DALARNA UNIVERSITY

Machine Learning

Supervised Learning
k-Nearest Neighbours for **Classification**

Roger G. Nyberg

Supervised Learning

DALARNA UNIVERSITY

- Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output
  `Y = f(X)`

- The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

X
Feature
Independent variable
Attribute
Predictor
Characteristic
Input (variable)

Y
Output (variable)
Dependent variable
Target
Label
Response
Outcome (variable)

Each collected set of data, often represented in **one row in a table**:
Case
Observation
Instance
Example

Model f((X): Refers to an algorithm as applied to a dataset, complete with its settings. The task for the model is to find the relationship in between the input features and the output targets

Supervised ***Regression*** Learning vs. Supervised ***Classification*** Learning

- Two kinds
  1. **Classification**: The output variable, called **target variable** or **label**, are categories, such as “red” or “blue” or “wine” and “beer”
     a. The dependent variable is categorical
  2. **Regression**: The target variable is continuous. Often the goal is to find out the line of the curve
     a. The dependent variable is numerical

- Both problems have as goal the construction of a model that can predict the value of the dependent attribute from the independent variables aka attribute variables

Training Dataset vs Testing Dataset

DALARNA UNIVERSITY

- The learning is based on data. The process is ***data-driven***
- Training Datasets
  - Training Dataset is the part of original dataset that you use to train your ML model.
  - The model learns on this data by running the algorithm and maps a function F(x) where “x” in the ***independent*** *variable* (inputs) for “y” where “y” is the ***dependent*** *variable* (output).
  - While Training your model on a dataset you provide both input and output variables to our model so that our model is able to ***learn to predict*** the output based on the input data.
- Testing Data
  - Is used for **validating the model**, and has been reserved from the original dataset
  - it is used to check the *accuracy* or *performance* of our model by comparing the predicted outcome of the model to the actual outcome of the test dataset.
  - You do not provide output variable to our model while testing although you know the outcome as Test data is extracted from the original dataset itself.
  - The model will provides you its predictions based on the learning from the training data
  - You then compare the predicted outcome with the original outcome to get the measure of ***accuracy*** or ***performance of our model*** based on ***unseen data***, i.e your testing data.

Never mix up training and testing data

DALARNA UNIVERSITY

- **Never train on test data!**
  - You must never let the model, i.e. the algorithm “see” the training data
  - If you are seeing surprisingly good results on your performance of your model, it might be a sign that you are accidentally training on the test set

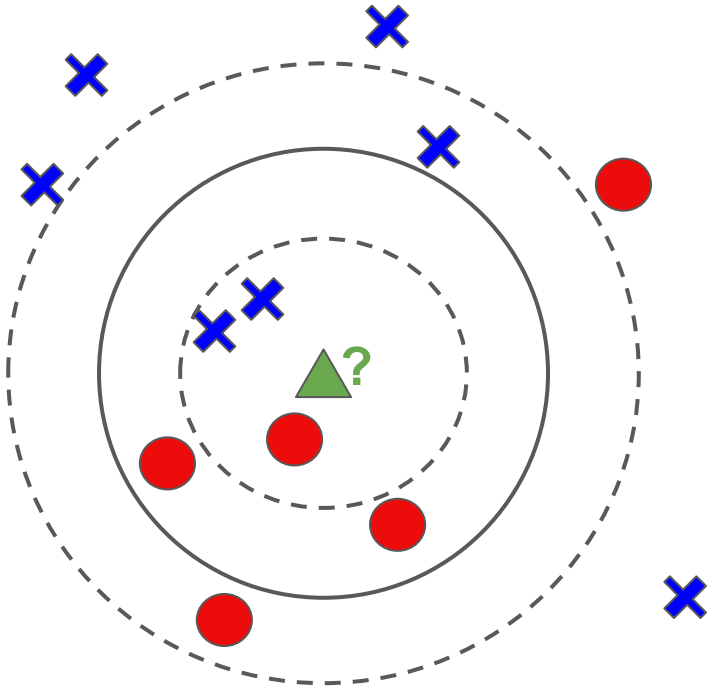
- Example: Consider a model that predicts whether an email is spam, using the subject line, email body, and sender's email address as features. We divide the data into training and test sets, with an 80-20 split. After training, the model achieves 99% precision on both the training set and the test set. We'd expect a lower precision on the test set, so we take another look at the data and discover that many of the examples in the test set are duplicates of examples in the training set (we neglected to scrub duplicate entries for the same spam email from our input database before splitting the data). We've inadvertently trained on some of our test data, and as a result, we're no longer accurately measuring how well our model generalizes to new data.

The k-Nearest Neighbors Algorithm

DALARNA UNIVERSITY

- Is a **non-parametric** and **lazy** algorithm used for *classification* and *regression*
- The input consists of the *k* closest training examples in the feature space
- The algorithm is **non-parametric**, i.e. it does not make any assumptions on the underlying data distribution, eg normal, or exponential distributions, gaussian mixtures, linearly separable et
- It is also a **lazy** algorithm, or a just-in-time learner
    - It doesn't learn a discriminative function from the training data but memorizes the training dataset instead. For example, the logistic regression algorithm learns the model weights during training time. In contrast, there is *no training time* in K-NN
- The output depends on whether k-NN is used for *classification* or *regression*
- Here we consider kNN for ***classification**, only*

Basic outline, k-NN



How K-Nearest Neighbors Classifier Works

https://www.youtube.com/watch?v=v6278Cjf_qA&feature=youtu.be (10 min)

How does kNN algorithm work in general?

- The algorithm classifies **a data point** based on **how close it is to its neighbours** AND **how they are classified**
- The algorithm stores all data classification cases. It uses these to classify new cases based on a similarity measure.
    - In case of classifying beer vs wine, this similarity measure could for example be colour (red, green, blue) and alcohol content (%).
- In forehand ***k*** has to be selected, i.e. an integer ≥ 1
    - Choose an odd number!
- Around *the new data point* to be classified, it counts ***k*** of already classified cases. 
- A commonly used **distance metric** is the Euclidean distance
- The new data point will be assigned to the class which has the most frequent class cases among the ***k*** nearest training samples

General Approach, Overview of steps

1. Data Acquisition
2. Explore Data
3. Normalise Data
4. Create Training Data and Test Data Sets
5. Build the classifier and Train a model on data
6. Assess the performance of the kNN classifier

- Each and every step will now follow.

DALARNA UNIVERSITY

1. Data Acquisition

- Machine learning typically starts from observed collected data.
- You can take your own data set or browse through other sources to find one

`iris` - Fisher's or Anderson's iris data set

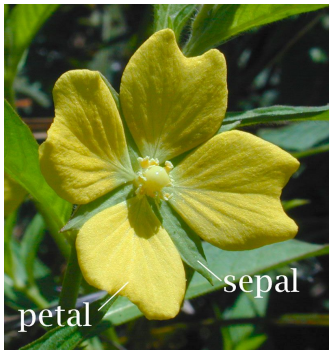
This famous (Fisher's or Anderson's) iris data set gives the measurements (in centimeters) of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *setosa, versicolor,* and *virginica.*

R. A. Fisher, 1890-1962

Usage:

```
# read in data as a Pandas dataframe
df = pd.read_csv('iris.csv')
```

Format: `iris` is a *data frame* with 150 cases (rows) and 4 features plus one target class (columns) named: `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`, and the target class `Species`



See: https://www.kaggle.com/uciml/iris

2. Explore Data: 4 Features and 3 Target classes

Features in the Iris data

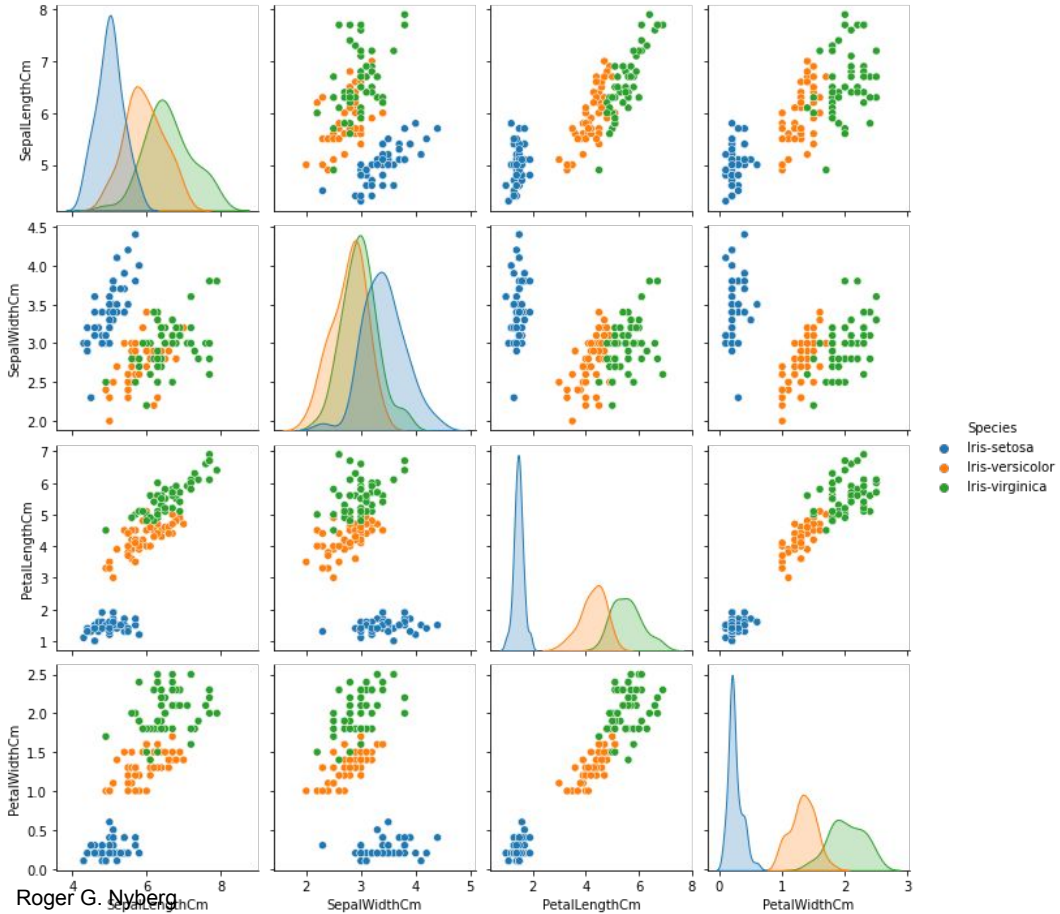
1. sepal length in cm, in `Sepal.Length`
2. sepal width in cm, in `Sepal.Width`
3. petal length in cm, in `Petal.Length`
4. petal width in cm, in `Petal.Width`

Target classes (or *labels*) to predict:

1. Iris Setosa
2. Iris Versicolour
3. Iris Virginica

} in `Species`

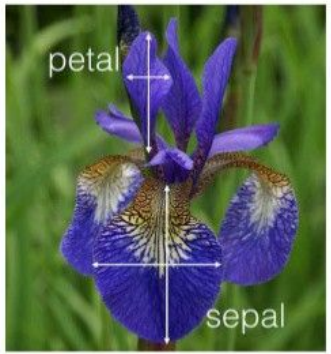
Each row having four features maps to one of the three target classes



2. Explore Data

```
# preview dataset
df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa



- Also, do note the pairwise graphical diagrame on the previous slide!!

2. Explore Data

```
# numerical statistics of dataset
round(df.describe(),1)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.0	150.0	150.0	150.0	150.0
mean	75.5	5.8	3.1	3.8	1.2
std	43.4	0.8	0.4	1.8	0.8
min	1.0	4.3	2.0	1.0	0.1
25%	38.2	5.1	2.8	1.6	0.3
50%	75.5	5.8	3.0	4.4	1.3
75%	112.8	6.4	3.3	5.1	1.8
max	150.0	7.9	4.4	6.9	2.5

For example, look at the range for each feature, e.g Sepal-Length [4.3, 7.9]
Should we normalise the data?

3. Normalise Data

- In short: Getting all data on the same scale
  - If the scales for different features are wildly different, this can have a knock-on effect on your ability to learn
- Feature scaling
  - Scales the data in the range of [0,1]
- Example: Normalising using Pandas column by column:
  - https://www.datasciencemadesimple.com/scaling-normalizing-column-pandas-dataframe-python/

$$X_{i,\ 0\ \text{to}\ 1} = \frac{X_i - X_{Min}}{X_{Max} - X_{Min}}$$

DALARNA UNIVERSITY

4. Create Training Data and Test Data Sets

- Why?
- We are going to train the kNN-classifier using **a part** of our collected data, i.e. in this context the training data is a fraction of the 150 iris observations, one observation per row.
- After the training we are going to **assess the performance** of the classifier. For this purpose we will use the test set, which (in this context) will be the remaining fraction of the 150 iris observations.
- Typically the collected data i splitted into two sets, training data (~70-80%) and test data (~20-30%).
-

DALARNA UNIVERSITY

5. Build the classifier and Train a model on data

- Find the k nearest neighbors of the training data set
- Create an instance of the KNeighborsClassifier class (scikit-learn), with the k number of neighbors
  - `knn = KNeighborsClassifier(n_neighbors=k)`
- Next, the `fit` method in k-NN stores the training data (X_train and y_train) within the classifier object, `knn`. Unlike many other machine learning algorithms, k-NN doesn't perform any complex computations or model training during this step. It just memorizes the training dataset
  - `knn.fit(X_train, y_train)`
- Next, the process of **finding the nearest neighbors occurs**!!
  - `y_pred = knn.predict(X_test)`

6. Assess the performance of the kNN classifier

One way of displaying its' performance is to make use of a **confusion matrix**

Based on the `test` data the kNN classified i.e predicted the specie to be:

```
confusion_matrix = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'])
print (confusion_matrix)
```

Predicted	Iris-setosa	Iris-versicolor	Iris-virginica
Actual			
Iris-setosa	13	0	0
Iris-versicolor	0	5	1
Iris-virginica	0	0	11

And this are the **correct** values, i.e the targets (or labels)!

Incorrect prediction!

kNN requires **Numbers:** How to prepare

- The KNN function classifies data points by calculating a distance between the points (often the Euclidean distance).
- Hence, this mathematical calculation requires ***numbers***.
- All variables in KNN must therefore be converted to numerics.
- The data preparation for KNN often involves:

1. Fix all **NA** or **""** values
2. **Convert all factors** into a set of booleans, one for each level in the factor
3. No variable's range should have too large impact on the distance measurement. Hence, **normalise** the values of each variable to the range [0,1]

References

Bishop, C. M., ***Pattern Recognition and Machine Learning*** (Information Science and Statistics), 2006. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Duda, R., ***Pattern Classification***, 2nd ed., 2001, Wiley-Interscience

k-NN on Iris Dataset https://towardsdatascience.com/k-nn-on-iris-dataset-3b827f2591e

Scaling and normalizing a column in Pandas python
https://www.datasciencemadesimple.com/scaling-normalizing-column-pandas-dataframe-python/

Split Your Dataset With scikit-learn's train_test_split() https://realpython.com/train-test-split-python-data/

The ABC of Machine Learning https://towardsdatascience.com/the-abc-of-machine-learning-ea85685489ef

Machine Learning Crash course https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data