of variables (Cov/Correlation matrix).

When we perform PCA manually, the `pca` with function in R is calculated on the realisations of X_1 and X_2 while the Eigenvalues of Σ are from the theoretical distribution of X_1 and X_2 . As we increase n , the two converge.

Dimensions are called PC1, PC2 and PC3 instead of x , y and z . There might exist “better” dimensions than the original ones to represent the data. It might be possible to use only a subset of dimensions (and skip/discard the ones that explain the least amount of variance/information about the data). The PCA transformation (rotation) ensures that the horizontal axis PC1 has the most variation. The vertical axis PC2 has the second-most and so on. Rotation consists of switching from a vector basis to another one.