Classification:
Linear Classifiers
Support Vector Machines
Decision Trees
K-Nearest Neighbor
Random Forest
Neural Networks
Naive Bayes

Regression: Predictive statistical process where the model attempts to find the important relationship between dependent and independent variables
$\hat{y} = w_0 \cdot x_0 + w_1 \cdot x_1 + ... + w_i \cdot x_i + b$
x = features, w & b = parameters

Goal: build a model on the training data and then be able to make accurate predictions on new, unseen data that has the same characteristics as the training set that we used. If a model is able to make accurate predictions on unseen data, we say it is able to generalize from the training data to the test data.

Accuracy = number of correct predictions / total number of predicitons

Overfitting We expect simple models to generalize better to new data. Therefore, we always want to find the simplest model. Building a model that is too complex for the amount of information we have, is called overfitting

Type	Description	Examples	Operations
Nominal	labels to distinguis one from another (= , ≠)	zip codes, ID numbers, eye color, sex	mode, entropy, contingency, correlation, χ^2 test
Ordinal	distinguish and rank/order (< , >)	grades, street numbers, [good better best]	median, percentiles, rank correlation, run tests, sign tests
Interval	meaningful differences between values (unit of measurment exists) (+ , -)	calender dates, temp in C/F	mean, standard deviation, Pearson's correlation, t and F tests
Ratio	both differences and ratios meaningful (* , /)	quantities, length, age, mass, weight, counts, temp in K	geometric mean, harmonic mea, percent variation

Noise refers to modification of original values: data = true signal + noise
Outliers are data objects with characteristics that are considerably different than most of the other data objects in the data set

Missing Data:
Eliminate: row or feature
Estimate:
- Constant value Imp.: (mean, median, mode constant value)
- Numerical Imp.: all possible values weighted by probabilty
- Categorical Imp.: max occured value or default value (other)
- Do Nothing: XGBoost (learn), LightGBM (ignore), does not work with LinReg
- kNN Imp.: find its k's closest neighbour (weighted avg on dist)

Regression Imp.: predict missing values by regressing it from other related vars
Cluster based Imp.:

Some algorithms (i.e., neural networks, SVMs, or distance based algorithms like k-NN or k-Means) are very sensitive to the scaling of the data. Transformations must be applied on the total (train+test) set to ensure equal scaling.

Standardization (or Z-score normalization):
It ensures that for each feature the mean μ_{new} is 0 and the standard deviation σ_{new} is 1
Useful for optimization algorithms (gradient descent, exploited for regression and neural networks), and for distance measurement based algorithms (k-Nearest-Neighbours)

+ brings all features to the same magnitude	$x_{new} = \frac{x - \mu}{\sigma}$
+ ensures statistical properties for each feature guaranteeing they are on the same scale	
- does not ensure any particular minimum and maximum values for the features	

Robust Standardization
It uses the median (50%ile) and IQR (75%ile - 25%ile), instead of mean and variance
The resulting variable has a zero mean and median and a standard deviation of 1

+ ignores data points that are very different from the rest	$x_{new} = \frac{x - median}{IQR}$
+ resulting data is not skewed by outliers	

Normalization (or MinMax Scaling):
It shifts and rescale the data such that all features are exactly between 0 and 1
good for: non-gaussian distributed data, algos which do not assume any dist (kNN, NN)

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

scaling of target values is generally not required.

Feature engineering:
One-out-of-N encoding: replace a categorical variable with *k* possible values with k binary features. (or *k-1*, the *k*th is deductible)
Categorical Variables: as integers, but must be treated as discrete

Binning:
make the model more robust; prevent overfitting; has a cost on the performance (binning means making data more regularized)

Unsupervised Discretization:
Equal Width Disc.: values interval is dived into *k* bins of equal length
Equal Freq Disc.: each bin contains the same number of elements

Supervised Discretization:

Entropy Based Approach finds the best split so that the bins are as pure as possible, that is the majority of the values in a bin correspond to have the same class label. Lower entropy is better, and a 0 value for entropy is the best. Select bin with highest gain.

$E(S) = -\sum_{i=1}^{C} p_i \cdot \log_2(p_i)$	$E(S, Split) = -\sum_{v \in Split}^{\square} \frac{\lvert S_v \rvert}{\lvert S \rvert} E(S_v)$	Gain = $E(S) - E(S,Split)$

Linear models (& naive Bayes) become much more flexible with binning, bc it will have a different value for each bin. They benefit greatly in expressiveness from the transformation of the data. Good if some features have non linear relationship with the output.
Tree based models don't benefit from binning, as they can learn whatever binning is most useful for predicting on this data. Decision trees look at multiple features at once, while binning is usually done on a per-feature basis.
SVMs, nearest neighbors, and neural networks might sometimes benefit from using binning, interactions, or polynomials, but the implications there are less clear than in the case of linear models.

Univariate Nonlinear Transformations:
Linear models and neural networks are very tied to the scale and distribution of each feature, and if there is a nonlinear relation between the feature and the target, that becomes hard to model —particularly in regression. log/exp to adjust relative scales, sin/cos for data with periodic patterns
These kinds of transformations are irrelevant for tree-based models but might be essential for linear models. Sometimes it is also a good idea to transform the target variable y in regression.

Curse of Dimentionality
More features make the model more expressive, but not all of the features are relevant.
Fewer features: easier to interpret, generalize better, faster and more efficient
Feature engineering can be used for either accuracy or explainability.

Feature selection:
- selection of the independent features with a stronger relation to the dependent feature
- selection of a sub-set of features, which exclude redundant or irrelevant features

this is an NP hard problem. practical approaches: unsupervised (heuristics), supervised

Supervised Feature Selection:
Univariate Statistics:
compute statistically significant relationship between each feature and the target, then select the features with the highest confidence value. Analysis of variance (ANOVA), each feature is considered individually and will be discarded if it is only informative in combination. Vert fast to compute, but independent of the model we want to apply. Good if really large number of features and/or we suspect many features completely uninformative

Model-Based Feature Selection:
Determine importance of each feature by a supervised ML model, then rank and select features accordingly. This considers all features together and can capture interactions.
Decision-Tree based models: provide a feature importance score.
Linear Models: the absolute value of the coefficients can be used to capture feature importance *(linear models with L1 penalty learn sparse coefficients, which only use a small*

subset of features. This can be viewed as a form of feature selection for the model itself, but can also be used as a preprocessing step to select features for another model)

Iterative Feature Selection:
Either start with no features and add one by one, or start with all features and remove. Recursive Feature Elimination starts with all features, builds a model, and discards the least important feature according to the model. Then a new model is built using all but the discarded feature, and so on until a certain number of features are left. Is more computationally expensive than other methods.