Machine Learning Introduction

Tourism Facing Digital Transition

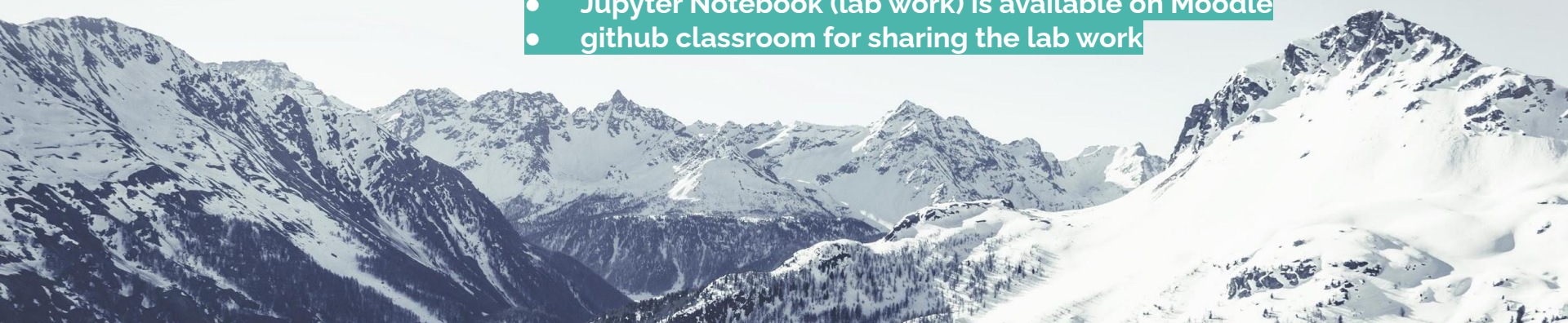
Emanuela Boros
University of La Rochelle, France

8-12 February 2021

Organization

1. 1.5h Course: Introduction to Machine Learning
2. 1.5h Lab work: Tourist Review Classification

- Course (slides and video) are available on Moodle
- Jupyter Notebook (lab work) is available on Moodle
- github classroom for sharing the lab work



Tourism Facing Digital Transition - Introduction

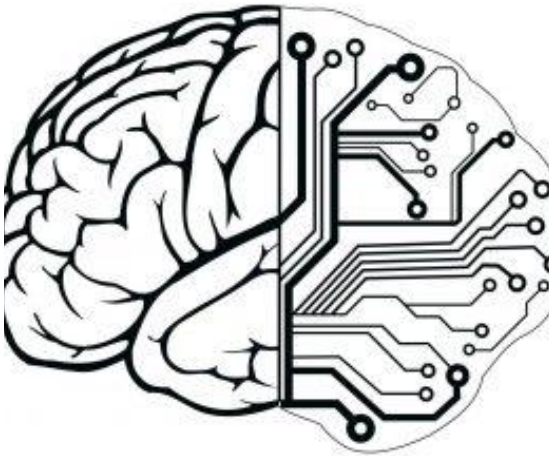
01

- **Tourism research** has entered the era of **big data**
- Academia and industry are now better positioned to understand and explore **tourist behavior** and the **tourism market**
- **blogs and social media posts** to online **travel review sites**
- Online travel review sites, such as **TripAdvisor**, generate large amounts of text-based **online travel review data**
- Ratings cannot provide any information about the specific **product characteristics** that visitors like or dislike, and such information is **typically included in the review text**
- In addition, many **users are overwhelmed by the enormous amount of review information** provided on travel online review sites
- **Large amounts of data** →
- It is necessary to establish an **effective mechanism** to help users identify the **main content and emotions** embedded in the review text

Text Classification: Introduction

01

- **Human brains** are wired to **recognize patterns** and classify objects for learning and making decisions
- .. they are not able to treat every object as **unique**
- .. we don't have a **lot of memory resources** to be able to process the world around us
- → our brains develop "concepts" or mental representations of "categories of objects"



- **Classification** is **fundamental** in language, prediction, inference, decision making and all kinds of environmental interactions
- **Language**: for example, how the meaning of words in a sentence can be contextualized by earlier words or concepts

QUICK COMMENT:

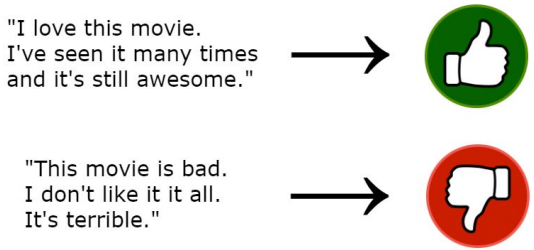
AI can be sexist and/or racist
Racist data? It's the human bias that is Infecting the AI development

Text Classification: Introduction

02

- The **classification of objects** consists of assigning a class to an object.
- These objects can be of the type:
  text image audio video

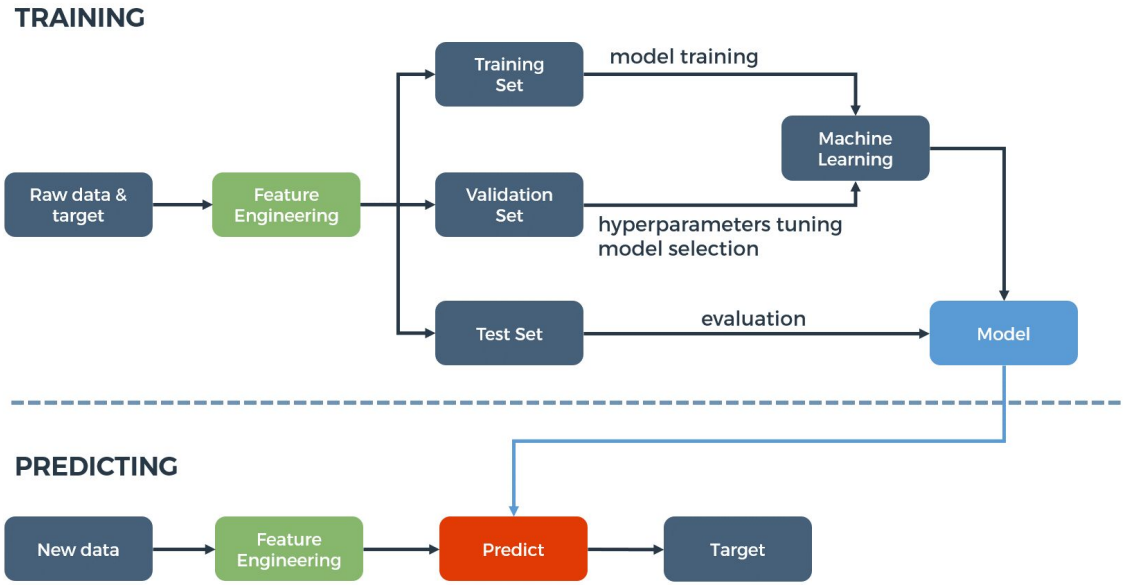
- We do classification all the time:
  - We can **recognize the way back home** from university
  - We can **recognize a cat that is black** even if we have only seen white and orange cats before
  - We can recognize when someone is **ironic or not**
  - We can recognize our **friends voices**
  - We can even distinguish between a **Chihuahua** and a **muffin**



Text Classification: Introduction

03

1. Recovery of the data (* .csv)
2. Text pre-processing (tokenization)
3. Text representations (bag of words, TF-IDF)
4. Data splitting (train, test)
5. Machine learning methods
6. Error analysis (evaluation, etc.)



How to classify hotel reviews:

Stayed here with husband and sons on the way to an Alaska Cruise. We all loved the hotel, great experience. Room service dinners were delicious! Heavenly beds were heavenly, too!

The Marriott hotel itself fell below the expectations. Housekeeping was just mediocre. The carpet in hallways very dirty. Rooms seems to be refinished, but very miserable in my opinion. I will not stay here again.

https://www.kaggle.com/harmanpreet93/hotelreviews
Hotel Reviews Dataset

How to classify hotel reviews:

Stayed here with husband and sons on the way to an Alaska Cruise. We all loved the hotel, great experience. Room service dinners were delicious! Heavenly beds were heavenly, too!

The Marriott hotel itself fell below the expectations. Housekeeping was just mediocre. The carpet in hallways very dirty. Rooms seems to be refinished, but very miserable in my opinion. I will not stay here again.

Happy

loved great delicious heavenly

Not Happy

mediocre dirty miserable

https://www.kaggle.com/harmanpreet93/hotelreviews
Hotel Reviews Dataset

How to classify hotel reviews:

Stayed here with husband and sons on the way to an Alaska Cruise. We all loved the hotel, great experience. Room service dinners were delicious! Heavenly beds were heavenly, too!

The Marriott hotel itself fell below the expectations. Housekeeping was just mediocre. The carpet in hallways very dirty. Rooms seems to be refinished, but very miserable in my opinion. I will not stay here again.

Check-in was **smooth** and **fast**, and the staff were **nice**. But there were some serious **flaws**. The room was **dirty** and had **great** noise. The smell of **smoke** was extremely strong.

Happy

loved great delicious heavenly

Not Happy

mediocre dirty miserable

https://www.kaggle.com/harmanpreet93/hotelreviews
Hotel Reviews Dataset

How to classify hotel reviews:

! We need to first install python 3.x and then run: jupyter notebook !

Let’s define some data first:

1. Load the data: pandas
2. Analyze the data: nltk & pandas
3. Represent the data in numerical form: scikit-learn

https://www.kaggle.com/harmanpreet93/hotelreviews
Hotel Reviews Dataset

Data representation: load the data

1. Load the data using **pandas** library

In [2]:
```
from pandas import read_csv

DATA_FILE = 'data/hotel_reviews.csv' # Read the data file of type .csv

data = read_csv(DATA_FILE)
data.head() # Show the first 5 entries in the dataset
```

Out[2]:

	review	class
0	The room was kind of clean but had a VERY stro...	not happy
1	I stayed at the Crown Plaza April -- - April -...	not happy
2	I booked this hotel through Hotwire at the low...	not happy
3	Stayed here with husband and sons on the way t...	happy
4	My girlfriends and I stayed here to celebrate ...	not happy

https://www.kaggle.com/harmanpreet93/hotelreviews
Hotel Reviews Dataset

Data representation: analyze the data

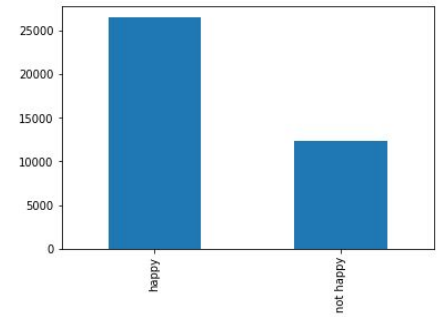
2. Let's look at a random example from our data

```
In [12]: data.loc[3]['review']
Out[12]: 'Stayed here with husband and sons on the way to an Alaska Cruise. We all loved the hotel, great experience. Ask f
         or a room on the North tower, facing north west for the best views. We had a high floor, with a stunning view of t
         he needle, the city, and even the cruise ships! We ordered room service for dinner so we could enjoy the perfect v
         iews. Room service dinners were delicious, too! You are in a perfect spot to walk everywhere, so enjoy the city. A
         lmost forgot- Heavenly beds were heavenly, too!'

In [13]: data.loc[3]['class']
Out[13]: 'happy'
```

3. How many **happy** and **not happy** reviews we have?

```
In [35]: _ = data['class'].value_counts().plot(kind='bar')
```



Data representation: transform the data

The representation of textual is important because it allows us not only to analyze this type of information, but also to transform the texts in **numerical data** for the machine learning algorithms!

- On the one hand, **some types of representations are very simple and quick to calculate**, on the other hand they do not contain much information about these words.
- On the other hand, the **more rich representations** are slower to calculate but they contain several characteristics on the words.

We will focus on the most famous representations:

1. **One-hot encoding**
2. **Bag of words**
3. **TF-IDF** (term frequency–inverse document frequency)
4. *Word embeddings*

Data representation: one-hot encoding

The **one hot encoding** representation is one of the simplest representations because all the words are independent from each other. The size of the representation increases with the number of the total words in a dataset.

“The hotel was OK. The room was clean.”

→ vocabulary = 6 unique words

the	hotel	was	OK	room	clean
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

Imagine that we have a vocabulary of 50,000. (There are about a million words in English.)

Each word is represented by 49,999 zeros and a single 1.

→ we need 50.000^2 = 2.5 billion units of memory space.
Not efficient in terms of calculation.

Data representation: bag-of-words

Differently from the **one hot encoding** representation, the **bag-of-words** is a representation of text that describes the occurrence of words within a document. It is called a "*bag*" of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

1. *"The hotel was OK, very very clean."*
2. *"The room was clean."*

→2 documents, vocabulary = 7 unique words

	the	hotel	was	OK	very	clean	room
1	1	1	1	1	2	1	0
2	1	0	1	0	0	1	1

Data representation: TF-IDF

The **TF-IDF** (Term Frequency-Inverse Document Frequency) statistical measure makes it possible to evaluate the importance of a term contained in a document, relative to a collection or a corpus. The weight increases in proportion to the number of occurrences of the word in the document. It also varies according to the frequency of the word in the corpus.

TF = (#Number of repetitions of a word in a document) / (#Number of words in a document)
IDF =Log[(#Number of documents) / (#Number of documents containing the word)]

1. *"The hotel was OK, very very clean."*→vocabulary = 7 total words

"the" → TF = 1/7 = 0,14	"the" → IDF = log(2/2) = 0	"the" → TF*IDF = 0
"hotel" → TF = 1/7 = 0,14	**"hotel" → IDF = log(2/1) = 0,30**	**"hotel" → TF*IDF = 0,14*0.30 = 0.042**
"was" → TF = 1/7 = 0,14	"was" → IDF = log(2/2) = 0	"was" → TF*IDF = 0
"OK" → TF = 1/7 = 0,14	**"OK" → IDF = log(2/1) = 0,30**	**"OK" → TF*IDF = 0,14*0.30 = 0.042**
"very" → TF = 2/7 = 0.28	**"very" → IDF = log(2/1) = 0,30**	**"very" → TF*IDF = 0,28*0,30 = 0,084**
"clean" → TF = 1/7 = 0.14	"clean" → IDF = log(2/2) = 0	"clean" → TF*IDF = 0

2. *"The room was clean."* →2 documents, vocabulary = 4 total words
   "the" → TF = 1/4 = 0,25
   "hotel" → TF = 1/4 = 0,25
   "was" → TF = 1/4 = 0,25
   "clean" → TF = 1/4 = 0,25 …..

TFIDF

For a term i in document j:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Data representation: TF-IDF

4. Let's represent the data with TF-IDF

In [56]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(min_df=.0025, max_df=.1, stop_words='english', ngram_range=(1,2))

tfidf_vectorizer_weights = tfidf_vectorizer.fit_transform(data['preprocessed_review_text'])
weights = np.asarray(tfidf_vectorizer_weights.mean(axis=0)).ravel().tolist()

weights_df = pd.DataFrame({'term': tfidf_vectorizer.get_feature_names(), 'weight': weights})
weights_df.sort_values(by='weight', ascending=False).head(20)
```

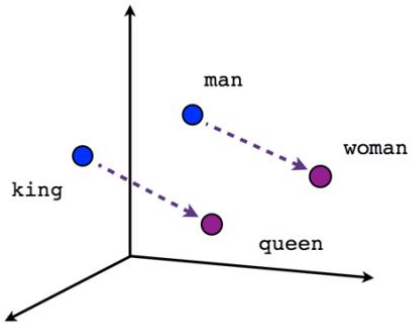
Out[56]:

	term	weight
2545	old	0.010244
1039	downtown	0.010121
4202	wonderful	0.009881
1197	experience	0.009845
2669	perfect	0.009774
3562	staff friendly	0.009598
2883	quiet	0.009441
2624	park	0.009441
494	car	0.009437
3394	shower	0.009366
1535	great location	0.009348
2496	noise	0.009251
2415	need	0.009188
3254	san	0.008994
2137	located	0.008984

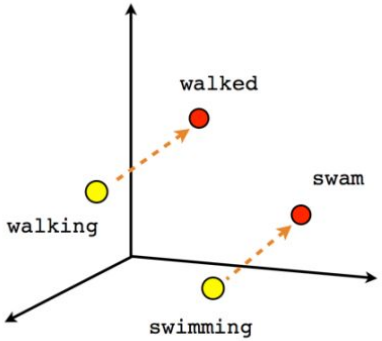
Data representation: More..

Word embeddings are a type of word representation that allows words with **similar meaning** to have a **similar representation**. Words or phrases from the vocabulary are mapped to vectors of **real numbers**.

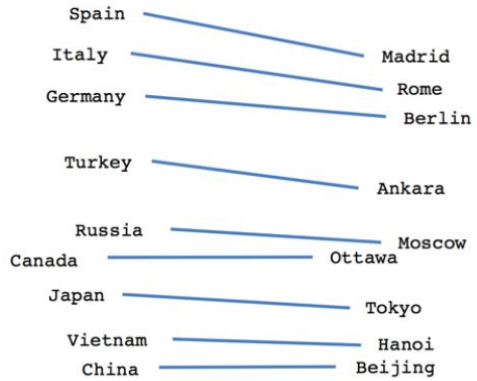
Conceptually it involves a mathematical embedding from space with many dimensions per word to a continuous vector space with a much lower dimension. The representations are usually generated by neural networks.



Male-Female



Verb tense



Country-Capital

Split the data.. For the machine learning algorithms

Before getting to the next step, let's see about data splitting.

The **train-test split** of the data procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

In [59]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data['preprocessed_review_text'], data['class'],
                                                    test_size=0.10, random_state=42)

print(len(X_train), len(X_test))
```

```
35038 3894
```

Never train on test data. If you are seeing surprisingly good results on your evaluation metrics, it might be a sign that you are accidentally training on the test set. For example, high accuracy might indicate that test data has leaked into the training set.

Machine learning: Naive Bayes

The **Naive Bayes** classification is a type of simple probabilistic Bayesian classification based on Bayes theorem with a strong independence of assumptions.

A **Naive Bayes** classifier assumes that the existence of a characteristic for a class is independent of the existence of other characteristics.

A fruit can be considered an apple if it is **red**, **round**, and **about ten centimeters**. Even if these characteristics are related in reality, **Naive Bayes** will determine that a random fruit is an apple by considering independently these characteristics of color, shape and size.

Likelihood

Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

Posterior Probability

Predictor Prior Probability

$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Machine learning: Naive Bayes

6 Happy reviews

"***great***" - 6 times, "***nice***" - 2 times, "***helpful***" - 3 times, "***dirty***" - 0 times, → 11 words

P("***great***"|Happy) = 6/11 = 0,54
P("***nice***"|Happy) = 2/11 = 0,18
P("***helpful***"|Happy) = 3/11 = 0,27
P("***dirty***"|Happy) = 0/11 = 0

3 Not Happy reviews

"***great***" - 1 time, "***nice***" - 1 time, "***helpful***" - 0 times, "***dirty***" - 2 times, → 4 words

P("***great***"|Not Happy) = ¼ = 0,25
P("***nice***"|Not Happy) = ¼ = 0,25
P("***helpful***"|Not Happy) = 0
P("***dirty***"|Not Happy) = 2/4 = 0,5

Machine learning: Naive Bayes

6 Happy reviews and 3 Not Happy reviews

P("great"|Happy) = 0,54
P("nice"|Happy) = 0,18
P("helpful"|Happy) = 0,27
P("dirty"|Happy) = 0

P("great"|Not Happy) = 0,25
P("nice"|Not Happy) = 0,25
P("helpful"|Not Happy) = 0
P("dirty"|Not Happy) = 0,5

Check-in was smooth and fast, and the staff were **nice**. But there were some serious flaws. The room was **dirty** and had **great** noise. The smell of smoke was extremely strong.

P(Happy) = 6/9 = 0,66
P(Not Happy) = 3/9 = 0,33

P(Happy) x P("nice"|Happy) x P("great"|Happy) x P("dirty"|Happy) =
= 0,66 x 0,18 x 0,54 x 0 = **0**

P(Not Happy) x P("nice"|Not Happy) x P("great"|Not Happy) x P("dirty"|Not Happy) =
= 0,33 x 0,25 x 0,25 x 0,5 = **0,01**

Not Happy

Machine learning: performance evaluation

Accuracy: in a classification problem, **accuracy** is the number of correct predictions made divided by the total number of predictions made, multiplied by 100 to turn it into a percentage.

Accuracy = Correct Predictions / Total Predictions * 100

A **true positive** is an outcome where the model correctly predicts the correct class. Similarly, a **true negative** is an outcome where the model correctly predicts the negative class.

A **false positive** is an outcome where the model incorrectly predicts the correct class. And a **false negative** is an outcome where the model incorrectly predicts the negative class.

Precision is the number of True Positives divided by the number of True Positives and False Positives.

Precision = True Positives / True Positives + False Positives

Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives.

Recall = True Positives / True Positives + False Negatives

The **F1 Score** is as follows:

2 * ((Precision * Recall) / (Precision + Recall))

The F1 score gives us the balance between our precision and recall values.

Machine learning: Naive Bayes

The scikit-learn library has an easy pipeline:

model.fit(train data) # Train

model.predict(test data) # Test

4. Let's train the Naive Bayes model with data represented with TF-IDF

```
In [37]: from sklearn.feature_extraction.text import TfidfVectorizer

         vectorizer = TfidfVectorizer()
         vectorizer.fit(X_train)

         X_train_vectorized = vectorizer.transform(X_train)
         X_test_vectorized = vectorizer.transform(X_test)

         X_train_vectorized.toarray()[0]

Out[37]: array([0., 0., 0., ..., 0., 0., 0.])
```

→ Represent the train and test set with TF-IDF

```
In [43]: from sklearn.naive_bayes import MultinomialNB
         clf = MultinomialNB()
         _ = clf.fit(X_train_vectorized, y_train)
```

→ Train a Naive Bayes model

```
In [44]: y_predicted = clf.predict(X_test_vectorized)
```

→ Predict the classes for the test set

```
In [45]: from sklearn.metrics import classification_report

         print(classification_report(y_test, y_predicted))
```

```
              precision    recall  f1-score   support

       happy       0.80      0.99      0.88      2649
   not happy       0.95      0.47      0.63      1245

    accuracy                           0.82      3894
   macro avg       0.88      0.73      0.76      3894
weighted avg       0.85      0.82      0.80      3894
```

→ Performance evaluation: accuracy = 82%

Machine learning: Naive Bayes

5. Let's take a review, represent it with TF-IDF and see what our model predicts

```
In [122]: review_vectorized = vectorizer.transform(["Check-in was smooth and fast, and the staff were nice." +
                                                    "But there were some serious flaws. The room was dirty " +
                                                    "and had great noise. The smell of smoke was extremely strong."])

In [123]: review_prediction = clf.predict(review_vectorized)

          review_prediction[0]

Out[123]: 'not happy'
```

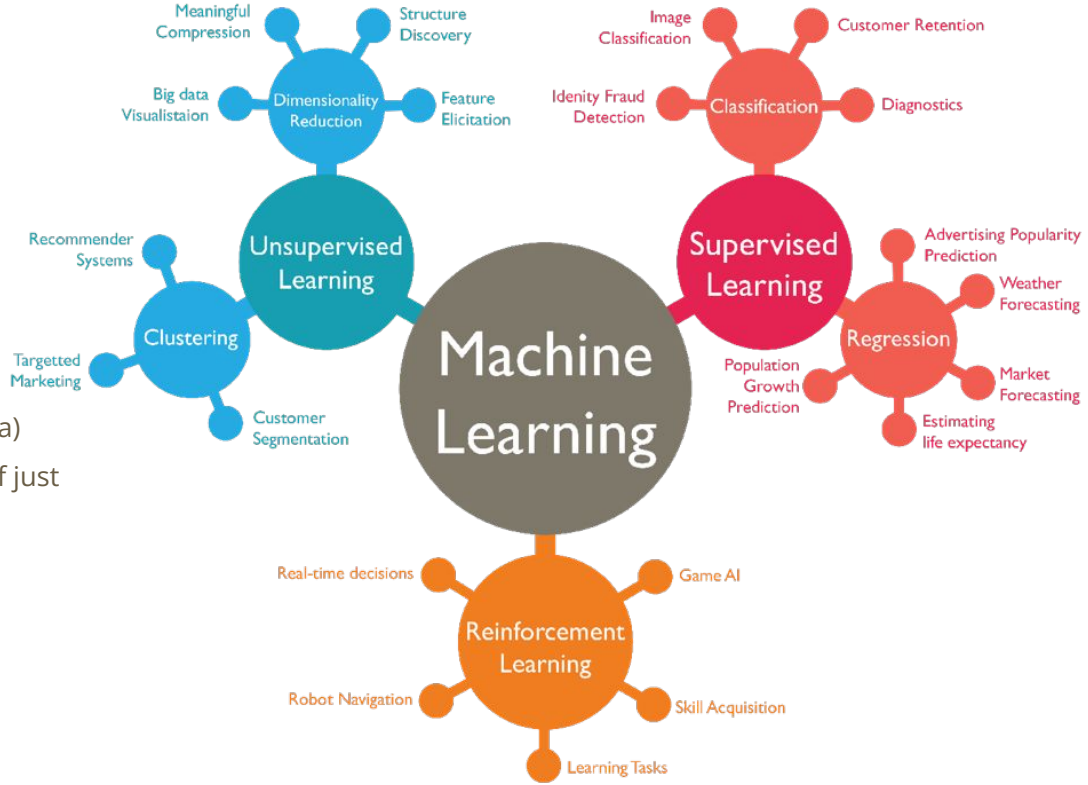
Check-in was **smooth** and **fast**, and the staff were **nice**. But there were some serious **flaws**. The room was **dirty** and had **great** noise. The smell of **smoke** was extremely strong.

Not Happy

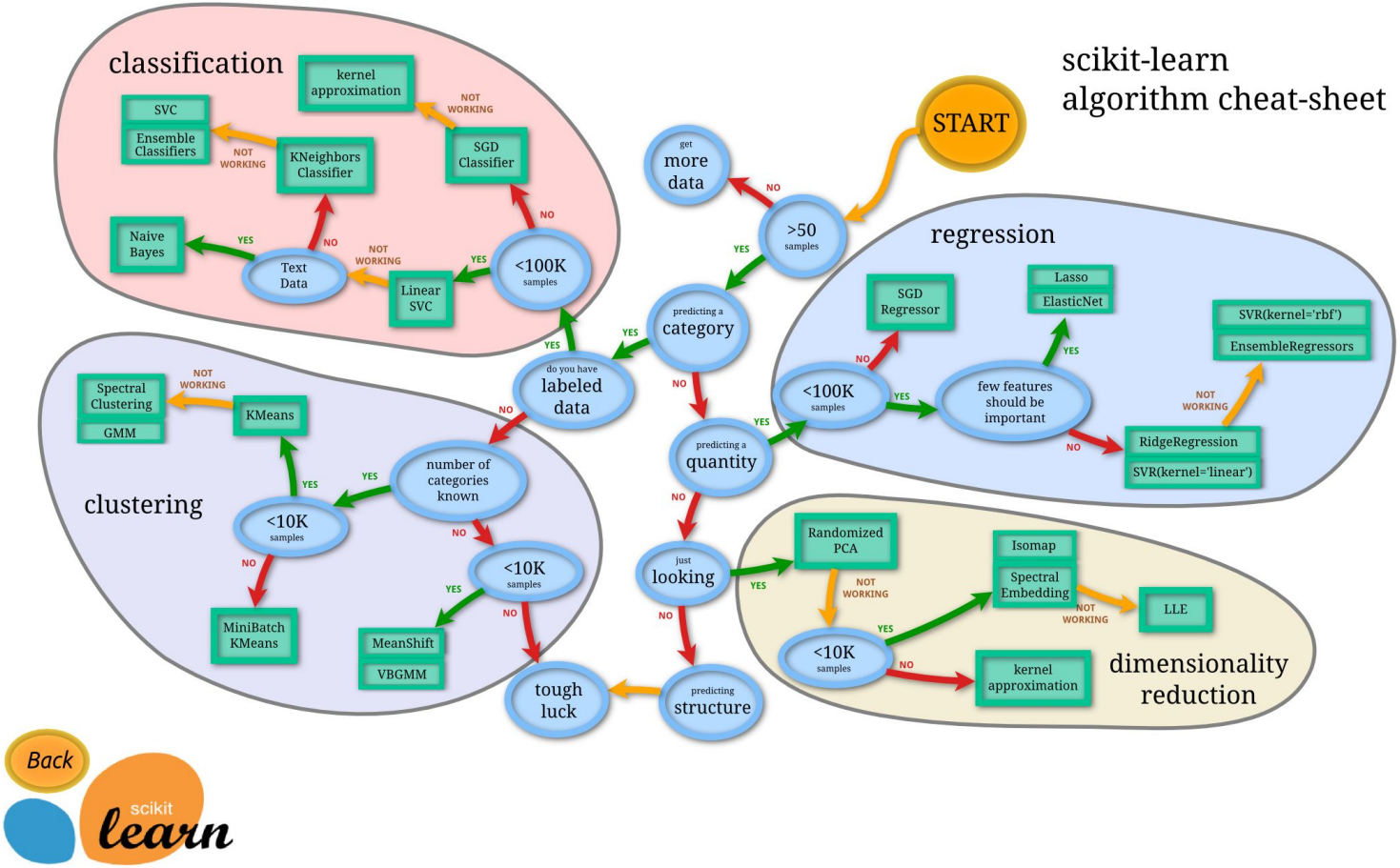
More..

Machine/Deep learning types of algorithms

- **Supervised** learning (labeled data)
- **Unsupervised** learning (unlabeled data)
- **Semi-supervised** learning (labeled+unlabeled data)
- **Reinforcement** learning (rules, a good example of just how broad the overall “learning” approach can be)



Scikit-learn



Links

https://www.kaggle.com/competitions

Machine learning Coursera famous courses, Andrew Ng, https://www.coursera.org/learn/machine-learning

Machine learning Coursera (on youtube), Andrew Ng, https://www.youtube.com/watch?v=PPLop4L2eGk

The most famous book on deep learning: https://www.deeplearningbook.org/ (Ian Goodfellow, Yoshua Bengio and Aaron Courville)

ML and DL People

Andrew Ng, Founder and CEO of Landing AI, Founder of deeplearning.ai.
Fei-Fei Li, Professor of Computer Science at Stanford University.
Andrej Karpathy, Senior Director of Artificial Intelligence at Tesla.
Demis Hassabis, Founder and CEO of DeepMind.
Ian Goodfellow, Director of Machine Learning at Apple.
Yann LeCun, Vice President and Chief AI Scientist at Facebook.
Jeremy P. Howard, Founding Researcher at fast.ai, Distinguished Research Scientist at the University of San Francisco.
Ruslan Salakhutdinov, Associate Professor at Carnegie Mellon University, Director of AI Research at Apple.
Geoffrey Hinton, Professor of Computer Science at the University of Toronto, VP and Engineering Fellow at Google
Rana el Kaliouby, CEO and Co-Founder of Affectiva.
Daphne Koller, Founder and CEO of insitro, Co-Founder of Coursera, Adjunct Professor of Computer Science and Pathology at Stanford.
Alex Smola, Director, Amazon Web Services.