1. **Data Exploration (EDA)**
    a. Data Import
        i. SQL pro tip: “Over(Partition by x)” allows to add sum(xyz) to ALL rows
    b. Stats
        i. Dimensions & Data Types
            1. Imbalanced classes
        ii. Univariate Analysis (each feature for itself)
            1. Variable Structure & Outliers (Boxplot, Histogram, Scatter Plot)
            2. Df.head, Df.describe
        iii. Bivariate Analysis
            1. Correlation Matrix
            2. Covariance Matrix
            3. Scatter Matrix
            4. Chi-Square
        iv. Missing values & wrong (df.isna.sum())
        v. Outliers
        vi. Distribution (histogram etc.)
2. **Data Preprocessing**
    a. Missing values
        i. Removing NaN values or
        ii. Imputing missing values
            1. Using variables from descriptive statistics (mean, median, mode)
            2. Constant value
            3. Predicted value (some model)
                a.
                    i. iIn time series data: interpolation (linear, polynomial, spline)
    b. Deal with outliers or imbalanced data
    c. Dimensionality Reduction / Feature Selection based on
        i. Adjusted R-Squared
            1. R-Squared adjusted for number of predictors in the model
            2. R-Squared = proportion of variance in predicted(dependent) variable that is explained by independent variable
        ii. Stepwise regression – maximizing model performance based on selection of features

            1. Forward selection starts with no features adds features sequentially to test for performance gain
            2. Backward elimination starts with all features and removes features sequentially to test for performance loss
            3. Bidirectional elimination is a combination of the above two
    iii. Principal Component Analysis (PCA) for dimensionality reduction, discovers relations between independent variables
            1. Linear, unsupervised way to dimensionality reduction by plotting data on a graph with several dimensions where the dimensions are the features of the data. Then, a line through the data is plot that maximizes the variance of the data
            2. Principal components are those that have eigenvectors with high eigenvalues. Components with low eigenvalues are less important
            3. Eigenvectors are directions of a line of data and eigenvalues tell us how spread out the data is along that line (similar to variance). Thus by excluding dimensions with eigenvalue of zero, features with low variance are excluded
            4. Eigenvectors times eigenvalues should be the same as eigenvectors times the matrix
            5. Randomized PCA performs best for high-dimensional datasets
    iv. Kernel PCA & Manifold Learning (non-linear PCA)
    v. T-SNE (supervised, non-linear)
            1. best visualization for high-dimensional datasets
            2. models high-dimension data points as lower dimensions
            3. means “t-Distributed Stochastic Neighbor Embedding”
    vi. LDA Linear Discriminant Analysis, for classification tasks
            1. linear, supervised, discovers relation between independent and dependent variable)
    vii. Truncated SVD (singular value decomposition, truncated: “verkürzt”)
            1. Calculates the eigenvectors of the covariance matrix
            2. Approximates matrix as a lower-rank matrix by matrix multiplication, thereby cutting out unnecessary features
    viii. L1 Regularization / Lasso Regression
    ix. Autoencoder
d. Encoding categorical data (labelencoder/dummy variables, one-hot encoder)

   e. Scaling features (Standard Scaler or MinMax Scaler) – changing range but NOT distribution
      i. Log() logarithm function can transform non-linear data to linear data (or at least close to normal distribution)
      ii. Normalization: from 0 to 1
      iii. Standardization: from -1 to 1, scaled with standard deviation
   f. Data Split (train_test_split) & K-fold Cross Validation (changing combinations of training and test set split)
      i. Training set for learning parameters (70% of data)
      ii. Validation set for testing for generalization error and preventing overfitting by tuning hyperparameters (20% of data, alternative to cross-validation, if enough data)
      iii. Testing set for testing performance (10% of data)

3. **Model Training & Inference**
   a. Model Selection
      i. Goals
      ii. Classification (supervised, discrete values), Regression (supervised, continuous values), Clustering (unsupervised), semi-supervised (only small amount of labeled data), association/similarity (unsupervised), induction (applying rules on new cases) vs. deduction (concluding rules from observations)
      iii. Linear, Non-linear
      iv. Size of dataset, processing power
         1. Small data – low bias, high variance algorithm (for higher model capacity, with the danger of overfitting)
         2. Large data – high bias, low variance algorithm (in order to prevent overfitting)
      v. Desired output
      vi. Greedy algorithms optimize all subparts independently instead of the whole system at once
   b. Model Parameter Tuning
      i. Grid Search CV (sklearn)
      ii. Tpot (genetic algorithm)
   c. Optimization function
      i. Adam, RMSProp, AdaDelta, SGD (Stochastic Gradient Descent)
   d. Hidden Layers
      i. Batch Normalization

4. **Model Evaluation**
   a. Model Evaluation

i. Bias vs. Variance, plotting model results
ii. Accuracy
iii. Confusion Matrix (shows Type I and II errors)
iv. ROC Curve (AUC area) or CAP Curve (slightly different)
    1. Superior for imbalanced classes
    2. ROC = Receiver Operating Characteristic curve
    3. Plotting true positives against false positives
    4. Straight line is bad, shaped curve is good
v. Regression: Mean Squared Error and Root Mean Squared Error
vi. Precision: true positives / all positive predictions
vii. Recall: true positives / all real positive values (= true positive + false negative)
viii. PR curve & F1-score: weighted average of precision and recall
ix. Crossentropy loss (in classification)
x. EM (expectation-maximization) algorithm (for Gaussian Mixture Models)
xi. Further clustering metrics: homogeneity, completeness, v-measure, adjusted Rand, Adjusted Mutual Info (AMI), silhouette scores

b. Plotting Model Fit, Listing Feature Importances

5. Model Optimization

= Reduce overfitting (high variance) or underfitting (high bias) by optimizing model Hyperparameters (Lambda learning rate, error penalty p i.e.). Convergence does not equal optimization.

a. Learning rate
    i. Finding the right learning rate is crucial
    ii. Learning rate decay means decreasing the learning rate over time to prevent getting stuck in local minima
b. Ensemble Techniques (all combining models and averaging decision)
    i. Boosting: combining models (weak leaners) in optimizing loss function of a strong learner to reduce bias (=higher capacity, higher risk of overfitting). Each new model instance is trained emphasizing (higher weights) the previously misclassified examples
        1. Sequential base learners / exploiting dependence between base learners – AdaBoost, Gradient Boosting
            a. Adaptive boosting – higher weights to wrong instances
            b. Gradient boosting – weak learner trains on the errors, contribution of weak learners is calculated based on their gradient descent performance

c. XGBoost additionally to gradient boosting controls overfitting through better regularization

2. Parallel base learners / exploiting independence between base learners– Random Forest

ii. Bagging (bootstrap aggregating): reduce variance by resampling data in k different smaller datasets (against overfitting) (example Random Forest)

iii. Stacking: meta-level function to combine predictions of models with each other and make final prediction

iv. Bootstrapping: randomnly replacing subsets of the data

c. Regularization (reduces/eliminates generalization error/overfitting)

i. Multicollinearity is the existence of near-linear relationships between independent variables

ii. L1 (Lasso Regression): Works as feature selector if large number of features

1. adds regularization term to the loss function,
2. consisting of penalty term lambda λ (hyperparameter) multiplied with each of the feature weights
3. can theoretically cut out features (zero weight). Thus only good for large features or creating simple models

iii. L2 (Ridge Regression or weight decay):

1. Same as L1 regularization, but weights are squared
2. The square forces the weights to be small, but does not cut them out
3. Not robust to outliers because of the square
4. Penalty added to the error of features

d. Normalization

i. MinMaxScaler: Transforming variables so they represent a fixed range (i.e. -1 to +1) to reduce variance and bias

e. Dropout (= taking out parts of the dataset to prevent overfitting)

i. As e.g. for Trees: Pruning (cutting leafs of the tree and thus reduce variance/overfitting)

f. Cross Validation (resampling of data in different datasets)

i. In scikitlearn – cross-validation means shuffling the data to reduce variance of a holdout (split of data in training and testing)

g. Early stopping (terminating algorithm aka no further learning epochs after optimized validation error has been achieved)

h. Feature Impact - measuring how a features is affecting predictions

i. SHAP Values

1. Comparing outcome of model having predetermined (to be tested) value for feature(s) XYZ with outcome of model having baseline value for feature(s) XYZ
2. SHAP Values seem more powerful/comprehensive than PDPs, which are limited to a max of 2 features, whil SHAP Values measure all features, but is limited to certain types of models which are considered by the authors of the library
3. https://www.kaggle.com/dansbecker/shap-values
4. import shap; explainer = shap.TreeExplainer(my_model); shap_values = explainer.shap_values(data_for_prediction)
5. Also has DeepExplainer and KernelExplainer, instead of TreeExplainer

ii. Partial Dependence Plots (PDPs)
   1. Is calculated after model has been fit
   2. Modifying the variable of a feature slightly and see the model prediction change (e.g. feature XYZ – 40, 50 and 60 – what is the outcome on Y?)
   3. 2D PDPs shows the impact of combinations of 2 features on the outcome
   4. from pdpbox import pdp, get_dataset, info_plots
   5. https://www.kaggle.com/dansbecker/partial-plots

i. Feature Importance - measuring how big the impact of each feature on predictions is
   i. Permutation Importance
      1. Is calculated after model has been fit
      2. Randomly shuffling one feature, ceteris paribus
      3. Feature importance is relative to the performance loss of the model caused by the shuffling (negative values mean that shuffling that feature actually causes better model performance)
      4. from eli5.sklearn import PermutationImportance
      5. https://www.kaggle.com/dansbecker/permutation-importance

j. Federated Learning
   i. Enables services to train a model on lots of user data while user data remains protected
   ii. Users download a model, train the model on their data, upload the gradients, gradients are summed and the model is updated
   iii. Platforms: Apple, Google, OpenMined (open-source)

            1. OpenMined uses smart contracts, a third-party oracle and homomorphic encryption, which allows to do mathematical operations on encrypted data

k. Transfer Learning
    i. Pre-trained model is applied to a new dataset to optimize the model and prevent overfitting
    ii. One-shot learning as a form of transfer learning means applying a trained model on a new dataset with only one labelled example
    iii. Zero-shot learning – no labelled input at all

l. Cascade Learning
    i. Applying first a simple model to pre-filter and reach posterior probability (probability of a random event occurring conditional on the evidence that the event has happened)
    ii. And then a complex model to reach high confidence for selected features/samples