2695 Introduction to Machine Learning
Masters Program in Economics, Finance and Management

NOVA SCHOOL OF BUSINESS & ECONOMICS

4a  Video Lecture

DATA PREPROCESSING

The Importance of Data Quality in Machine Learning

- **Garbage in, garbage out:** output of the machine learning model will not be correct or useful if the input data is invalid, regardless of how well we performed the model training.
- Insufficient quantity of training data
- Lack of informative features
- Presence of irrelevant features
- Non-representative training data
- Poor-quality data

What are good features?

- Better features usually help more than a better model.
- Complexity in features allows us to use less-complex models that are faster to run, easier to understand and easier to maintain.
- Good features
  - Provide useful information about the target variable
  - Generalize well to new data (not unique ids with no pattern, not exact timestamps)
  - Balance between simplicity and expressiveness

Types of Features

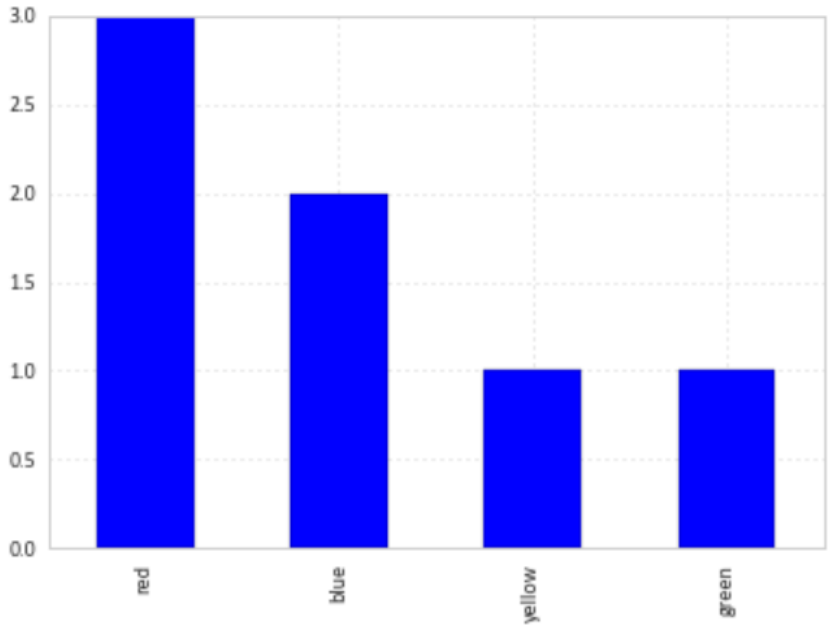
- **Categorical** feature represents categories.
  - **Nominal:** values do not have any intrinsic ordering.
    - Example: color, school from which a person graduated
  - **Ordinal:** values can be ranked.
    - Example: ratings of a restaurant's new menu: dislike, neutral, like

- **Numerical** features represents numbers.
  - **Discrete:** Have only a finite set of values.
    - Examples: number of children, number of students in a class
    - Often represented as integer variables.
    - Note: binary features are a special case of discrete features.
  - **Continuous:** Have real numbers as feature values.
    - Examples: temperature, height, or weight
    - Continuous features are typically represented as floating-point variables.

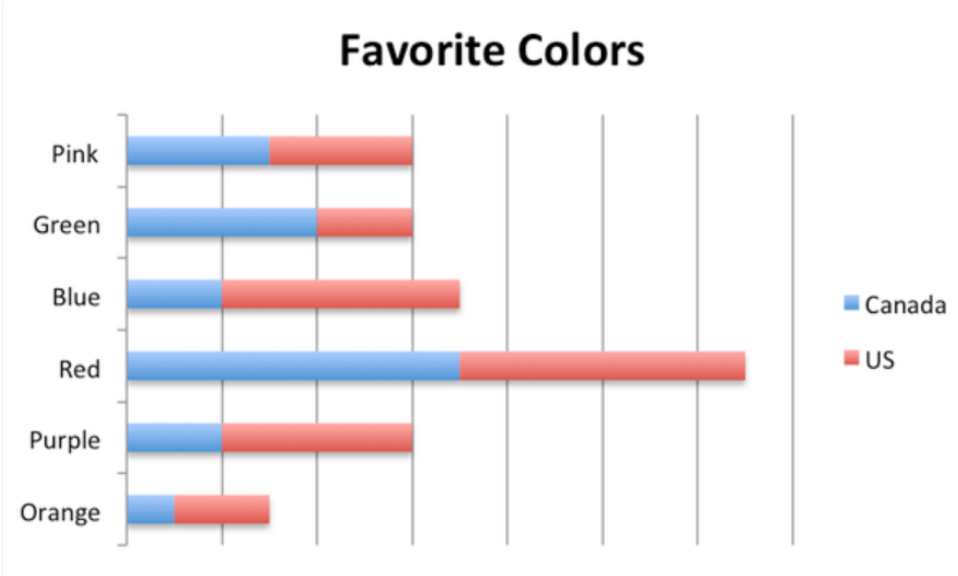
Analyzing categorical features

For a categorical feature we need to look at the:

- distribution
- relationship with the target
- potential data issues: missing values or imbalances



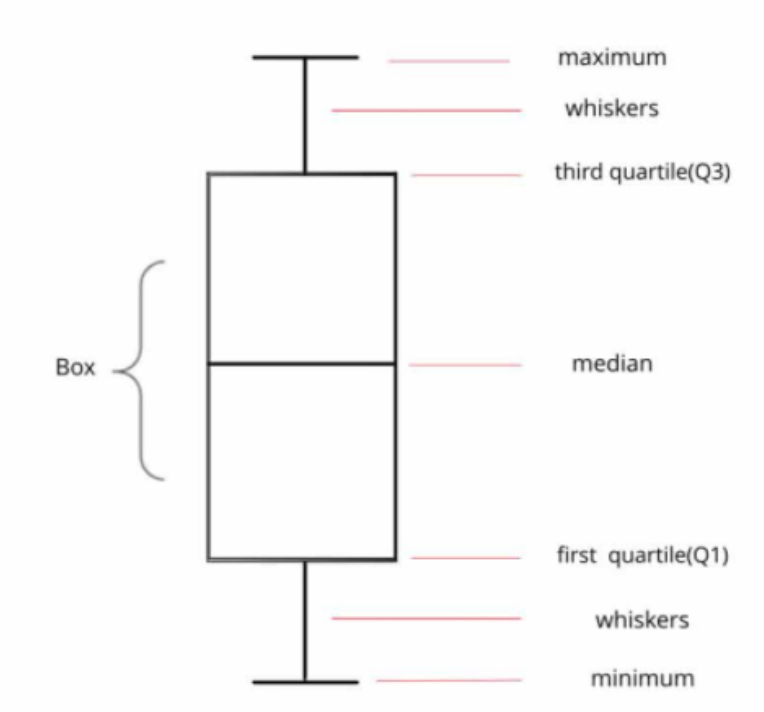
Bar plot



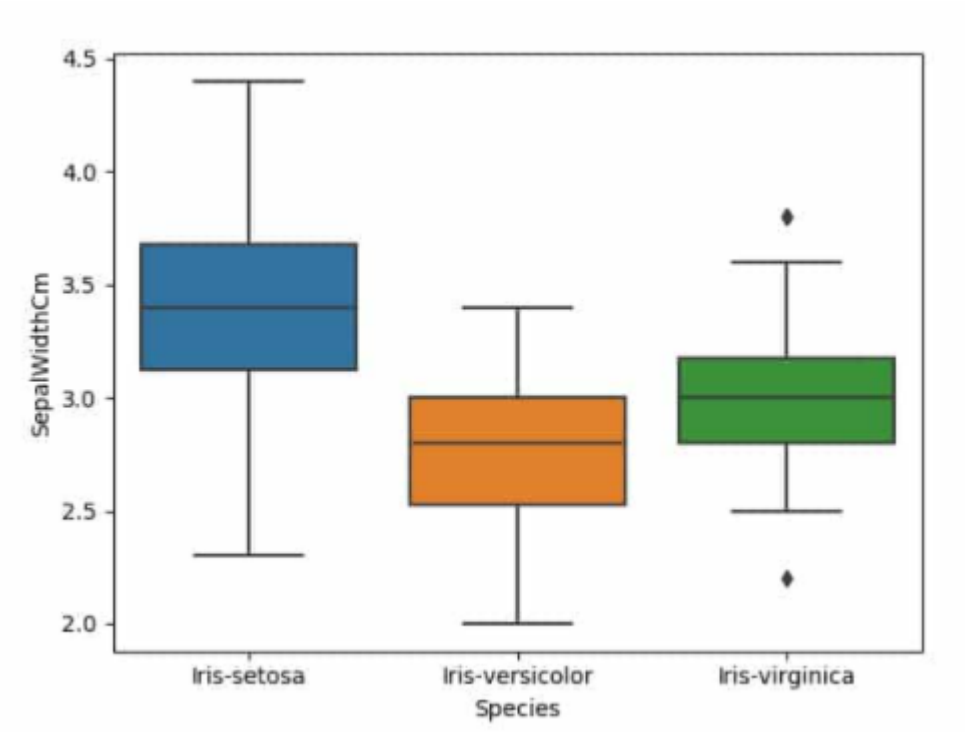
Stacked bar plot

Relationship between a categorical and a numerical feature

The relationship between a categorical and a numerical variable can be visualized with a box plot.



box plot



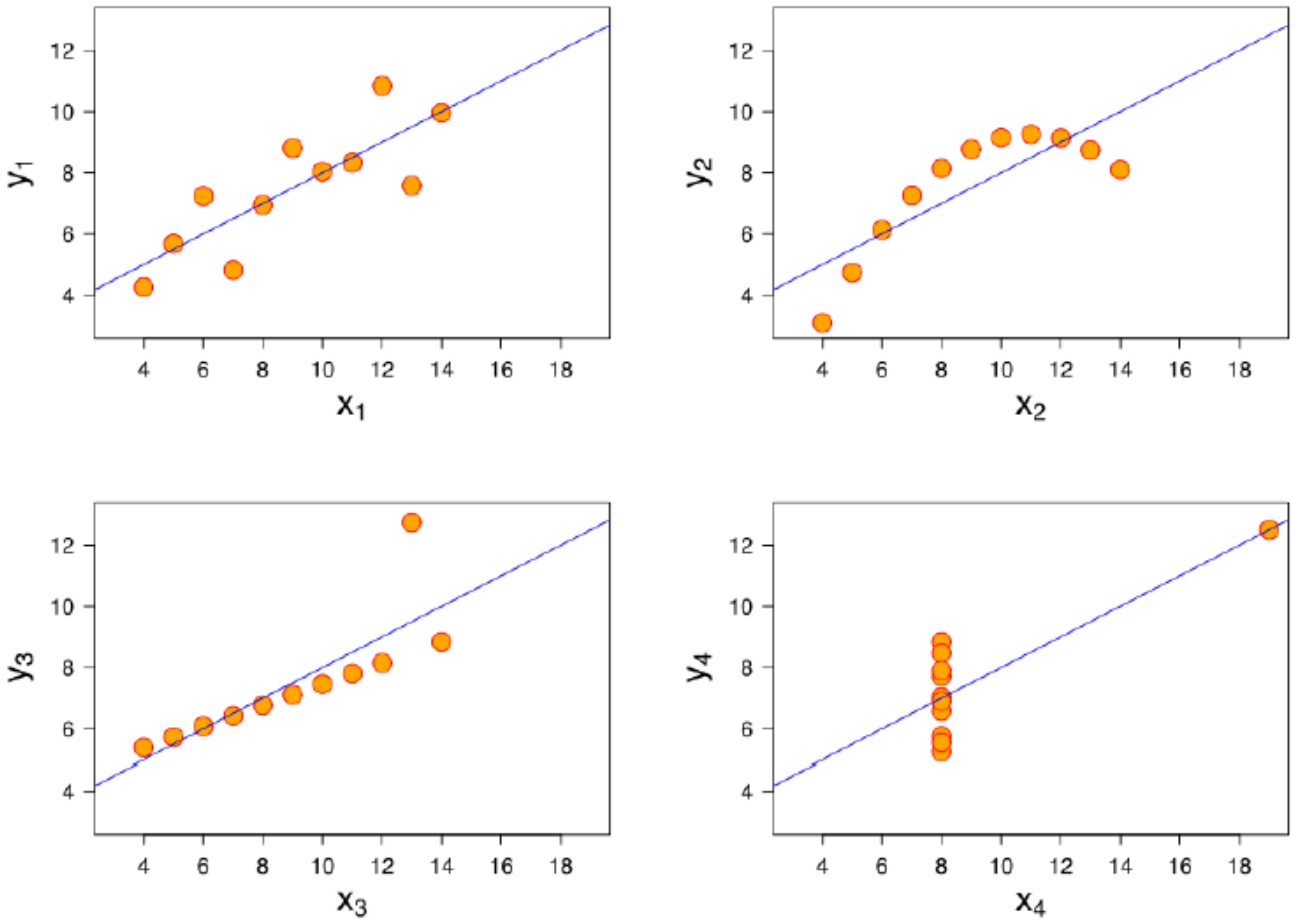
Summary statistics of continuous features

- Mean: average value
- Median: value such that half values are larger/smaller
- Quantiles: value such that 'k' fraction of values are larger
- Range: minimum and maximum values
- Variance: measures how far values are from mean
- Standard deviation: square root of variance

- Data: [0 1 2 3 3 5 7 8 9 10 14 15 17 200]
- Measures of location:
  - Mean(Data) = 21
  - Mode(Data) = 3
  - Median(Data) = 7.5
  - Quantile(Data,0.5) = 7.5
  - Quantile(Data,0.25) = 3
  - Quantile(Data,0.75) = 14
- Measures of spread:
  - Range(Data) = [0 200].
  - Std(Data) = 51.79
  - IQR(Data,.25,.75) = 11
- Notice that mean and std are more sensitive to extreme values ("outliers").

Slide credit: Mark Schmidt

Summary statistics sometimes can be misleading



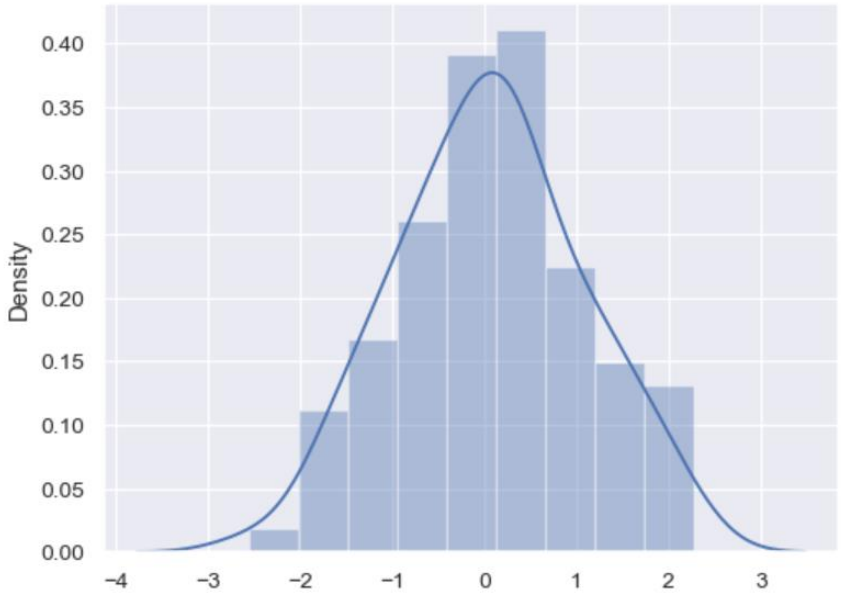
Amcomb's quartet:

- Almost same means
- Almost same variances
- Almost same correlations
- Look completely different

https://en.wikipedia.org/wiki/Anscombe%27s_quartet

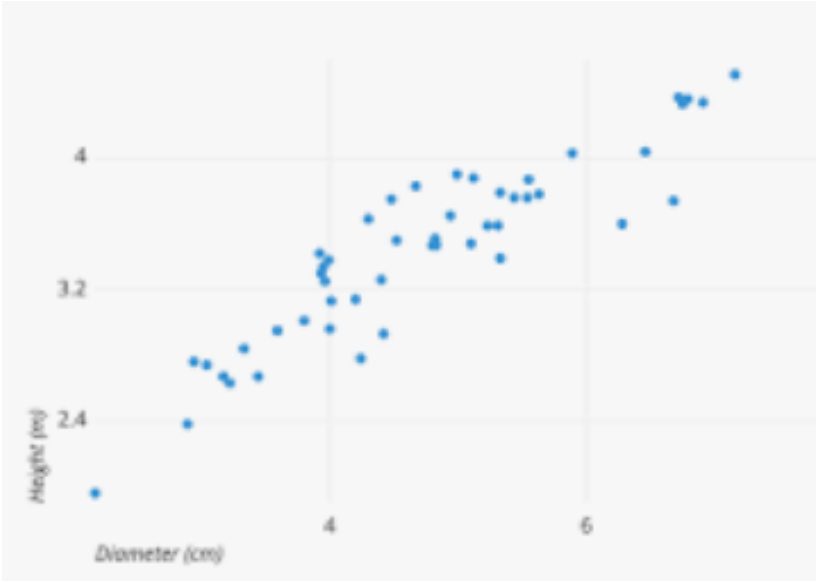
Visualizing continuous features

To examine the distribution of a continuous variable, we can use a histogram and a probability density function plot.



histogram

To see the relationship between two numerical variables, we can use a scatter plot.



scatter plot

Missing values in data

- A common characteristic of real-life data is the presence of missing values (incomplete dataset).
- Most of the existing machine learning models do not work well with missing values.
- The causes of missing values can be categorized into three primary types:
  - **Value missing completely at random**
    The probability of any value in that feature being missing is the same, and the value missing has nothing to do with the observation being studied.
  - **Value missing at random**
    The probability of an observation being missing depends on the other features in the dataset.
  - **Value missing, but not at random**
    The fact that the data is missing depends either on the hypothetical value or on the target value.

Complete removal of rows or columns of missing values

- One of the most intuitive and simple methods.
- Can be used regardless if the feature is numerical or categorical.
- Should be applied only if data is missing completely at random, and relatively small number of examples have missing values.

- Typically, columns of missing values can be completely removed when the there are many more missing values than the other values.
- The removal of rows and columns could mean losing important information about the data.
- When the model is put in production, it will not automatically know how to handle missing data.
- Real life dataset: amount of missing data is never small, and therefore complete removal of rows is not an option.

Mean/Median & Mode Imputation

Imputation is the act of replacing missing data with statistical estimates of the missing values.

- Numerical variables:

Missing values are substituted with a mean or a median of the **training data** feature values: **Data leakage if test data values used to calculate the statistic.**
Mean: normally-distributed distribution
Median: skewed distribution

- Categorical values

Mode imputation means replacing missing values by the mode, or the most frequent- category value.

- Advantage: simple
- Disadvantage: distorts the distribution of the dataset, decreases the variance, leads to an over-representation of the most frequent category.

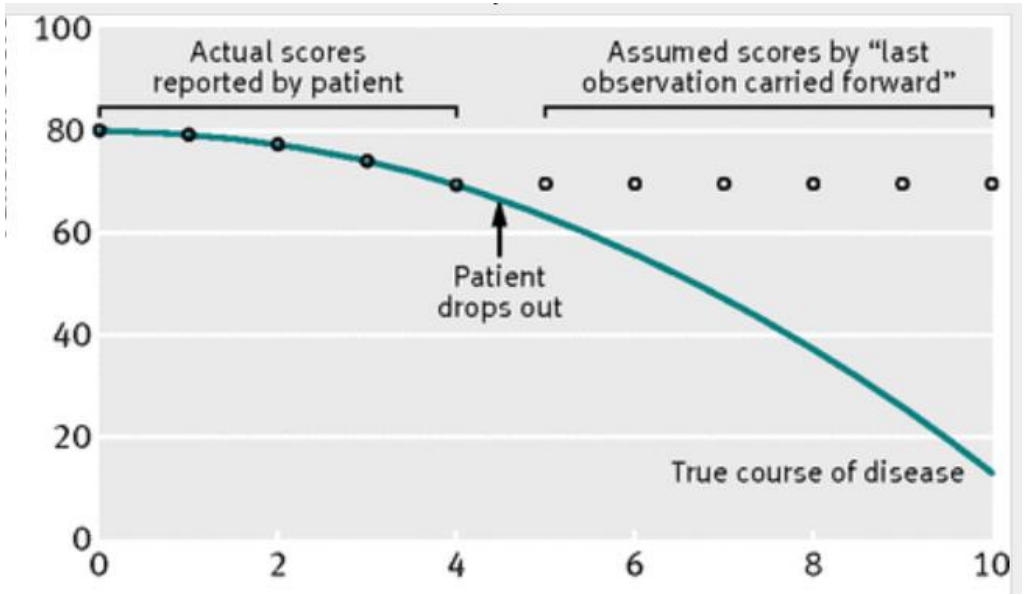
- Alternative: Using predictive models to impute missing data

Imputing with a value not present in the values of the feature

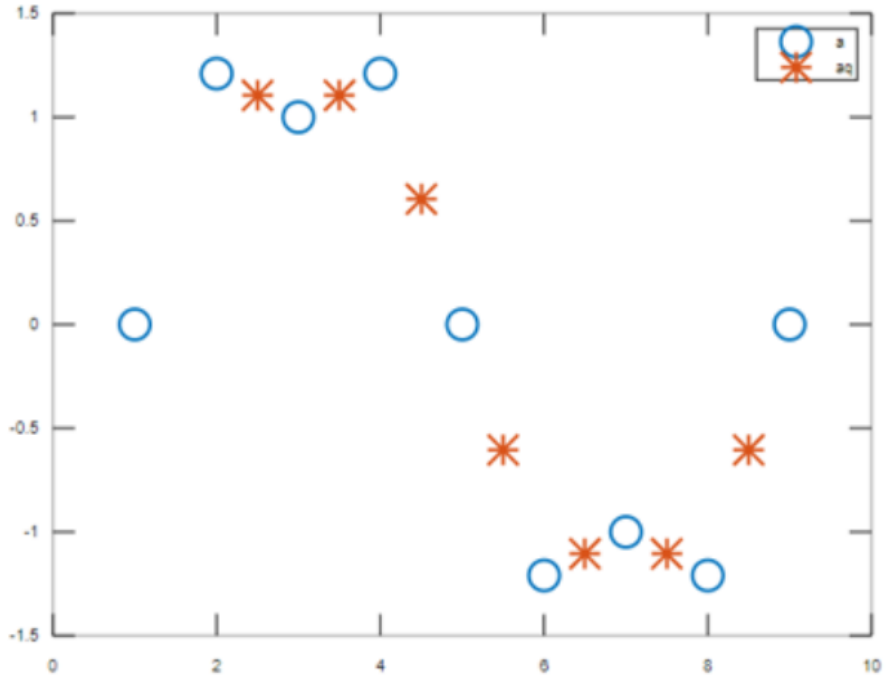
- Can be used when the data is not missing completely or at random
- For numerical features: pick a value outside the normal range
- For categorical: add a category “missing”

Special case: Time series specific methods

- Last Observation Carried Forward (LOCF) & Next Observation Carried Backward (NOCB)
- Linear Interpolation



Last Observation Carried Forward



Linear Interpolation
orange points interpolated

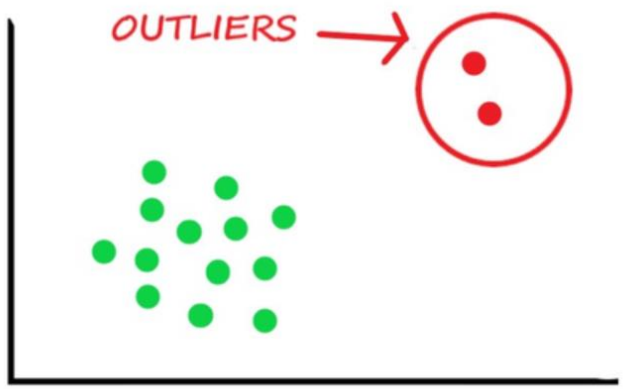
Outliers

- **Outlier** is a data point that is distant from other observations. It is an atypical point that lies far away from other values.

- Causes of outliers:
  - Data errors (data entry errors, measurement errors, data processing errors)
  - Variance in data (natural outliers)

- Sometimes main goal is identification of anomalous data points: fraud, intrusion, sensor failure detection.
- This context: finding outliers during data preparation for another task.

- Outliers: impact many statistics: mean, standard deviation.
- Some machine learning algorithms: sensitive to the outliers, causing longer training times, less accurate models and ultimately poorer results.

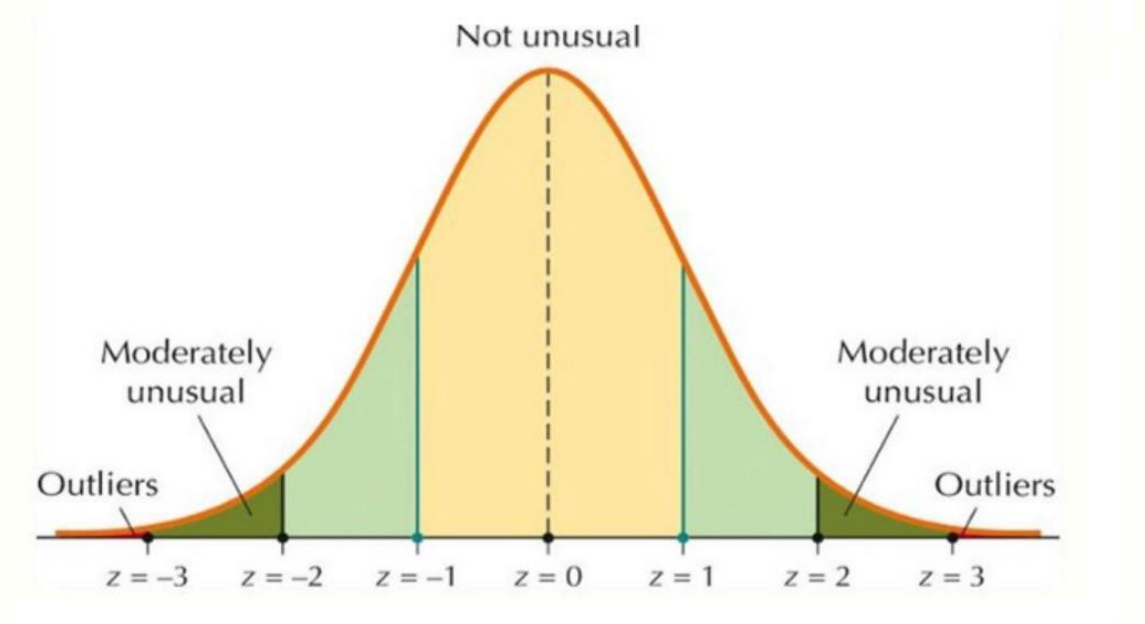


Detecting outliers with Z-score

- Z score: measure of how far a data point is from the mean.
- Approach assumes a Gaussian distribution of the data.
- Data points are standardized.
- Outliers are the data points that are in the tails of the distribution and therefore far from the mean.
- Threshold: commonly used values 2.5, 3.0 and 3.5 .

$$Z = \frac{x - \mu}{\sigma}$$

Score, Mean, SD



Detecting Outliers with Z-scores. Image source: https://laptrinhx.com/

Handling outliers

- If the cause for the outlier is a data entry error (age of a participant or number of children too large):
  - Remove the row with the outlier
  - Impute with mean/median/mode
  - **Top coding (bottom coding):** value of outliers is substituted with an upper bound (lower bound)
    Example: set all ages of sprint runners above 80 to 80, or number of children below 0 to 0.

- **NOT acceptable** to drop an observation just because it is an outlier.

- If the model used is sensitive to outliers, top and bottom encoding could be used.

Transforming categorical to numerical features

- Many machine learning algorithms require that their input is numerical and therefore categorical features must be transformed into numerical features before we can use any of these algorithms.

- There are many different ways to transform categorical to numerical feature:
  - One hot encoding
  - Ordinal encoding
  - Target encoding

One Hot Encoding

- For each category of a categorical feature, we create a new binary variable.
- These newly created binary features are known as **dummy variables.**
- Map each feature value to a corresponding dummy variable: 0 represents the absence, and 1 represents the presence of that category.
- Used for transforming nominal features (features that do not have ranking associated).

Index	Animal
0	Dog
1	Cat
2	Sheep
3	Horse
4	Lion

One-Hot code

Index	Dog	Cat	Sheep	Lion	Horse
0	1	0	0	0	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	0	1
4	0	0	0	1	0

One Hot Encoding (Dummy Encoding)

- Feature with k categories: k new features.
- We do not need all k new features: sufficient to use k-1.
- **Multicollinearity:** dependency between the independent features, serious issue in some machine learning models, but not all
- In tree-based models, typically this last column is not dropped.
- One hot encoding: k columns
- Dummy encoding: k-1 columns (but sometimes these terms are used interchangeably)

Index	Animal
0	Dog
1	Cat
2	Sheep
3	Horse
4	Lion

One-Hot code

Index	Dog	Cat	Sheep	Lion	Horse
0	1	0	0	0	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	0	1
4	0	0	0	1	0

Ordinal Encoding

- Used for ordinal variables.
- Each category is assigned a value from 1 through k, where k is the number of categories for the feature.
- Ordinal encoding: ensures that the encoding retains the ordinal nature of the feature.

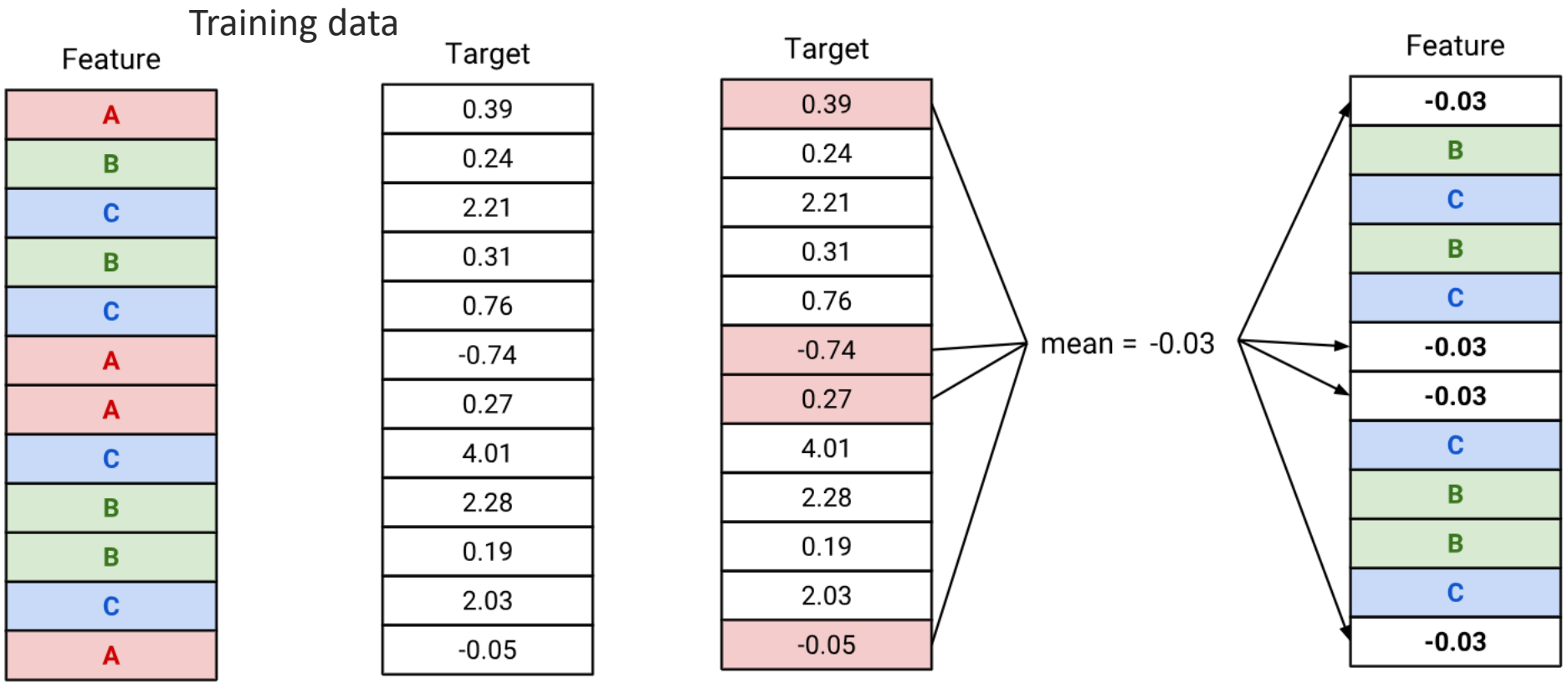
Rating
Bad
Very Good
Good
Good
Very Bad
Good
Medium

Rating
2
5
4
4
1
4
3

- In the example above: very bad< bad < medium < good < very good
- “Bad” to “Medium” distance is similar to “Good” to “Very Good” distance.

Target (mean) Encoding

- For each category in the categorical feature, the **mean value of the target variable** is calculated on the **training data**.
- Can be used if the number of categories is high, helps in faster learning.
- It might lead to overfitting.



Value A is encoded as the mean of target values for all examples where feature has value A

https://brendanhasz.github.io/2019/03/04/target-encoding

Discretization of continuous numerical features

Discretization transforms: transforming numerical variables to discrete ordinal variables.

It is especially relevant for counting based methods such as decision tree type of approaches. But it is not recommended for distance-based methods.

- improved interpretability
- reducing sensitivity to small variations

With discretization, we sacrifice information and make data more regularized.

Age
23
23
22
25
19
22

< 20	>= 20, < 25	>= 25
0	1	0
0	1	0
0	1	0
0	0	1
1	0	0
0	1	0

Feature Scaling

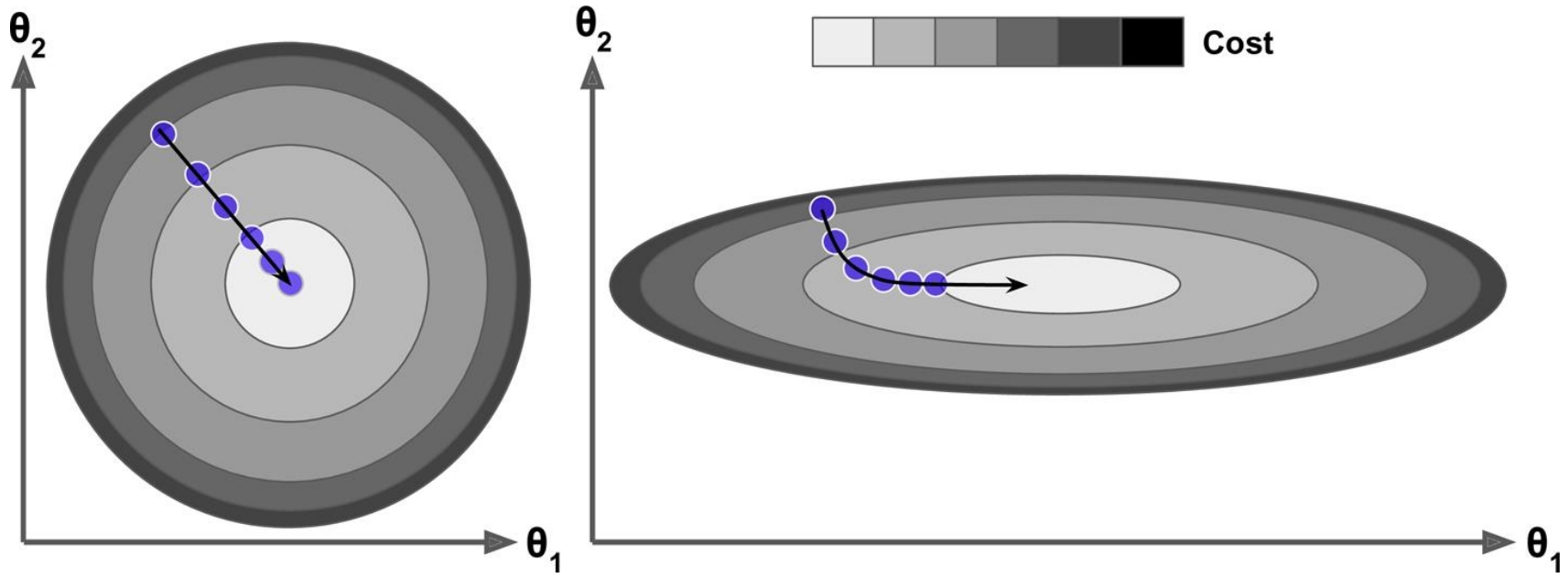
- **One of the most important transformations.**
- Distance-based methods: it is important that features have the same scale so that all the features contribute equally to the result.
- If features have different scales, there is a chance that higher weight is given to features with higher magnitude.
- Without feature scaling, the model will pay too much attention to the features having a wider range.

Attribute	Person A	Person B
Age	23	40
Years at current address	2	10
Residential status (1=Owner, 2=Renter, 3=Other)	2	1
Income	50,000	90,000

Age might have a range from 18 to 100, while Income might have a range from $1000 to $1,000,000.
Without scaling, our distance metric would consider ten dollars of income difference to be as significant as ten years of age difference.

Feature Scaling: Impact on gradient descent

- Having features on a similar scale can help the gradient descent converge more quickly towards the minima.



Normalization vs Standardization

Normalization: values are shifted and rescaled so that they end up ranging between 0 and 1.
Also known as Min-Max scaling.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Standardization: zero mean, a unit standard deviation, the values are not restricted to a particular range.

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$$

- Standardization is less affected by outliers.
- Scaling method is yet another hyperparameter of the model.
- It is important to fit the scalers to the training data only, not to the full dataset (including the test set).

Feature Aggregation

- Combining two or more features into a new feature.
- Reasons:
  - Data reduction
    Reduce the number of features
  - Change of scale
    Cities aggregated into regions, states, countries
  - More “stable” data
    Aggregated data tends to have less variability

- Different aggregation functions (count, min, max, average, standard deviation, etc.) to summarize several values into one feature, aggregating over varying windows of time and space.

- Date and amount differences: (number of days since last important event, or difference in two monthly bills).

Cyclical Features

- Many features commonly found in datasets are cyclical in nature.

Examples:

- Time: minutes, hours, seconds
- Day of the week
- Month
- Week of the month

- Using ordered numbers is not enough, as we should include information about closeness between certain values:
  - Sunday to Monday
  - December to January
  - 11 at night to 1 in the morning

- We can encode such features with two new features:

$$x_{cos} = \cos\left(\frac{2*\pi*x}{\max(x)}\right) \qquad x_{sin} = \sin\left(\frac{2*\pi*x}{\max(x)}\right)$$

- Disadvantage: converting one information into two features.

4b Video Lecture

MODEL SELECTION

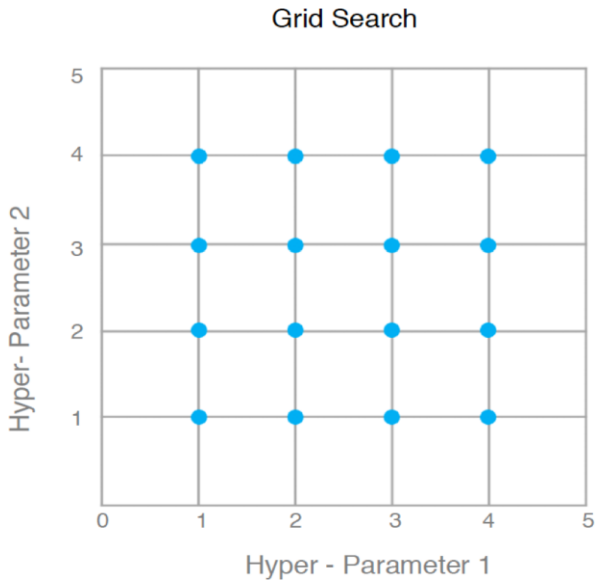
Hyperparameters

- Different design choices when creating ML model
- **Hyperparameters:** parameters which define the model architecture, but not learnt directly from the training data:
  - What degree of polynomial features should I use for my linear model?
  - What should I set my learning rate to for gradient descent?
  - What regularization strength should I choose in regularization?

- No single optimal design
- **Hyperparameter tuning:** process of searching for the ideal model architecture, part of model selection
  - Grid Search
  - Random Search
  - Bayesian Optimization

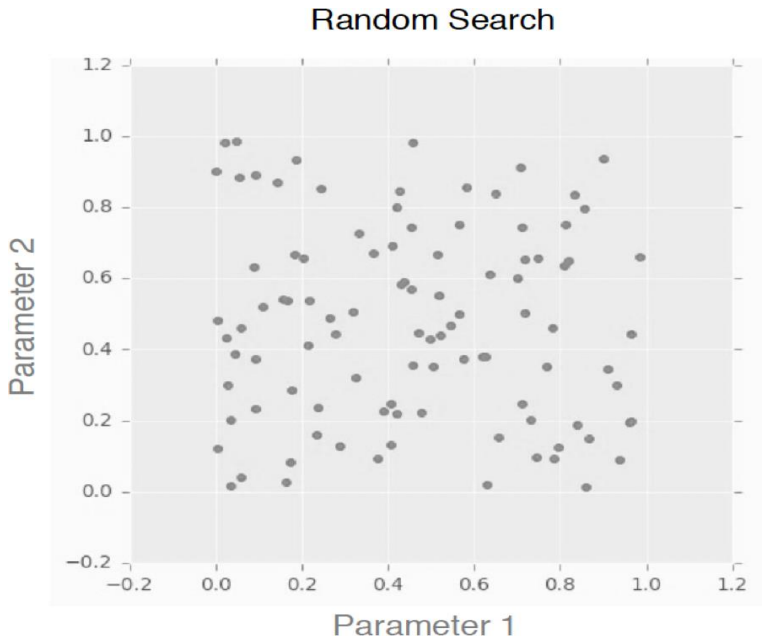
Grid search

- Form a grid of hyper-parameter values
- Grid search is a costly and time-consuming approach
- This method works ok when the number of hyperparameters is relatively small.



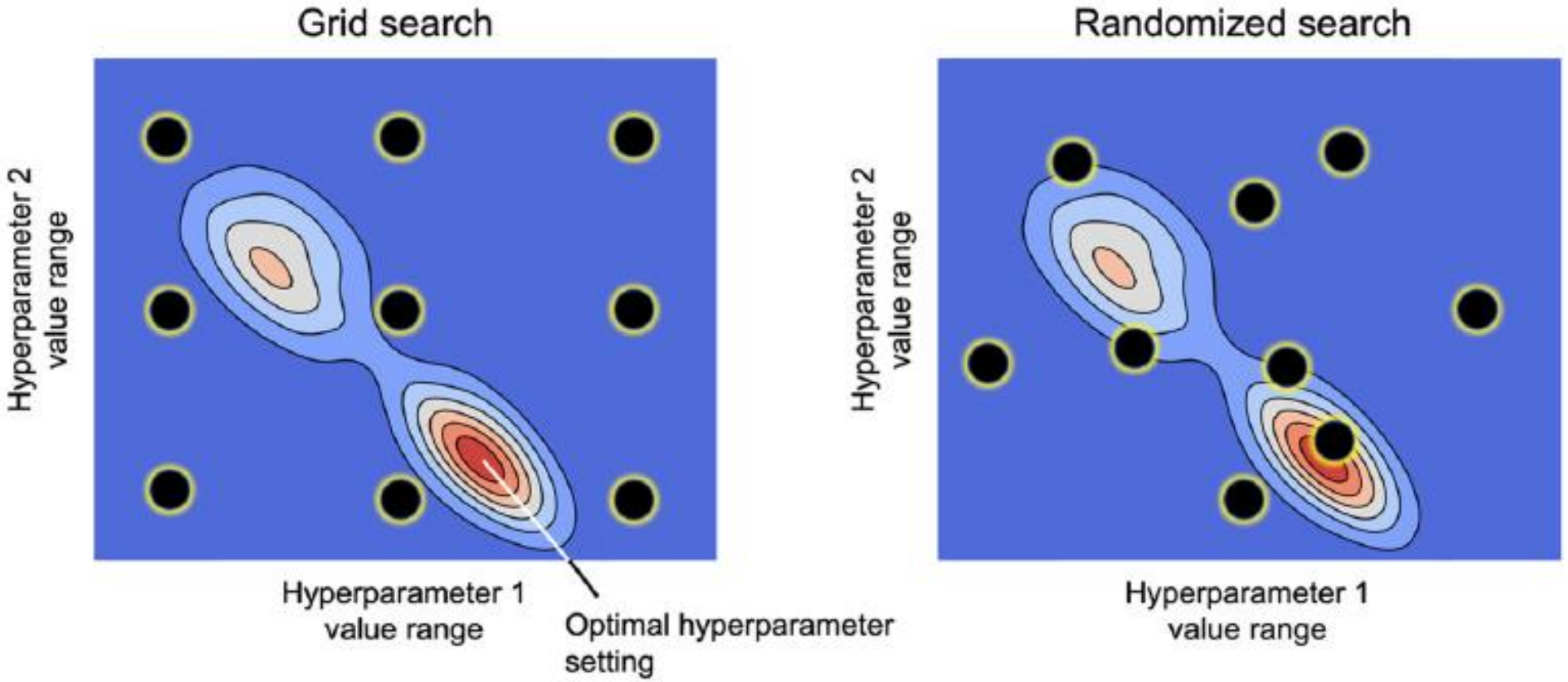
Randomized search

- Instead of trying all possible specified combinations we will just use randomly selected subset of the hyperparameters.
- It has been found to be more effective in high-dimensional spaces than exhaustive search.
- Often not all hyperparameters have an equal impact on model performance



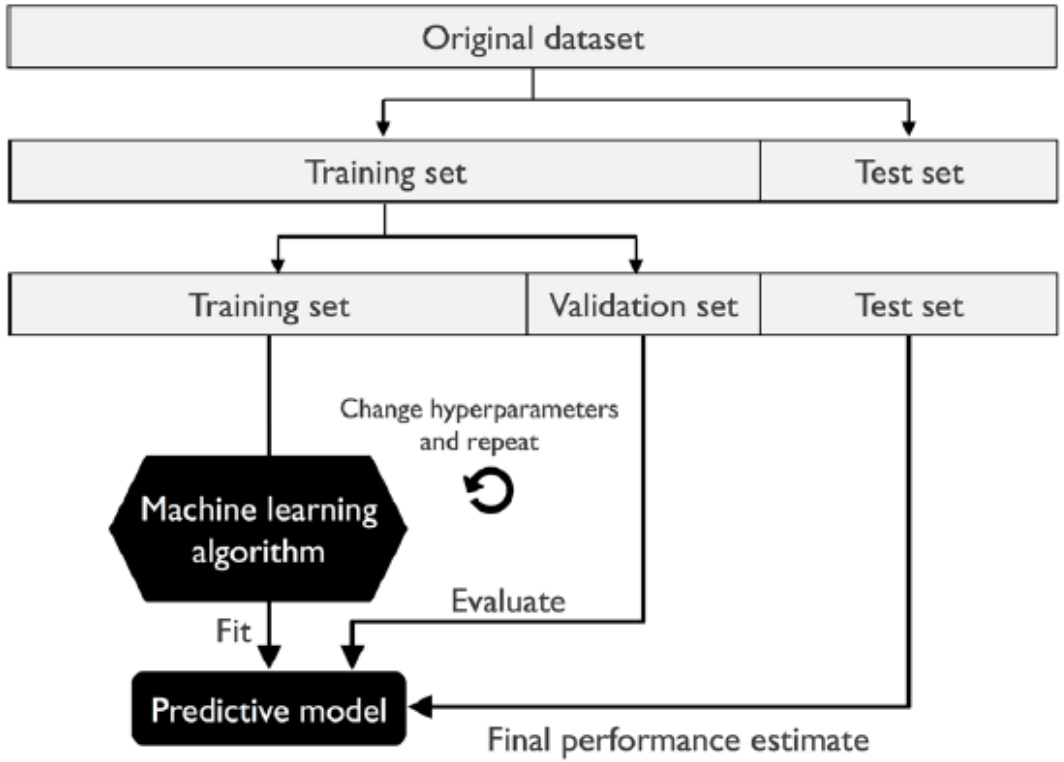
Methods for Hyperparameter Tuning

- **Grid search**: exhaustive search of the user-specified grid
- **Randomized search**: drawing hyperparameter configurations randomly from distributions
- **Bayesian Optimization**: intelligently selects the most promising hyperparameter values



Hyperparameter tuning NOT DONE on the test dataset

- Final evaluation of the chosen model performance is done on the independent test set: not used in training, not used in hyperparameter selection



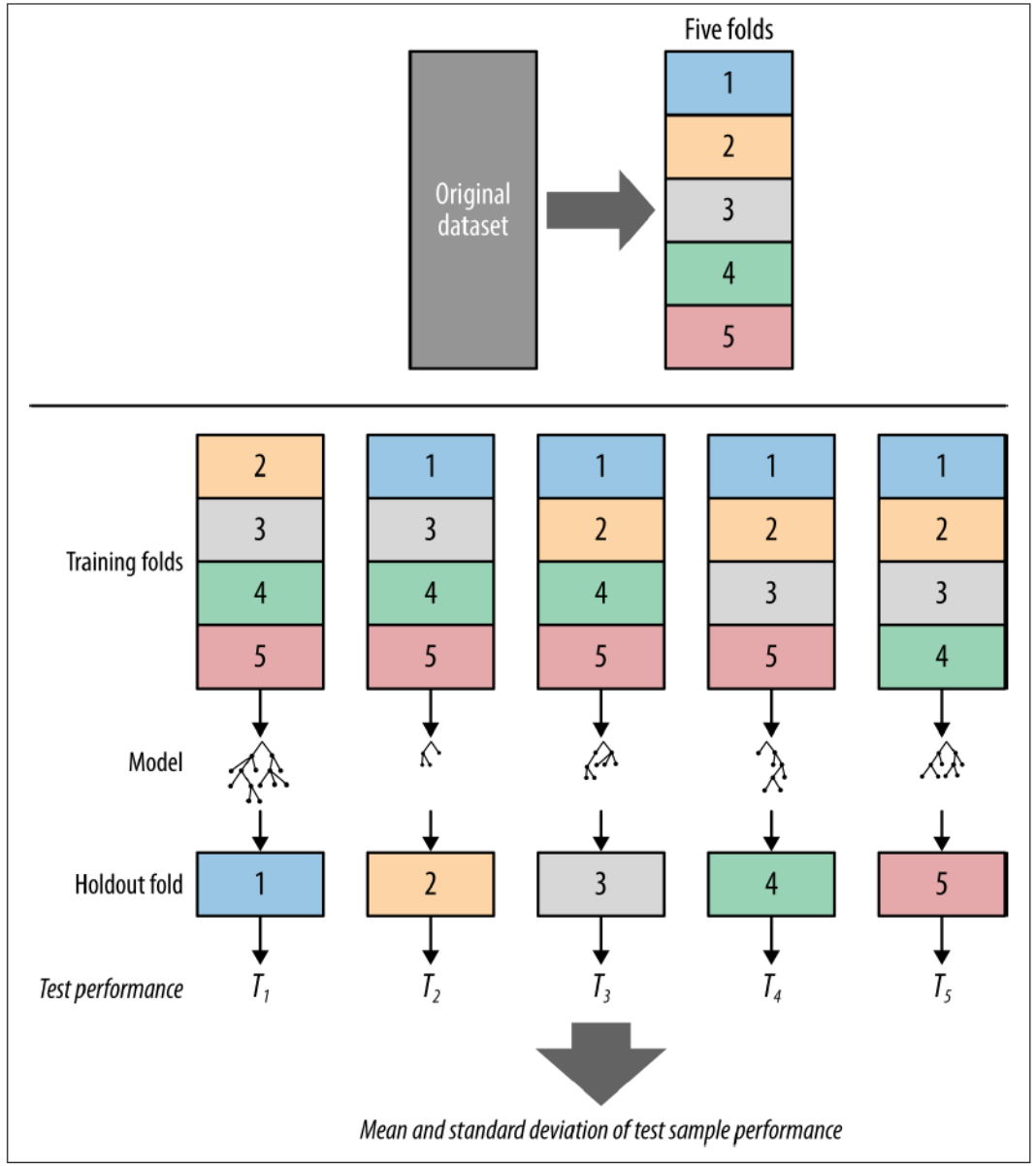
The final test set is not used to make any modeling decisions, instead, it is used to evaluate the performance of the final selected mode.

The test data cannot influence the training phase in any way.

From Holdout Evaluation to Cross-Validation

- Evaluate model performance on the holdout data.
- Hold out data just a single estimate. The performance estimate may be very sensitive to how we partition the training dataset into the training and validation subsets; the estimate will vary for different examples of the data.
- **Cross-validation** is a more sophisticated holdout training and testing procedure.
  - Not a single split of the dataset into training, validation and test, but multiple.
  - Provides statistics on how performance varies across datasets.

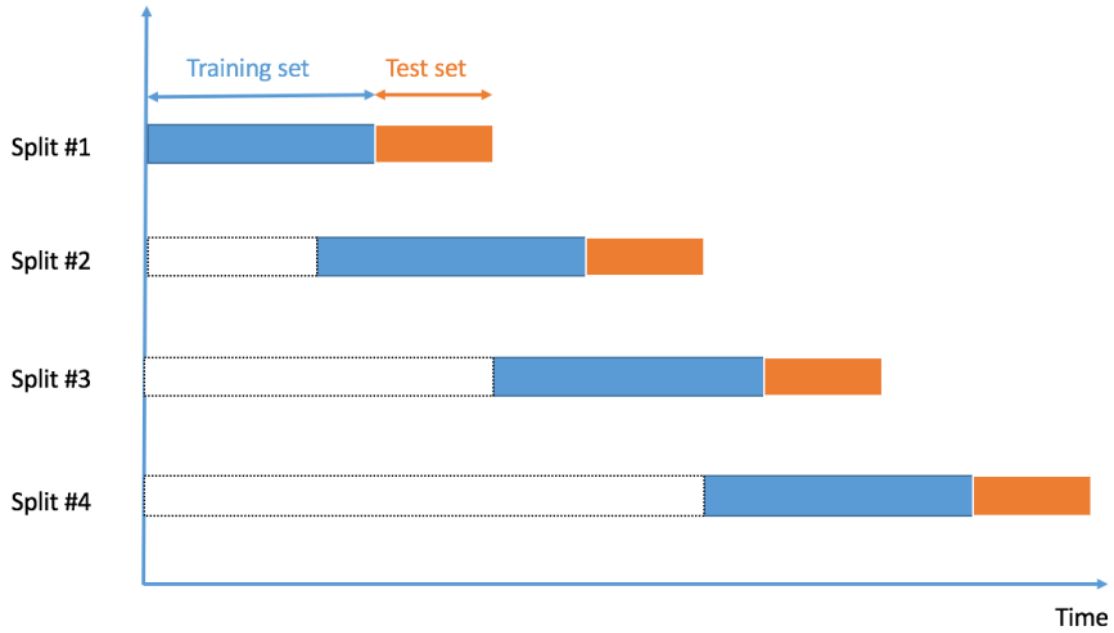
Cross-Validation procedure



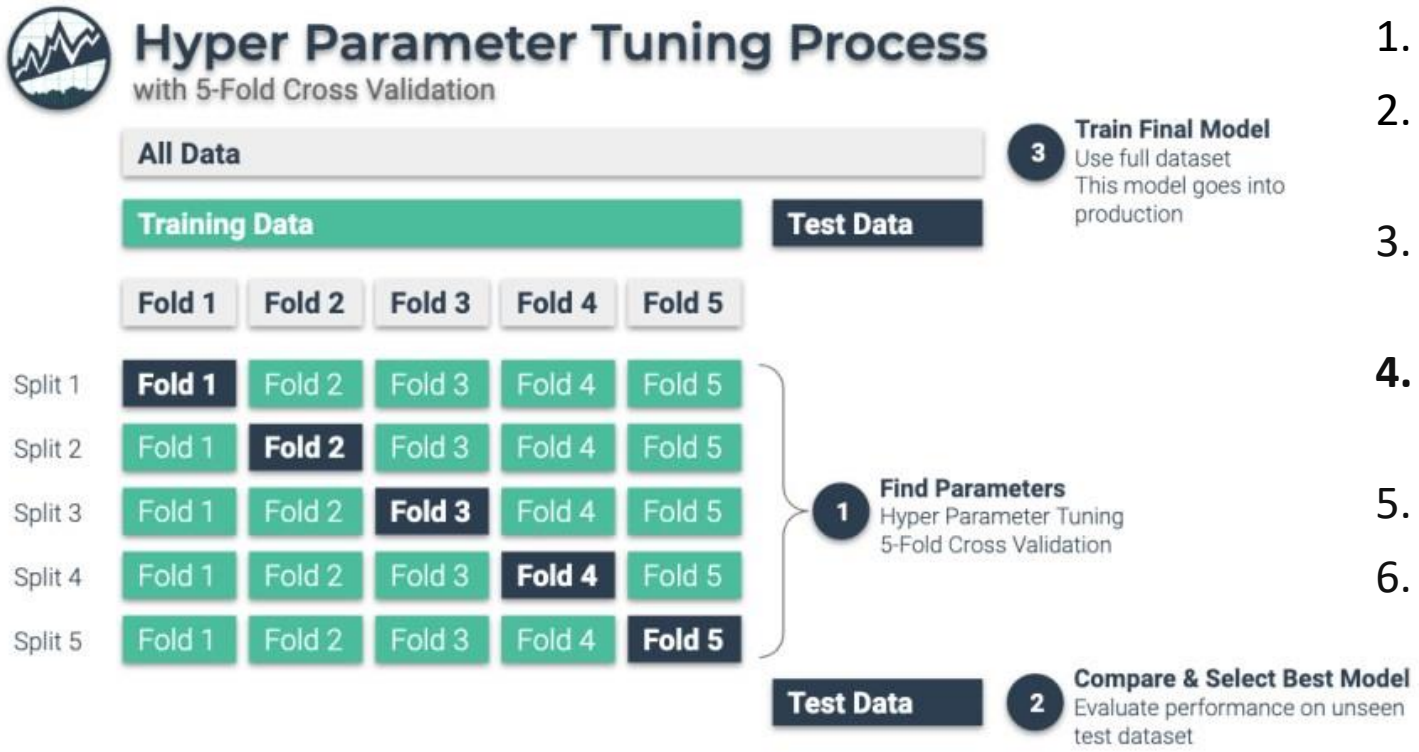
- Divide dataset into groups (K folds)
- Use K-1 folds for training
- Evaluate on the left-out fold
- Swap and repeat, use each fold once for testing and K-1 times for training
- Get mean and variance of performance over folds
- Typical value of K= 5 or 10

Cross validation for time-series

- Special cross-validation for time series
- Simulation of real-world environment
- Folds are ordered chronologically
- NEVER use future data in training, train on PAST, evaluate on (relative) FUTURE



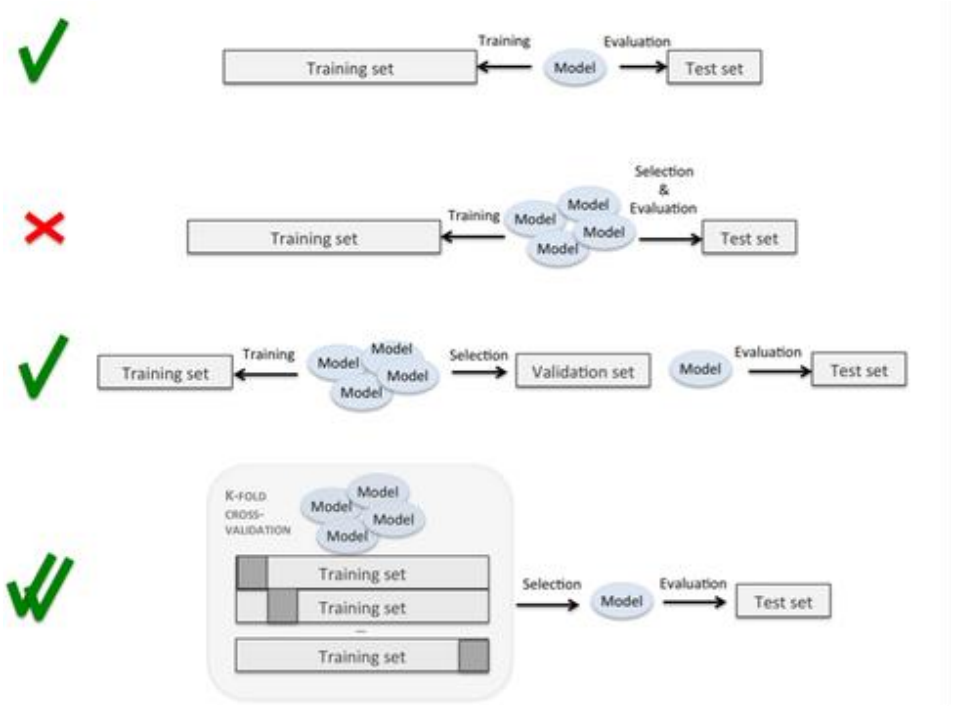
Hyperparameter tuning with Cross Validation



1. Split train and test
2. Use cross validation on the training data to test different values of the hyperparameter
3. Choose hyperparameters with the best performance
4. **Retrain the model with the chose parameters on the whole training set**
5. Evaluate on the test set
6. Retrain the model with all the data test and train before going into production, but with the same parameters chosen in step 3

Model selection using Cross validation

- There's no universally best ML algorithm for all the problems
- Use cross validation
  - Try and test different algorithms
  - Try and test different values of the hyperparameters



Benchmarks for model performance

- It is useful to compare a model to some benchmark.
- What is a suitable benchmark depends on the actual application
- Coming up with suitable baselines part of the business understanding phase:
  - random classifier
  - model based on a simple business strategy
    to identify clients that are more likely to buy a new product we can choose the clients with high previous revenue
  - simple model based on some average
    predict how many "stars" a particular customer would give to a particular movie use the average number of across the population and the average number of stars a particular customer gives to movies
  - model that only considers a very small number of features