Introduction to Business Analytics

Lecture 12: Intro to Machine Learning in R

Iegor Vyshnevskyi

Woosong University

May 15/16, 2023

Agenda

1. Intro to Machine Learning
2. Machine Learning Process
3. Machine Learning Applications
4. Machine Learning Algorithms / Methods
5. Machine Learning in Practice
6. In-class Assignment

Acknowledgment: Used a number of open sources and materials from the web.

1. Intro to Machine Learning

*What is **Machine Learning?***

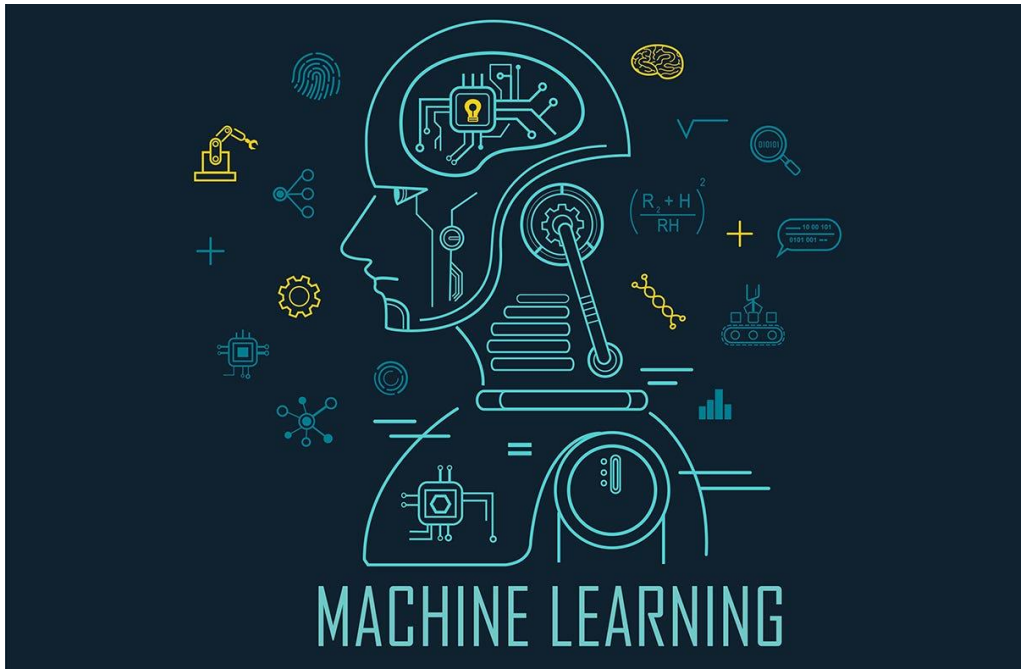
Machine Learning is a branch in computer science that studies the design of algorithms that can learn.

Arthur Samuel, a pioneer in the field of artificial intelligence and computer gaming, coined the term “Machine Learning”. He defined machine learning as – “Field of study that gives computers the capability to learn without being explicitly programmed”.

AI vs. ML

Although Artificial intelligence (AI) and machine learning (ML) are used interchangeably, but they differ with uses, data sets, and more.

- While AI encompasses the idea of a machine that can mimic human intelligence, ML does not.
- ML aims to teach a machine how to perform a specific task and provide accurate results by identifying patterns.



*What is **Machine Learning?** (cont.)*

- In general, *Machine Learning(ML)* can be explained as automating and improving the learning process of computers based on their experiences without being actually programmed i.e. without any human assistance.

- ML can be seen as a form of artificial intelligence (AI).

- “Machine learning is a scientific discipline that is concerned with the design and development of algorithms that allow computers to evolve behaviours based on empirical data, such as from sensor data or databases.” (Wikipedia)

- Primary goal of a ML implementation is to develop a general purpose algorithm that solves a practical and focused problem.

- Important aspects in the process include *data*, *time*, and *space* requirements.

- The goal of a learning algorithm is to produce a result that is a rule and is as accurate as possible.

How Do Machines Learn?

- Traffic prediction, face recognition, product recommendations, self-driving cars—machine learning is at the core of the future of technology.

- With machine learning, machines can match and even transcend human information processing and learning capabilities to solve complex problems.

- Machines and their learning accuracy determine how good they are at solving everyday problems. Understanding how machines learn should be a priority for organizations worldwide, so they can leverage their ability and take their operations to the next level.

Usefulness of ML for Business Analytics

- In today's quickly-evolving corporate landscape, companies must often engage in intense competition to secure users and customers.
- In the age of big data and in-depth analysis of customer behavior, machine learning (ML) solutions are emerging as the de facto way for companies to gain a competitive edge.
- ML was discovered to be a good fit for the corporate landscape, providing cost-effective solutions to problems that previously required a lot of resources.
- Driven by market leaders, such as Google, Microsoft, and Amazon, the corporate landscape as a whole has gravitated towards using ML.
- Smart automation has enabled businesses to effectively deploy low-cost, high-accuracy ML solutions to replace low-skilled workers.
- Machine learning has a variety of applications in the corporate sector, as its capabilities have made it a natural fit for the requirements of an ever-increasing market.

Usefulness of ML for Business Analytics

- 56% of organizations today are using machine learning in at least one business function, according to a recent McKinsey survey. That means that and ML (Machine Learning) will benefit more than half of companies in 2022.

Overall, ML is used in business to improve scalability and develop company operations for organisations worldwide and in any industry.

Advantages of using R for ML

- It provides ***good explanatory code***. For example, if you are at the early stage of working with a machine learning project and you need to explain the work you do, it becomes easy to work with R language comparison to python language as it provides the proper statistical method to work with data with fewer lines of code.
- R language is ***perfect for data visualization***. R language provides the best prototype to work with machine learning models.
- R language has the ***best tools and library packages to work with machine learning projects***. Developers can use these packages to create the best pre-model, model, and post-model of the machine learning projects. Also, the packages for R are more advanced and extensive than python language which makes it the first choice to work with machine learning projects.

2. Machine Learning Process

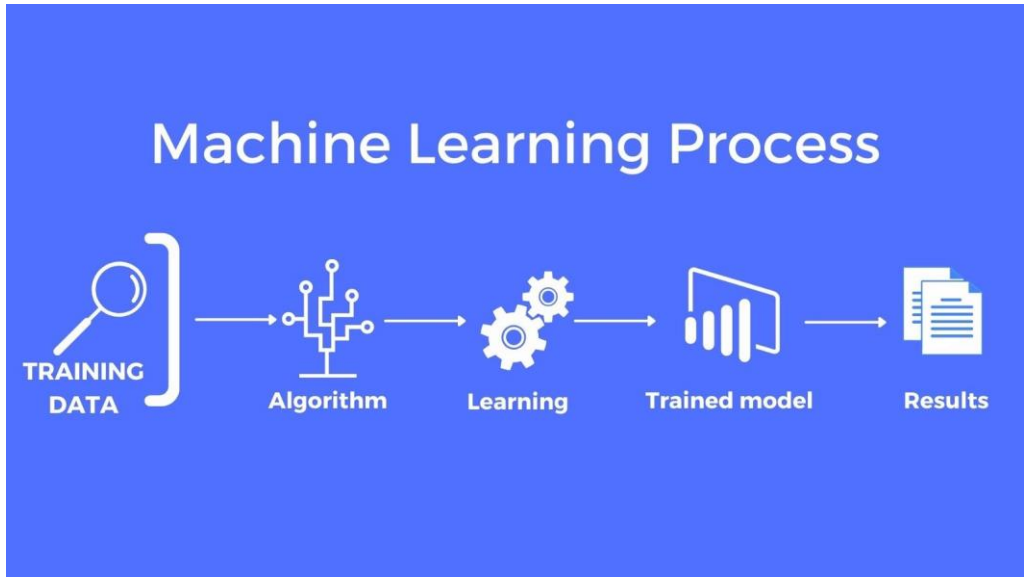
Machine Learning Process

The process starts with feeding good quality data and then training our machines(computers) by building machine learning models using the data and different algorithms.

The choice of algorithms depends on what type of data do we have and what kind of task we are trying to automate.

As such, there are three main phases in an ML process:

- Training Phase: Training Data is used to train the model by using expected output with the input. Output is the learning model.
- Validation/Test Phase: Measuring the validity and fit of the model. How good is the model? Uses validation dataset, which can be a subset of the initial dataset.
- Application Phase: Run the model with real world data to generate results



3. Machine Learning Applications

ML Applications

- ***Image Classification:*** the process by which algorithms are trained to analyze images and find out what they contain. Image classification as a machine learning solution has become highly popular in the enterprise sector, mainly due to its capability to disrupt existing systems created for the same purpose. Previously, human labor was required to go through vast amounts of data and label them. Today, giants like Facebook, Twitter, and Google are using image classification to prevent unwanted content from going viral.
- ***Predictive Modeling:*** is a category of ML solutions that mines large amounts of data to predict the outcomes of potential scenarios. These predictions can then be utilized to make informed business decisions. Predictive modeling algorithms essentially provide a forecast of the future based on past data, allowing companies to make business moves in preparation for these predictions.
- ***Real-time chatbot systems:*** one of the foremost forms of automation. ML enables chatbots to be more productive and more interactive (Alexa, Google Assistant, Siri, Watson Assistant).
- ***Customer recommendation engines:*** ML powers the customer recommendation engines built to deliver customized experiences and improve the overall customer experience. Here, algorithms analyze data points about each customer, including the customer's previous purchases, and other data sets like demographic trends, an organization's present inventory, and the purchase histories of other customers in order to know what services and products to offer as recommendations to each individual customer (Amazon, Walmart, Netflix, and YouTube).

And many more: https://www.techfunnel.com/information-technology/machine-learning-business-applications/

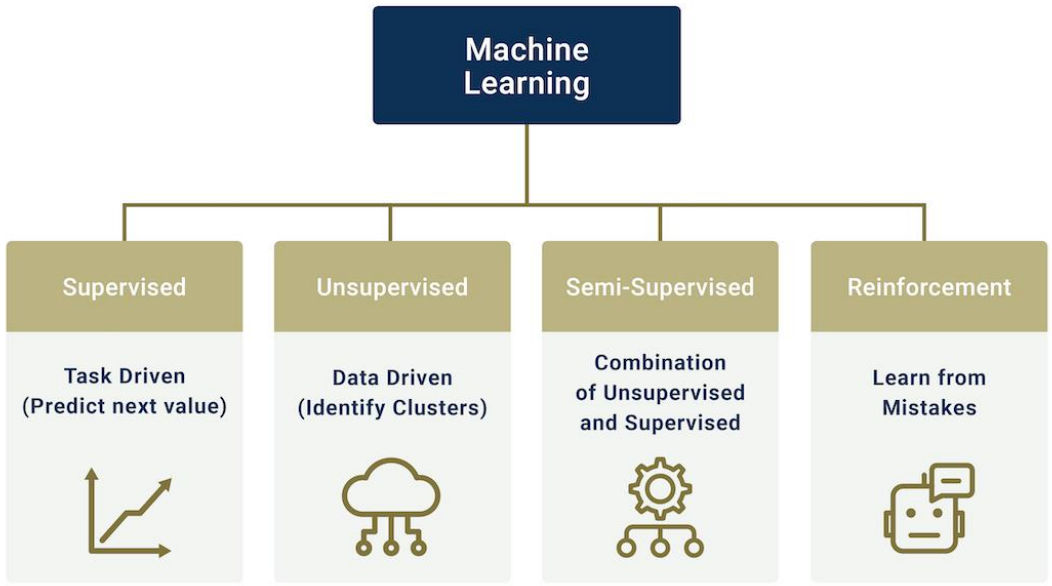
4. Machine Learning Algorithms / Methods

ML Algorithms

- Algorithms are *sets of rules* created to help computers perform problem-solving operations. Machine learning algorithms are systems that run the machine learning process. They are *the key* to how computers interact with data to learn how to interpret it and put it to use.
- Interacting with, interpreting, and using data can happen in many different ways. Machine learning algorithms must be trained to learn how to perform any task. This training often requires the machine *to be fed large volumes of data multiple times over*.
- The machine learns the specific representation after applying an algorithm to a dataset is referred to as a machine learning model or hypothesis.
- Upon multiple instances of using algorithms, computers get better at achieving desirable results. This type of zero interference model makes up the core of all machine learning processes.
    - Therefore, all machine learning relies on the availability of data to train the machines. This data must be well-organized and labeled to ensure the outcomes are accurate.
- There are *many types of machine learning algorithms* based on how the machine interacts with the data and what it needs to accomplish.

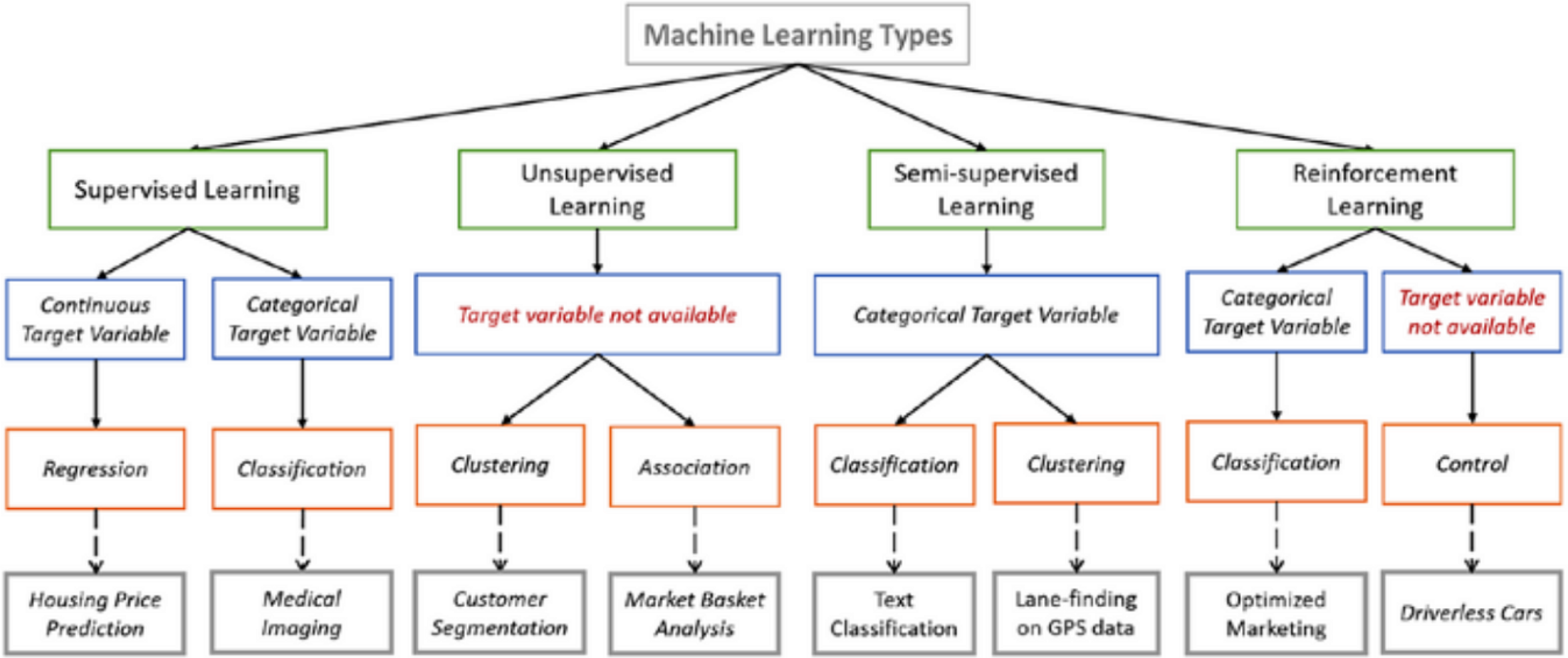
*Overview of **ML Algorithms***

There are mainly four types of learning.

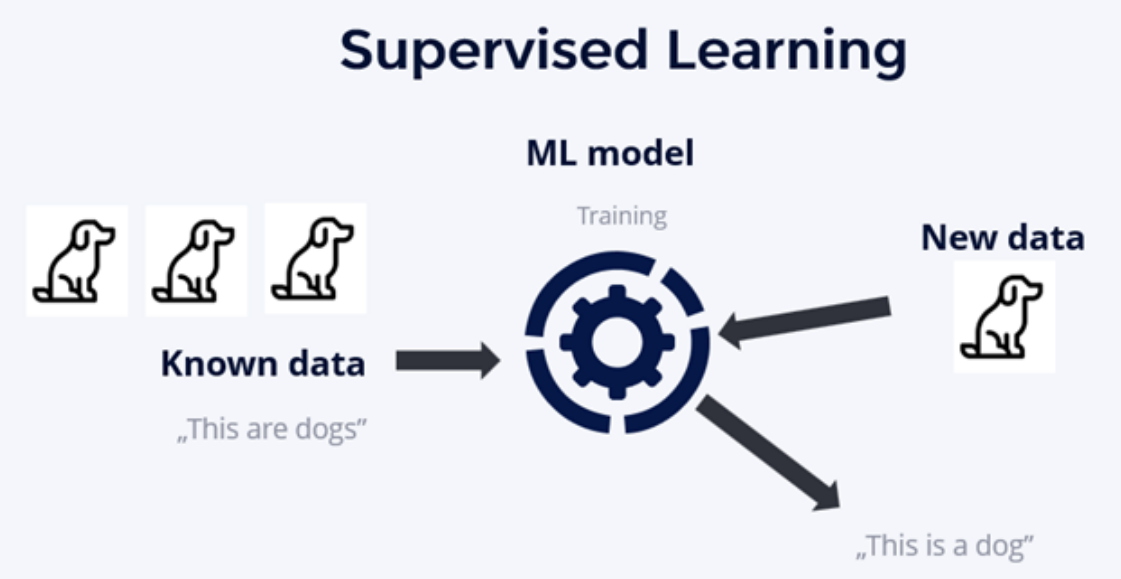


- In **supervised learning** (SML), the learning algorithm is presented with labelled example inputs, where the labels indicate the desired output. SML itself is composed of **classification**, where the output is categorical, and **regression**, where the output is numerical.
- In **unsupervised learning** (UML), no labels are provided, and the learning algorithm focuses solely on detecting structure in unlabeled input data.
- Note that there are also **semi-supervised learning** approaches that use labelled data to inform unsupervised learning on the unlabeled data to identify and annotate new classes in the dataset (also called novelty detection).
- **Reinforcement learning**, the learning algorithm performs a task using feedback from operating in a real or synthetic environment.

*Overview of **ML Algorithms** (cont.)*



Supervised *Learning vs.* ***Unsupervised*** *Learning*

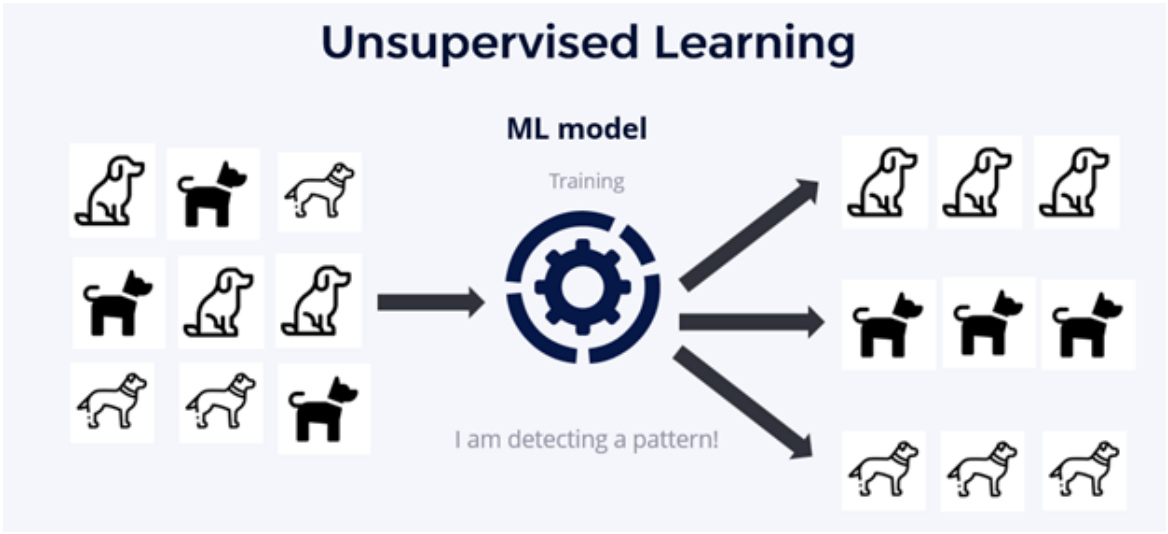


•**Classification**: A classification problem is when the output variable is a category, such as “Red” or “blue” or “disease” and “no disease”.

•**Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

•**Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.

•**Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.



5. Machine Learning in Practice

What we do today

k-Nearest Neighbors (KNN)

- The KNN or k-nearest neighbors algorithm is one of the simplest machine learning algorithms and is an example of instance-based learning, where new data are classified based on stored, labeled instances.
- It is a *Supervised Non-linear classification* algorithm. KNN is a Non-parametric algorithm i.e. it doesn't make any assumption about underlying data or its distribution. It is one of the simplest and widely used algorithm which depends on it's k value(Neighbors) and finds it's applications in many industries like finance industry, healthcare industry etc.
- More specifically, the distance between the stored data and the new instance is calculated by means of some kind of a similarity measure. This similarity measure is typically expressed by a distance measure such as the Euclidean distance, cosine similarity or the Manhattan distance.
- In other words, the similarity to the data that was already in the system is calculated for any new data point that you input into the system. Then, you use this similarity value to perform predictive modeling.
- Predictive modeling is either classification, assigning a label or a class to the new instance, or regression, assigning a value to the new instance. Whether you classify or assign a value to the new instance depends of course on your how you compose your model with KNN.

Example from: https://www.r-bloggers.com/2015/03/machine-learning-in-r-for-beginners/

Example Dataset

Edgar Anderson's Iris Data

- From the *iris* manual page:
  - 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 Iris setosa, versicolor, and virginica.

Loading packages and the data

Install and load needed packages first.

```r
# Install libraries
install.packages('tidyverse')
install.packages('DT')
install.packages('ggvis')
install.packages('gridExtra')
install.packages("class")
install.packages('gmodels')


# Load libraries
library(tidyverse)
library(DT)
library(ggvis)
library(gridExtra)
library(class)
library(gmodels)
```

Load data.

```r
# Load data
data(iris)

iris
```

```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
10           4.9         3.1          1.5         0.1     setosa
11           5.4         3.7          1.5         0.2     setosa
12           4.8         3.4          1.6         0.2     setosa
13           4.8         3.0          1.4         0.1     setosa
14           4.3         3.0          1.1         0.1     setosa
15           5.8         4.0          1.2         0.2     setosa
```

Initial Overview of the Data Set

```
# Dataset information
datatable(iris)

glimpse(iris)

summary(iris)
```

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

Show 10 entries

Search:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	5.1	3.5	1.4	0
2	4.9	3	1.4	0
3	4.7	3.2	1.3	0
4	4.6	3.1	1.5	0
5	5	3.6	1.4	0
6	5.4	3.9	1.7	0
7	4.6	3.4	1.4	0

Showing 1 to 10 of 150 entries

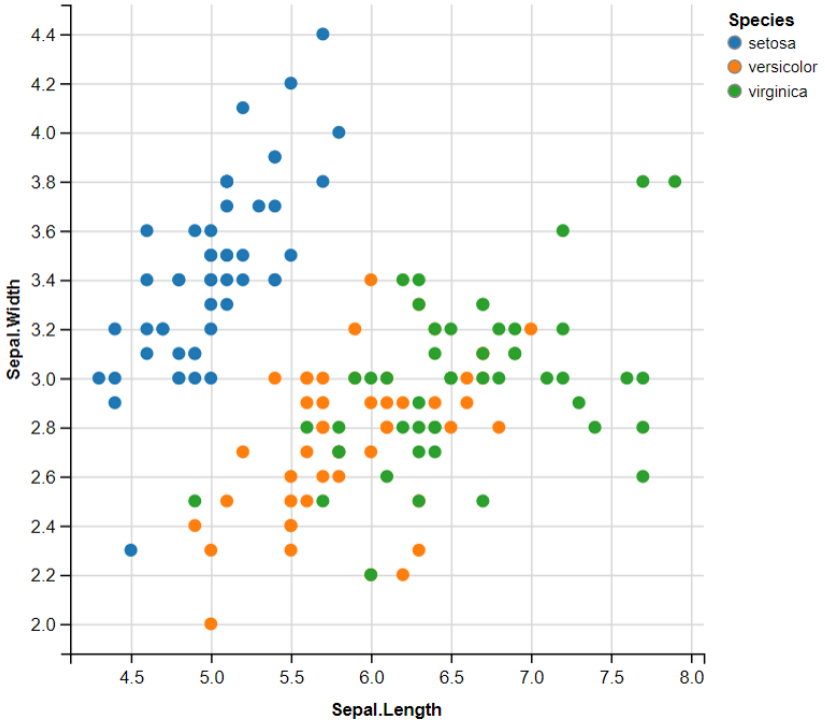
Previous 1 2 3 4 5 … 15 Next

```
> glimpse(iris)
Rows: 150
Columns: 5
$ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.8, 4.8, 4.3, 5.8, 5.7, 5.4, 5.1, 5.7, 5.1, 5.4, 5.1, …
$ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.4, 3.0, 3.0, 4.0, 4.4, 3.9, 3.5, 3.8, 3.8, 3.4, 3.7, …
$ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.6, 1.4, 1.1, 1.2, 1.5, 1.3, 1.4, 1.7, 1.5, 1.7, 1.5, …
$ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.2, 0.1, 0.1, 0.2, 0.4, 0.4, 0.3, 0.3, 0.3, 0.2, 0.4, …
$ Species      <fct> setosa, setosa, setosa, setosa, setosa, setosa, setosa, setosa, setosa, setosa, setosa, setosa, setosa, setosa…
```

```
# For more details, see
?iris
```

Initial Overview of the Data Set (cont.)

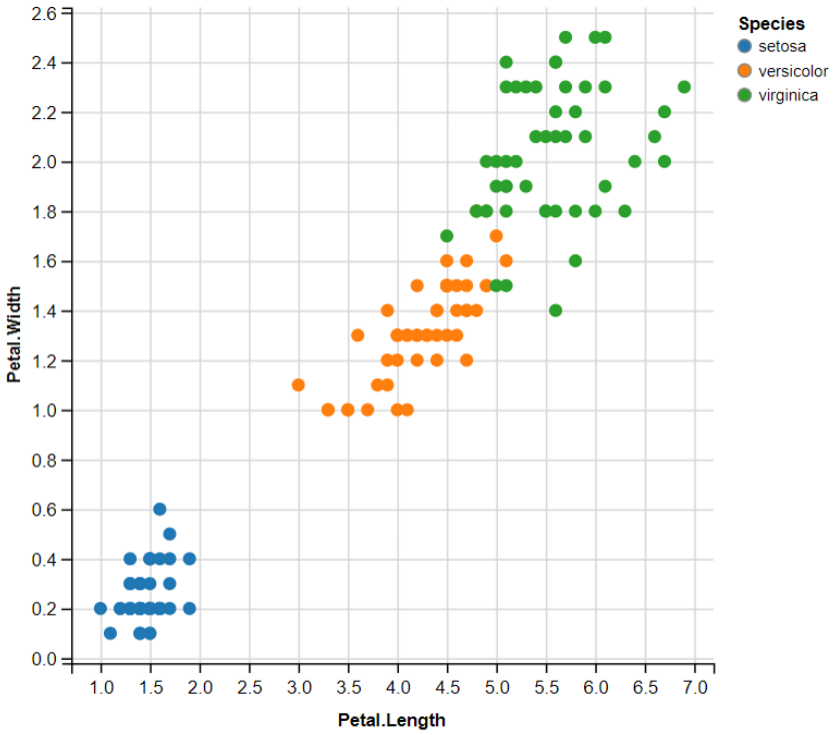
```
# Iris scatter plot (1)
iris %>% ggvis(~Sepal.Length, ~Sepal.Width, fill = ~Species) %>% layer_points()
```



You see that there is a high correlation between the sepal length and the sepal width of the Setosa iris flowers, while the correlation is somewhat less high for the Virginica and Versicolor flowers: the data points are more spread out over the graph and don't form a cluster like you can see in the case of the Setosa flowers.

Initial Overview of the Data Set (cont.)

```
# Iris scatter plot (2)
iris %>% ggvis(~Petal.Length, ~Petal.Width, fill = ~Species) %>% layer_points()
```



The scatter plot that maps the petal length and the petal width tells a similar story:
You see that this graph indicates a positive correlation between the petal length and the petal width for all different species that are included into the Iris data set.

Normalization

The Iris data set doesn't need to be normalized.

- As a part of your data preparation, you might need to normalize your data so that its consistent. For this introductory tutorial, just remember that normalization makes it easier for the KNN algorithm to learn.
- So when do we need to normalize a dataset?
  - In short: when we suspect that the data is not consistent.
  - We can easily see this when we go through the results of the *summary()* function. Look at the minimum and maximum values of all the (numerical) attributes. If we see that one attribute has a wide range of values, we will need to normalize a dataset, because this means that the distance will be dominated by this feature.
  - For example, if a dataset has just two attributes, X and Y, and X has values that range from 1 to 1000, while Y has values that only go from 1 to 100, then Y's influence on the distance function will usually be overpowered by X's influence. When you normalize, you actually adjust the range of all features, so that distances between variables with larger ranges will not be overemphasised.

For more: https://www.r-bloggers.com/2015/03/machine-learning-in-r-for-beginners/

Training and Test Sets

- We need to make sure that all three classes of species are present in the training model.
- What's more, the amount of instances of all three species needs to be more or less equal so that you do not favour one or the other class in your predictions.

In order to assess the model's performance later, we will need to divide the data set into two parts: a training set and a test set.

- the most common splitting choice is to take 2/3 of the original data set as the training set, while the 1/3 that remains will compose the test set.

```
# Training and test sets
set.seed(1234)
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.67, 0.33))
iris.training <- iris[ind==1, 1:4]
iris.test <- iris[ind==2, 1:4]
iris.trainLabels <- iris[ind==1, 5]
iris.testLabels <- iris[ind==2, 5]
```

- To make a training and test sets, we first set a seed. This is a number of R's random number generator.
- Then, we want to make sure that the Iris data set is shuffled and that we have the same ratio between species in the training and test sets. We use the *sample()* function to take a sample with a size that is set as the number of rows of the Iris data set, or 150. We sample with replacement: we choose from a vector of 2 elements and assign either 1 or 2 to the 150 rows of the Iris data set. The assignment of the elements is subject to probability weights of 0.67 and 0.33.
  - *Note* that the replace argument is set to TRUE: this means that we assign a 1 or a 2 to a certain row and then reset the vector of 2 to its original state. This means that, for the next rows in the data set, we can either assign a 1 or a 2, each time again. The probability of choosing a 1 or a 2 should not be proportional to the weights amongst the remaining items, so we specify probability weights.
- *Remember* that we want the training set to be 2/3 of the original data set: that is why we assign "1" with a probability of 0.67 and the "2"s with a probability of 0.33 to the 150 sample rows.

Training and Test Sets: Remember

- In addition to the 2/3 and 1/3 proportions specified above, we don't take into account all attributes to form the training and test sets.
- Specifically, you only take Sepal.Length, Sepal.Width, Petal.Length and Petal.Width.
- This is because **we actually want to predict the fifth attribute, Species: it is our target variable**.
- However, we do want to include it into the KNN algorithm, otherwise there will never be any prediction for it. We therefore need to store the class labels in factor vectors and divide them over the training and test sets.

```
> iris.training
   Sepal.Length Sepal.Width Petal.Length Petal.Width
1           5.1         3.5          1.4         0.2
2           4.9         3.0          1.4         0.2
3           4.7         3.2          1.3         0.2
4           4.6         3.1          1.5         0.2
6           5.4         3.9          1.7         0.4
```

```
> iris.test
   Sepal.Length Sepal.Width Petal.Length Petal.Width
5           5.0         3.6          1.4         0.2
11          5.4         3.7          1.5         0.2
14          4.3         3.0          1.1         0.1
16          5.7         4.4          1.5         0.4
26          5.0         3.0          1.6         0.2
```

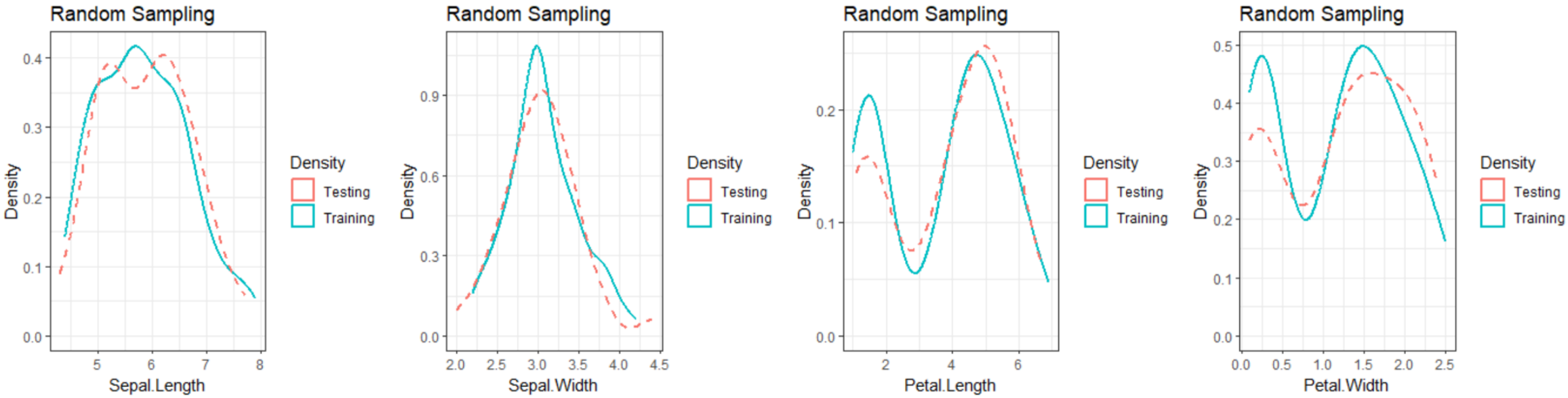
Training and Test Sets: sampling check

```
# Plots
# Sepal.Length
p1 = ggplot(iris.training, aes(x = Sepal.Length)) +
  geom_density(trim = TRUE,
               aes(color = "Training"), size = 1) +
  geom_density(data = iris.test, aes(x = Sepal.Length,
                                     color = "Testing"), trim = TRUE, size = 1, linetype = 2)
(p1 = p1 + theme_bw() + labs(color = "Density", title = "Random Sampling",
                             x = "Sepal.Length", y = "Density"))
# Sepal.Width
p2 = ggplot(iris.training, aes(x = Sepal.Width)) +
  geom_density(trim = TRUE,
               aes(color = "Training"), size = 1) +
  geom_density(data = iris.test, aes(x = Sepal.Width,
                                     color = "Testing"), trim = TRUE, size = 1, linetype = 2)
(p2 = p2 + theme_bw() + labs(color = "Density", title = "Random Sampling",
                             x = "Sepal.Width", y = "Density"))
# Petal.Length
p3 = ggplot(iris.training, aes(x = Petal.Length)) +
  geom_density(trim = TRUE,
               aes(color = "Training"), size = 1) +
  geom_density(data = iris.test, aes(x = Petal.Length,
                                     color = "Testing"), trim = TRUE, size = 1, linetype = 2)
(p3 = p3 + theme_bw() + labs(color = "Density", title = "Random Sampling",
                             x = "Petal.Length", y = "Density"))

# Petal.Width
p4 = ggplot(iris.training, aes(x = Petal.Width)) +
  geom_density(trim = TRUE,
               aes(color = "Training"), size = 1) +
  geom_density(data = iris.test, aes(x = Petal.Width,
                                     color = "Testing"), trim = TRUE, size = 1, linetype = 2)
(p4 = p4 + theme_bw() + labs(color = "Density", title = "Random Sampling",
                             x = "Petal.Width", y = "Density"))

# Combine all four plots
grid.arrange(p1, p2, p3, p4,  nrow = 1)
```

- We want to visually check the distribution of both sets.

Training and Test Sets: sampling check (cont.)

- Combine all four plots for Sepal.Length, Sepal.Width, Petal.Length and Petal.Width.
- Notice some differences between the lines due to sample size (better to have more data).



The Actual KNN Model

```
# The Actual KNN Model
# Building the Classifier
iris_pred <- knn(train = iris.training, test = iris.test, cl = iris.trainLabels, k=3)
iris_pred
```

- To build your classifier, you need to take the *knn()* function and simply add some arguments to it.
  - the k parameter is one that you set yourself. It is often an odd number to avoid ties in the voting scores.

```
> iris_pred
 [1] setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa
[12] setosa     versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
[23] versicolor versicolor virginica  virginica  virginica  virginica  versicolor virginica  virginica  virginica  virginica
[34] virginica  virginica  virginica  virginica  virginica  virginica  virginica
Levels: setosa versicolor virginica
```

- You store into *iris_pred* the *knn()* function that takes as arguments the training set, the test set, the train labels and the amount of neighbours you want to find with this algorithm. The result of this function is a factor vector with the predicted classes for each row of the test data.

The Model Evaluation

- We make a cross tabulation or a contingency table.
  - This type of table is often used to understand the relationship between two variables.
  - In this case, we want to understand how the classes of the test data, stored in *iris.testLabels* relate to the model that is stored in *iris_pred.*

```
# The Model Evaluation
CrossTable(x = iris.testLabels, y = iris_pred, prop.chisq=FALSE)
```

- From this table, we can derive the number of correct and incorrect predictions:
  - one instance from the testing set was labeled Versicolor by the model, while it was actually a flower of species Virginica.
  - We can see this in the first row of the “Virginica” species in the *iris.testLabels* column.
- In all other cases, correct predictions were made.
- **We can conclude that the model's performance is good enough and that we don't need to improve the model!**

```
> CrossTable(x = iris.testLabels, y = iris_pred, prop.chisq=FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  40


                | iris_pred
iris.testLabels |     setosa | versicolor |  virginica |  Row Total |
----------------|------------|------------|------------|------------|
         setosa |         12 |          0 |          0 |         12 |
                |      1.000 |      0.000 |      0.000 |      0.300 |
                |      1.000 |      0.000 |      0.000 |            |
                |      0.300 |      0.000 |      0.000 |            |
----------------|------------|------------|------------|------------|
     versicolor |          0 |         12 |          0 |         12 |
                |      0.000 |      1.000 |      0.000 |      0.300 |
                |      0.000 |      0.923 |      0.000 |            |
                |      0.000 |      0.300 |      0.000 |            |
----------------|------------|------------|------------|------------|
      virginica |          0 |          1 |         15 |         16 |
                |      0.000 |      0.062 |      0.938 |      0.400 |
                |      0.000 |      0.077 |      1.000 |            |
                |      0.000 |      0.025 |      0.375 |            |
----------------|------------|------------|------------|------------|
   Column Total |         12 |         13 |         15 |         40 |
                |      0.300 |      0.325 |      0.375 |            |
----------------|------------|------------|------------|------------|
```

The Model Evaluation (cont.)

- An alternative way – to get confusion matrix

```
# Confusiin Matrix
cm <- table(iris.testLabels, iris_pred)
cm
```

```
> cm
                iris_pred
iris.testLabels setosa versicolor virginica
     setosa         12          0         0
     versicolor      0         12         0
     virginica       0          1        15
```

- We see the same results as before:
  - one instance from the testing set was labeled Versicolor by the model, while it was actually a flower of species Virginica.
- In all other cases, correct predictions were made.
- **We can conclude that the model's performance is good!**

- The model achieved 97.5% accuracy with k is 1.

```
# Calculate out of Sample error
misClassError <- mean(iris_pred != iris.testLabels)
print(paste('Accuracy =', 1-misClassError))
```

```
> print(paste('Accuracy =', 1-misClassError))
[1] "Accuracy = 0.975"
```

6. In-class Assignment