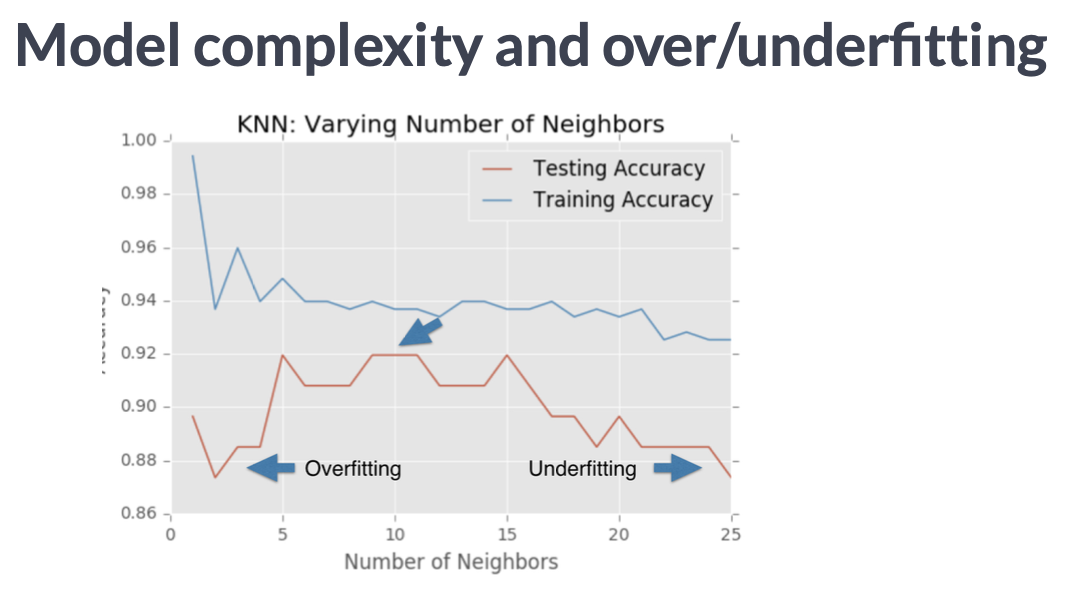
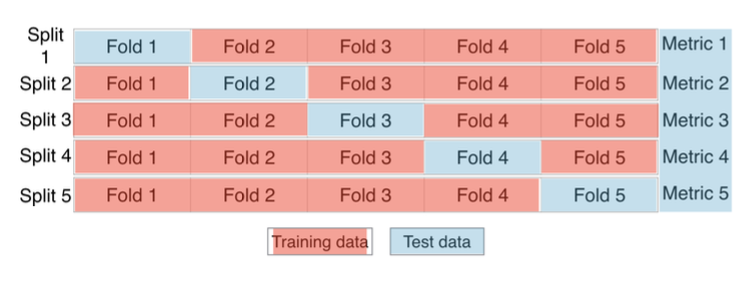
Chapter 1 - Supervised Learning with Scikit-learn

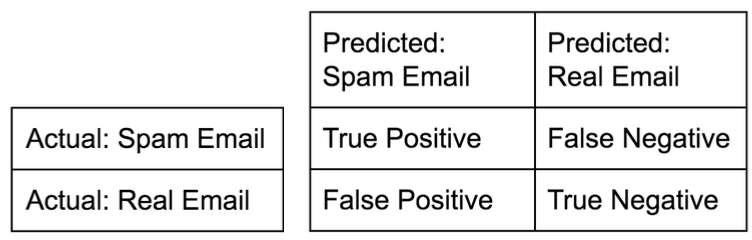
* 1. Classification
* What is machine learning?
  + Definition: giving computers the ability to learn to make decisions from data without being explicitly programmed
  + Unsupervised learning: uncovering hidden patterns from unlabeled data
  + Reinforcement learning: software agents interact with an environment
    - Learn how to optimize their behavior
    - Given a system of rewards and punishments
    - Draws inspiration from behavioral psychology
  + Supervised learning: predictor variables (X) and a target variable (y)
    - Aim: predict the target variable, given the predictor variables
    - Classification: target variable consists of categories
    - Regression: target variable is continuous
    - Automate time-consuming or expensive manual tasks
    - Make predictions about the future
  + Naming conventions:
    - Features = predictor variables = independent variables
    - Target variable = dependent variable = response variable
* Exploratory data analysis
  + Explore data via “print(df.head())”
* Visual EDA
  + Goes with scatter\_matrix / scatter\_plot
* K-Nearest Neighbors
  + Basic idea: predict the label of a data point by
    - looking at the ‘k’ closest labeled data points
    - taking a majority vote
* Scikit-learn fit and predict
  + All machine learning models implemented as Python classes
    - They implement the algorithms for learning and predicting
    - Store information learned from the data
  + Training a model on the data =’fitting’ a model to the data with *.fit()*
  + To predict the labels of new data use the *.predict()* method
* Measuring model performance
  + In classification, accuracy is a commonly used metric
  + Accuracy = Fraction of correct predictions
  + Split data into training and test set
  + Fit/train the classifier on the training set
  + Make predictions on test set
  + Compare predictions with the known labels > Train/test split



* 1. Regression
* Intro
  + You create feature and target arrays by dropping the target variable from the data and name this X
  + You take the dropped variable and define it as y
* The basics of linear regression
  + Regression mechanics
    - Y=ax+b
      * Y = target
      * X = single feature
      * A, b = parameters of model
    - Choose a and b by selecting an error function for any given line; choose the line that minimizes the error function
  + The loss function sometimes calculated by OLS (ordinary least squares): that minimizes the sum of squares of residuals
* Linear regression in higher dimensions
  + Y = a1x1 + a2x2 + b > here you need to specify three variables
* Cross validation
  + Model performance is dependent on way the data is split
  + Not representative of the model’s ability to generalize
  + Solution: cross-validation
  + Basics: split the data in different folds, and use folds for training and testing > the more folds the longer the testing takes



* Regularized regression
  + Why regularize linear regression minimizes a loss function
  + It chooses a coefficient for each feature variable
  + Large coefficients can lead to overfitting
  + Penalizing large coefficients: Regularization
* Ridge Regression
  + Loss function = OLS loss function
  + Alpha: parameter we need to choose
  + Picking alpha here is similar to picking k in k-NN
  + Hyperparameter tuning
  + Alpha controls model complexity
    - Alpha = 0: we get back OLS (can lead to overfitting)
    - Picking alpha here is similar to picking k in k-NN
    - Alpha controls the model complexity
    - Very high alpha: Can lead to underfitting
  + Lasso regression:
    - Can be used to select important features of a dataset
    - Shrinks the coefficients of less important features to exactly 0
  1. Fine-tuning the model
* Classification metrics: measuring model performance accuracy may not always be a good metric, e.g. when class imbalances take place
* Therefore use confusion matrices:



* + Accuracy: tp + tn / (tp+tn+fp+fn)
  + Precision: tp / (tp+fp)
  + Recall: tp(tp+fn)
  + F1score: 2\* (precision \* recall / (precision+recall))
  + High precision: not many real emails predicted as spam
  + High recall: predicted most spam emails correctly
* Logistic regression and the ROC curve
  + Logistic regression outputs probabilities
  + If the probability ‘p’ is greater than .5: the data is labeled ‘1’; else ‘0’
* Area under the ROC curve
  + Larger area under the ROC curve = better model
* Hyperparameter tuning
  + Linear regression: choosing parameters
  + Ridge/lasso regression: choosing alpha
  + K-Nearest Neighbors: choosing n\_neighbors
  + Parameters like alpha and k: Hyperparameters
* Choosing the correct hyperparameter
  + Try a bunch of different hyperparameter values
  + Fit all of them separately
  + See how well each performs
  + Choose the best performing one
  + It is essential to use cross-validation
  + Testing different hyperparameters works with “Grid Search Cross-Validation” where you try different parameters (e.g., C or alpha)
* Hold-out set for final evaluation
  + How well can the model perform on never before seen data?
  + Using all data for cross-validation is not ideal, hence split data in train and test
  + Perform grid-search cross-validation on training set
  + Choose best parameters and evaluate them on the hold-out set
  1. Preprocessing and pipelines
* Dealing with categorical features
  + Scikit-learn will not accept categorical features by default
  + They must be encoded numerically > 0 if observation was not in that category, 1 if it was
  + Dummy variables: if you got three classes, you can delete one, but still use it as it is not in the two residual classes
  + Dealing with categorical features in Python:
    - Scikit-learn: OneHotEncoder()
    - Pandas: get\_dummies()
* Handling missing data:
  + Dropping missing data: df.replace.(0, np.nan, inplace=True), df = df.dropna()
* Imputing missing data: is making an educated guess about the missing values
* Imputing with a pipeline by e.g. setting steps: imputation and then doing a logistic regression
* Centering and scaling:
  + Why scaling:
    - Many models use some form of distance to inform them
    - Features on larger scales can unduly influence the model
    - Example: k-NN uses distance explicitly when making predictions
    - We want features to be on a similar scale
    - Normalizing (or scaling or centering)
  + Ways to normalize data:
    - Standardization: subtract the mean and divide by variance
    - All features are centered around zero and have variance one
    - Can also subtract the minimum and divide by the range
    - Minimum zero, max will be one
    - Can also normalize so the data ranges from -1 to 1