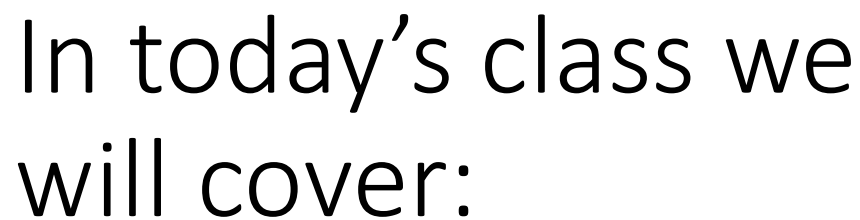


# Predictive Data Analysis

Lecturer: Marina Iantorno

E-mail: [miantorno@cct.ie](mailto:miantorno@cct.ie)





- ☐ Downloading Anaconda
- ☐ Getting started in Jupyter Notebook
- ☐ Data Preparation

# Introduction to Machine Learning in Python




---

## GETTING STARTED

# Introduction to Machine Learning in Python

The first thing we have to do is to install a platform in our computer. It is called “Anaconda”, and it is great to have it because it displays different tools that you may use in your Data Science path. Just follow the link below and pick the option that matches your device operator:

<https://www.anaconda.com/products/individual>

Windows 	MacOS 	Linux 
Python 3.8 64-Bit Graphical Installer (477 MB) 32-Bit Graphical Installer (409 MB)	Python 3.8 64-Bit Graphical Installer (440 MB) 64-Bit Command Line Installer (433 MB)	Python 3.8 64-Bit (x86) Installer (544 MB) 64-Bit (Power8 and Power9) Installer (285 MB) 64-Bit (AWS Graviton2 / ARM64) Installer (413 M) 64-bit (Linux on IBM Z & LinuxONE) Installer (292 M)

# Introduction to Machine Learning in Python

Now that we downloaded the software, let's trace our next steps.

1. Libraries
2. Import Datasets
3. Take care of missing data
4. Encode data
5. Splitting dataset into the Training Set and Test Set

You will find on Moodle a file called "Data.csv". Please download it because we will use it as an example.

# Introduction to Machine Learning in Python

---

LIBRARIES



# Introduction to Machine Learning in Python

## Libraries

There are 3 libraries that we will always use for Machine Learning models:

➤ Numpy: “Numerical Python”

*Code: `import numpy as np`*

**Functionality:** Provides support for complex mathematical algorithms such as multidimensional arrays and matrices. Numpy gives structure to the data and makes the calculations efficient.

We could consider Numpy as the primary library for data science and it is essential when we work with Machine Learning algorithms.

# Introduction to Machine Learning in Python

## Libraries

There are 3 libraries that we will always use for Machine Learning models:

### ➤ Matplotlib

*Code: `import matplotlib.pyplot as plt`*

**Functionality:** This is a plotting library and a numerical mathematic extension. It allows us to add different features to our plots, such as create figures, plot lines on certain areas, make an interactive visualization, among other things.



# Introduction to Machine Learning in Python

## Libraries

There are 3 libraries that we will always use for Machine Learning models:

### ➤ Pandas

*Code: import pandas as pd*

**Functionality:** This library helps us with the analysis of data structure. Pandas has 3 structures of data:

One dimension (series), two dimensions (data frame or tables), three dimensions (cubes). We need this library to read the datasets with the correct structure in Python.

# Introduction to Machine Learning in Python

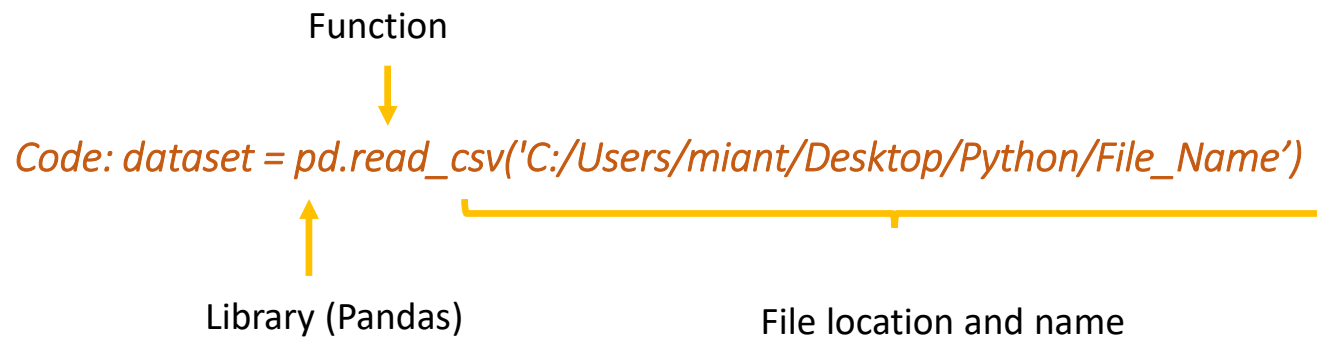
---

## IMPORTING DATASET

# Introduction to Machine Learning in Python

## Import Dataset

Now it is time to import the dataset that contains our data. Usually, the name of the dataset is dataset, data\_frame, df. We try to simplify our code as much as possible. Let's analyse the code



# Introduction to Machine Learning in Python

## Import Dataset

Something we need to know about the dataset, is that it will have a range of variables (columns) and observations (rows).

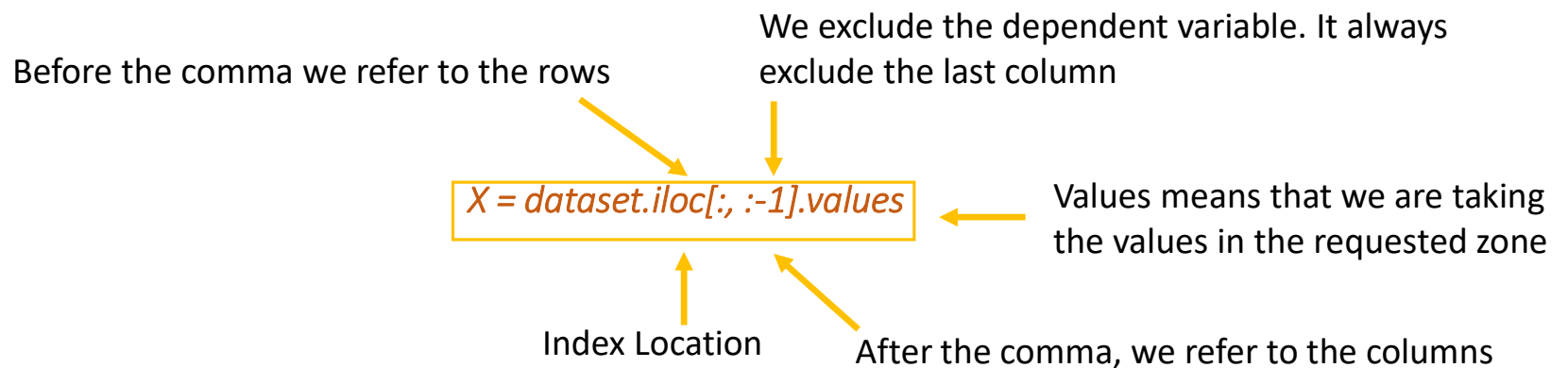
Features	X1	X2	Y (Dependent variable)
A	100	10	Yes
B	200	20	No
C	300	30	Yes
...	...	...	...

With the features and independent variables (X1, X2, ..., Xn), we will predict the result of the dependent variable.

# Introduction to Machine Learning in Python

## Import Dataset

Now, let's go back to understand the code. We need to determine the variables of our dataset. We will start with X. Following the previous table, we need to take the first 3 columns, or, in other words, we need all the columns of the dataset except the last one.

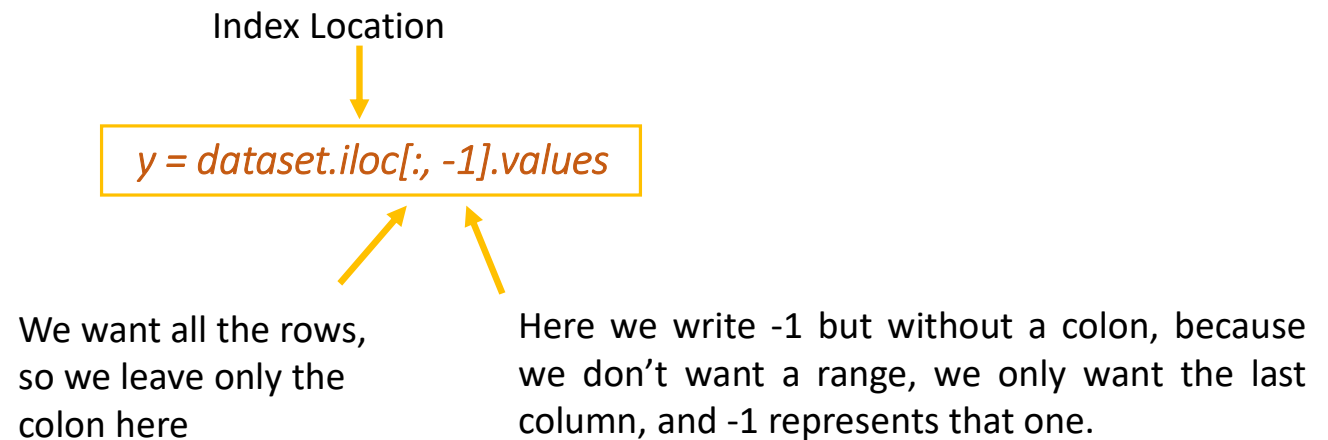


*To sum up: iloc is a Pandas function and stands for Index Location. Within the brackets we use a colon to mark a range and a comma to split between rows and columns. In the columns, we keep all the columns before the colon and exclude the last one with -1, that is the reference for the last column.*

# Introduction to Machine Learning in Python

## Import Dataset

Let's work with the dependent variable (Y) now. We will use a similar structure, but the difference will be in the range of values.



To sum up: We select again the full range for the rows, and we do not use the colon in the columns space, we only use -1 to name the column that we need, which is the dependent variable.

# Introduction to Machine Learning in Python

## Import Dataset

Till here, we covered:

- Import dataset
- Create a matrix of features (X)
- Create a dependent variable vector.

Code:

```
dataset = pd.read_csv('C:/Users/miant/Desktop/Python/File_Name')
```

```
X = dataset.iloc[:, :-1].values
```

```
y = dataset.iloc[:, -1].values
```

# Introduction to Machine Learning in Python

---

TAKING CARE OF MISSING DATA



# Introduction to Machine Learning in Python

On some occasions, the datasets could present some missing values. We have two ways to sort this issue:

1. Deleting the rows/column
2. Populating the empty cells

Both options have their positive and negative aspects. Let's explore a bit more.

# Introduction to Machine Learning in Python

## 1. Deleting the rows

This is probably the first option that comes to our mind, and it is not bad if we are working with a large dataset, and it will not affect our final figure. We could say that we can delete up to 1% of the data to avoid leading to wrong conclusions in our predictions. To do this in Python, the code is straight forward.

*Code: `dataset.dropna`*

With this code, we are indicating the software to drop all the rows that contain a missing value.

# Introduction to Machine Learning in Python

## 1. Deleting the rows

This is probably the first option that comes to our mind, and it is not bad if we are working with a large dataset, and it will not affect our final figure. We could say that we can delete up to 1% of the data to avoid leading to wrong conclusions in our predictions. To do this in Python, we need to implement a method:

*Code: `dataset.dropna`*

With this code, we are indicating the software to drop all the rows that contain a missing value.

# Introduction to Machine Learning in Python

## 1. Deleting the columns

Sometimes we have an entire column with missing values, and we want to remove it from our dataset. In that case, we need to follow the code below:

*Code: `dataset.dropna(axis = "columns")`*

With this code, we are indicating the software to drop all the columns that contain a missing value.

*\*Note: Instead of using "columns" you can use "1", because in Python 0 means rows and 1 means columns.*

# Introduction to Machine Learning in Python

## 2. Populating the empty cells

When we talk about populating the empty cells, we mean to use an alternative value to fill that blank space. We have two options: replacing the missing values by the average or by the median.

Here we will use one of the most important libraries in Machine Learning: Scikit-learn. We will use this library many times during the module. The short cut of this library in Python is “sklearn”.

# Introduction to Machine Learning in Python

## 2. Populating the empty cells

The most common alternative value is the average (in Statistics called “mean”), but it is important to take into consideration that to calculate the average we need all the values of the column, therefore, if there are many missing values the best option would be the median. Let’s see an example replacing the data with the average.

*Code:*

```
from sklearn.impute import SimpleImputer
```

Here we define our strategy indicating  
what values will replace the missing data

```
imputer = SimpleImpute(missing_values=np.nan, strategy='mean')
```

```
imputer.fit(X[:, 1:3])
```

The “fit” method will connect the object imputer to the matrix of features

I include the columns that contain numerical values

```
X[:, 1:3] = imputer.transform(X[:, 1:3])
```

# Introduction to Machine Learning in Python

## 2. Populating the empty cells

```
In [12]: print(X)
```

```
[['France' 44.0 72000.0]
 ['Spain' 27.0 48000.0]
 ['Germany' 30.0 54000.0]
 ['Spain' 38.0 61000.0]
 ['Germany' 40.0 63777.77777777778]
 ['France' 35.0 58000.0]
 ['Spain' 38.77777777777778 52000.0]
 ['France' 48.0 79000.0]
 ['Germany' 50.0 83000.0]
 ['France' 37.0 67000.0]]
```

Here we can see the missing value replaced by the average

# Introduction to Machine Learning in Python

---

## ENCODING CATEGORICAL DATA



# Introduction to Machine Learning in Python

Data could be divided in two main categories:

- Categorical Data: This refers to everything that contains attributes of the variable (Countries, Names, Status, etc).
- Numerical Data: As it names clearly says, it refers to numbers.

When we code, and especially in Machine Learning, it is necessary to transform categorical data into numerical data. Using our original example, we can start with the first column.

# Introduction to Machine Learning in Python

A	B	C	D
Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40		Yes
France	35	58000	Yes
Spain		52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes


The easiest way would be to assign numbers like 0, 1, 2, and so on, but careful! Our Machine Learning Model could understand that there is a kind of order between the countries and that the order matters, but this is not the case. So to help our ML Model to understand it, we need to encode the data. In other words, we will assign a code to each value of the column, for the model to use it to predict data.

# Introduction to Machine Learning in Python


This process is known as “One Hot Encoder”, and it will assign different codes to each of our categorical variables, or we could say that we will transform the columns. Let’s see:

*Code:*

```
from sklearn.compose import ColumnTransformer  
from sklearn.preprocessing import OneHotEncoder  
ct = ColumnTransformer(transformers=[("encoder", OneHotEncoder(), [0])], remainder="passthrough")
```



This is the column that  
we want to transform



This will ensure that the countries take  
the values of the other variables as they  
are

```
X = np.array(ct.fit_transform(X))
```

# Introduction to Machine Learning in Python

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[("encoder", OneHotEncoder(), [0])], remainder="passthrough")
X = np.array(ct.fit_transform(X))
print(X)
```

```
[1.0 0.0 0.0 44.0 72000.0]
[0.0 0.0 1.0 27.0 48000.0]
[0.0 1.0 0.0 30.0 54000.0]
[0.0 0.0 1.0 38.0 61000.0]
[0.0 1.0 0.0 40.0 63777.77777777778]
[1.0 0.0 0.0 35.0 58000.0]
[0.0 0.0 1.0 38.77777777777778 52000.0]
[1.0 0.0 0.0 48.0 79000.0]
[0.0 1.0 0.0 50.0 83000.0]
[1.0 0.0 0.0 37.0 67000.0]
```

Now, we can see that each country has a different code.

<b>France</b>	1	0	0
<b>Spain</b>	0	0	1
<b>Germany</b>	0	1	0

# Introduction to Machine Learning in Python

Now we need to encode the Dependent Variable, which is also categorical.

A	B	C	D
Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40		Yes
France	35	58000	Yes
Spain		52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes

In this case, we have a Binary outcome (“Yes” or “No”). When this is the case, it is common to assign the values 0 and 1. This process is known as “Label Encoding”.

# Introduction to Machine Learning in Python

We will use the sklearn library again. Let's see how it works:

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
y = le.fit_transform(y)
```

We transform the content in "y"  
into 0 and 1.

The dependent  
variable is  
always called "y"

# Introduction to Machine Learning in Python

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[("encoder", OneHotEncoder(), [0])], remainder="passthrough")
X = np.array(ct.fit_transform(X))
print(X)
```

```
[1.0 0.0 0.0 44.0 72000.0]
[0.0 0.0 1.0 27.0 48000.0]
[0.0 1.0 0.0 30.0 54000.0]
[0.0 0.0 1.0 38.0 61000.0]
[0.0 1.0 0.0 40.0 63777.77777777778]
[1.0 0.0 0.0 35.0 58000.0]
[0.0 0.0 1.0 38.77777777777778 52000.0]
[1.0 0.0 0.0 48.0 79000.0]
[0.0 1.0 0.0 50.0 83000.0]
[1.0 0.0 0.0 37.0 67000.0]
```

Now, we can see that each country has a different code.

<b>France</b>	1	0	0
<b>Spain</b>	0	0	1
<b>Germany</b>	0	1	0

# Introduction to Machine Learning in Python

```
In [7]: from sklearn.preprocessing import LabelEncoder  
le = LabelEncoder()  
y = le.fit_transform(y)
```

```
In [8]: print(y)  
[0 1 0 0 1 1 0 1 0 1]
```

Here is our output and we can see that “No” became 0 and “Yes” became 1



# Introduction to Machine Learning in Python

---

SPLITTING DATA INTO TRAINING  
SET AND TEST SET

# Introduction to Machine Learning in Python

This is the last step of our Data Preprocessing. In Machine Learning we will make two different sets.

- **Training Set:** In this set we will train our Machine Learning Model on existing observations.
- **Test Set:** In this set we will evaluate the performance of our model in new observations.

Note: After splitting the data, in some models we will need to scale all our variables to ensure that all of them are taking values in the same scale. We will use see it in some models during this module.

# Introduction to Machine Learning in Python

To split the data into the Train and Test set, we will need to have:

X Train Set

X Test Set

y Train Set


y Test Set

# Introduction to Machine Learning in Python

We cannot divide our data in equal parts, we will need many observations in our Training set, and a few in our Test set. Usually, the proportion is 20% Test Set and 80% Train Set.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state= 1)
```



We indicate here  
that 20% will be  
Test Set

# Introduction to Machine Learning in Python

---

HANDS ON!

# Introduction to Machine Learning in Python

You will find on Moodle a file called “Data\_Prep\_Exercise.csv”, which shows a survey done to people from different nationalities who were following a diet.

Follow the steps we previously covered and define the X matrices and the dependent variable.

Prepare the data to start working with it.

**THAT'S ALL FOR TODAY**

**THANK YOU**

