

Studentnumber: **500917714**

Studentname: **Bonnaud Laura**



FINAL PORTFOLIO

"Whatever proofs your progress on your individual learning journey ..."

21/06/2023

TABLE OF CONTENT

I. Introduction

II. General concepts

- a. Machine Learning
- b. Coding

III. Machine learning

- a. What is machine learning?
- b. Type of machine learning
- c. Supervised learning
- d. Unsupervised learning
- e. Choosing an algorithm

IV. Data Scientist and Engineer

- a. Data collecting
- b. Data cleaning
- c. Data storage
- d. ML Model building
- e. Visualization
- f. Operations

V. Appreciation

VI. Sources

I. INTRODUCTION

Why did I choose the Big Data semester?

At ESIEA, I want to choose the **Artificial Intelligence and Data Science major**. We can find data being used in almost every industry; be it Healthcare, Finance, or Technology. And AI is present in all sectors, from health to security to education.

First of all, my choice turned by analyzing which areas I prefer the most among those that we have learned since the 1st year. Being very interested in the fields of **statistics, algorithms, computer science** and **physics**, I decided to come for my semester abroad at **HVA** for the **Big Data semester**. In addition, for my next two years at ESIEA, I wish to follow the **apprenticeship course** in a company in a position in the field of big data and artificial intelligence, for reasons of business experience, remuneration to become more financially independent from my parents, and to have training in Big Data.

I currently have an internship contract for 2 years at Nokia. The purpose of this work will be the design of a machine learning model on software codes.

This is why this semester in Big Data is important to be able to be effective in my future job because I will already have some basics in the field and to already have the basics in my major for the next two years.

What do I want to learn during this semester?

During this semester, I want to learn the basics of Big Data. I want to acquire knowledge about the different types of machine learning algorithms and, in several cases, know how to implement them. For example, having a foundation in supervised learning, unsupervised learning, neural networks and machine learning in the real world.

But I particularly want to learn how to implement a Learning machine for software code reviews, because this is the area in which I may have an opportunity for work-study work.

I want to have a solid foundation of data analysis and be able to use Python for the whole process and acquire basic statistical knowledge necessary for data analysis.

II. GENERAL CONCEPT

a. MACHINE LEARNING

	Insufficient	Marginal	Good	Excellent
Machine Learning	The student has no idea about different models to be used for data science	The student can explain the role of supervised and unsupervised machine learning during a ML implementation. Moreover the student can explain the differences between regression and classification.	Additional to Marginal: The student knows the pro's and cons of at least 3 ML algorithms	Additional to Good: Student has done research on ML algorithms and can use arguments for using a particular one beyond the mandatory literature

*In this part, in my opinion I have accomplished the level **Good**.*

- ❖ I can explain the basics of [machine learning](#) and the different [types of machine learning](#).
- ❖ At first, I particularly learned [supervised machine learning](#) models. I can explain what this type of machine learning is, why it is used. I researched the different [models of supervised learning and their pros and cons](#), especially as part of the onboarding assignment 1.
- ❖ Then, I focused on [unsupervised learning](#): after studying the subject of my project, I learned about the best type of machine learning to use. I particularly learned [what clustering is](#), which will be the basis of our problem, and I listed in a table the [models of unsupervised learning and their pros and cons](#) that we can use.
- ❖ For any machine learning problem, I have studied 4 steps that allow you to [select the most suitable machine learning model](#).
- ❖ Once the model has been created, it is possible to evaluate the model. For this, many indicators can be used, the majority depending on the task chosen. I can explain the [5 main performance measures](#), which will be useful to me in projects.

b. CODING

	Insufficient	Marginal	Good	Excellent
Coding	The student has done no coding beyond the onboarding assignment.	The student has done some coding beyond the onboarding assignment and can explain some basic lines of coding.	Additional to Marginal: The student has done coding for one of the following aspects : Data import/ cleaning /storage , ML algorithm , Visualization with Python and can explain all produced lines of code	Additional to Good: The student has done coding for at least 2 of the following aspects Data import/ cleaning / storage, ML algorithm, Visualization with Python and only map, filter and reduce functions are used on datasets.

*In this part, in my opinion I have accomplished the level **Good**.*

Data import:

Beyond the onboarding assignment:

- ❖ I have done some training of importing dataset from SVC file by using the Pandas **read_csv** function [fig1] with for example [the Iris Flowers dataset](#).
- ❖ I learned how to **examine** the dataset with different method, it's important to understand what we're working on, like **head()** [fig2], **describe()** [fig3], **groupby()** [fig3] functions.

Data cleaning:

[1] Beyond the courses taught in class, I have learned [two NLP techniques](#) to reduce words to their base or root form: **stemming** and **lemmatization**.

[2] Beyond the courses taught in class, I learned how to use a [lambda function and the filter\(\) function](#) to remove stopwords , which is more memory-efficient. It also learn how to use a **map() function** with a lambda function to apply both stemming and lemmatization [fig10] to each word in the list of words. This replaces the separate loops that apply stemming and lemmatization, which makes the code more concise and easier to read.

[3] As part of my project, I created a [python script who performs text cleaning on CSV files found in a specified directory](#):

- ❖ A '[clean text](#)' function to clean a given text string.
- ❖ A '[clean csv files](#)' function who iterates through all files in the input directory. I learn how to work with the 'os' library.

Data storage:

[1] Beyond the courses taught in class, I learned the theoretical part of data storage:

- ❖ what is [the importance to back up the original data to a storage system](#)
- ❖ how to [choose between SQL and NoSQL](#)
- ❖ how to [use MySQL database with Python](#)
- ❖ what is [SQLAlchemy](#) and how to use it
- ❖ what is [SQLite](#) and how to use it

[2] As part of my project, I use my knowledge [for the project](#) and did some other research, I learned:

- ❖ [what is phpMyAdmin and how to use it](#): configuration of a file named "config.json" that contains the necessary information to connect to a MySQL database, creation of a "config.py" file to allow us to access the database configuration parameters stored in the JSON file, creation of a "sql.py" file to define a class called MySQL who contains methods to establish a connection to the MySQL database and to execute SQL queries and retrieve results from the database
- ❖ [how to create an instance of the MySQL class and connect it](#) to establish a connection to the database
- ❖ I started coding a [script with some SQL queries](#) knowing the requirements of the product owner. For now, I filled in the table some false data because I need to wait to get the data from my teammate, but he is still working on.
- ❖ After my teammate gave me all the new CSV files, I create a [script 'storeCleanDataInDB'](#) who essentially reads CSV files from a directory, extracts data from the files, and inserts the data into a MySQL database table.
- ❖ Then, I adapt the previous code of the SQL Queries to use it in the dashboard. I create a [script 'fct_queries'](#) who interact with the MySQL database and performs queries to retrieve data from a table

ML Model building:

For the onboarding assignment and the project, I did **a lot of research** and built several machine learning models.

Beyond the courses taught in class:

- ❖ I built machine learning models on the Iris Flower dataset : **Logistic regression**, **k-nearest neighbors** and **support vector machine** [\[fig4\]](#) and I made **prediction** on the test dataset [\[fig5\]](#) | [fig6](#).
- ❖ I know [how to improve the accuracy of the models](#) : I have used **feature scaling** and **hyperparameter tuning** for all the algorithms.
- ❖ I learned [what is K-fold cross-validation](#) and how to use it.
- ❖ I search [why it is important to use pipeline](#) and [what means scaling the data](#).
- ❖ I can discuss about the results of a model : for example for the [results of KNN model](#).

Visualization:

I learned how to do:

- ❖ a bar chart [\[fig7\]](#)
- ❖ an interactive visualization of scatter plots with a dropdown menu [\[fig8\]](#)
- ❖ an interactive visualization to help us to understand the relationship between some variables with an interactive click on a data point to show more info [\[fig9\]](#).
- ❖ a scatter matrix plot using the Plotly Express library, which shows the pairwise relationships between the four features (sepal length, sepal width, petal length, and petal width) and the class labels. [\[fig10\]](#)

Operations:

- ❖ I can explain the role of [git repositories](#) and [python environments](#) and can use of these tools for team collaboration effectively.
- ❖ I did some research and did a summary of what I learned:
 - [\[1\] Theoretical part of Docker](#)
 - [\[2\] Docker's basic concepts](#)

III. MACHINE LEARNING

a. What is Machine Learning?

Machine learning is the process whereby computers learn to make decisions from data without being explicitly programmed. In essence, Machine Learning involves using computer programming to improve performance by analyzing example data or past experience. So it is a model is created with defined parameters, and the learning process involves executing a computer program to optimize those parameters using training data or past experience. Fundamentally, Machine Learning involves extracting knowledge from data by analyzing it, and then fine-tuning the parameters of the model to continually improve performance in a given task. Put simply, it's about using data to gain knowledge, and using that knowledge to optimize performance.

b. Type of machine learning:

There are many different types of machine learning systems:

- ❖ Depending on whether the learning takes place under human supervision (**supervised, unsupervised, semi-supervised** or **reinforced** learning)
- ❖ Depending on whether learning takes place gradually or not, gradually (**e-learning** or **group learning**)
- ❖ Depending on whether it is content to compare the new data with data known, or that It detects, on the contrary, structuring elements in the training data and builds a **predictive** model like a scientific (learning from observations or learning from a model)

c. Supervised learning:

Supervised Learning is inferring a function from labeled training data. For Supervised Learning, we have both inputs, and we're using those to predict the outputs. This model was concerned with predicting the values of one or more outputs, or response variables for a given set of input or predicted variables. So the predictions were based on the training sample of previously solved cases, and the joint values of all the variables are known.

Predicting the values of one or more outputs for a given set of inputs → $\Pr(X, Y) = \Pr(Y|X) \cdot \Pr(X)$

There is a distinction of Supervised Learning between the outputs, we have qualitative outputs and quantitative outputs, and this distinction has led to a naming convention for the prediction task:

- ❖ **Regression:** when we predict quantitative outputs: continuous values
- ❖ **Classification:** when we predict qualitative outputs: the label, or category, of an observation

ML algorithms of supervised learning:

For the onboarding assignment, we had to make machine learning models for the sentiment analysis problem, we can study the different classification models with their pros and cons in order to find the most relevant to the problem.

Model	Description	Pros	Cons
Random Forests	type of ensemble learning method where multiple decision trees are trained on subsets of the training data, and their predictions are combined to make a final prediction.	<ul style="list-style-type: none">❖ Capable of handling non-linear relationships between features and target variables❖ Can handle interactions between features❖ Can reduce overfitting and improve generalization performance	<ul style="list-style-type: none">❖ Can be computationally expensive, especially with large datasets and many features
K-Nearest Neighbors	simple and non-parametric machine learning algorithm the set of training data constitutes the model the class of an unknown data point is determined by the class of its k nearest neighbors in the training set.	<ul style="list-style-type: none">❖ Simple and easy to implement❖ Can handle non-linear relationships between features and target variables	<ul style="list-style-type: none">❖ Sensitive to the scaling of the data and may require normalization of the input data❖ Can be computationally expensive❖ May not work well with high-dimensional data
Logistic Regression	linear model that is commonly used for binary classification problems. The goal is to create a linear function of the variables that assigns positive numbers to the data of the positive class and negative values to the negative class.	<ul style="list-style-type: none">❖ Easy to implement❖ Linear model that can be easily trained and optimized❖ Often used in combination with feature extraction techniques, such as bag-of-words, to extract meaningful features from text data	<ul style="list-style-type: none">❖ Assumes a linear relationship between the features and the target variable❖ May not perform well with non-linear data
Naïve Bayes	probabilistic classifier that assumes that the presence of a particular feature is independent of the presence of other features.	<ul style="list-style-type: none">❖ Simple and fast❖ Can handle high-dimensional data❖ Can work well with text data and feature extraction techniques, such as bag-of-words	<ul style="list-style-type: none">❖ Assumes that the presence of a particular feature is independent of the presence of other features, which may not be true in some cases
Support Vector Machines (SVMs)	set of techniques that generalize linear classifiers to handle nonlinear decision boundaries by mapping the data into a higher-dimensional feature space	<ul style="list-style-type: none">❖ Can handle complex datasets with high accuracy❖ Capable of handling non-linear relationships between features and target variables	<ul style="list-style-type: none">❖ Can be computationally expensive, especially with large datasets❖ Requires careful tuning of parameters to avoid overfitting

		<ul style="list-style-type: none"> ❖ Can work well with both linear and non-linear data ❖ Effective in high-dimensional spaces ❖ Provides flexibility in choosing different kernel functions ❖ Can handle multi-class classification problems with appropriate modifications ❖ Performs well with small to medium-sized datasets 	<ul style="list-style-type: none"> ❖ May not perform well with noisy data ❖ Can be sensitive to the choice of kernel function ❖ Training time can be slow for large datasets
Support Vector Classifier (SVC)	a variant of Support Vector Machine (SVM) that is simpler and faster, and is primarily used for binary classification or multi-class classification problems.	<ul style="list-style-type: none"> ❖ Simple and fast ❖ Suitable for binary classification or multi-class classification problems ❖ Can handle complex datasets with high accuracy ❖ Can work well with both linear and non-linear data ❖ Performs well with small to medium-sized datasets 	<ul style="list-style-type: none"> ❖ May not perform as well as SVM on some datasets ❖ Limited flexibility in choosing different kernel functions compared to SVM ❖ May not be suitable for very large datasets ❖ Requires careful tuning of parameters to avoid overfitting.

d. Unsupervised learning

The alternative of this type of machine learning is **unsupervised learning**, in this case, one has a set of observations of a random factor with a joint probability density. And the goal is to directly infer the properties of this probability density without the help of a supervisor providing correct answers or degree of error for each observation. No labels are given to the learning algorithm, leaving it on its own to find structure in its input.

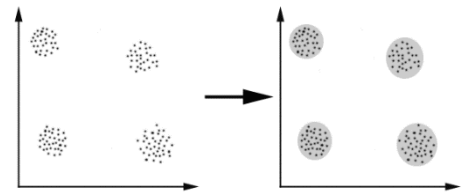
Unsupervised learning is a type of machine learning in which the algorithm learns to identify patterns and relationships in data without being explicitly told what to look for. Unlike supervised learning, there is no labeled data or target variable to guide the learning process. Instead, the algorithm attempts to discover structure or clusters in the data based on inherent similarities or differences in the features or attributes of the data.

Unsupervised Learning can be divided into two types:

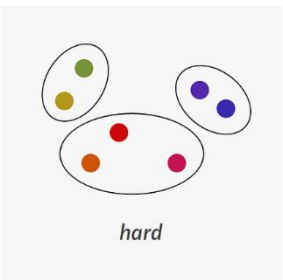
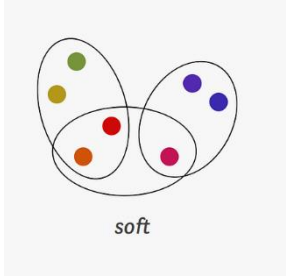
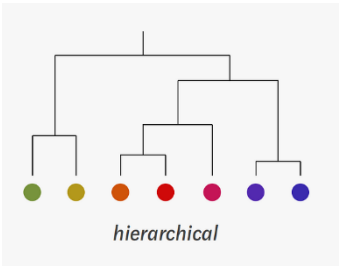
- ❖ **Parametric Unsupervised Learning** assumes a parametric distribution of the data, where the sample data follows a probability distribution based on a fixed set of parameters. It involves constructing Gaussian Mixture Models and using the Expectation-Maximization algorithm to predict the class of the sample.
- ❖ **Non-parametric Unsupervised Learning** groups the data into clusters to identify categories and classes present in the data. This method is commonly used to analyze data with small sample sizes and does not require any assumptions about the population distribution, making it a distribution-free method.

What is Clustering?

Clustering is an unsupervised learning problem that involves finding a structure in a collection of unlabeled data by organizing objects into groups based on their similarities. The goal is to form clusters of objects that are similar to each other while being dissimilar to objects in other clusters.



Clustering methods:

Hard clustering	Soft clustering	Hierarchical clustering
<p>Every object belongs to <i>exactly one</i> cluster</p>  <p style="text-align: center;"><i>hard</i></p>	<p>An object can belong to <i>one or more</i> clusters</p>  <p style="text-align: center;"><i>soft</i></p>	<p>Clusters are iteratively combined in a hierarchical manner, finally ending up in one root</p>  <p style="text-align: center;"><i>hierarchical</i></p>

Different clustering models:

Model	Description	Pros	Cons
K-Means Clustering	algorithm that partitions data into K clusters based on their similarity	<ul style="list-style-type: none"> ❖ Simple and easy to implement ❖ Computationally efficient for large datasets 	<ul style="list-style-type: none"> ❖ Requires the user to specify the number of clusters in advance ❖ Sensitive to initialization conditions ❖ Works best when clusters have a spherical shape, which might not be the case for text data
Hierarchical Clustering	creates a hierarchy of clusters. It can be agglomerative, starting with individual documents and merging them into clusters, or divisive, starting with all documents in one cluster and dividing them into smaller clusters.	<ul style="list-style-type: none"> ❖ Does not require the user to specify the number of clusters in advance ❖ Can produce a dendrogram(like the figure) that provides a visual representation of the clustering hierarchy 	<ul style="list-style-type: none"> ❖ Can be computationally expensive for large datasets ❖ Can produce non-optimal results if the distance measure or linkage criterion is not appropriate

DBSCAN	algorithm that groups together points that are close to each other in high-density regions and separates points that are in low-density regions.	<ul style="list-style-type: none"> ❖ Does not require the user to specify the number of clusters in advance ❖ Can handle non-spherical clusters and noise points 	<ul style="list-style-type: none"> ❖ Can be sensitive to the choice of distance metric and neighborhood radius parameter ❖ Can be computationally expensive for large datasets
Latent Dirichlet Allocation (LDA)	probabilistic topic modeling algorithm that extracts latent topics from a corpus of documents. Each document is assumed to be a mixture of topics, and each topic is a distribution over words in the vocabulary.	<ul style="list-style-type: none"> ❖ Can provide insights into the underlying topics in the documents ❖ Can handle large datasets 	<ul style="list-style-type: none"> ❖ Requires the user to specify the number of topics in advance ❖ Does not guarantee that each topic will be coherent or meaningful.

e. Choosing an algorithm:

When selecting a machine learning algorithm, there are four key factors to consider:

- ❖ the **size**, **quality**, and **nature** of the data
- ❖ the available computational **time**
- ❖ the **urgency** of the task
- ❖ the specific **goal** of the project (prediction, inference, or a sequence of actions)

The type of data will ultimately determine what machine learning approaches are available. Additionally, the urgency of the task and available computational resources will affect which algorithm is most suitable for the job. Finally, understanding the specific goal of the project will help guide the selection of the appropriate machine learning algorithm.

Step 1 : choosing the learning Method depending on the type of data and the goal to accomplish

For prediction tasks, classification, regression, or anomaly detection may be used, depending on the type of data available (labeled vs. unlabeled, or quantitative vs. qualitative). Unsupervised learning techniques such as dimensionality reduction and clustering may also be used in prediction tasks. Reinforcement learning, on the other hand, is used for generating a sequence of actions and is particularly useful for skill acquisition, task learning, and robot navigation.

Step 2 : Consider accuracy and training time

Before selecting a machine learning algorithm, it's important to consider the accuracy needed for the task at hand. In some cases, an approximation may be sufficient, and less accurate algorithms with lower training times may be more appropriate. Training time is closely tied to accuracy, with neural networks having the highest training time but also the highest accuracy and simple algorithms such as linear support vector machines having low training times but lower accuracy. The chosen algorithm will depend on the balance between required accuracy and available computational resources and time.

Step 3 : Consider the number of parameters

Doing that can avoid overfitting and a poor ability to generalize that comes along with that.

f. Select a performance measure:

Once the model has been created, it is possible to evaluate the model. For this, many indicators can be used, the majority depending on the task chosen. In this project, I will use these indicators:

❖ **Confusion Matrix:** In the classification task, the confusion matrix is the main indicator of model quality. It is a double-entry table matching the actual classes and the classes predicted by the model. In the matrix, there are 4 cells: true positives (VP), true negatives (VN), false positives (FP), false negatives (FN).

❖ **Accuracy:** Accuracy is the first indicator deviated from this matrix. It consists of looking at which portion of the predictions is correct. Its formula is therefore for a binary classification: $\text{accuracy} = (\text{VP} + \text{VN}) / \text{total}$. The goal is to classify tweets as positive or negative with at least 80% accuracy, so the minimum performance required would be 80% accuracy.

Accuracy, which does not distinguish the type of error made, is supplemented by two other indicators:

❖ **Recall:** it is the proportion of positive class detected. A strong recall therefore indicates that almost all cases in the positive class have been detected.

❖ **Precision:** it is the proportion of true positives in all positives detected. This makes it possible to estimate the number of positive tweet.

❖ **F1-score:** it is based on precision and recall; it gives a unique indicator.

IV. Data Scientist and Engineer

For all of my project, I use **Python** through **Anaconda**, which is an open-source distribution of Python and R. Anaconda is great for data processing and predictive analytics and comes with Conda, a package and environment manager that makes it easy to download and install necessary packages. Additionally, Anaconda includes **Jupyter**, an open-source software for creating collaborative notebooks that allow for sharing code across platforms and users.

a. Data import

	Insufficient	Marginal	Good	Excellent
3. Data collecting	The student uses only one (provided) dataset and has little or no understanding of its content	The student uses only one (provided) dataset and has full understanding of its content	Additional to Marginal: The student has combined several (external) datasets. The student knows how to get data from an API or by webscraping	Additional to Good: The student has implemented jobs to gather data from an API or by webscraping continuously

*In this part, in my opinion I have accomplished the level **Good**.*

Beyond the onboarding assignment:

- ❖ I have done some training of importing dataset from SVC file by using the Pandas **read_csv** function [fig1] with for example [the Iris Flowers dataset](#).
- ❖ I learned how to **examine** the dataset with different method, it's important to understand what we're working on, like **head()** [fig2], **describe()** [fig3], **groupby()** [fig3] functions.

For the little project I do by my own, I will be utilizing the Iris Flowers dataset, which comprises 150 observations of iris flowers and four columns of measurements along with the corresponding species of the flower. This dataset is widely used in statistics and machine learning as it serves as a standard. I

load this dataset from the UCI machine learning repository through an URL.

I load the data from the SVC file by using the Pandas **read_csv** function.

```
import pandas as pd
#load dataset
url="https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names=['sepal_leght', 'sepal_width', 'petal_length', 'petal_width', 'class']
dataset= pd.read_csv(url, names=names)
```

```
print(dataset.shape)
(150, 5)
```

Figure 1

```
print(dataset.head())
```

	sepal_leght	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Figure 2

I can examine the first five lines using the **head()** method of the training dataset.

I can study more characteristics of each variable using the describe() function and the groupby() function.

```
print(dataset.describe())
```

	sepal_leght	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
print(dataset.groupby('class').size())
```

class	size
Iris-setosa	50
Iris-versicolor	50
Iris-virginica	50

dtype: int64

Figure 3

For the project, we need to retrieve data from an article which is in the form of a PDF files. In the beginning, before we divided up the roles within the team, I had started researching it:

- ❖ Convert the PDF file to text format using a PDF to text conversion tool, such as Adobe Acrobat or an online PDF converter. Ensure the text is of high quality and images and tables are also included in the text file
- ❖ Use a Python library to extract the data from the text file (Pandas or Numpy) to perform data cleaning and transformation operations and then normalize data by putting it in a form that can be easily processed by machine learning algorithms. This can include normalizing numeric values, converting categories to binary variables, and handling missing values.

Which library choose?

Library	Description	Pros	Cons
PyPDF2	a library that can be used to extract text and metadata from PDF files	It can handle encrypted PDF files and allows you to extract text from individual pages or the entire document.	it may not work well with scanned PDFs or PDFs with complex layouts.
pdfminer	A library for extracting text and metadata from PDF files	it can handle more complex PDFs and is able to extract information about fonts, images, and other elements in the PDF.	It is more low-level than PyPDF2 and requires more code to get started.
textract	a library that can extract text from a variety of file formats, including PDFs	It uses a combination of other libraries, including PyPDF2 and pdfminer, to extract text. It also has support for OCR (optical character recognition) to extract text from scanned PDFs	
Tabula-py	a library that can be used to extract tables from PDFs. It uses a combination of OCR and pattern matching to identify tables in the PDF and extract the data	It can handle simple and complex tables and can output the data in CSV or JSON format	

b. Data cleaning

The data preparation phase makes it possible to move from raw data (as extracted from data sources) to data that can be used by the various Machine Learning algorithms. Here, machine learning algorithms can only work properly with good data.

	Insufficient	Marginal	Good	Excellent
Data cleaning	The student can not turn a real life datasets into a usable dataset for Machine Learning.	The student as done some basic cleaning of a (provided) dataset, like set datatypes, label encoding and hot encoding	Additional to Marginal: The student has done advanced cleaning on several datasets, using text, date, and number conversions or translations.	Additional to Good: The student has used for instance complex regex expressions and lambda functions to clean the data

*In this part, in my opinion I have accomplished the level **Excellent**.*

[1] In the onboarding assignment, I did a data cleaning process:

- ❖ I have done some basic cleaning like delete columns using. drop function, stopwords...
- ❖ Limit text and normalization using stemming and lemmatization methods.
- ❖ Vectorize the text data using CountVectorizer from sklearn.feature_extraction.text.

[2] Beyond the courses taught in class:

- ❖ I have learned [two NLP techniques](#) to reduce words to their base or root form: **stemming** and **lemmatization**.
- ❖ I learned how to use a [lambda function and the filter\(\) function](#) to remove stopwords, which is more memory-efficient. It also learn how to use a **map() function** with a lambda function to apply both stemming and lemmatization [\[fig10\]](#) to each word in the list of words. This replaces the separate loops that apply stemming and lemmatization, which makes the code more concise and easier to read.

[3] As part of my project, I created a [python script who performs text cleaning on CSV files found in a specified directory](#).

- ❖ A [‘clean text’](#) function to clean a given text string.
- ❖ A [‘clean csv files’](#) function who iterates through all files in the input directory. I learn how to work with the **‘os’** librairy.

Natural language processing:

There are two natural language processing (NLP) techniques to reduce words to their base or root form with the aim of standardizing words and reducing the dimensionality of the vocabulary:

❖ **Stemming:** involves removing the suffix from a word to reduce it to its base form. For example, the stem of "walking", "walked", and "walks" is "walk". This process is relatively simple and fast, but it can also result in words that are not actually words (e.g. "walk" is a valid word, but "walks" and "walked" are not).

❖ **Lemmatization:** involves reducing a word to its base form (called a lemma) using a dictionary based approach that considers the part of speech of the word. For example, the lemma of "am", "is", "are", "was", and "were" is "be". This process is more accurate than stemming, but it is also more complex and computationally expensive.

```
# stem the words
stemmer = DutchStemmer()
words = cleaned_text.split()
words = [stemmer.stem(word) for word in words]
cleaned_text = ' '.join(words)
# lemmatize the words
lemmatizer = WordNetLemmatizer()
words = cleaned_text.split()
words = [lemmatizer.lemmatize(word) for word in words]
cleaned_text = ' '.join(words)

return cleaned_text
```

The stemmed words are first split into a list of individual words, and then each word is stemmed using the DutchStemmer from the nltk library. After stemming, the words are joined back together into a single string. The lemmatizer also receives the cleaned text as a string, and it splits the string into individual words. The lemmatizer then applies its algorithm to each word and returns its lemma. Finally, the lemmatized words are joined back together into a single string.

figure10

Lambda, map & filter function:

I wanted to learn and use a **lambda function**, which could lead to **more optimal** and **less expensive** code. So on the new code, stemming and lemmatization are done using a single **map()** function along with a lambda function.

- ❖ The **filter()** function is used to remove elements from a sequence that do not satisfy a certain condition. In this case, the **filter()** function is used to remove all stop words from the list of words.
- ❖ The **map()** function applies a function to each element of a sequence and returns a new sequence with the results. In this case, the **map()** function is used to apply both stemming and lemmatization to each word in the list of words using a lambda function.

```
# apply both stemming and lemmatization using a lambda function
stemmer = DutchStemmer()
lemmatizer = WordNetLemmatizer()
words = map(lambda word: stemmer.stem(lemmatizer.lemmatize(word)), words)
cleaned_text2 = ' '.join(words)
```

figure 11

The lambda function takes a single argument (a word), and applies both the stemming and lemmatization functions to it. This replaces the two separate loops that apply stemming and lemmatization with a single **map()** function and lambda function, which is more concise and easier to read in some cases.

Part of the project: a python script who performs text cleaning on CSV files found in a specified directory.

1. The required libraries are imported:

- **pandas** (imported as **pd**) is used for data manipulation and analysis.
- **re** is used for regular expression operations.
- **stopwords** from the NLTK library is used to remove common words that do not carry much meaning.
- **DutchStemmer** from the NLTK library is used for stemming Dutch words.
- **WordNetLemmatizer** from the NLTK library is used for lemmatizing words.
- **word_tokenize** from the NLTK library is used to tokenize text into words.
- **os** provides a way of interacting with the operating system.

2. The 'clean_text' function: is defined to clean a given text string.

```
def clean_text(text_d):
    #check if the input is a string
    if type(text_d) == str:
        # remove HTML tags
        text_d = re.sub(r'<.*?>', '', text_d)
        # remove leading and trailing whitespace
        text_d = text_d.strip()
        # remove consecutive whitespace (tabs/newlines)
        text_d = re.sub(r'\s+', ' ', text_d)
        # convert to lowercase
        text_d = text_d.lower()
        # remove email addresses
        text_d = re.sub(r'\S+@(\S+)?(\.\S+)+', '', text_d)
        # remove http URLs
        text_d = re.sub(r'http\S+', '', text_d)
        # remove www URLs
        text_d = re.sub(r'www.\S+', '', text_d)
        # remove non-alphanumeric charact: punctuation/special characters
        text_d = re.sub(r'[^a-zA-Z0-9\s]', '', text_d)
        # remove phone numbers
        text_d = re.sub(r'\b[0-9]+\b\s*', '', text_d)
```

- ❖ It first checks if the input is a string.
- ❖ HTML tags are removed using regular expressions.
- ❖ Leading and trailing whitespace is removed.
- ❖ Consecutive whitespace (tabs/newlines) is replaced with a single space.
- ❖ The text is converted to lowercase.
- ❖ Email addresses, http URLs, www URLs, non-alphanumeric characters (punctuation/special characters), and phone numbers are removed using regular expressions.

```
# tokenize the string into individuals words
# allow to apply techniques/perf
words = word_tokenize(text_d)
found_stopwords = ['fysiopraxis', 'fysio', 'praxis', 'fysiotherapie', 'therapie', 'fysiotherapeut',
                    'therapeut', 'patient', 'jar', 'jaargang', 'januari', 'februari', 'maart', 'april',
                    'mei', 'juni', 'juli', 'augustus', 'september', 'oktober', 'november', 'december',
                    'nederland', 'artikel', 'email', ]
words = [word for word in words if word not in found_stopwords]
# remove stopwords
dutch_stopwords = stopwords.words('dutch')
words = [word for word in words if word not in dutch_stopwords]
# apply both stemming and lemmatization using a lambda function
stemmer = DutchStemmer()
lemmatizer = WordNetLemmatizer()
# map() applies lambda() to each word and returns a new sequence
# lambda() takes a word and applies stemm&lemm methods [lambda arg: exprss]
words = map(lambda word: stemmer.stem(lemmatizer.lemmatize(word)), words)
# convert the map object to a list
words = list(words)
# check if words is not empty
text_d = ' '.join(words)
if not text_d:
    text_d = "N"
else:
    text_d = "N"
return text_d
```

- ❖ The text is tokenized into individual words.
- ❖ Certain stopwords and specific words are removed from the list of words: when we studied the magazines, we noted a list of words that came up often and that didn't have any meaning afterwards.

- ❖ Dutch stopwords are removed.
- ❖ Both stemming and lemmatization are applied to each word using a lambda function.
- ❖ The resulting words are joined back into a string.
- ❖ If the resulting string is empty, it is assigned the value "N" & if the input is not a string, the output is assigned the value "N": The letter N has no meaning, however, a problem was occurring later in the project for the lines where there was no data in the text column. However, I was unable to remove the entire line, so I replaced the blank with a letter that will be removed in the next function.

3. The 'clean csv files' function: iterates through all files in the input directory.

First, I had to learn a way to work with file paths and directory operations: how to interact with the operating system to iterate through all the files in a directory and add new files to a directory.

I learned about the 'os' library:

The **os** library in Python provides a way to interact with the operating system. It offers various functions and methods to perform operations related to file and directory manipulation, environment variables, process management, and more.

```
# Set the directory path
directory = "data/csv_files/old_csv_files"
directory2="data/csv_files/cleaned_csv_files2"
# Iterate through all files in the directory
# Perform text cleaning on any CSV files found
def clean_csv_files():
    #return list of all the files and dir in the dir specified
    for filename in os.listdir(directory):
        # Check if the file ends with the extension ".csv"
        if filename.endswith(".csv"):
            # Load the CSV file
            # Joins dir path with filename var to create new path that points to CSV in the dir
            file_path = os.path.join(directory, filename)
            df = pd.read_csv(file_path)
            # Clean the 'text_d' column using the function
            df["text_d"] = df["text_d"].apply(clean_text)
            df = df[df['text_d'] != 'N']
            df = df[df['page_number'] > 5]
            # Create a new dataframe
            cleaned_df = df.copy()
            cleaned_df["text_d"] = df["text_d"]
            # Save the new df to a new CSV file with the same name as the original file with _cleaned add
            output_filename = filename.replace(".csv", "_cleaned.csv")
            output_file_path = os.path.join(directory2, output_filename)
            cleaned_df.to_csv(output_file_path, index=False)
            # Check the first five rows of the new df to confirm that the cleaning has been applied correctly
            print(f"{output_file_path}:")
            print(cleaned_df["text_d"].head(50))
        else:
            continue
```

For each file:

- ❖ It checks if the file ends with the extension ".csv".
- ❖ If so, it loads the CSV file into a pandas DataFrame (df).
- ❖ The clean_text function is applied to the "text_d" column of the DataFrame using the apply method, which cleans the text in each row.
- ❖ Rows where the cleaned text is "N" are removed from the DataFrame.

- ❖ Rows where the "page_number" column value is less than or equal to 5 are removed from the DataFrame.
- ❖ A new DataFrame (cleaned_df) is created as a copy of df, with the cleaned text in the "text_d" column.
- ❖ The new DataFrame is saved as a new CSV file with the same name as the original file but with "_cleaned" added to the filename.
- ❖ The first 50 rows of the cleaned text column are printed to confirm that the cleaning has been applied correctly.
- ❖ If a file does not end with the ".csv" extension, the iteration continues to the next file.

c. Data storage

	Insufficient	Marginal	Good	Excellent
Data storage	The student can not explain the differences between a SQL and a NOSQL database	The student can store datasets into a database and can argue the choice between a SQL and a NOSQL Database	Additional to Marginal: Student knows how to handle incremental uploads of data in the database In case a SQL Database is used , the student can explain the schema of the database In case a NOSQL database is used, the student can explain the nesting of JSON strings by referring to the most common use cases for searching	Additional to Good: In case a SQL Database is used : for communication with the database a ORM framework is used In case a NOSQL Database is used: All queries are parametrized

*In this part, in my opinion I have accomplished the level **Marginal**.*

I learned step by step this part.

[1] Beyond the courses taught in class, I learned the theoretical part of data storage:

- ❖ what is [the importance to back up the original data to a storage system](#)
- ❖ how to [choose between SQL and NoSQL](#)
- ❖ how to [use MySQL database with Python](#)
- ❖ what is [SQLAlchemy](#) and how to use it
- ❖ what is [SQLite](#) and how to use it

[2] As part of my project, I use my knowledge [for the project](#) and did some other research, I learned:

- ❖ [what is phpMyAdmin and how to use it](#): configuration of a file named "config.json" that contains the necessary information to connect to a MySQL database, creation of a "config.py" file to allow us to access the database configuration parameters stored in the JSON file, creation of a "sql.py" file to define a class called MySQL who contains methods to establish a connection to the MySQL database and to execute SQL queries and retrieve results from the database
- ❖ [how to create an instance of the MySQL class and connect it](#) to establish a connection to the database
- ❖ I started coding a [script with some SQL queries](#) knowing the requirements of the product owner. For now, I filled in the table some false data because I have to wait to get the data from my teammate, but he is still working on.
- ❖ After my teammate gave me all the new CSV files, I create a [script 'storeCleanDataInDB'](#) who essentially reads CSV files from a directory, extracts data from the files, and inserts the data into a MySQL database table.
- ❖ Then, I adapt the previous code of the SQL Queries to use it in the dashboard. I create a [script 'fct_queries'](#) who interact with the MySQL database and performs queries to retrieve data from a table called "magazine_clean".

[1] Beyond the courses taught in class, I learned the theoretical part of data storage:

1. Importance to back up the original data to a storage system: (like a database)

- Manage large data
- Ensure data consistency
- Access data faster (databases are opti & use of SQL queries)
- Make it easy to maintain and update data

2. Choosing between SQL and NoSQL:

The choice between a SQL and a NoSQL database will depend on the specific requirements and use case, the general factors to consider are:

	SQL databases	NoSQL databases
Data structure	best suited for structured data that can be represented in tables with rows and columns	better for semi-structured or unstructured data, such as text.
Scalability	better suited for vertical scaling	typically better suited for scaling horizontally
Data consistency	enforce strict data consistency rules, which can be an advantage in some use cases but can also lead to performance issues	typically more flexible in terms of data consistency, which can be an advantage in some scenarios
Querying flexibility	better suited for complex queries involving multiple tables	better suited for ad-hoc queries and fast data access

The data in our project is considered **structured**: it is organized into tables with well-defined columns and rows, even if the text column can be consider semi-structured data. → **SQL databases**

3. O'REILLY COURSE: Using MySQL Databases with Python

- Learn to use MySQL databases with Python
- understanding the basics of using MySQL with Python
- explore the technique of creating tables for safely storing data
- Get to grips with adding, sorting and pulling specific data

```
import mysql.connector

#create instance of var to connect to the db
mydb=mysql.connector.connect(
    #url of the database
    host="",
    #user &password when we set up the db
    user="",
    passwd="!",
)

print(mydb)
```

Connect to Database in Python:

```
dir(s7big_data_Project/coursecoreilly/database.py
<mysql.connector.connection_cext.CMySQLConnection object at 0x000001A452D4E8D0>
```

Create a cursor and database:

```
#create cursor
my_cursor=mydb.cursor()

my_cursor.execute("CREATE DATABASE test2")

my_cursor.execute("SHOW DATABASES")
for db in my_cursor:
    print(db)
```

Create table:

```
my_cursor.execute("CREATE TABLE user(name VARCHAR(255), email VA
```

Insert one record into table:

```
sqlStuff = "INSERT INTO user (name, email, age) VALUES(%s, %s, %s)"
records1= ("John", "john@d.com", 40)
#put into db
my_cursor.execute(sqlStuff, records1)
#commit a change to the db
mydb.commit()
```

4. BOOK: Essential SQLAlchemy

SQLAlchemy helps to map Python objects to database tables without substantially changing the existing Python code.

- **SQLAlchemy Core:** Provide database services to your applications in a Pythonic way with the SQL Expression Language
- **SQLAlchemy ORM:** Use the object relational mapper to bind database schema and operations to data objects in our application

	SQLAlchemy Core	SQLAlchemy ORM
Definition	low-level SQL abstraction layer that provides a set of Python classes and functions for communicating with databases	high-level Object-Relational Mapping (ORM) layer built on top of the Core.
Allows to	write raw SQL queries and provides a way to execute them through Python code	to interact with databases using Python objects, rather than raw SQL
Syntax	write SQL statements using Python functions and classes	write Python code that interacts with the database, and the ORM handles the SQL generation
Abstraction	provides a low-level, fine-grained abstraction over the database	provides a higher-level, object-oriented abstraction (work with Python objects that map to database tables and provides methods for querying and manipulating those objects)
Complexity	simpler and easier to learn: requires knowledge of SQL and database schema design	requires knowledge of both SQL and Python, as well as the ORM's API
Flexibility	more flexible : allows for more complex SQL queries and operations	more convenient for common CRUD operations, as it provides a simpler, higher-level API

Defining schema:

SQLAlchemy Core	SQLAlchemy ORM
create a metadata container and then declared a Table object associated with that metadata	define a class that inherits from a special base class called the <code>declarative_base</code> . The <code>declarative_base</code> combines a metadata container and a mapper that maps our class to a database table. It also maps instances of the class to records in that table if they have been saved.

Learning about the coding part of SQLAlchemy ORM:

Defining Schema: I learned how to:

- ❖ Define tables via ORM Classes
- ❖ Define keys
- ❖ Create database tables : using 'create_all' method on the metadata within our Base instance (metadata refers to the information about the database and its objects, such as tables, columns, indexes, views, and constraints)

- Imports the sessionmaker class.
- Defines a Session class with the bind configuration supplied by sessionmaker.
- Creates a session for our use from our generated Session class.

- ❖ Use the session: is the way SQLAlchemy ORM interacts with the database: is a transactional conversation between an application and a database
- ❖ Insert Data: adding instance to the session and then commit the session

- Inherit from the declarative_base object.
- Contain __tablename__, which is the table name to be used in the database.
- Contain one or more attributes that are Column objects.
- Ensure one or more attributes make up a primary key.

5. Using SQLite:

SQLite is a serverless database, meaning it doesn't require a separate server process. It's easier to set up and maintain. SQLite is a self-contained library that doesn't require external dependencies,

making it compatible with various operating systems and programming languages. MySQL, being a client-server database, requires a separate MySQL server and client libraries to be installed and configured.

I did a script that demonstrates the usage of SQLAlchemy, a Python SQL toolkit, to interact with a SQLite database:

First, we create a base class for declarative models and then we define the Magazine class that inherits from **Base**.

It set the name of the table in the database and define the columns of the table with their respective data types.

Then, we create an engine object with the connection string pointing to a SQLite database file named **magazine.db**.

We create the **magazine** table in the SQLite database using the metadata defined in the **Base** class and a session class bound to the engine to interact with the database.

```
from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.orm import sessionmaker
from sqlalchemy.orm import declarative_base

# create a base class for declarative SQLAlchemy models
Base = declarative_base()

# create 'Magazine' class that inherits from declarative_base
# define the columns in the magazine table
class Magazine(Base):
    __tablename__ = 'magazine'

    id = Column(Integer, primary_key=True)
    file_name = Column(String(255))
    page_number = Column(Integer)
    text_d = Column(String(1000))
    tag = Column(String(10))
    size = Column(Float)

# create a SQLAlchemy engine
# passing in the connection string to connect to the SQLite
engine = create_engine('sqlite:///magazine.db')

# create the magazine table in the SQLite database
# metadata is a SQLAlchemy object that holds metadata about
Base.metadata.create_all(engine)

# create session object to interact with the database
Session = sessionmaker(bind=engine)
session = Session()
```


Part of the project:

For the project, we wanted to use an **online database management tool**. We can access to phpMyAdmin with the school. phpMyAdmin is a tool that helps people manage and control their databases. It's like a user-friendly website where we can perform different tasks, such as creating, changing, or deleting databases and tables.

While local database management systems like SQLite or the MySQL shell have their advantages, such as faster performance for certain use cases or more control over the system configuration, online tools like phpMyAdmin provide a convenient and accessible solution for managing databases, particularly for users who prefer a visual interface or need remote access.

Using phpMyAdmin:

'config.json': We have create a configuration file named "config.json" that contains the necessary information to connect to a MySQL database. The "config.json" file includes details such as the host, username, password, and database name. this file has keywords for the company that no one should have but in this case, it was not ignore to be share to the group.

```
config.json > {} mysql > password
1  {
2      "mysql":{
3          "host":"127.0.0.1",
4          "user":"root",
5          "password":"Zxcvbn123456",
6          "db":"Zxcvbn"
7      }
8  }
```

'Config.py': In the "config.py" file, I import the JSON module and load the content of the "config.json" file into the **config** variable. This allows us to access the database configuration parameters stored in the JSON file.

```
import json

config = {}

with open('C:\\Users\\laura\\Documents\\ESIEA\\3A\\AMS\\config.json') as f:
    config = json.load(f)
```

'Sql.py': this file define a class called **MySQL**. This class has methods to establish a connection to the MySQL database using the provided host, username, password, and database name. It also includes methods to execute SQL queries and retrieve results from the database.

```
import mysql.connector as mysql

class MySQL:
    def __init__(self, host, user, password, database):
        self.host = host
        self.user = user
        self.password = password
        self.database = database
        self.connection = None
        self.cursor = None

    def connect(self):
        self.connection = mysql.connect(
            host=self.host,
            user=self.user,
            password=self.password,
            database=self.database
        )
        if self.connection.is_connected():
            self.cursor = self.connection.cursor()
```

```
def close(self):
    if self.connection.is_connected():
        self.cursor.close()
        self.connection.close()

def execute(self, query):
    self.cursor.execute(query)

def fetchall(self):
    return self.cursor.fetchall()

def fetchone(self):
    return self.cursor.fetchone()
```


Creation of an instance of the MySQL class and connection to the database:

```
import mysql.connector as mysql
from config import config as config
from sql import MySQL as sql

db = sql(host=config['mysql']['host'], user=config['mysql']['user'], password=config['mysql']['password'], database=config['mysql']['db'])
db.connect()
```

Then, to use that, we need to import the necessary modules, including the **config** dictionary from the "config" file and the **MySQL** class from the "sql" file. We then need to create an instance of the **MySQL** class, passing the relevant values from the **config** dictionary to the constructor. Afterward, we call the **connect()** method on the **db** object to establish a connection to the database.

Once the connection is established, we can proceed to execute SQL queries using the **execute()** method and retrieve the results using the **fetchall()** or **fetchone()** methods.

The product owner has some requirement:

- Know the part of the text that are dealing with complexity
- Take one specific year and see the words
- Make analysis peer magazine / peer year
- Comparison to another selection, to find patterns between years : summary peer year/magazine
- Excel (or something like that): show for some words how the frequency change peer magazine / year
- Search for some specific words : he want to input, enter the word and see when he is used / enter a set of words, synonyms so he can see the complexity

SQL QUERIES:

For that, I am going to make some SQL queries with the table I create to show what he want.

```
db.execute(create_magazines_table_query)
#db.execute(alter_table_query)

db.close()
```

```
# Create the magazines table
create_magazines_table_query = """
CREATE TABLE magazines (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name_file TEXT,
    year_jaargang INT,
    number_month_in_year INT,
    year YEAR,
    text TEXT
);
```

[1] I just define 2 functions that I am going to use right after:

'print_results': takes the query results and headers as input and formats them into a grid-shaped table using the **tabulate** library. The table is then printed to the console, so directly to the application.

'get_available_years': executes a SQL query to retrieve distinct years from the "magazines" table. It uses the **db.execute()** method to execute the query.

```
def print_results(results, headers):
    table = tabulate(results, headers, tablefmt="grid")
    print(table)

def get_available_years():
    query = """
    SELECT DISTINCT year FROM magazines;
    """
    db.execute(query)
    results = db.fetchall()
    years = [str(result[0]) for result in results]
    return years
```

[2] For all the queries, I decided to do a sort of a menu, so the user can choose between all the possibilities. Here for example:

(For now, I filled in the table some false data because I have to wait to get the data from my teammate but he is still working on.)

Menu:

1. View all magazines
2. View magazines for a specific year
3. Search for a specific word
4. View magazines for a range of years
5. Exit

Option 1: view all the data of the magazines: it execute a SQL query to retrieve all the magazines from the "magazines" table using the **SELECT * FROM magazines** query.

```
Enter your choice (1-5): 1
```

Magazines:

ID	Name File	Year Jaargang	Number Month in Year	Year	Text
1	fysio1	2	1	2012	hello, oui, trois, quoi
2	fysio2	20	2	2025	oui, toto, soleil
3	fysio3	14	3	2022	hva, meda
4	fysio4	25	5	2013	fysio, project, toto
5	fysio5	5	4	2014	lol, oui, non

```
if choice == "1":
    select_all_query = """
    SELECT * FROM magazines;
    """
    db.execute(select_all_query)
    results = db.fetchall()
    headers = ["ID", "Name File", "Year Jaargang", "Number Month in Year", "Year", "Text"]
    print("\nMagazines:")
    print_results(results, headers)
    print()
```

Option 2: The user can choose to see data from a specific year. The code retrieves the available years from the database using the **get_available_years** function. It then prompts the user to enter a year. The entered year is used in a SQL query to retrieve magazines for that specific year from the "magazines" table.

```
Enter your choice (1-5): 2
```

Available years:

1. 2012
2. 2025
3. 2022
4. 2013
5. 2014

Enter the year you want to view: 2012

Results for year 2012:

Name File	Year	Text
fysio1	2012	hello, oui, trois, quoi

```
elif choice == "2":
    available_years = get_available_years()

    if available_years:
        print("\nAvailable years:")
        for idx, year in enumerate(available_years):
            print(f"{idx+1}. {year}")
        while True:
            selected_year = input("\nEnter the year you want to view: ")

            if selected_year in available_years:
                select_text_query_year = f"""
                SELECT name_file, year, text FROM magazines WHERE year = {selected_year};
                """
                db.execute(select_text_query_year)
                results = db.fetchall()
                headers = ["Name File", "Year", "Text"]
                print(f"\nResults for year {selected_year}:")
                print_results(results, headers)
                print()
                break
            else:
                print("\nInvalid year. Please enter a valid year from the available options.")
                print()
```

Option 3: The user can enter a specific word to see in which year it appears. The code prompts the user to enter a word. It uses the entered word in a SQL query with a **LIKE** condition to search for magazines that contain that word in the text column of the "magazines" table.

```
Enter your choice (1-5): 3
```

Enter the word you want to search: toto

Results for the word 'toto':

Name File	Year	Text
fysio2	2025	oui, toto, soleil
fysio4	2013	fysio, project, toto

```
elif choice == "3":
    word = input("\nEnter the word you want to search: ")
    select_text_query_word = f"""
    SELECT name_file, year, text FROM magazines WHERE text LIKE '%{word}%';
    """
    db.execute(select_text_query_word)
    results = db.fetchall()
    headers = ["Name File", "Year", "Text"]

    if results:
        print(f"\nResults for the word '{word}':")
        print_results(results, headers)
    else:
        print(f"\nNo results found for the word '{word}'.")
        print()
```

Option 4: the user can choose a range of years to see the data. This can help him to find patterns between years for example. The code prompts the user to enter a start year and an end year. It uses these values in a SQL query with a **BETWEEN** condition to retrieve magazines for the specified year range from the "magazines" table.

```
Enter your choice (1-5): 4
Enter the start year: 2012
Enter the end year: 2014

Results for the year range 2012-2014:
+-----+-----+-----+
| Name File | Year | Text |
+-----+-----+-----+
| fysio1    | 2012 | hello, oui, trois, quoi |
+-----+-----+-----+
| fysio4    | 2013 | fysio, project, toto |
+-----+-----+-----+
| fyzio5    | 2014 | lol, oui, non |
+-----+-----+-----+
```

```
elif choice == "4":
    start_year = input("\nEnter the start year: ")
    end_year = input("Enter the end year: ")

    select_text_query_year_range = f"""
    SELECT name_file, year, text FROM magazines WHERE year BETWEEN {start_year} AND {end_year};
    """

    db.execute(select_text_query_year_range)
    results = db.fetchall()
    headers = ["Name File", "Year", "Text"]

    if results:
        print(f"\nResults for the year range {start_year}-{end_year}:")
        print_results(results, headers)
    else:
        print("\nNo results found for the specified year range.")
    print()
```

Updates on the SQL queries and store the magazines data:

Now that my teammate gave me all the new CSV files created, I can first store them in the database. The new CSV files are of the form 'page_number' and 'text'. Each file name corresponds to the date of the magazine. So I will select the title of each file to store the data in the date column of the table. After creating the new table which is in the form of 3 columns: 'date', 'page_number' and 'text', I will create a script allowing to browse the desired folder to store all the data of each .csv file of the file in the database. I also store the old csv files before cleaning in the database.

```
page_number,text
1,Fysio praxisvakblad voor de fysiotherapeut JAARGANG 2
2,002_Dvdk_adv_210x297_OL.indd 1 29/01/15 14:52
3,"FysioPraxis | maart 20153 REACTIES KUNT U MAILEN NA
4,"Accreditatievrijstelling voor een deelregister naar
5,"FysioPraxis | maart 20155 OP DE COVER AGENDA Maart
```

Script: 'storeCleanDataInDB' essentially reads CSV files from a directory, extracts data from the files, and inserts the data into a MySQL database table.

After importing the necessary modules, establishing a database connection and defining the CSV directory path, we iterate over each file in the directory using the 'os' library as learned before. Then we process each CSV file:

- ❖ The **date_str** variable is assigned the date part of the filename by splitting on the "." character and taking the first part.
- ❖ The **year**, **month**, and **day** variables are assigned the corresponding parts of the date string by splitting on the "-" character.
- ❖ The **formatted_date** variable is created by formatting the date as 'YYYY-MM-DD' for MySQL, padding the month and day with zeros if necessary.

```
# Iterate over each file in the directory
for filename in os.listdir(csv_directory):
    if filename.endswith('.csv'):
        file_path = os.path.join(csv_directory, filename)
        date_str = filename.split('.')[0] # Extract the date string
        # Extract the year, month, and day from the date string
        year, month, day = date_str.split('-')
        # Format the date as 'YYYY-MM-DD' for MySQL
        #formatted_date = f"{year}-{month}-{day}"
        formatted_date = f"{year}-{month.zfill(2)}-{day.zfill(2)}"
```

Then the CSV file is opened: the `insert_query` variable is assigned the SQL query to insert data into the `magazine_clean` table, with placeholders (%) for the values. The data are insert into the database. Then we commit the changes and close the database connection.

```
# Open the CSV file with the correct encoding
with open(file_path, 'r', newline='', encoding='utf-8-sig') as file:
    reader = csv.reader(file)
    header = next(reader) # Read the header row
    # Extract the column names from the header
    column_names = [column.strip() for column in header]
    # Prepare the SQL query to insert data
    insert_query = f"INSERT INTO magazine_clean (date_column, page_number, text) VALUES (%s, %s, %s)"
    # Iterate over each row in the CSV file
    for row in reader:
        # Extract the row values and trim any leading/trailing whitespace
        page_number = row[0].strip()
        text = row[1].strip()
        row_values = (formatted_date, page_number, text)
        # Execute the SQL query to insert the row into the table
        cursor = db.connection.cursor()
        cursor.execute(insert_query, row_values)
        cursor.close()
    # Commit the changes and close the database connection
    db.connection.commit()
    db.close()
```

Now I need to adapt the previous [code of the SQL Queries](#) to use it in the dashboard. I will therefore create a new script to be able to implement the queries in the main.

Script: 'fct_queries' interacts with the MySQL database and performs queries to retrieve data from a table called "magazine_clean".

- ❖ Importing necessary modules
- ❖ Establishing a database connection

Defining 2 functions to help us for the main queries function:

- ❖ Defining a helper function to convert query results into a DataFrame: the `get_results_as_dataframe` function takes two parameters: **results** (the query results) and **headers** (column names for the DataFrame). Inside the function, a list called **data** is created. For each **row** in **results**, a dictionary is created by mapping each column name from **headers** to the corresponding value in the row. This dictionary represents a single row of data. The dictionaries are appended to the **data** list. Finally, a DataFrame is created from the **data** list, using the `pd.DataFrame()` function, and returned.

```
def get_results_as_dataframe(results, headers):
    data = []
    for row in results:
        data.append(dict(zip(headers, row)))
    df = pd.DataFrame(data)
    return df
```

- ❖ Defining a function to retrieve available years from the database: The `get_available_years` function doesn't take any parameters. It defines a SQL query that selects distinct years from the `date_column` of the `magazine_clean` table. The query is executed and the query results are fetched. The years are extracted from the results and converted to strings and returned as a list.

```
def get_available_years():
    query = """
    SELECT DISTINCT YEAR(date_column) FROM magazine_clean;
    """
    db.execute(query)
    results = db.fetchall()
    years = [str(result[0]) for result in results]
    return years
```

Function 'perform_query': to perform a query based on optional parameters: The function takes three optional parameters: **first_year_select**, **last_year_select**, and **word_select**.

```
def perform_query(first_year_select=None, last_year_select=None, word_select=""):
    select_query = """
    SELECT date_column, page_number, text FROM magazine_clean
    WHERE 1=1
    """

    if first_year_select and last_year_select:
        select_query += f" AND YEAR(date_column) BETWEEN {first_year_select} AND {last_year_select}"

    if word_select:
        select_query += f" AND text LIKE '{word_select}%"

    db.execute(select_query)
    results = db.fetchall()
    headers = ["Date", "Page Number", "Text"]
    df = get_results_as_dataframe(results, headers)
    return df
```

- ❖ It starts by defining a base select query that selects the **date_column**, **page_number**, and **text** columns from the **magazine_clean** table.
- ❖ The **WHERE 1=1** condition is included as a placeholder for potential additional conditions.
- ❖ If **first_year_select** and **last_year_select** are provided, the query is extended to filter the results based on the year range.
- ❖ If **word_select** is provided, the query is extended to filter the results based on a word appearing in the **text** column.

d. ML Model building

	Insufficient	Marginal	Good	Excellent
ML Model building, beyond the onboarding assignment	The student cannot explain any of the different statements in the code used to build a classifier	Student can explain only the basic statements in the code behind a classifier	Additional to Marginal: Student knows how to explain all the ins and outs of the pieces of code involved. In particular how to succeed in improving the overall accuracy, for instance by using GridSearch	Additional to Good: Advanced tweaking of the parameters involved in the used classifiers has been used

*In this part, in my opinion I have accomplished the level **Good**.*

- ❖ I can explain all statements in the code behind a classifier.
- ❖ I know [how to improve the accuracy of the models](#) : I have used **feature scaling** and **hyperparameter tuning** for all the algorithms.
- ❖ I learned [what is K-fold cross-validation](#) and how to use it.
- ❖ I search [why it is important to use pipeline](#) and [what means scaling the data](#).
- ❖ I can discuss about the results of a model : for example for the [results of KNN model](#).

[1] In my onboarding assignment, I did a lot of research and built several machine learning models.

[2] Beyond the courses taught in class, to deepen what I learned during the first weeks and the onboarding assignment, I built machine learning models on a dataset that I found on the internet to practice.

The task at hand is to curate the dataset by creating a validation dataset and setting up cross-validation. To create the validation dataset, the loaded dataset will be split into two.

```
from sklearn.model_selection import train_test_split
#split out test dataset
array=dataset.values
X=array[:,0:4] #first 4 columns
Y=array[:,4]#last
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.20, random_state=7)

#test and evaluate metric
acc='accuracy'

models=[]
models.append(('Logistic Regression', LogisticRegression()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('SVM', SVC()))
```

Figure 5

```
#make prediction on test dataset
results=[]
names=[]
for name, model in models:
    model.fit(X_train, Y_train)
    pred=model.predict(X_test)
    print(name)
```

Figure 4

Logistic Regression 0.8666666666666667				
	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	0.83	0.83	0.83	12
Iris-virginica	0.82	0.82	0.82	11
accuracy			0.87	30
macro avg	0.88	0.88	0.88	30
weighted avg	0.87	0.87	0.87	30
KNN 0.9				
	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	0.85	0.92	0.88	12
Iris-virginica	0.90	0.82	0.86	11
accuracy			0.90	30
macro avg	0.92	0.91	0.91	30
weighted avg	0.90	0.90	0.90	30
SVM 0.8666666666666667				
	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	0.83	0.83	0.83	12
Iris-virginica	0.82	0.82	0.82	11
accuracy			0.87	30
macro avg	0.88	0.88	0.88	30
weighted avg	0.87	0.87	0.87	30

Figure 6

Logistic regression performed at 80% accuracy on the validation data, **k-nearest neighbors** at 90% accuracy, and **support vector machine** remains the most accurate with 93.3%. The precision for each class was also analyzed and it was found that the third class was the most difficult for all models except for k-nearest neighbors. The **SVM**, however, achieved 100% precision for both Iris-setosa and Iris-versicolor in the validation data. With the validation step completed, it can be confidently concluded that the SVM with its RBF kernel is the best model for the classification problem, as it not only avoided overfitting during training, but also demonstrated the ability to generalize to new data with 93% accuracy - a better performance than the other models.

How to improve the accuracy of the models:

For my models, I can use two ways:

- ❖ **Feature selection:** Try selecting the most relevant features for the model. I can use techniques like correlation analysis, feature importance, or principal component analysis (PCA) to identify the most important features.
- ❖ **Hyperparameter tuning:** Try experimenting with different hyperparameters for your models. For example, you can try different values for the number of neighbors in the KNN algorithm or the regularization parameter in the logistic regression algorithm.

For learning and training that, I did an updated version of my code : I have added feature scaling and hyperparameter tuning for all the algorithms.

LOGISTIC REGRESSION MODEL:

```
models = []
models.append(('LR', Pipeline([('scaler', StandardScaler()), ('lr', LogisticRegression())]))
models.append(('KNN', Pipeline([('scaler', StandardScaler()), ('knn', KNeighborsClassifier())]))
models.append(('SVM', Pipeline([('scaler', StandardScaler()), ('svm', SVC())]))
```

First, I create a list of tuples, where each tuple contains a model name and a pipeline that includes a data scaler and the corresponding machine learning model.

Why use pipeline?

The difference between the first sets of model definitions [\[fig4\]](#) is that the second set includes a **pipeline** that scales the data using **StandardScaler** before applying the machine learning algorithm.

- ❖ In the first set of model definitions, the algorithms are used directly, without any pre-processing of the data.
- ❖ In the second set of model definitions, a Pipeline object is used to combine the scaling step with the machine learning algorithm. This has the advantage of making it easier to perform multiple preprocessing steps in a repeatable way, and to combine them with different machine learning algorithms.

The **StandardScaler** function is used to standardize the features by removing the mean and scaling to unit variance. This can be important for some algorithms, especially those that are sensitive to the scale of the input features.

In my code, I need to use pipeline for scaling the data before feeding it into the machine learning models. **StandardScaler** scales the input features to zero mean and unit variance, which is important for some machine learning algorithms to work well. Without scaling the data, some features may dominate others, leading to biased results. Using pipeline ensures that scaling is applied consistently during both training and testing, avoiding data leakage from the test set into the training set, and making the models more reliable.

What means scaling the data?

Scaling data means transforming the values of the features so that they have a similar scale. This is important because some machine learning algorithms, such as K-Nearest Neighbors and Support Vector Machines, are sensitive to the scale of the input features. By scaling the data, we ensure that all features contribute equally to the analysis and that the algorithm can focus on the patterns in the data rather than the differences in scale.

```
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=7, shuffle=True)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

Next, for each model in the list, I use k-fold cross-validation to evaluate its performance. It iterate through the list of models and for each model, it performs **10-fold cross-validation** using the `model_selection.cross_val_score()` function, which takes the model pipeline, the training data, the target variable, the number of folds for cross-validation, and the performance metric (here is **accuracy**) to evaluate the models. During each iteration, the code appends the cross-validation accuracy results to the **results** list and the model names to the **names** list.

The mean and standard deviation of accuracy across all folds are printed for each model.

```
LR: 0.966667 (0.040825)
KNN: 0.966667 (0.040825)
SVM: 0.975000 (0.038188)
```

What is K-fold cross-validation?

K-fold cross-validation is a method used to evaluate the performance of a machine learning model on a limited sample size. In this method, the original data set is randomly partitioned into k equal-sized subsets, or folds. Then, k iterations are performed, each time using one fold as the validation set and the remaining k-1 folds as the training set. This process is repeated until each fold has been used as the validation set once.

At the end of k iterations, the average performance across all k-folds is calculated to estimate the overall performance of the model. K-fold cross-validation is commonly used to evaluate the performance of models when there is a limited amount of data available, to help reduce overfitting, and to select optimal hyperparameters.

```
print("\nCHOOSING BEST LR MODEL ON TEST SET\n")
# Tune logistic regression algorithm by testing different values of C
lr_pipeline = Pipeline([('scaler', StandardScaler()), ('lr', LogisticRegression())])
c_values = [0.001, 0.01, 0.1, 1, 10, 100]
param_grid = {'lr__C': c_values}
lr_grid = model_selection.GridSearchCV(lr_pipeline, param_grid=param_grid, scoring='accuracy', cv=kfold)
lr_grid.fit(X_train, Y_train)
print("Best LR accuracy: ", lr_grid.best_score_)
print("Best LR parameters: ", lr_grid.best_params_)
```

Now, we perform the task of choosing the best logistic regression model using the **GridSearchCV** method, which helps to tune hyperparameters to find the best performing model. The **GridSearchCV** function is used to perform a grid search over a range of **C** values specified in the **c_values** list to find the best value of C.

The model is first defined using a **pipeline** that scales the data using the **StandardScaler** and then fits a logistic regression model.

CHOOSING BEST LR MODEL ON TEST SET

Best LR accuracy: 0.9833333333333332

Best LR parameters: {'lr__C': 100}

Here, C is the inverse of the regularization strength, which determines the amount of regularization applied to the model. Regularization is a technique used to prevent overfitting by adding a penalty term to the loss function. In **scikit-learn's LogisticRegression** class, C is the parameter used to control the strength of regularization, with smaller values indicating stronger regularization. **C is a hyperparameter which is tuned by trying different values to obtain the best accuracy for the logistic regression model.**

```
# Test the best LR model on the test set
best_lr_model = lr_grid.best_estimator_
best_lr_model.fit(X_train, Y_train)
lr_pred = best_lr_model.predict(X_test)
print("Best LR accuracy on test set: ", accuracy_score(Y_test, lr_pred))
print(classification_report(Y_test, lr_pred))
```

After the best hyperparameters are found, the **best model** is trained on the training set, and the accuracy of the model is evaluated on the test set. Finally, the classification report of the model is printed which includes metrics such as precision, recall, and f1-score. **The goal of this code is to find the best logistic regression model that can accurately classify the data.**

Best LR accuracy on test set:	0.8666666666666667			
	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	0.83	0.83	0.83	12
Iris-virginica	0.82	0.82	0.82	11
accuracy			0.87	30
macro avg	0.88	0.88	0.88	30
weighted avg	0.87	0.87	0.87	30

Discuss about the results of Logistic Regression model:

The output suggests that the best logistic regression model has an **accuracy of 0.9833** when trained on the **training data**, using the value of **C=100**. This is the highest accuracy that was achieved during cross-validation of the logistic regression model. The model is then tested on the **test set** and it achieved an **accuracy of 0.8667**, which is still quite high. So, the logistic regression model achieved a high accuracy during cross-validation, but its accuracy on the test dataset was slightly lower.

The classification report shows precision, recall, and F1-score for each class in the test set, as well as the weighted average precision, recall, and F1-score, and the overall accuracy. It shows that the model performed well on the iris-setosa class, but slightly worse on the iris-versicolor and iris-virginica classes.

Comparing the results of the first method [fig4] and this one:

As a result, we can see that the second set of models performs better than the first set of models. The accuracy of the LR model in the first set is 0.8, while the accuracy of the LR model in the second set is 0.983. Furthermore, the precision, recall, and f1-score are also higher in the second set of models, indicating that the second set of models is more accurate and robust.

Now, I do the same thing for the KNN and SVC models:

KNN MODEL:

```
print("\nCHOOSING BEST KNN MODEL ON TEST SET\n")
# Tune KNN algorithm by testing different numbers of neighbors
knn_pipeline = Pipeline([('scaler', StandardScaler()), ('knn', KNeighborsClassifier())])
k_values = list(range(1, 31))
param_grid = {'knn__n_neighbors': k_values}
knn_grid = model_selection.GridSearchCV(knn_pipeline, param_grid=param_grid, scoring='accuracy', cv=kfold)
knn_grid.fit(X_train, Y_train)
print("Best KNN accuracy: ", knn_grid.best_score_)
print("Best KNN parameters: ", knn_grid.best_params_)
# Test the best KNN model on the test set
best_knn_model = knn_grid.best_estimator_
best_knn_model.fit(X_train, Y_train)
knn_pred = best_knn_model.predict(X_test)
print("Best KNN accuracy on test set: ", accuracy_score(Y_test, knn_pred))
print(classification_report(Y_test, knn_pred))
```

First, a KNN pipeline is defined with two steps: scaling and the KNN algorithm. The **parameter grid** for KNN is defined as **a range of values** for the **number of neighbors**. Then, **GridSearchCV** is used to search for the best KNN model based on the provided parameter grid, the scoring metric, and the cross-validation strategy. The best KNN model is then trained on the training set, and the accuracy is evaluated on the test set.

```
CHOOSING BEST KNN MODEL ON TEST SET
Best KNN accuracy: 0.975
Best KNN parameters: {'knn__n_neighbors': 7}
Best KNN accuracy on test set: 0.9333333333333333
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	0.86	1.00	0.92	12
Iris-virginica	1.00	0.82	0.90	11
accuracy			0.93	30
macro avg	0.95	0.94	0.94	30
weighted avg	0.94	0.93	0.93	30

Discuss about the results of KNN model:

In the result, we see that the **best KNN accuracy is 0.975**, which means that the KNN model was able to correctly classify 97.5% of the samples in the training set. The best KNN parameters are {'knn__n_neighbors': 7}, indicating that the **optimal number of neighbors** for this problem is 7.

When we test the best KNN model on the test set, we see that the accuracy **drops slightly to 0.933**, indicating that the model is still able to generalize well to new data. This means that the model was able to correctly predict the class of **28 out of the 30 samples** in the test set.

Looking at the classification report, we can see that the model was **perfect** at predicting the **Iris-setosa** class. For the **Iris-versicolor** class, the precision was **0.86** and recall was **1.0**. This means that out of all the samples predicted to be Iris-versicolor, 86% were actually of that class, and the model was able to identify all of the Iris-versicolor samples in the test set. For the **Iris-virginica** class, the precision was **1.0** and **recall** was **0.82**. This means that out of all the samples predicted to be Iris-virginica, **100%** were actually of that class, but the **model missed 2 of the 11** Iris-virginica samples in the test set.

Comparing the results of the first method [fig4] and this one:

Here, we can see that the accuracy of the KNN model has increased **from 0.9 to 0.975**, indicating that the feature scaling has a positive effect on the model's performance. Furthermore, the precision, recall, and f1-score have also improved in this set of models, showing that this set of models is more accurate and reliable.

The best KNN model chosen on the test set in the second set also **performs better** than the KNN model in the first set. The accuracy of the best KNN model is **0.933**, which is higher than the accuracy of the KNN model in the first set (0.9). Moreover, the precision, recall, and f1-score of the best KNN model are also higher, indicating that the second set of models is more accurate and robust.

Therefore, we can conclude that scaling the data using StandardScaler and hyperparameter tuning to choose the best model on the test set can significantly improve the performance of machine learning algorithms, such as LR and KNN, in classification tasks.

SUPPORT VECTOR MACHINE (SVM) MODEL:

```
print("\nCHOOSING BEST SVM MODEL ON TEST SET\n")
# Define SVM pipeline
svm_pipeline = Pipeline([('scaler', StandardScaler()), ('svm', SVC())])
# Tune SVM algorithm by testing different values of C and gamma
C_values = [0.1, 1, 10, 100]
gamma_values = [0.1, 1, 10, 100]
param_grid = {'svm__C': C_values, 'svm__gamma': gamma_values}
svm_grid = model_selection.GridSearchCV(svm_pipeline, param_grid=param_grid, scoring='accuracy', cv=kfold)
svm_grid.fit(X_train, Y_train)
print("Best SVM accuracy: ", svm_grid.best_score_)
print("Best SVM parameters: ", svm_grid.best_params_)
# Test the best SVM model on the test set
best_svm_model = svm_grid.best_estimator_
best_svm_model.fit(X_train, Y_train)
svm_pred = best_svm_model.predict(X_test)
print("Best SVM accuracy on test set: ", accuracy_score(Y_test, svm_pred))
print(classification_report(Y_test, svm_pred))
```

I start by defining an SVM pipeline that includes scaling the data using **StandardScaler()** and an SVM model. Then, it tunes the SVM algorithm by testing different values of C and gamma using **GridSearchCV**.

- ❖ C is a regularization parameter that controls the trade-off between achieving a low training error and a low testing error by adjusting the width of the margin. A smaller value of C creates a wider margin and a simpler decision boundary, whereas a larger value of C creates a narrower margin and a more complex decision boundary.
- ❖ Gamma is a kernel coefficient that controls the influence of each training example in the decision boundary. A small gamma will result in a decision boundary that is smoother and simpler, whereas a large gamma will result in a decision boundary that is more complex and can potentially lead to overfitting.

The accuracy of the best SVM model and its corresponding parameters are printed. After that, the best SVM model is fitted to the training set, and its accuracy is calculated on the test set using **accuracy_score()**. Finally, the classification report is printed.

Discuss about the results of SVM model:

The result shows that the best SVM model achieved an accuracy of **0.983** on the training set, with a regularization parameter C of 100 and a gamma value of 0.1. However, when this model was tested on the independent test set, it achieved an accuracy of 0.867, which is slightly lower than the training set accuracy.

The precision, recall, and F1-scores were also calculated for each class. The precision measures the proportion of true positive predictions out of all positive predictions, while the recall measures the proportion of true positive predictions out of all actual positive instances in the dataset. The F1-score is the harmonic mean of precision and recall.

Looking at the precision, recall, and F1-score for each class, we can see that the SVM classifier performed well in classifying Iris-setosa with a perfect precision, recall, and F1-score of 1.0. However, the classifier performed less accurately on the other two classes, with a precision of 0.83 and 0.82, respectively. This indicates that the classifier made some incorrect predictions for these classes. The weighted average F1-score of the classifier was 0.87, indicating that the overall performance of the classifier was good, but not perfect.

```
CHOOSING BEST SVM MODEL ON TEST SET
Best SVM accuracy: 0.9833333333333332
Best SVM parameters: {'svm_C': 100, 'svm_gamma': 0.1}
Best SVM accuracy on test set: 0.8666666666666667
      precision    recall  f1-score   support

   Iris-setosa       1.00       1.00       1.00         7
  Iris-versicolor    0.83       0.83       0.83        12
   Iris-virginica    0.82       0.82       0.82        11

 accuracy                   0.87         30
  macro avg              0.88         30
 weighted avg            0.87         30
```

Comparing the results of the first method [fig4] and this one:

The accuracy of SVM in the first set is 0.8666666666666667, which is lower than the accuracy of the same algorithm in the second set, which is 0.9833333333333332.

e. Visualization

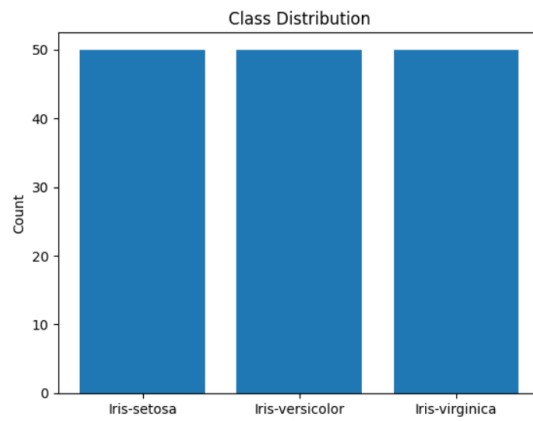
	Insufficient	Marginal	Good	Excellent
Visualization	There is no dashboard at all. The student has not made any visualizations	There is either <ul style="list-style-type: none">A Python IDE with interactive plots orAn out of the box package like Tableau or Power BI is used.	Additional to Marginal: Student had coded a dashboard in a modern programming language. In an interactive way the mantra of visualization is implemented: Overview first /Zoom and filter / Then Details on Demand	Additional to Good: Sophisticated interactive elements are implemented by coding

*In this part, in my opinion I have accomplished the level **Good**.*

Always the same as the other parts, not having realized an interactive visualization during the onboarding assignment, I trained on my side:

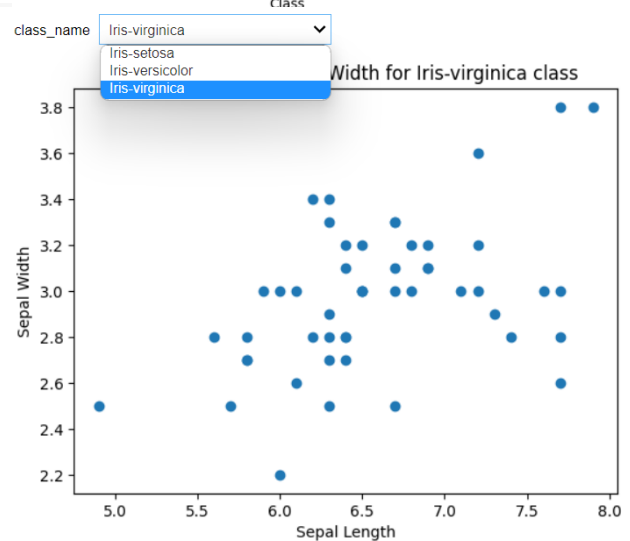
```
'''overview'''
#show the class distribution using a bar chart
class_counts = dataset.groupby('class').size()
plt.bar(class_counts.index, class_counts.values)
plt.title('Class Distribution')
plt.xlabel('Class')
plt.ylabel('Count')
plt.show()
```

Figure 7



```
'''Zoom and filter'''
#allow the user to select class using dropdown menu
from ipywidgets import interact, Dropdown
@interact(class_name=Dropdown(options=list(dataset['class'].unique())))
def show_scatter(class_name):
    #create scatter plot of sepal length vs sepal width for the selected class
    filtered = dataset[dataset['class'] == class_name]
    plt.scatter(filtered['sepal_length'], filtered['sepal_width'])
    plt.title(f'Sepal Length vs. Sepal Width for {class_name} class')
    plt.xlabel('Sepal Length')
    plt.ylabel('Sepal Width')
    plt.show()
```

Figure 8



This will help us understand the relationship between some of the variables.

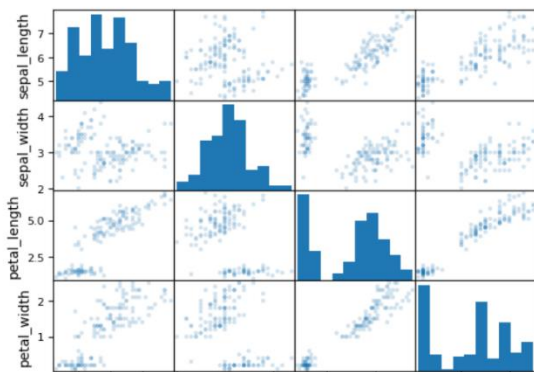


Figure 9

```
'''Details on Demand'''
#allow the user to click on a data point to show more info
def on_click(event):
    #check if the click was on a data point
    if event.artist.get_label() == 'data points':
        #get the index of the clicked data point
        index = event.ind[0]
        #show more info about the selected data point
        selected = dataset.iloc[index]
        print(f'Sepal Length: {selected["sepal_length"]}')
        print(f'Sepal Width: {selected["sepal_width"]}')
        print(f'Petal Length: {selected["petal_length"]}')
        print(f'Petal Width: {selected["petal_width"]}')
        print(f'Class: {selected["class"]}')

#create scatter matrix plot of all features with data points that can be clicked
scatter_matrix(dataset, alpha=0.2, label='data points')
#add an event listener for clicks on the plot
plt.gcf().canvas.mpl_connect('button_press_event', on_click)
plt.show()
```

So petal.width and petal.length are interrelated, they're dependent on each other. And again, we see that linear nature in some of these other ones up here too. And that relationship between variables is going to help our machine learning algorithms actually perform better.

I tried to write a new code using the Plotly Express library to create a scatter matrix plot of the Iris dataset, which shows the pairwise relationships between the four features (sepal length, sepal width, petal length, and petal width) and the class labels.

```
import pandas as pd
import plotly.express as px

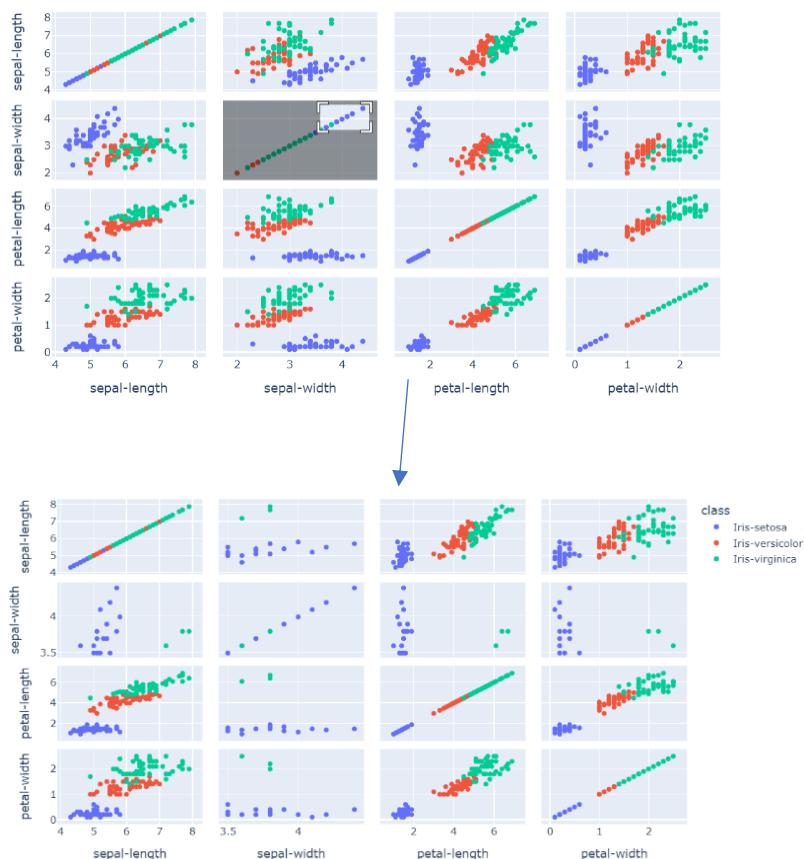
url="https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names=['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset=pd.read_csv(url, names=names)

def on_click(trace, points, state):
    if points.point_inds:
        index = points.point_inds[0]
        selected = dataset.iloc[index]
        print(f'Sepal length: {selected["sepal-length"]}')
        print(f'Sepal width: {selected["sepal-width"]}')
        print(f'petal length: {selected["petal-length"]}')
        print(f'petal width: {selected["petal-width"]}')
        print(f'Class: {selected["class"]}')

fig = px.scatter_matrix(dataset, dimensions=['sepal-length', 'sepal-width', 'petal-length', 'petal-width'], color='class')
fig.for_each_trace(lambda trace: trace.on_click(on_click))
fig.update_layout(dragmode='zoom')
fig.show()
```

- ❖ **fig.for_each_trace(lambda trace: trace.on_click(on_click))**: This line sets up an event listener for the scatter plot, such that when a data point is clicked, the **on_click** function is called with information about the selected point.
- ❖ **fig.update_layout(dragmode='zoom')**: This line sets the plot's dragmode to "zoom", which allows the user to zoom in and out of the plot using the mouse.
- ❖ **fig.show()**: This line displays the plot in a new window or tab in the default web browser.

The code creates an interactive scatter matrix plot of the Iris dataset, which allows the user to click on individual data points to display their feature values and class label:



f. Operations

	Insufficient	Marginal	Good	Excellent
8. Operations	The student has no idea of the processes involved in application development, testing, deployment and operations.	The student can explain the role of git repositories and python environments, and can use of these tools for team collaboration effectively.	Additional to Marginal: The student can run a local docker container with some of the coding beyond the onboarding assignment locally. The docker can run everywhere.	Additional to Good: The student has taken care of the whole devops pipeline. Whenever a change in the code is made and pushed to a git repo, a new image is being built and the image can run everywhere

In this part, in my opinion I have accomplished the level **Marginal**.

- ❖ I can explain the role of [git repositories](#) and [python environments](#) and can use of these tools for team collaboration effectively.
- ❖ I did some research and did a summary of what I learned:
 - [\[1\] Theoretical part of Docker](#)
 - [\[2\] Docker's basic concepts](#)

Since the beginning of the semester, I learned how to use git as well as the tools for managing Python virtual environments:

GIT:

Git is a version control system used for tracking changes in software code and other digital files. It allows multiple developers to work on the same codebase, while keeping track of changes made by each contributor.

- ❖ A Git repository is a collection of files and folders that are being tracked by Git. It is a centralized location where developers can access and make changes to the codebase. Git allows developers to create multiple repositories, each of which can have different branches or versions of the codebase.
- ❖ Commits in Git are used to save changes to the codebase. A commit represents a snapshot of the code at a specific point in time. Each commit has a unique identifier that can be used to reference the changes made in that commit.
- ❖ Branches in Git are used to create parallel versions of the codebase. A branch is essentially a copy of the codebase that can be modified independently of the original code. Developers can create new branches to work on new features or bug fixes, without affecting the main codebase.
- ❖ Merges in Git are used to combine changes from different branches. When a developer wants to incorporate changes from one branch into another, they can use Git to merge those changes. Git will automatically resolve any conflicts that arise, and create a new commit that includes the merged changes.

There is two popular **tools for managing Python virtual environments** that I use during this semester:

Anaconda:

Anaconda is a popular distribution of the Python programming language that includes a package manager, environment manager, and other tools for data science and scientific computing. It comes with a large collection of pre-built packages and libraries, including the popular NumPy, pandas, and matplotlib libraries, making it a popular choice for data science projects.

Anaconda provides its own virtual environment manager called **conda**, which allows users to create and manage virtual environments for Python and other programming languages. Conda allows users to create virtual environments with specific versions of Python and other packages, making it easy to manage dependencies and ensure reproducibility.

Pipenv:

Pipenv is another popular tool for managing Python virtual environments. It combines the functionality of pip (Python's default package manager) and virtualenv (Python's built-in tool for creating virtual environments) into a single tool. This tool simplifies the process of creating and managing virtual environments by automatically creating a new virtual environment whenever a new project is created. It also manages package dependencies for the project by automatically creating a **Pipfile** that lists all the required packages and their versions. It also provides other features, such as automatically updating dependencies, managing different environments for production and development, and enabling easy deployment of the project.

Docker:

I did some research and did a summary of what I learned:

[1] Theoretical part of Docker:

- ❖ Docker is an open-source platform for creating, deploying, and managing containerized applications. It was created by Solomon Hykes in France and has been available since 2013.
- ❖ **Containers and Virtualization:** Containers provide a standardized way to package and run programs on a host system. They isolate programs in a dedicated namespace and resource space. Unlike virtual machines, containers run directly on the host system, sharing its kernel for lightweight and fast execution.
- ❖ **Advantages and Use Cases of Docker:** Docker enables the creation, execution, and distribution of containers across various operating systems. It improves software development speed, simplifies application deployment at scale, and supports different host systems.
- ❖ **Standardization:** Docker's popularity led to the Open Container Initiative (OCI) in 2015, which defines specifications for building, executing, and distributing containers. It includes Image, Runtime, and Distribution Specifications.
- ❖ **Portability:** Docker provides standards for building and distributing containers. Containers include all dependencies necessary to run an application on a specific host system. Container images are defined using a manifest and can be stored in registries for easy distribution and execution.
- ❖ **Security:** Docker ensures the isolation of containerized programs through dedicated namespaces, resource limitations, and fine-grained kernel access control. It supports trusted images, kernel-level permissions, integration with third-party security software, and benefits from a secure underlying kernel.
- ❖ **Use Cases:** Docker is widely used for sharing reproducible work environments, running automated tests, deploying applications in production, and scaling horizontally.

[2] Docker's basic concepts:

Then I learned an overview of Docker's basic concepts:

Docker Image:

- Docker images are read-only file systems that contain all the necessary elements for running a program, including the file system, system libraries, dependencies, and executable files.

Anatomy of an Image:

- Docker images are built using layers, where each layer represents modifications made compared to the previous layer. Layers are stacked to form a unified file system from which the program can execute.

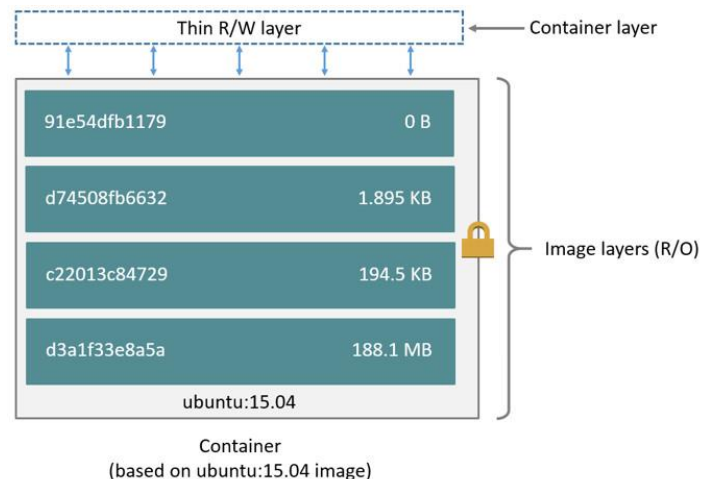


Image Creation:

- Docker images can be created using a Dockerfile, which contains instructions for building the image, or by modifying an existing container and committing the changes to a new image.

Image Registry:

- Docker images can be stored in image registries for distribution. Docker Hub is the default registry, hosting a wide range of images developed by Docker, partner companies, and the community. Images are identified using a naming convention of "<name>:<tag>".

Docker Containers:

- Docker containers represent executable processes and follow a lifecycle of creation, execution, stoppage, and deletion.

Container Creation and Execution:

- Containers are created and executed from a Docker image. If the image is not present locally, Docker can automatically retrieve it from the public Docker Hub registry.

As part of our project, the command lines are:

To download the project image:

```
docker pull wattbreak/physio:clean
```

To launch the project image:

```
docker run -p 8050:8050 wattbreak/physio:clean
```

V. Appreciation

I would like to express my gratitude to all the teachers for this Big Data semester. I am grateful for the valuable knowledge and skills I have acquired under your guidance.

I particularly want to highlight how much I enjoyed the learning process, especially the portfolio assignments and the presentations. The portfolio system, which encourages independent learning and values our work, was a refreshing change from the learning process I was accustomed to in France. I was pleasantly surprised by how effective it was for me. Throughout the semester, I learned so much. While there is still room for improvement, reflecting on where I started at the beginning of the semester with no knowledge of Big Data and seeing my progress now feels like a significant accomplishment. I am truly grateful for the teaching methods employed by the teachers and the emphasis on independent work.

Additionally, I want to express my appreciation for the last big project we worked on. It was a concrete project with clear goals, and I found it immensely fulfilling, who again, was a lot different than in France. Knowing that the project would be useful for the product owner added a sense of purpose and brought the project to life. It was an excellent opportunity for me to apply the knowledge and skills I had acquired throughout the semester and to learn more depending on which task I was working on.

VI. Sources

1. Géron, A. (2019). Machine learning avec Scikit-Learn: Mise en œuvre et cas concrets [Machine learning with Scikit-Learn: Implementation and practical examples]. (Authorized French translation of material from the English edition of Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2E, ISBN 9781492032649). O'Reilly Media, Inc.
2. Mathivet, V. (n.d.). Machine Learning: Implémentation en Python avec Scikit-Learn [Machine Learning: Implementation in Python with Scikit-Learn]. ENI. ISBN: 9782409023516.
3. Grus, J. (2016). K-means and hierarchical clustering with Python. O'Reilly Media, Inc.
4. Mannambeth, M. (2018). Docker for the Absolute Beginner - Hands-On [Tutorials course]. Packt Publishing. O'Reilly Media.
5. Into Data, J. (2021). Python for Data Analysis: Step-By-Step with Projects [Course]. Packt Publishing. O'Reilly Media
6. Aggarwal, C. C., & Reddy, C. K. (2013). Data Clustering. Chapman and Hall/CRC. O'Reilly Media
7. Author, A. (2020, July 14). A Friendly Introduction to Text Clustering. Towards Data Science. <https://towardsdatascience.com/a-friendly-introduction-to-text-clustering-fa996bcefd04>
8. Docker. (n.d.). What is a Container? Docker. Retrieved April 10, 2023, from <https://www.docker.com/resources/what-container/>
9. Myers, J., & Copeland, R. (2015). Essential SQLAlchemy (2nd ed.). O'Reilly Media, Inc.
10. Elder, J. (2019). Using MySQL Databases with Python. Packt Publishing.
11. Wilke, C. O. (2019). Fundamentals of Data Visualization. O'Reilly Media, Inc.
12. Myers, J. (Co-Author of Essential SQLAlchemy) & Software Engineer. (n.d.). Introduction to Databases in Python & Advanced SQLAlchemy Queries [DataCamp course].
13. (n.d.). Using MySQL Databases with Python [O'Reilly course].
14. Grossenbacher, T. (Head of Newsroom Automation at Tamedia). (n.d.). Introduction to Relational Databases in SQL [Interactive course].
15. Docker Documentation. (n.d.). Storage drivers. Retrieved from <https://docs.docker.com/storage/storagedriver/>
16. Docker Labs. (n.d.). Docker cheat sheet. Retrieved from <https://dockerlabs.collabnix.com/docker/cheatsheet/>