Bil 470 / YAP 470

Introduction to Machine Learning (Yapay Öğrenme)

Batuhan Bardak

Lecture 2: Basic concepts of ML & ML by example (kNN)

Date: 19.09.2022

Plan for today

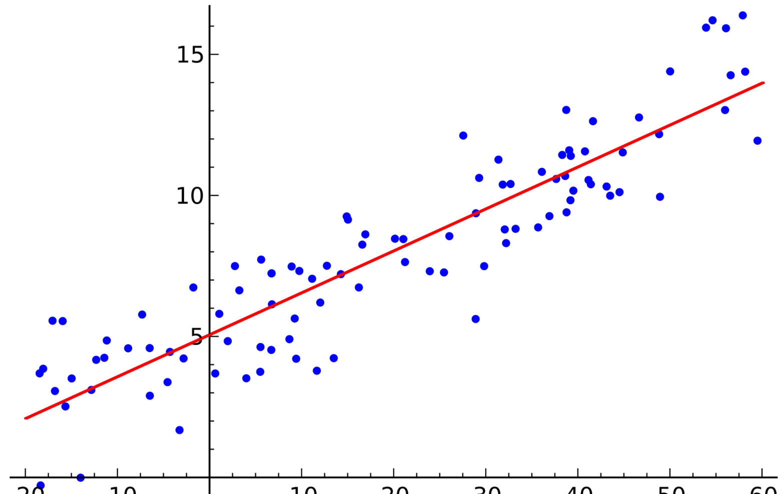
- Basic concepts of Machine Learning
- KNN
- Overfitting / Underfitting
- Train / Test split
- Evaluation metrics

Types of Machine Learning Systems

- Whether or not they are trained with human supervision
  - Supervised
  - Unsupervised
  - Semi-supervised
  - Reinforcement Learning
- Whether or not they can learn incrementally on the fly
  - Batch Learning
  - Online Learning
- Whether they work by simply comparing new data points to known data points, or instead detect patterns in the training data and build a predictive model
  - Instance-based Learning
  - Model Based Learning

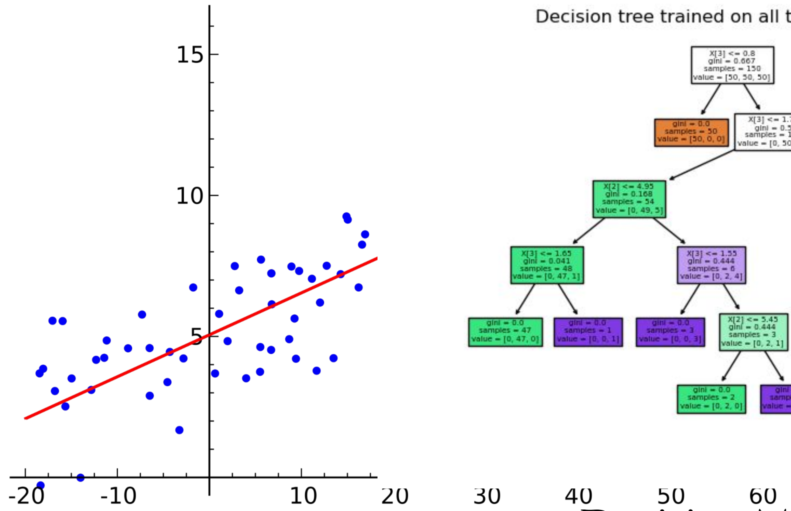
Slide Credit: Aurelien Geron, Hands-On ML with Scikit-Learn

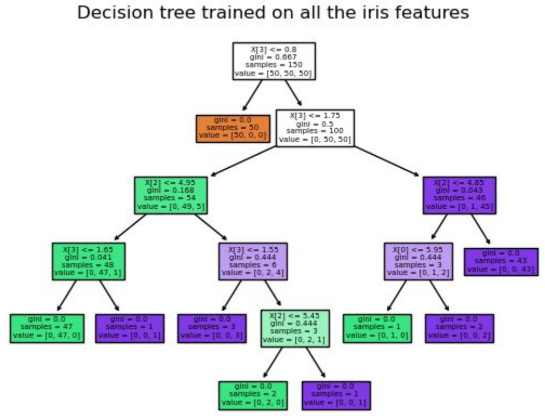
Supervised/Unsupervised Learning



Linear Regression

Supervised/Unsupervised Learning

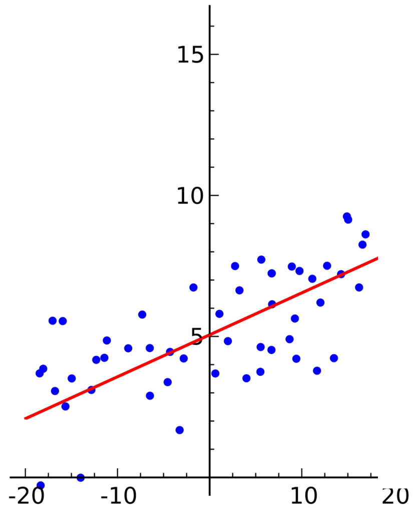


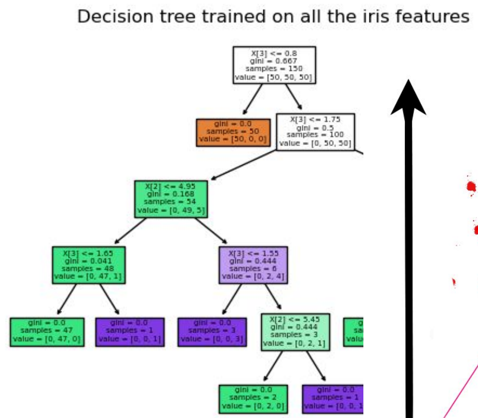


Decision Tree

Linear Regression

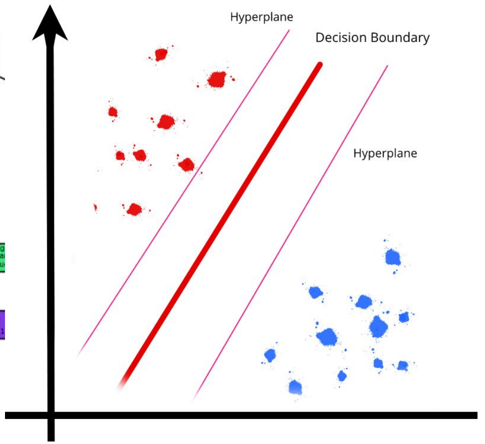
Supervised/Unsupervised Learning





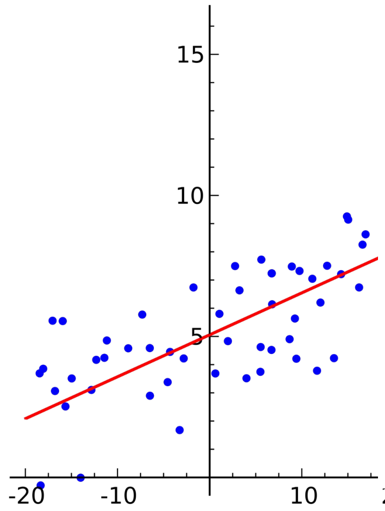
Decision Tree

Linear Regression

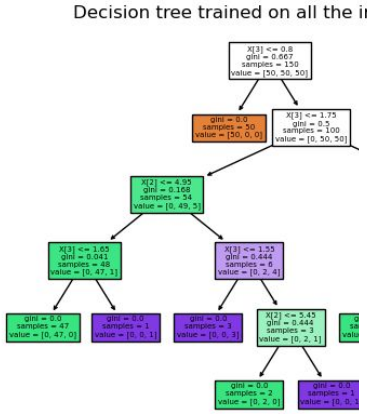


SVM

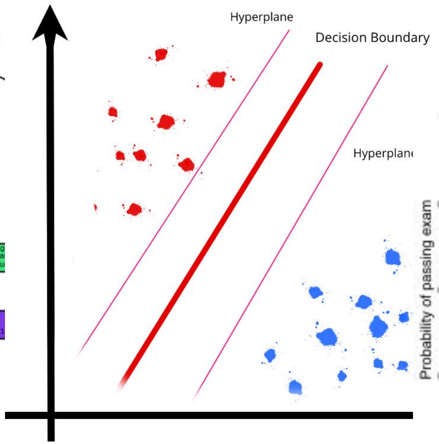
Supervised/Unsupervised Learning



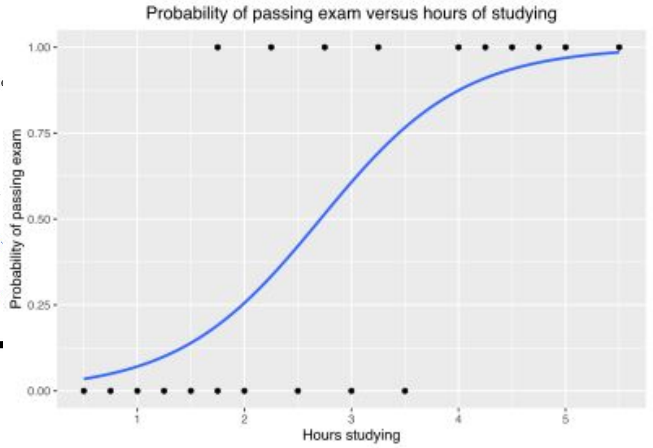
Linear Regression



Decision Tree



SVM

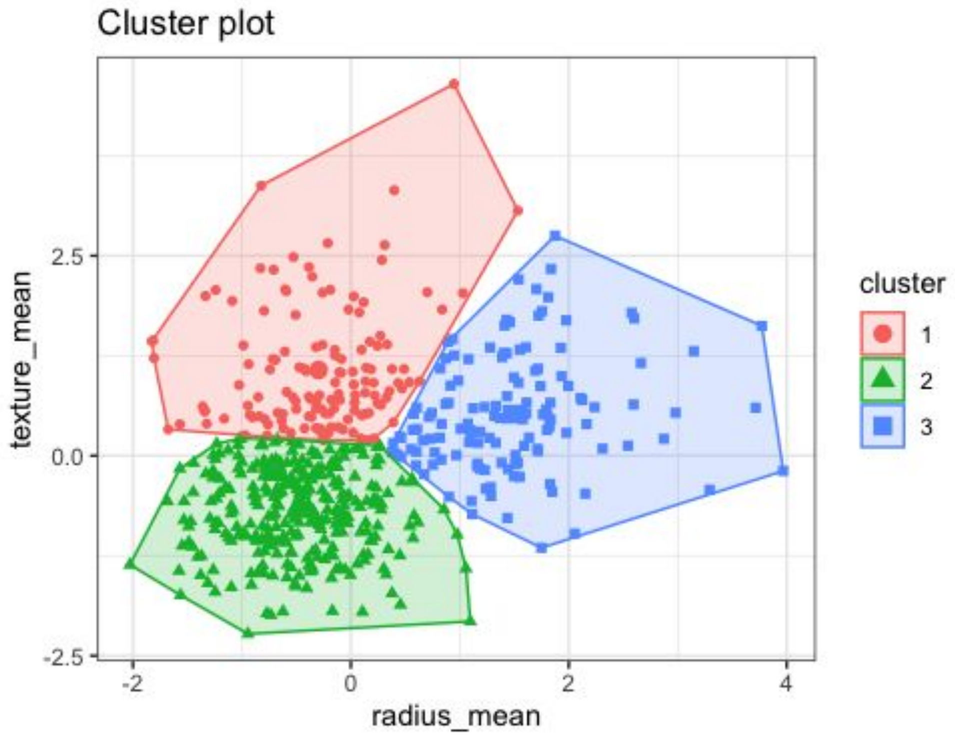


Logistic Regression

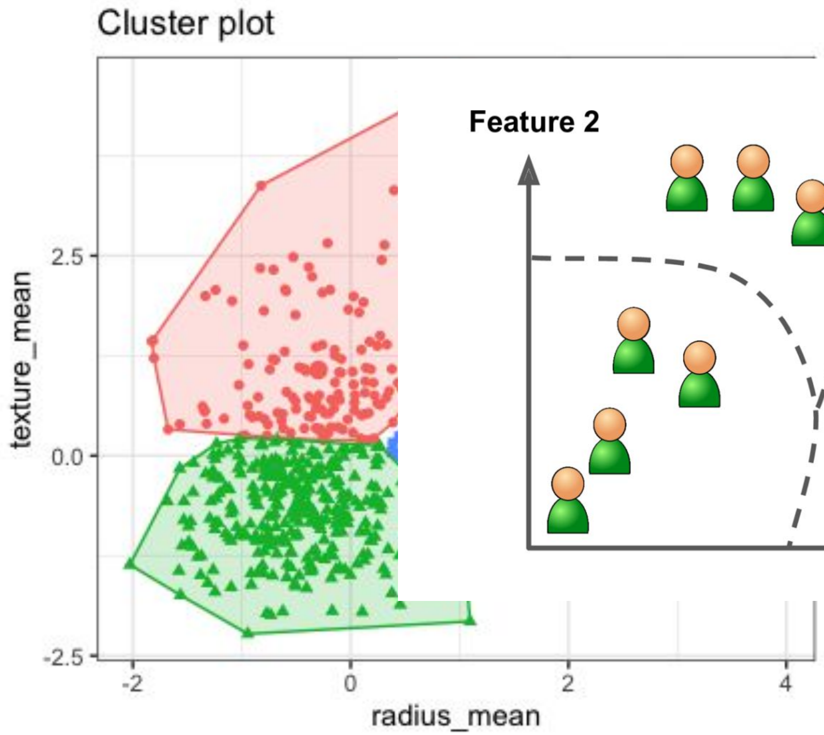
Supervised Learning

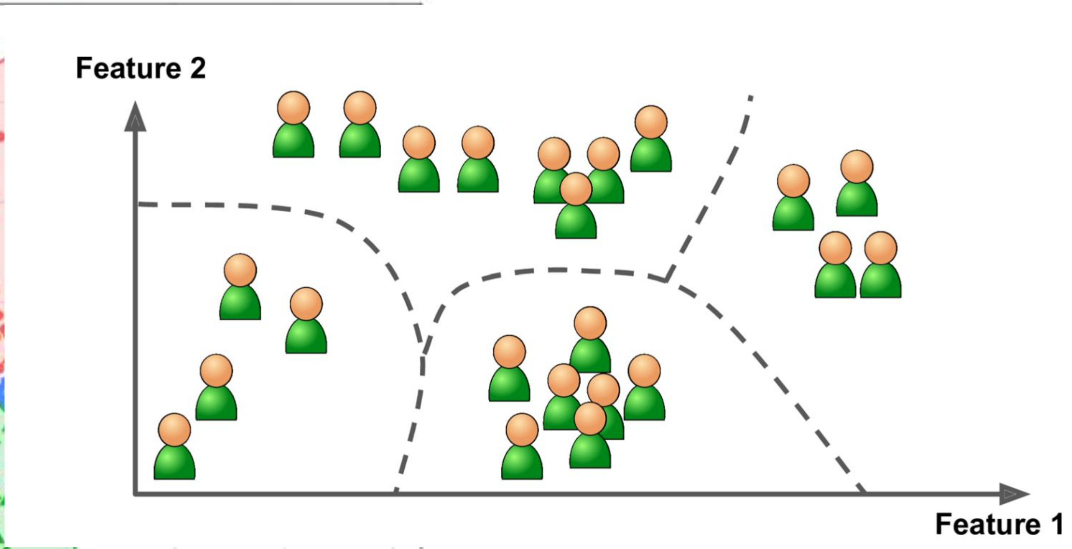
- Some of the most important supervised learning algorithms
  - k-Nearest Neighbors
  - Linear Regression
  - Logistic Regression
  - Support Vector Machines (SVMs)
  - Decision Trees and Random Forests
  - Neural Networks (some of them can be unsupervised)

Supervised/**Unsupervised Learning**



Supervised/**Unsupervised Learning**





Slide Credit: Aurelien Geron, Hands-On ML with Scikit-Learn

Unsupervised Learning

- Some of the most important unsupervised learning algorithms
  - Clustering
    - K-means
    - Hierarchical Clustering Analysis
    - Expectation Maximization
  - Visualization and dimensionality reduction
    - Principal Component Analysis (PCA)
    - Kernel PCA
    - T-distributed Stochastic Neighbor Embedding (t-SNE)
  - Association rule learning
    - Apriori
    - Eclat

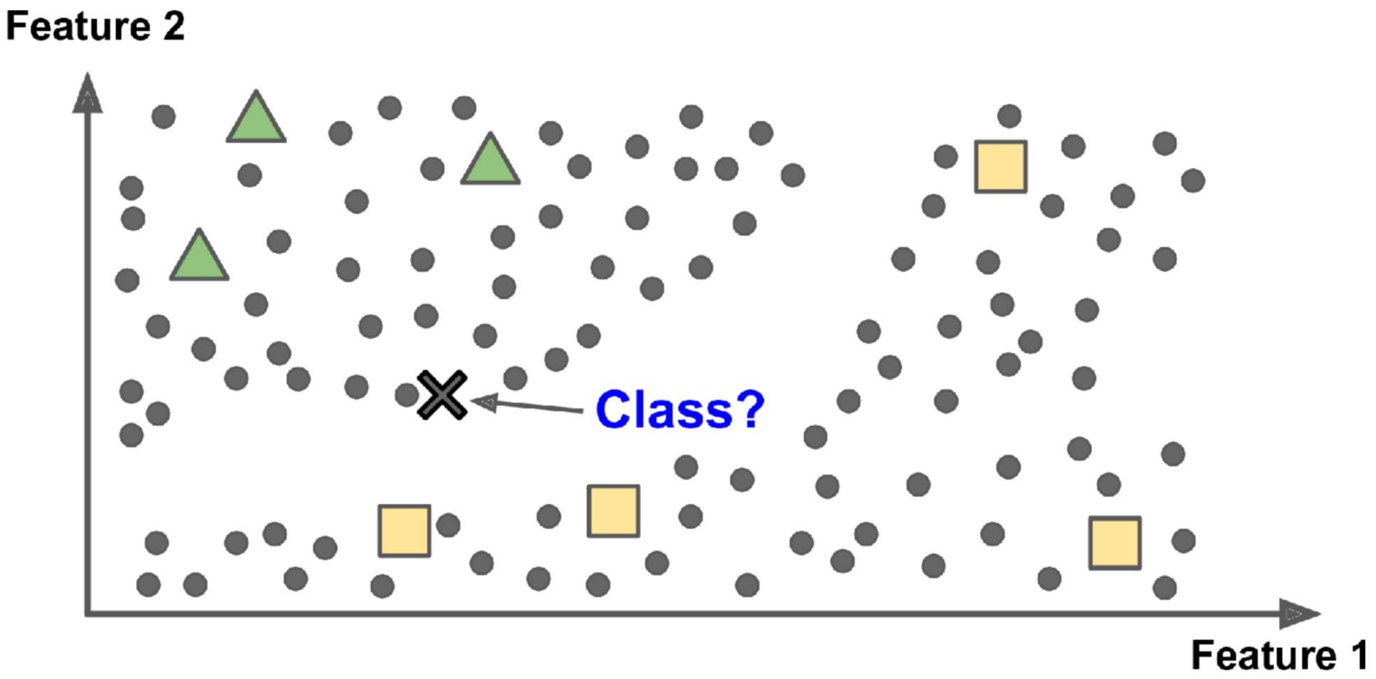
Slide Credit: Aurelien Geron, Hands-On ML with Scikit-Learn

Semisupervised Learning

- Some algorithms can deal with partially labeled training data
  - Usually a lot of unlabeled data
  - A little bit of labeled data

Slide Credit: Aurelien Geron, Hands-On ML with Scikit-Learn

Semisupervised Learning

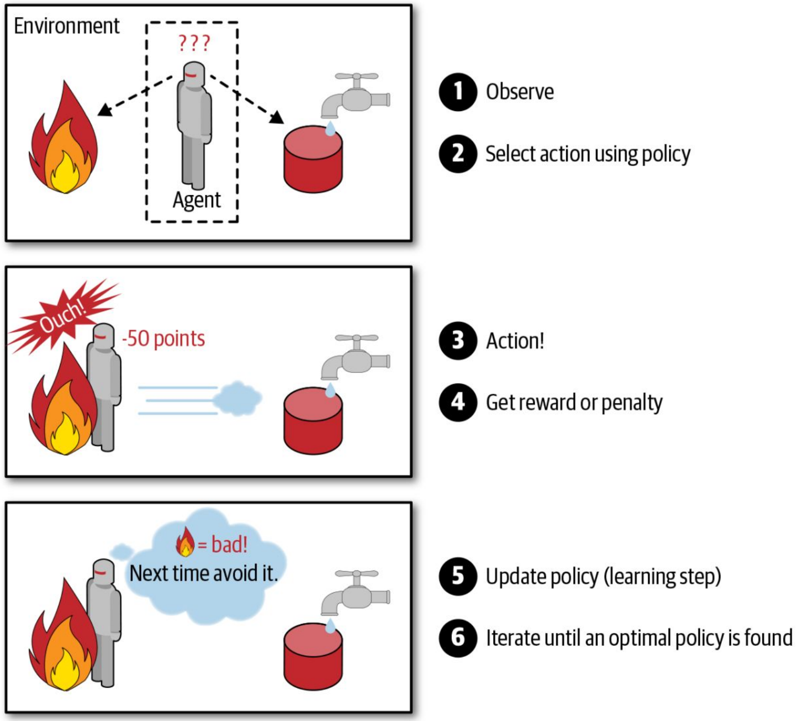


Slide Credit: Aurelien Geron, Hands-On ML with Scikit-Learn

Reinforcement Learning

- The learning system, called an *agent*
  - Can observe the *environment*
  - Select and perform *actions*
  - Get *rewards* (or *penalties*)
  - It must then learn by itself what is the best strategy, called a *policy*

Reinforcement Learning


Environment
???
Agent
1 Observe
2 Select action using policy
Ouch!
-50 points
3 Action!
4 Get reward or penalty
= bad!
Next time avoid it.
5 Update policy (learning step)
6 Iterate until an optimal policy is found

Slide Credit: Aurelien Geron, Hands-On ML with Scikit-Learn

Batch and Online Learning

- Batch Learning
  - The system must be trained using all the available data.
  - Generally take a lot of time and computing resources.
  - Also called *offline learning*.
- Online Learning
  - Also called *Incremental learning*.
  - Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives.
  - Good option for systems that need to adapt to change rapidly or autonomously.
  - Important parameter for online learning is *learning rate*.
    - Rapidly adapt new data, but will also tend to quickly forget old data (**high learning rate**)
    - It will learn more slowly, but it will also be less sensitive to noise in the new data (**low learning rate**)
    - **Big challenge:** if bad (noisy) data is fed to the system, the system's performance will gradually decline.

Slide Credit: Aurelien Geron, Hands-On ML with Scikit-Learn

Batch and Online Learning

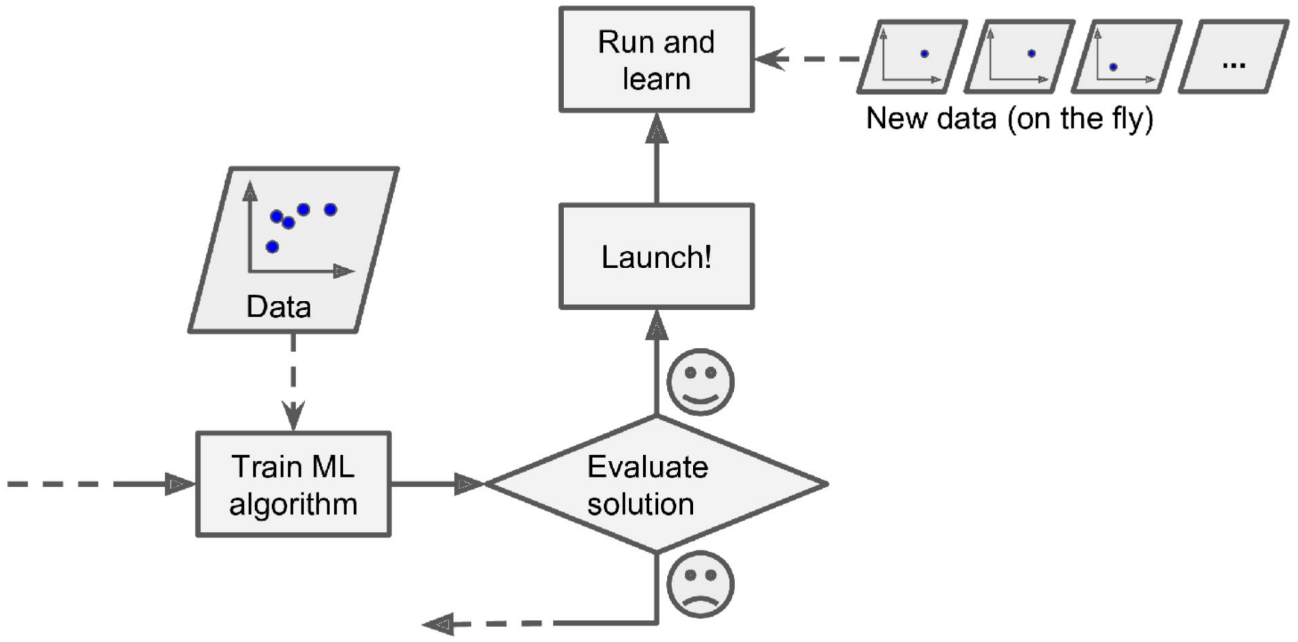
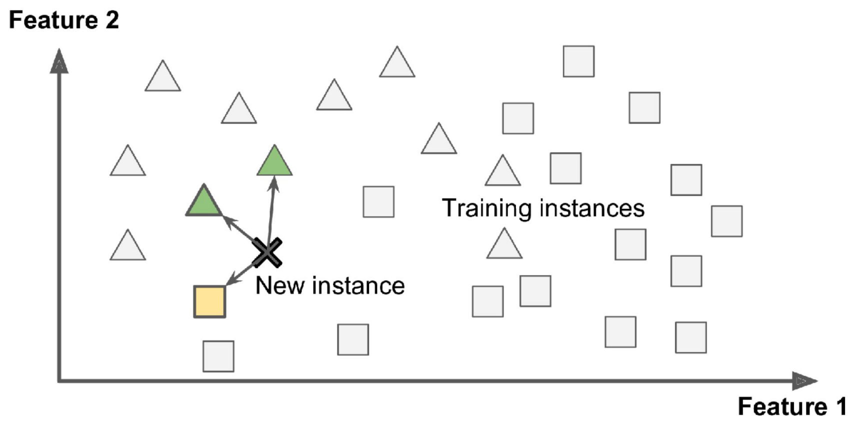


Figure 1-13. In online learning, a model is trained and launched into production, and then it keeps learning as new data comes in

Slide Credit: Aurelien Geron, Hands-On ML with Scikit-Learn

Instance-Based vs Model-Based Learning

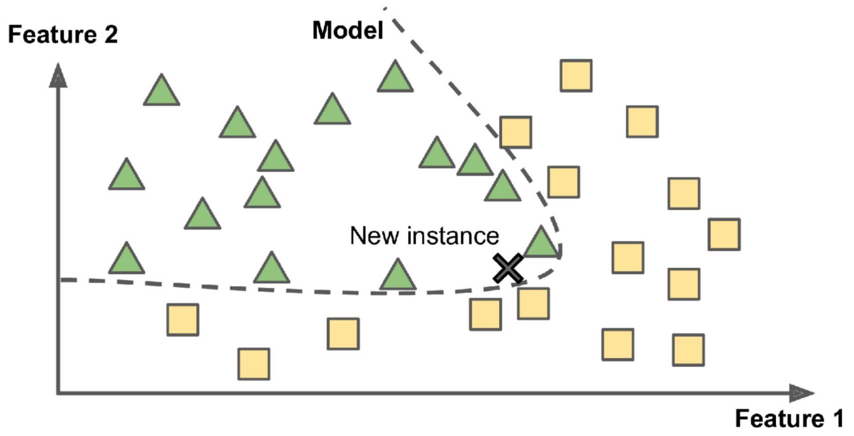
- **Instance-based learning**
  - Trivial form of learning
  - Requires a *measure of similarity*.
  - Sometimes called *memory-based learning*.
  - Hypothesis complexity can grow with the data in the worst case, a hypothesis is a list of *n* training items and the computational complexity of classifying a single new instance is *O(n)*.



Slide Credit: Aurelien Geron, Hands-On ML with Scikit-Learn

Instance-Based vs Model-Based Learning

- **Model-based learning**
  - Build a model of these examples and then use that model to make *predictions*.



Slide Credit: Aurelien Geron, Hands-On ML with Scikit-Learn

Main Challenges of ML

- Insufficient Quantity of Training Data
  - *The unreasonable effectiveness of data*
- Nonrepresentative Training Data
  - *Sampling bias.*
- Poor Quality Data
  - If your training data is full of errors, outliers, and noise.
- Irrelevant Features
  - Garbage in - garbage out.
  - Feature engineering
    - Feature selection, extraction
- Overfitting the Training Data
- Underfitting the Training Data

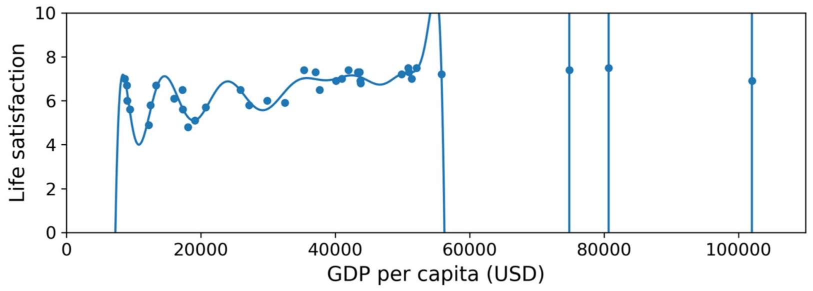


Figure 1-22. Overfitting the training data

Slide Credit: Aurelien Geron, Hands-On ML with Scikit-Learn

k-Nearest Neighbors

- One of the simplest machine learning algorithm.
- Building the model consists only of storing the training dataset.
- A lazy learner
    - Instead of learning a discriminative function from training data, it memorizes the training data.
- To make a prediction for a new data point, the algorithm finds the closest data points in the training dataset.
- Instance-based learning (subcategory of nonparametrics models)
    - Can’t be characterized by a fixed set of parameters, and the number of parameters grows with the training data.

Distance Metrics

- Euclidean Distance

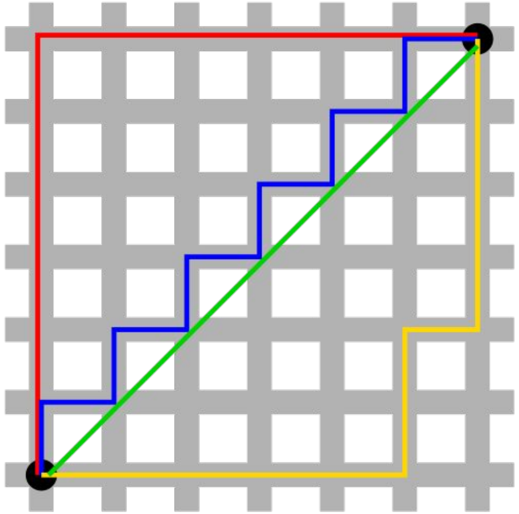
$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

- Manhattan Distance

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

- Minkowski Distance

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}.$$



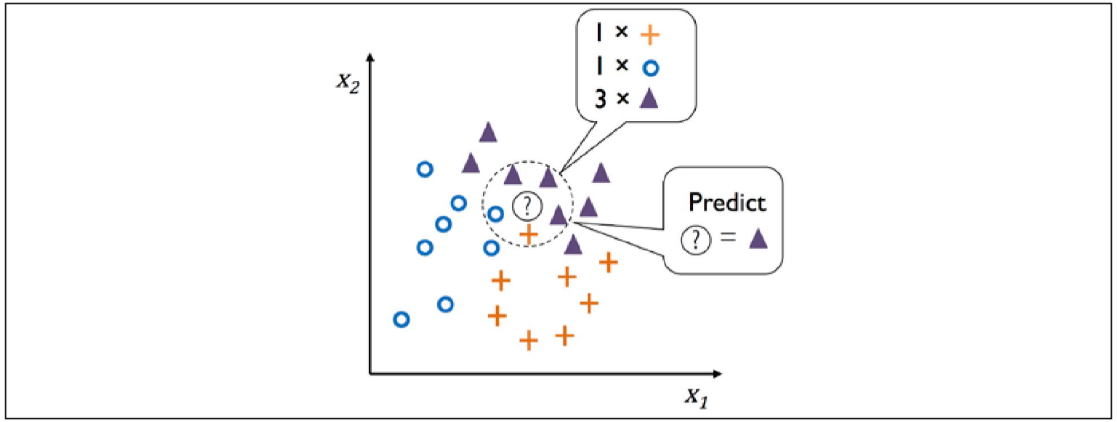
scikit-learn

p : *int, default=2*

Power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2. For arbitrary p, minkowski_distance (l_p) is used.

k-Nearest Neighbors

- Choose the number of *k* and a *distance metric*.
- Find the k-nearest neighbors of the data record that want to classify.
- Assign the class label by majority vote.
- Check this link for other hyperparameters of kNN.
- Demo: http://vision.stanford.edu/teaching/cs231n-demos/knn/



Slide Credit: **Sebastian** Raschka, Python Machine Learning

k-Nearest Neighbors

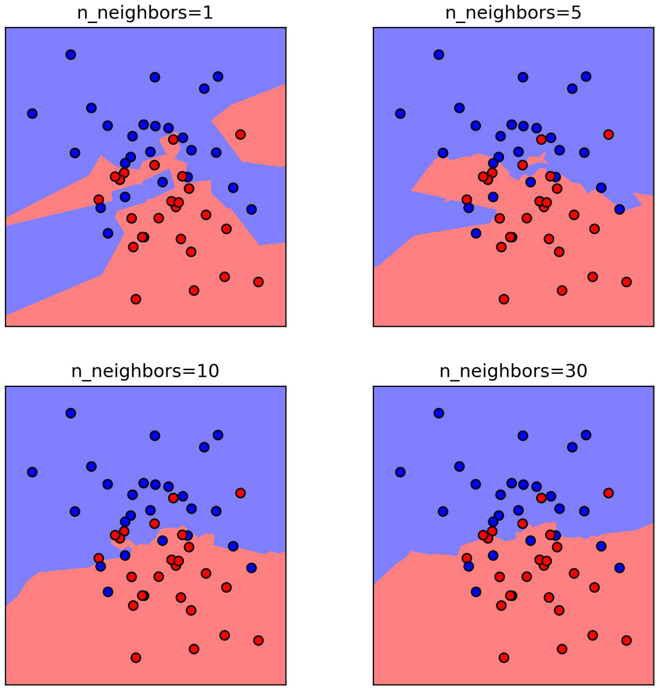
- Main advantage
  - Immediately adapts as we collect new training data
- Downside
  - The computational complexity for classifying new examples grows linearly with the number of examples in the training dataset in the worst-case scenario
  - We can't discard training examples since no training step is involved. Thus, storage space can become a challenge if we working with large datasets.

Slide Credit: Sebastian Raschka, Python Machine Learning

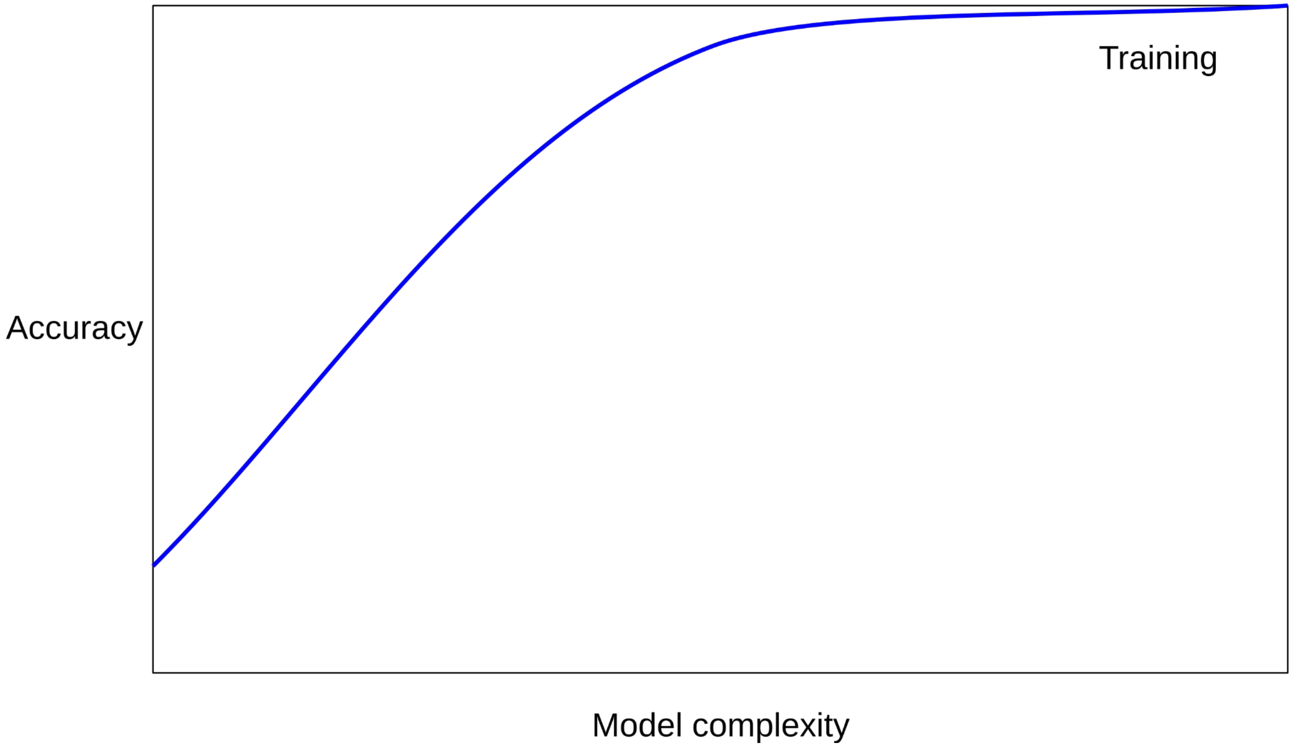
First ML Project with k-NN

- Check out the "**lecture-2-knn.ipynb**" notebook!
- The topics that we will cover on this notebook
  - KNN model training
  - Train/test split
  - Standardization of Data
  - Evaluation Metrics
  - Overfitting / Underfitting

k-Nearest Neighbors

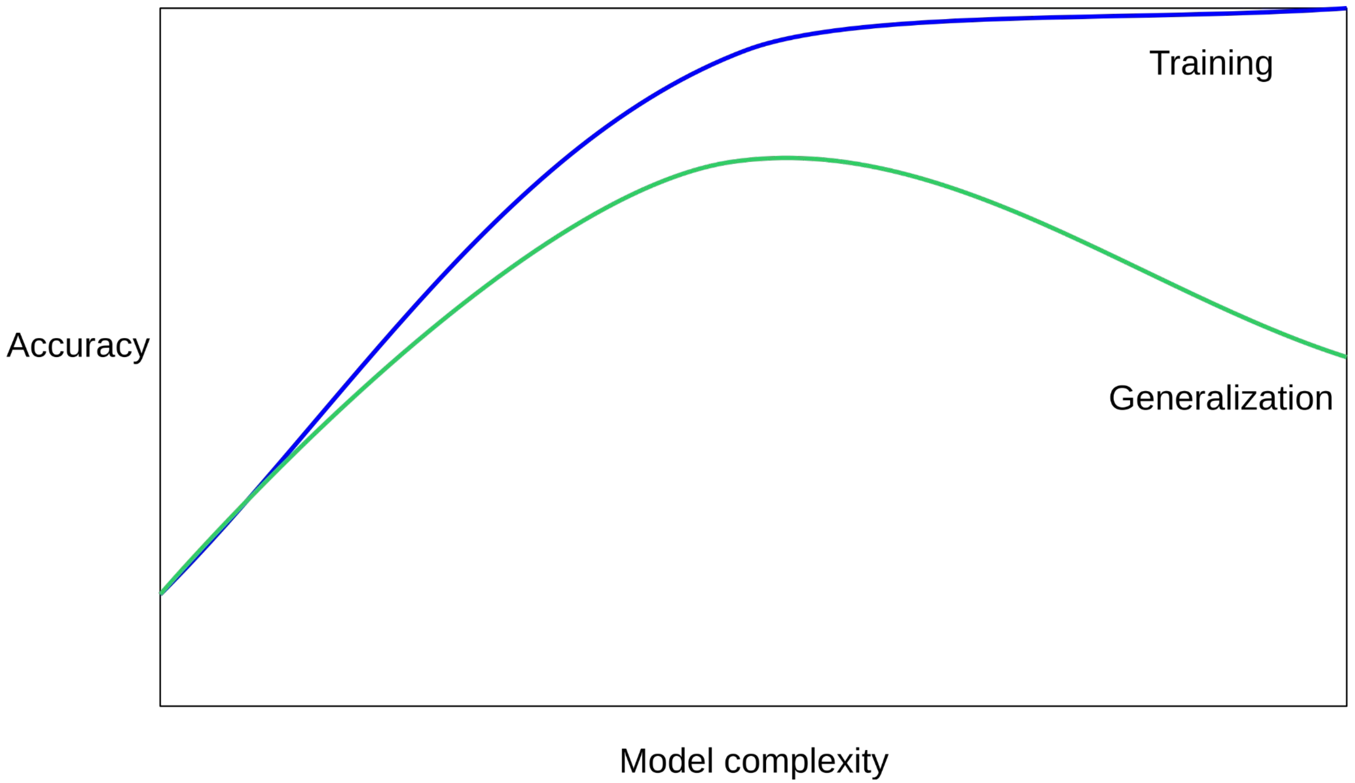


Overfitting / Underfitting



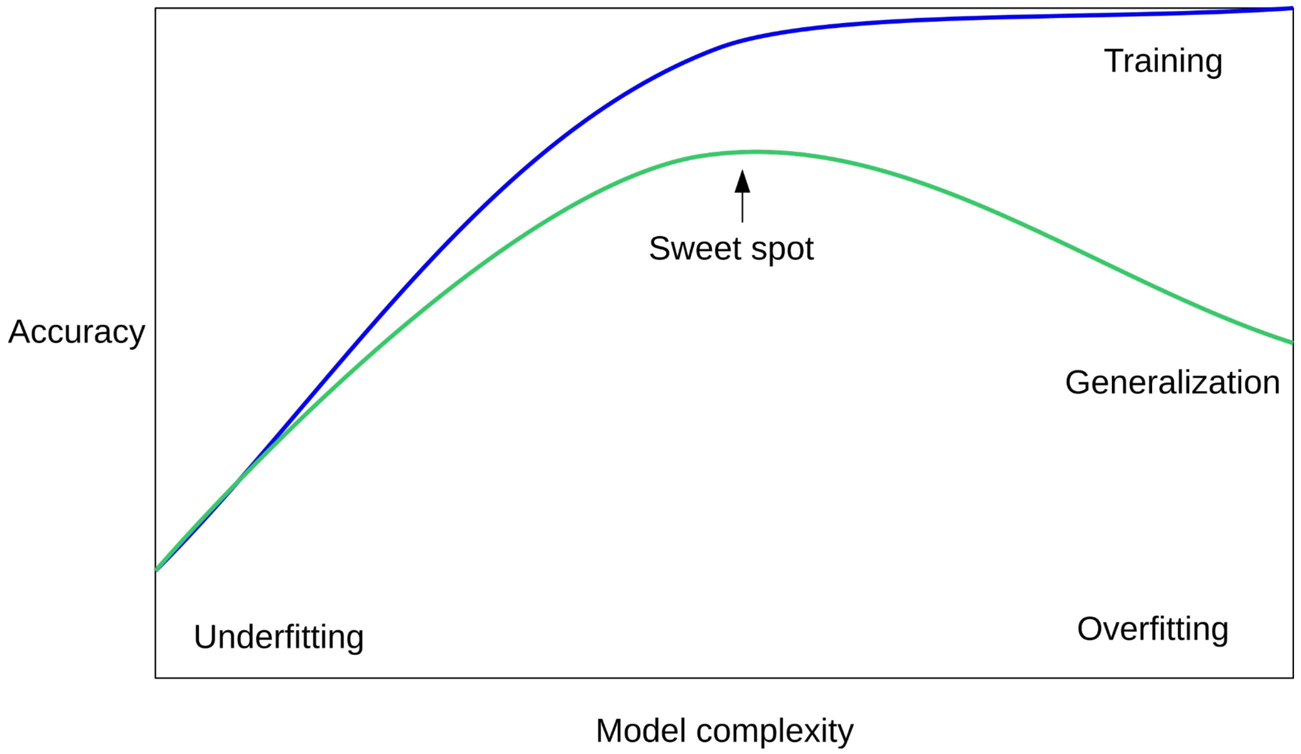
Slide Credit: Amuller, COMS4995

Overfitting / Underfitting

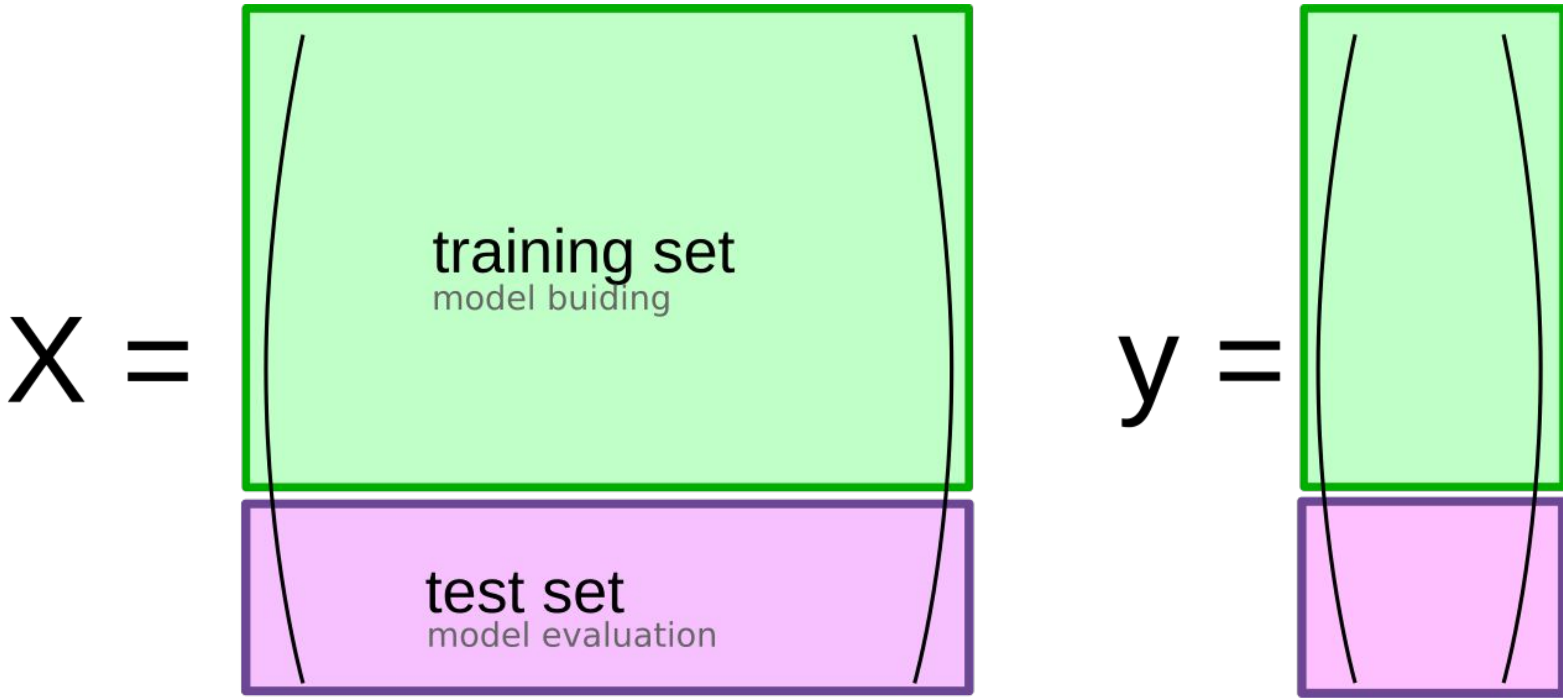


Slide Credit: Amuller, COMS4995

Overfitting / Underfitting

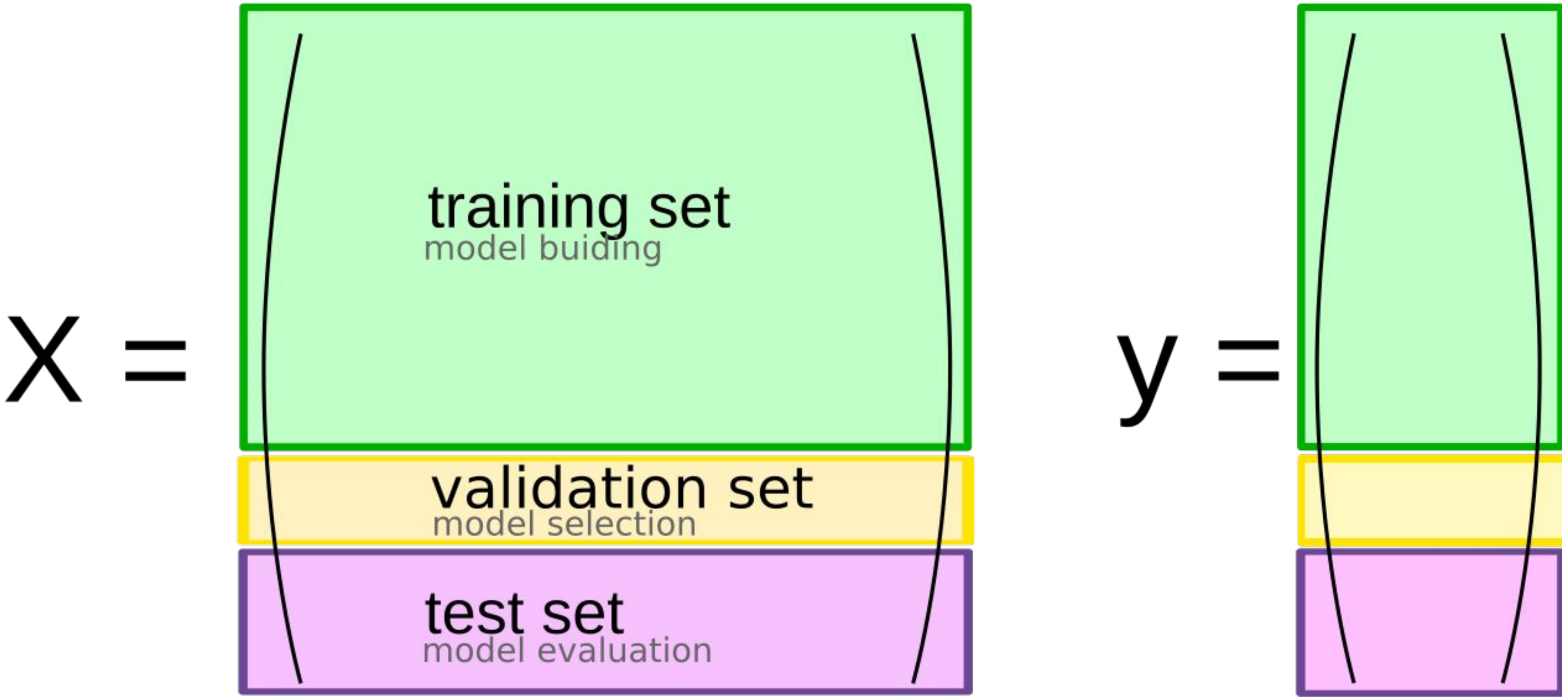


Train / Test Split



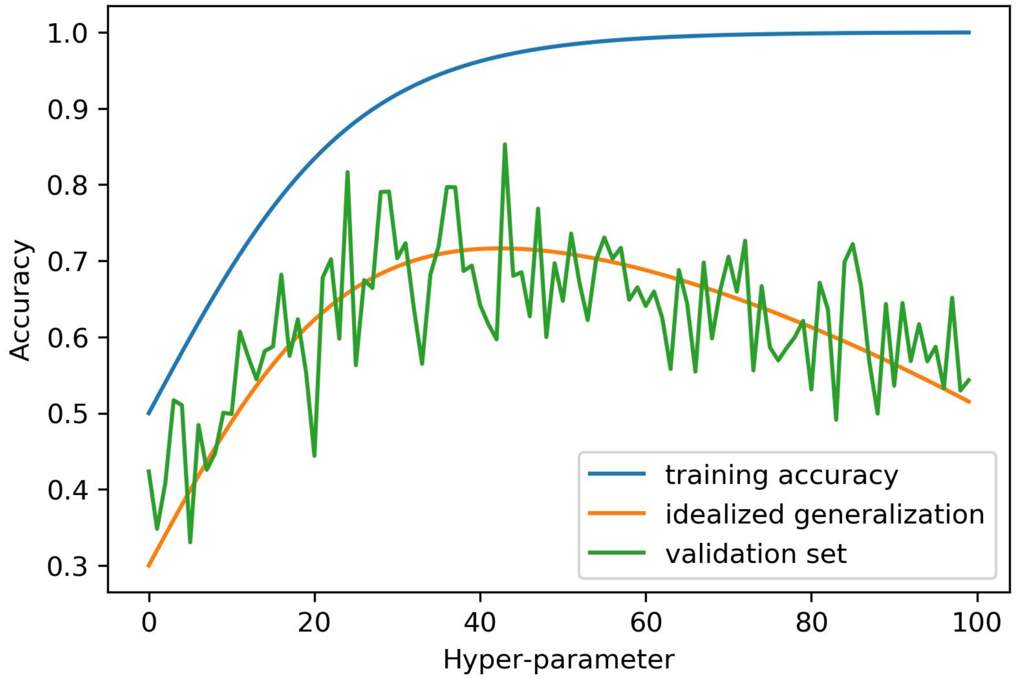
Slide Credit: Amuller, COMS4995

Train / Test Split



Slide Credit: Amuller, COMS4995

Train / Test Split



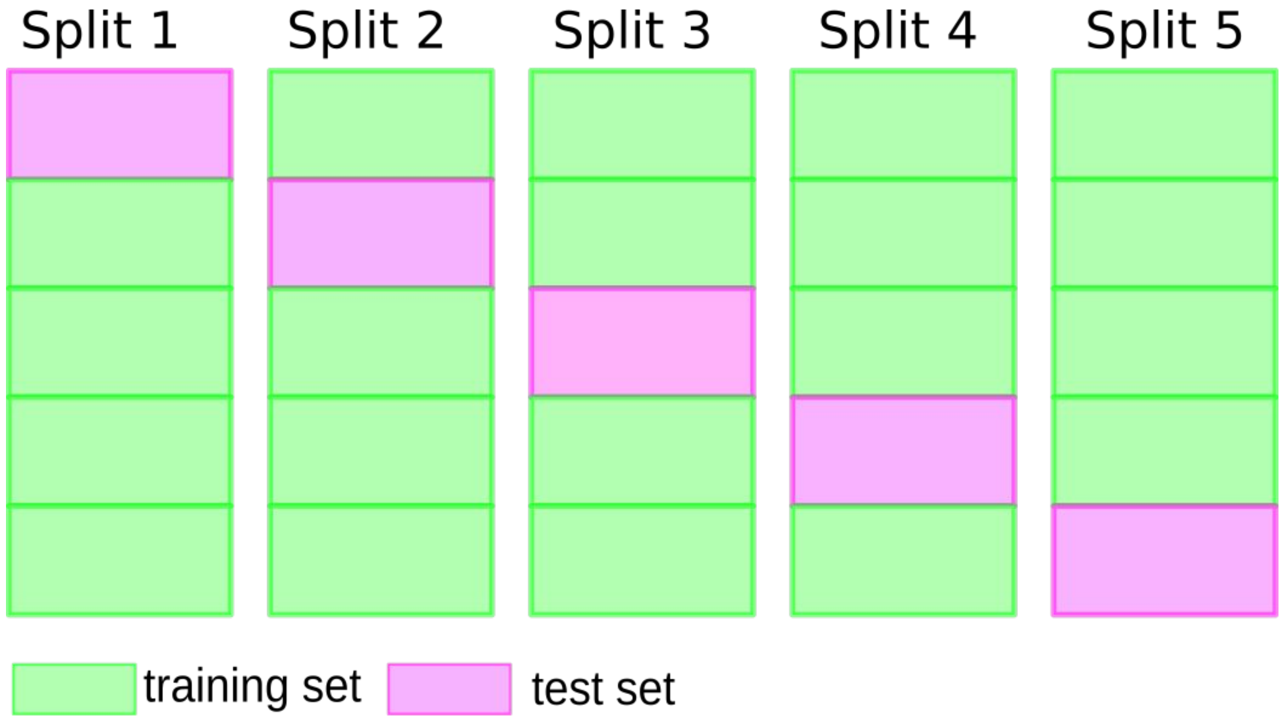
Train / Test Split

```python
X_trainval, X_test, y_trainval, y_test = train_test_split(X, y)
X_train, X_val, y_train, y_val = train_test_split(X_trainval, y_trainval)

val_scores = []
neighbors = np.arange(1, 15, 2)
for i in neighbors:
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    val_scores.append(knn.score(X_val, y_val))
print(f"best validation score: {np.max(val_scores):.3}")
best_n_neighbors = neighbors[np.argmax(val_scores)]
print("best n_neighbors:", best_n_neighbors)

knn = KNeighborsClassifier(n_neighbors=best_n_neighbors)
knn.fit(X_trainval, y_trainval)
print(f"test-set score: {knn.score(X_test, y_test):.3f}")
```

```
best validation score: 0.991
best n_neighbors: 11
test-set score: 0.951
```

Slide Credit: Amuller, COMS4995

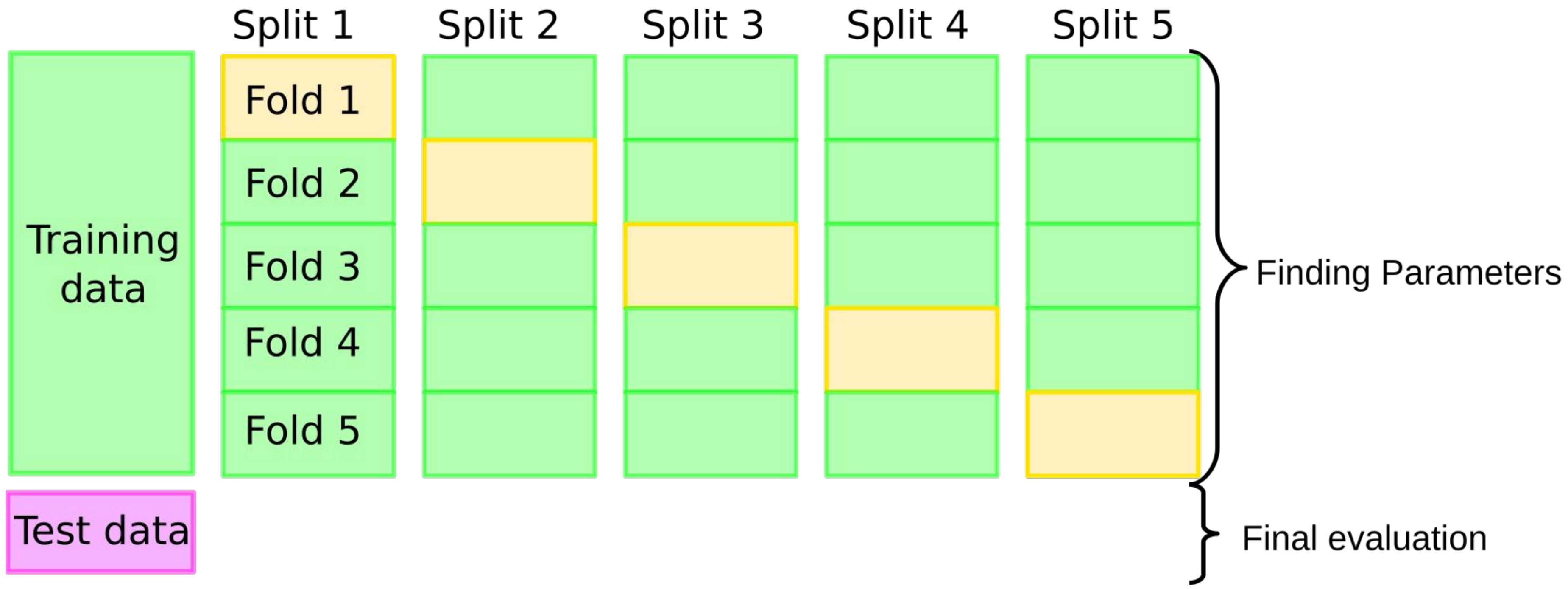
Cross-Validation



Adv: more stable, more data
Disadv: slower

Slide Credit: Amuller, COMS4995

Cross-Validation



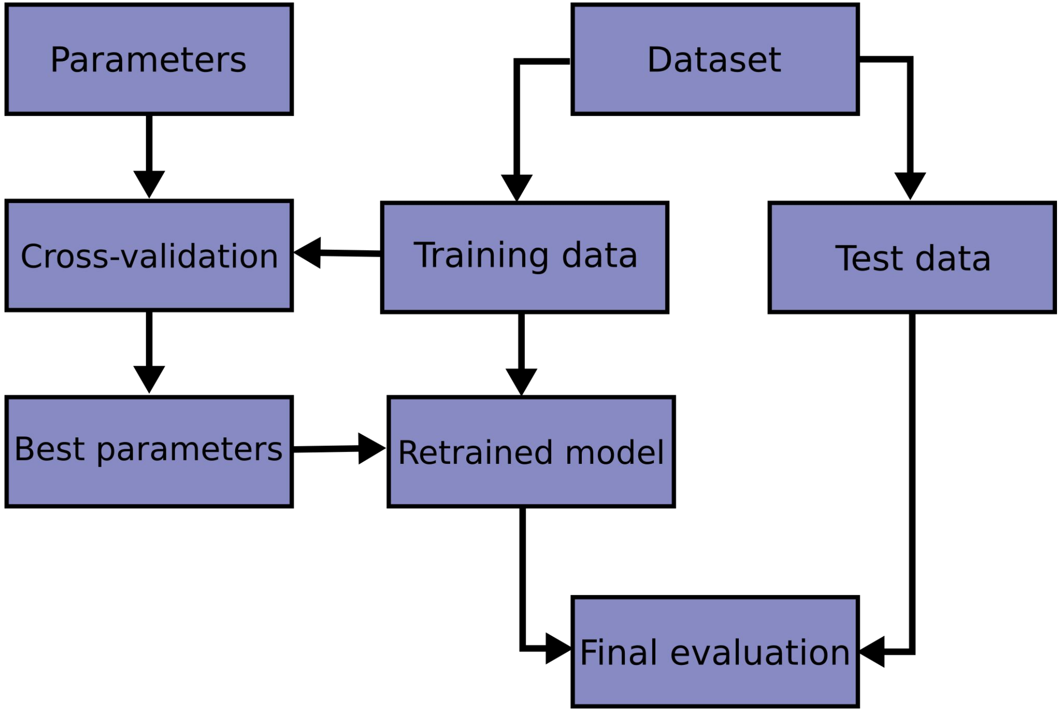
Grid-Search with Cross-Validation

```python
from sklearn.model_selection import cross_val_score

X_train, X_test, y_train, y_test = train_test_split(X, y)

cross_val_scores = []

for i in neighbors:
    knn = KNeighborsClassifier(n_neighbors=i)
    scores = cross_val_score(knn, X_train, y_train, cv=10)
    cross_val_scores.append(np.mean(scores))

print(f"best cross-validation score: {np.max(cross_val_scores):.3}")
best_n_neighbors = neighbors[np.argmax(cross_val_scores)]
print(f"best n_neighbors: {best_n_neighbors}")

knn = KNeighborsClassifier(n_neighbors=best_n_neighbors)
knn.fit(X_train, y_train)
print(f"test-set score: {knn.score(X_test, y_test):.3f}")
```

```
best cross-validation score: 0.967
best n_neighbors: 9
```

Slide Credit: Amuller, COMS4995

Pipeline for hyper parameter tuning



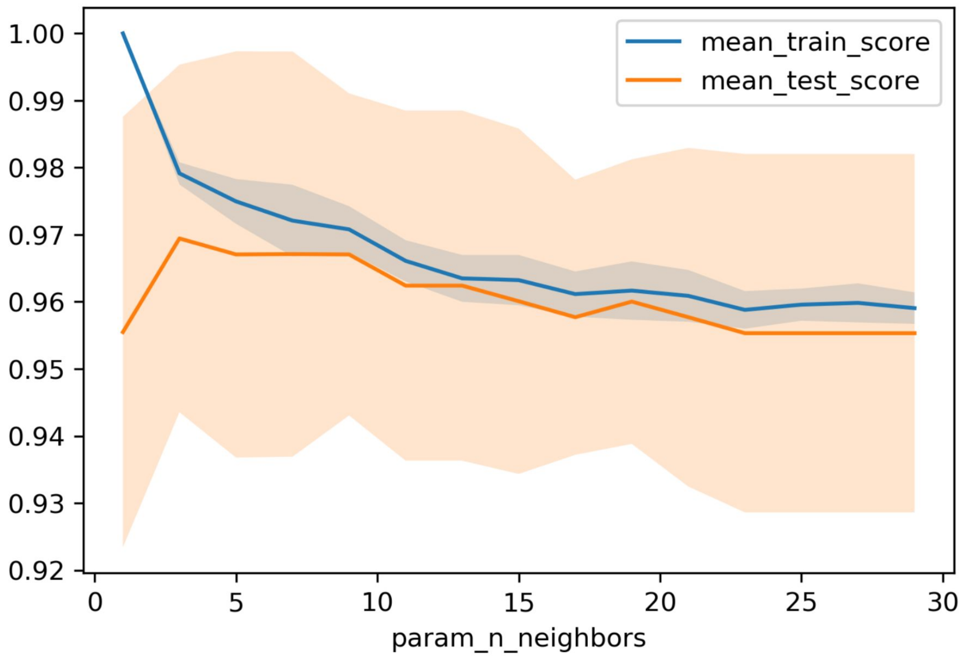
Parameter Optimization

```python
from sklearn.model_selection import GridSearchCV

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y)


param_grid = {'n_neighbors':  np.arange(1, 30, 2)}
grid = GridSearchCV(KNeighborsClassifier(), param_grid=param_grid, cv=10,
                    return_train_score=True)
grid.fit(X_train, y_train)
print(f"best mean cross-validation score: {grid.best_score_}")
print(f"best parameters: {grid.best_params_}")
print(f"test-set score: {grid.score(X_test, y_test):.3f}")
```

```
best mean cross-validation score: 0.967
best parameters: {'n_neighbors': 9}
test-set score: 0.993
```

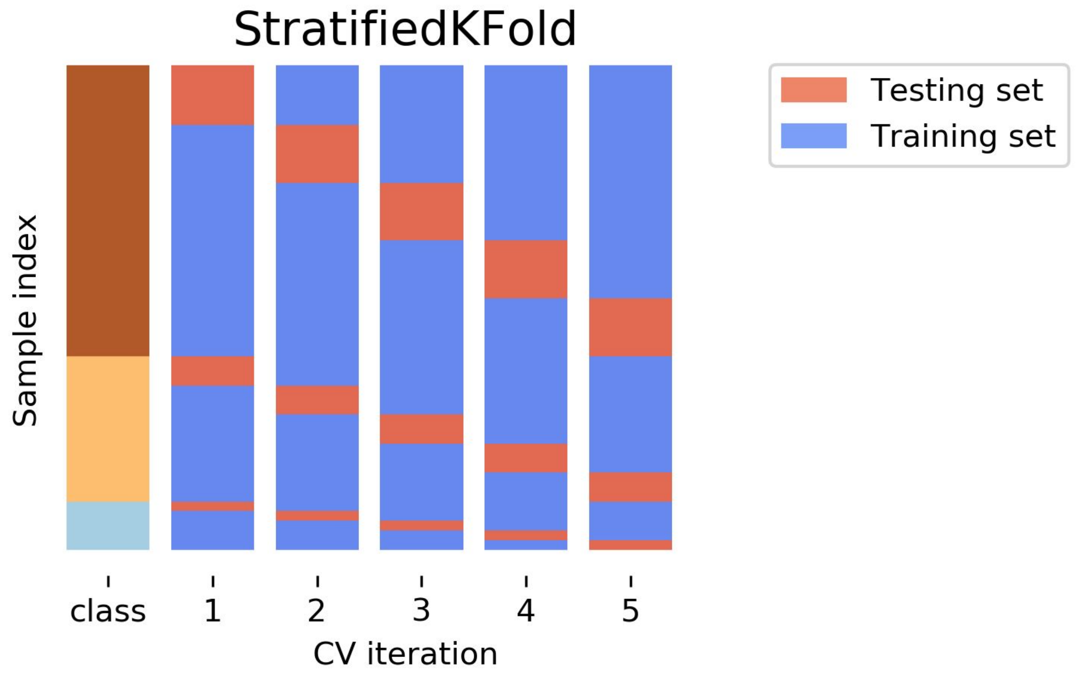
Slide Credit: Amuller, COMS4995

Grid Search Result (n_neighbors)



Slide Credit: Amuller, COMS4995

Stratified KFold



Stratified: Ensure relative class frequencies in each fold reflect relative class frequencies on the whole dataset.

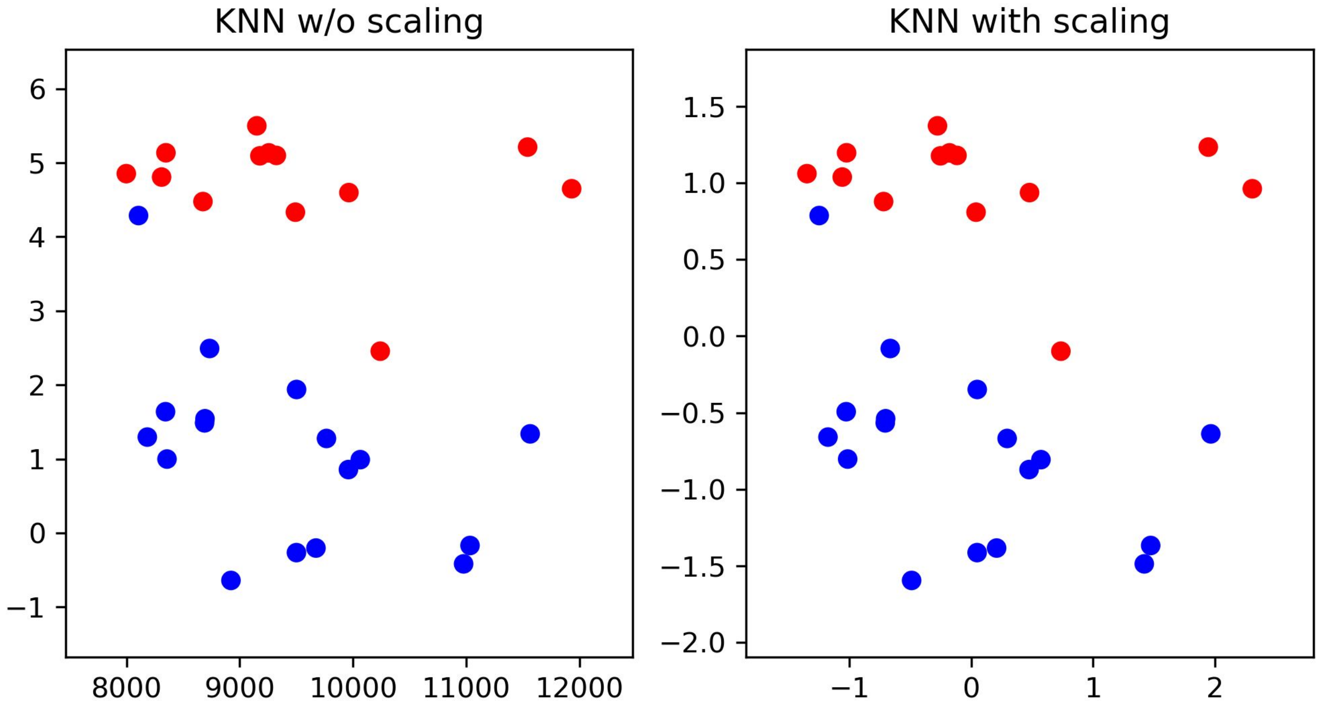
Slide Credit: Amuller, COMS4995

Further Reading on model evaluation & selection

- Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning
  - https://arxiv.org/pdf/1811.12808.pdf
- Cross-validation failure: small sample sizes lead to large error bars
  - https://hal.inria.fr/hal-01545002/file/paper.pdf

Feature Scaling

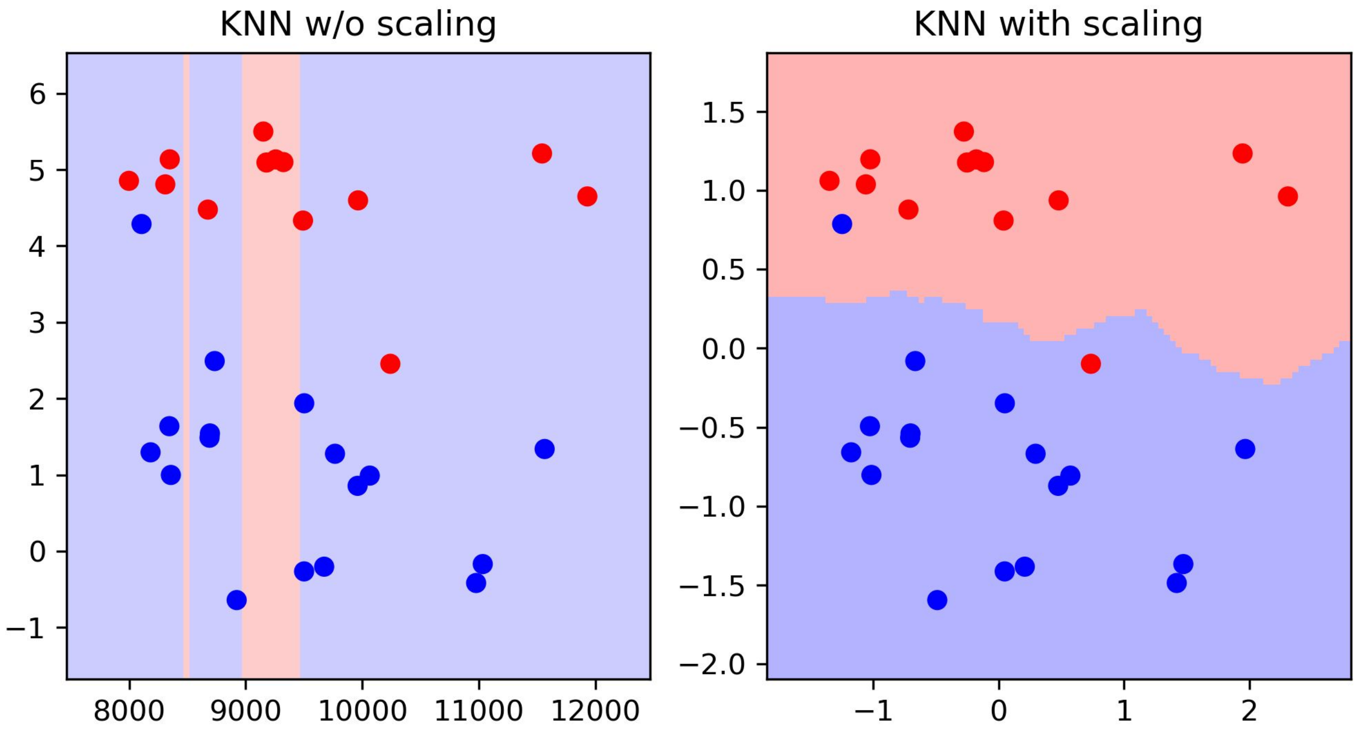
-



Slide Credit: Amuller, COMS4995

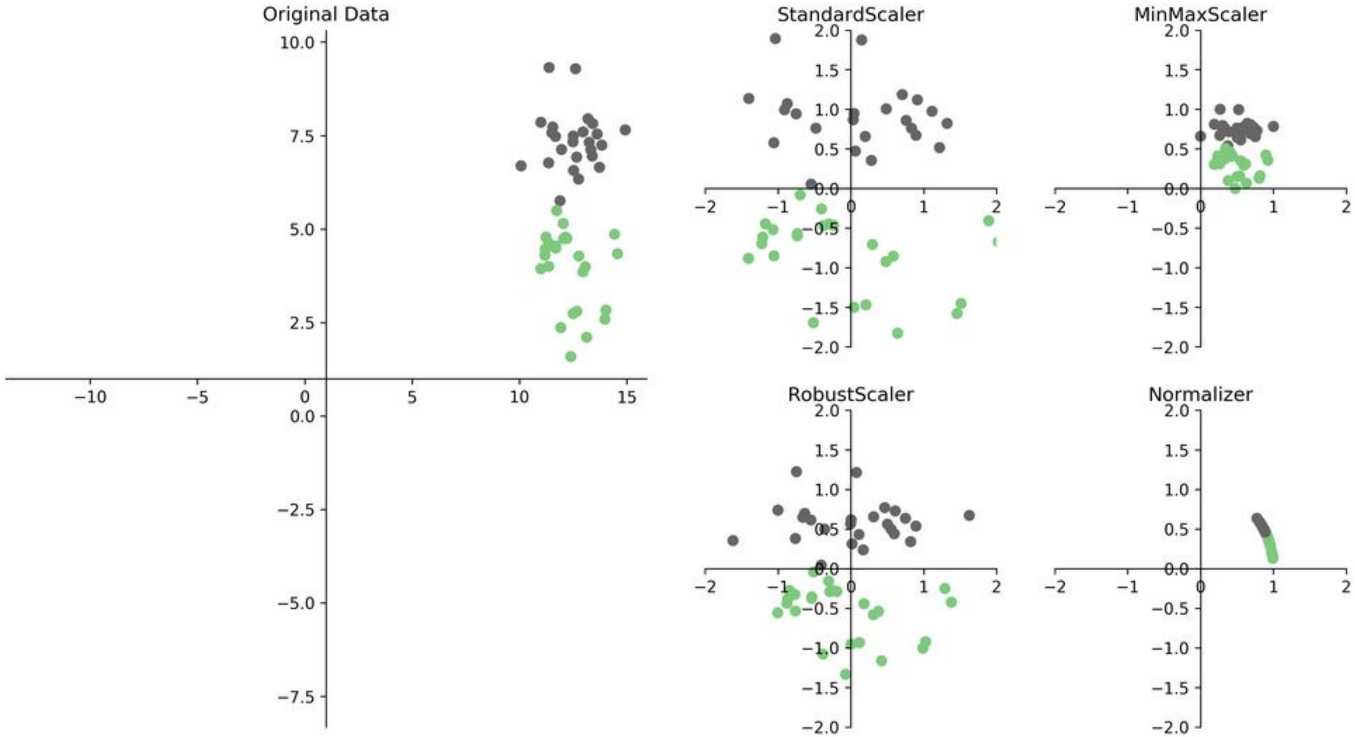
Feature Scaling

-



Slide Credit: Amuller, COMS4995

Different Methods for Scaling Data



Slide Credit: Amuller, COMS4995

Feature Scaling (Normalization)

- MinMax Scaler
  - Transformers features between 0 and 1.

$$\frac{x_i - \min(x)}{\max(x) - \min(x)}$$

- Standard Scaler
  - Assumes a normal distribution for data within each feature.
  - Scaling makes the distribution centered around 0, with standard deviation of 1 and the mean removed.

$$\frac{x_i - \text{mean}(x)}{sd(x)}$$

- Robust Scaler
  - Scale features that are robust to outliers. Similar to MinMax Scaler but it uses the interquartile range (rather than the min-max). The median and scales of the data are removed by this scaling algorithm according to the quantile range.

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$

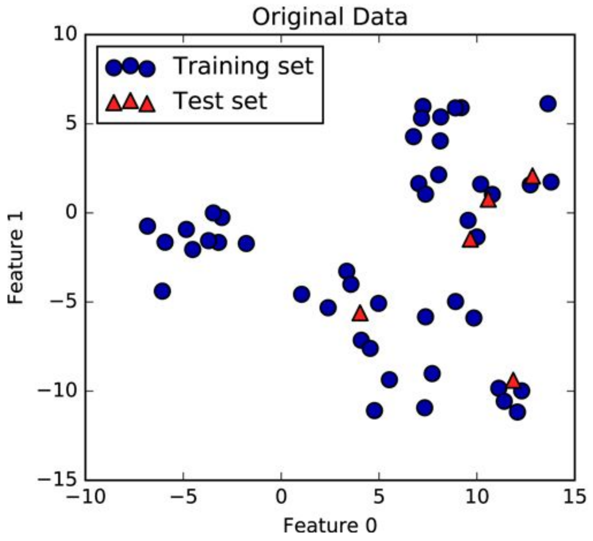
- For comparing methods
  - https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html

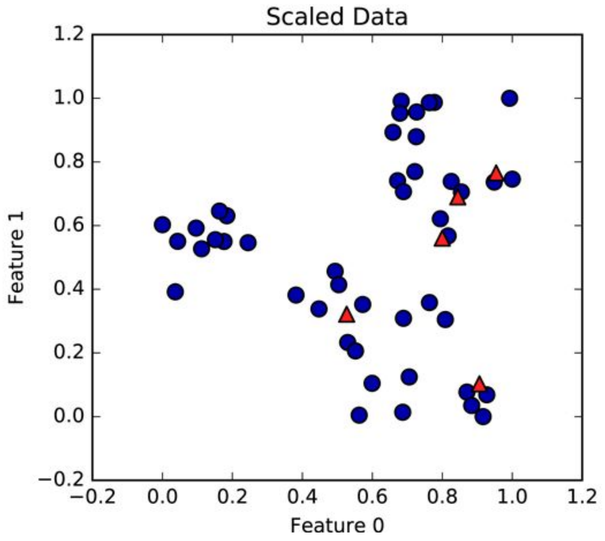
Slide Credit: machinelearninggeek.com/feature-scaling-minmax-standard-and-robust-scaler

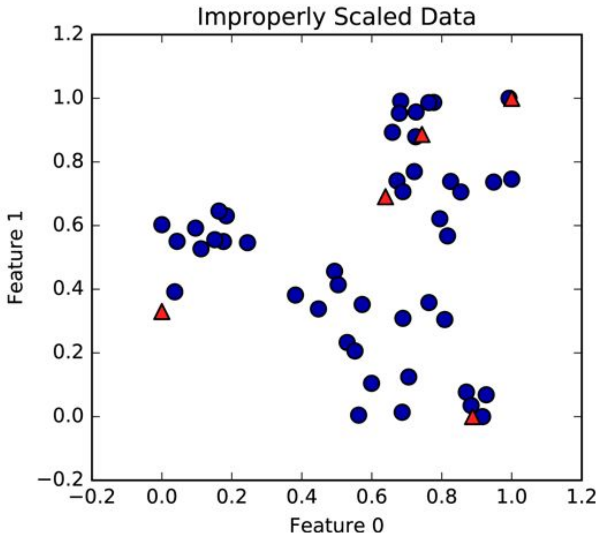
Improperly Scaled Data

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

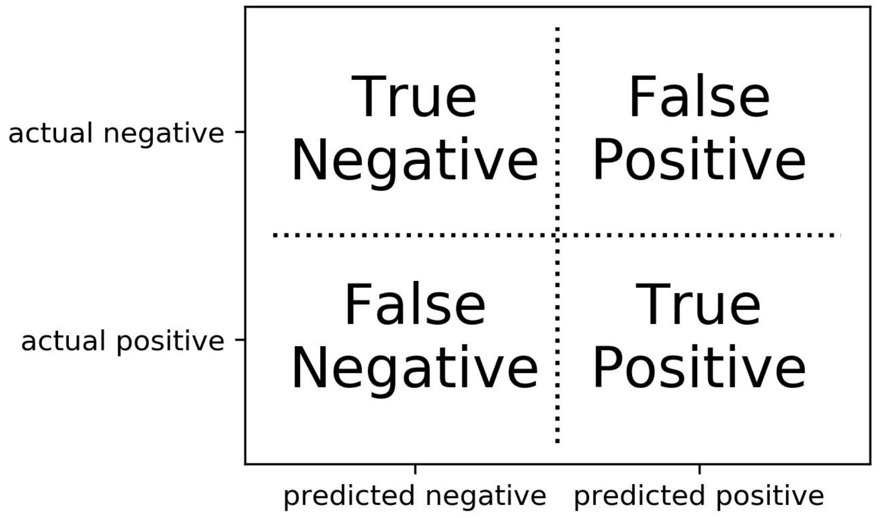
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)







Evaluation Metrics for Binary Classification



$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

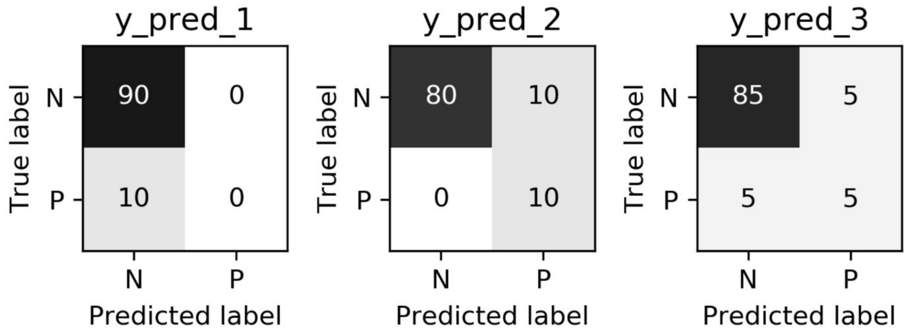
Slide Credit: Amuller, COMS4995

Problems with Accuracy

Data with 90% negatives:

```python
from sklearn.metrics import accuracy_score
for y_pred in [y_pred_1, y_pred_2, y_pred_3]:
    print(accuracy_score(y_true, y_pred))
```

```
0.9
0.9
0.9
```



Slide Credit: Amuller, COMS4995

Evaluation Metrics for Classification

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Positive Predicted Value (PPV)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Sensitivity, coverage, true positive rate.

$$\text{F} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Harmonic mean of precision and recall

Slide Credit: Amuller, COMS4995

Administrative

- You can do the projects in teams of at most 2 people.
- Please email the team member(s) till 26.09.2022 and send your project proposal till 30.09.2022.
- The project proposal should be max one page (min. 300 words) .Your project proposal should describe:
  - What is the problem that you will be investigating? Why is it interesting?
  - What reading will you examine to provide context and background?
  - What data will you use? If you are collecting new data, how will you do it?
  - What method or algorithm are you proposing? If there are existing implementations, will you use them and how? How do you plan to improve or modify such implementations? You don't have to have an exact answer at this point, but you should have a general sense of how you will approach the problem you are working on.
  - How will you evaluate your results? Qualitatively, what kind of results do you expect (e.g. plots or figures)? Quantitatively, what kind of analysis will you use to evaluate and/or compare your results?

Please submit your proposal as a PDF on Gradescope. **Only one person on your team should** submit. Please have this person add the rest of your team as collaborators as a "Group Submission".

Slide Credit: Stanford CS231n, project proposal , http://cs231n.stanford.edu/project.html

Administrative

- Expect some questions from the paper below in the midterm exam. (Reading assignment)
  - A Few Useful Things to Know about Machine Learning, *P. Domingos*
    - *https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf*

Next Class:

Linear Regression, Gradient Descent, and Regularization