Artificial Intelligence For Physics

M2 OPHO, MAX

PHY2446M

2023-2024

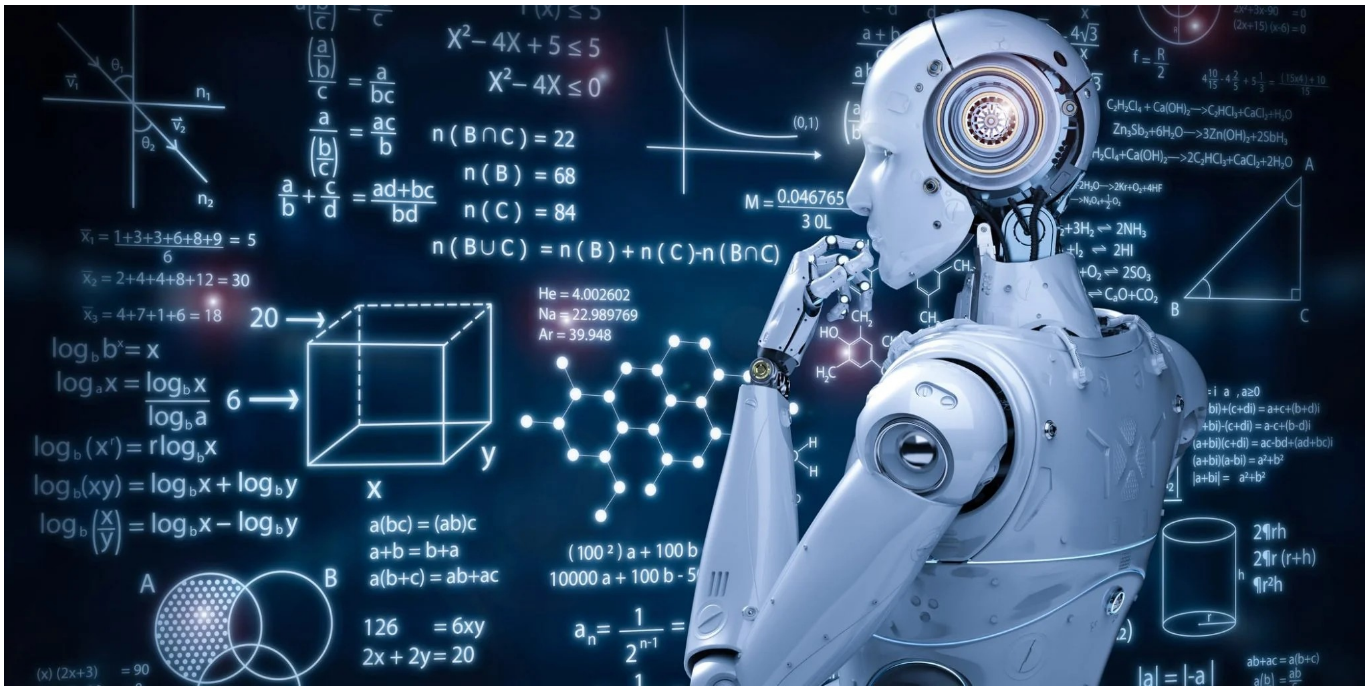
The exercises in green are optional.

The Projects, in red, are graded. You must return the code of each project one week after the practical work where we begin its development

All necessary data available in **moodle.univ-lyon1.fr**

On **jupyter.univ-lyon1.fr**, use this path :

/home/pers/allouche/PHY2446M/Data/nameOfFile



Contents

1. Python (~1h30)

1.1 Basic (~ 40 mn)

1) Write a Python Code which accepts the radius of a circle and computes its area.

Sample Output :

r = ...

Area = ...

NB : used input ("Input the radius of the circle : ") to ask a value for r

2) Write a Python Code to display the first and last colors from the following list.
color_list = ["Red","Green","White" ,"Black"]

3) Write a Python Code to print the documents (syntax, description etc.) of Python abs function.

4) Write a Python Code to count the number 4 in a given list.

5) Write a Python Code to remove the first item from a specified list.

6) Write a Python function that takes a positive value and returns the sum of the cube of all the positive values smaller the specified value.

Ex.: 8 : $7^3+6^3+5^3+4^3+3^3+2^3+1^3 = 784$

7) Write a Python Code to compute the sum of first n given prime numbers.

8) Write a Python function to find the maximum and minimum numbers in a list.

Note: Do not use built-in functions.

9) Write a Python Code to remove duplicates from a list. Note : use for and append.

10) Write a Python Code to split a given list into two parts where the length of the first part of the list is given. Note : Do not use any loop.

11) Write a Python Code to find the item with maximum occurrences in a given list. You can use count for an element of a list. You can use also collections.Counter and c.most_common()[0][0]

Original list: [2, 3, 8, 4, 7, 9, 8, 2, 6, 5, 1, 6, 1, 2, 3, 4, 6, 9, 1, 2]
Item with maximum occurrences of the said list: 2

12) Write a Python script to concatenate following dictionaries to create a new one

Sample Dictionary :

dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

13) Write a Python Code to create a dictionary of keys x, y, and z where each key has as value a list from 11-20, 21-30, and 31-40 respectively. Access the fifth value of each key from the dictionary.

Note : use list and range

14) Write a Python Code to find the key of the maximum value in a dictionary.

Sample Output:
Original dictionary elements: {'Jean': 19, 'Claire': 22, 'Mathew': 21, 'Pierre': 20}
Finds the key of the maximum and minimum value of this dictionary: ('Claire', 'Jean')

1.2 Functions (~30 mn)

1) Write a Python function that takes a positive integer and returns the sum of the cube of all the positive integers smaller than the specified number.

2) Write a Python Code to find the key of the maximum value in a dictionary.
Sample dictionary: {'Jean': 19, 'Claire': 22, 'Mathew': 21, 'Pierre': 20}
Expected Output : 'Claire'

3) Write a Python function to multiply all the numbers in a list.
Sample List : [8, 2, 3, -1, 7]
Expected Output : -336

4) Write a Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number as an argument.

5) Write a Python function that takes a list and returns a new list with unique elements of the first list.
Sample List : [1,2,3,3,3,3,4,5]
Unique List : [1, 2, 3, 4, 5]

6) Write a Python function that takes a number as a parameter and check the number is prime or not.

7) Write a Python function to create and print a list where the values are square of numbers between 1 and 30 (both included)

8) Write a Python Code to execute a string containing Python code. Note : use exec

1.2 File Input Output (~20 mn)

1) Write a Python Code to read first n lines of a file. Use readlines

2) Write a Python Code to read a file line by line and store it into a list.

3) Write a Python Code to read a random line from a file. Note : use random.choice

4) Write a Python Code to write a list to a file.

2. NumPy (~30 mn)

2.1 Basic (~ 25 mn)

1) Write a NumPy code to convert a list of numeric values into a 1-D NumPy array.

Expected Output:

Original List: [12.23, 13.32, 100, 36.32]

One-dimensional NumPy array: [12.23 13.32 100. 36.32]

2) Write a NumPy code to create a null vector of 20 elements and update forth value to 12.

3) Write a NumPy code to create 1-D array containing values from 10 to 48

4) Write a NumPy code to find the indices of the maximum and minimum values. (use argmax or argmin).

Original array: [3 8 4 1 9 6]

Index of Maximum Values: 4

Index Minimum Values: 3

5) Write a NumPy code to get the values and indices of the elements bigger than 10 in a 1-D array. Note : Do not use a loop

6) Write a NumPy code to save a NumPy array to a text file. Note use savetxt

7) Write a NumPy code to create an array of ones and an array of zeros. Note use zeros and ones

8) Write a NumPy code to change a 1D array on 2D array.

9) Write a NumPy code to create a contiguous flattened array (convert 2D in 1D)

10) Write a NumPy code to collapse a 3-D array into 1-D one.

Expected Output:

3-D array:

[[2,17], [45, 78]],

[[88, 92], [60, 76]],

[[76,33] , [20,18]]])

One dimension array:

[2 17 45 78 88 92 60 76 76 33 20 18]

11) Write a NumPy code to replace the negative values in a NumPy array with 0. Note : Dot not use loop

12) Write a NumPy code to remove all rows in a NumPy array containing nan values

Note : use np.isnan and any(axis=1), and ~ operator. ***Do not use loop.***

```
Expected Output:
Original array:
    [[ 3. 1. 2.]
    [ 9. 3.  nan]
    [ 8. 7. 1.]
    [ 2. 1. 2.]]
Removed all  nan elements :
    [[ 3. 1. 2.]
    [ 8. 7. 1.]
    [ 2. 1. 2.]]
```

13) Write a NumPy code to count the frequency of unique values in NumPy array. ***Note use np.unique, not a loop***

14) Write a NumPy code to get the index of a maximum element in a NumPy array along one axis

15) Write a NumPy code to stack 1-D arrays as columns. Note : not loop, use column_stack

```
Original arrays:
    Array-1 [11 12 13]
    Array-2 [12 13 14]
Stack 1-D arrays as columns:
    [[11 12]
     [12 13]
     [13 14]]
```

16) Write a NumPy code to calculate averages along a given array.

```
    Original array:
    [[10. 20. 30.]
    [40. 50. 0]
    [0    6.   0]
    [0    0    0]]
Averages along the said array:
    [20.  45.   6.  0]
```

17) Write a NumPy code to create a random 8x8x3 array and extract any array of shape (3,3,2) from it. Note use np.random.random_sample.

18) Write a NumPy code to remove a column from an array. Note : Use np.delete

2.2 Statistics (~ 5 mn)

1) Write a NumPy code to calculate the difference between the maximum and the minimum values of a given array along the second axis.

2) Write a NumPy code to compute the mean, standard deviation, and variance of a given array along the second axis. Note : No loop. Use mean, average, std, ….

```
Sample output:
Original array: [0 1 2 3 4 5]
Mean: 2.5
std: 1
variance: 2.9166666666666665
```

3) Write a NumPy code to compute the covariance matrix of two given arrays. ***Note : no loop***

4) Write a NumPy code to compute the histogram of nums against the bins. ***Use np.histogram***

```
Sample Output:
nums: [0.5 0.7 1. 1.2 1.3 2.1]
bins: [0 1 2 3]
Result: (array([2, 3, 1], dtype=int64), array([0, 1, 2, 3]))
```

3. Pandas (~2h00)

3.1 Getting & Knowing your data.

Write a Python Code to read occupations.csv file and analyze corresponding data :

- Import the necessary libraries
- Read occupations.csv
- See the first 25 entries
- See the last 10 entries
- What is the number of observations in the dataset?
- Print the name of all the columns.
- How is the dataset indexed?
- What is the data type of each column?
- Print only the occupation column
- How many different occupations are in this dataset?
- What is the most frequent occupation?

- Summarize the DataFrame.
- Summarize all the columns
- Summarize only the occupation column
- What is the mean age of users?

3.2 Filtering & Sorting

Write a Python Code to read , filter and sort corresponding data :

- Import the necessary libraries
- Read Euro_2012_stats_TEAM.csv. The dataFrame will be named euro12.
- Show only the Goal column.
- What is the number of teams participated in the Euro2012?
- How many columns in the dataset?
- Show the columns : Team, "Yellow Cards" and "Red Cards". Assign them to a dataframe called discipline.
- Sort the teams by "Red Cards", then to "Yellow Cards", in discipline dataframe
- Compute the mean "Yellow Cards" given by Team
- Filter teams scored with more 6 goals, using euro12 dataFrame.
- Show the teams that start with G.
- Show the first 7 columns.
- Show all columns except the last 3.

3.3 Grouping

Write a Python Code to read , filter and sort corresponding data :

- Import the necessary libraries & read data from occupations.csv file.
- Show mean age per occupation
- For each occupation, calculate the minimum and maximum ages
- For each combination of occupation and gender, calculate the mean age
- For each occupation present the percentage of women and men

3.4 Apply

Write a Python Code to read, filter and sort corresponding data:

- Import the necessary libraries & read data from student-mat.csv file. df is the name of dataframe.
- For the purpose of this exercise slice the dataframe from 'school' until the 'guardian' column. The name of the new datafarme is studAlc
- Create a function capitalizer, with x as argument and return x.capitalize()
- Capitalize both Mjob and Fjob, using .apply and capitalizer
- Print the last elements of the data set studAlc.
- You notice that the original dataframe is still lowercase?
  Why is that? Fix it and capitalize Mjob and Fjob.
- Create a function called majority that returns True if age >17 & False if not.
- Create in studAlc a new column called legal_drinker (Consider majority as older than 17 years old)
- Create a finction timesTen with x as argument and return 10*x if x is a real or a int. That returns x if not.
- Multiply every number of the dataset by 10. (use applymap & timesTen)

3.5 Merge

Write a Python Code to merge dataFrame as is :

- Create the 3 DataFrames (named data1, data2 and data3) based on the following raw data :

```
raw_data_1 = {
            'subject_id': ['1', '2', '3', '4', '5'],
'first_name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
'last_name': ['Anderson', 'Ackerman', 'Ali', 'Aoni', 'Atiches']}
raw_data_2 = {'subject_id': ['4', '5', '6', '7', '8'],
    'first_name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
    'last_name': ['Bonder', 'Black', 'Balwner', 'Brice', 'Btisan']
}
raw_data_3 = {
    'subject_id': ['1', '2', '3', '4', '5', '7', '8', '9', '10', '11'],
```

'test_id': [51, 15, 15, 61, 16, 14, 15, 1, 61, 16]

}

- Join data1 & data2 along rows and assign all_data (use concat)
- Join data1 & data2 along columns and assign to all_data_col (concat & axes=1)
- Print data3
- Merge all_data and data3 along the subject_id value (use merge)
- Merge only the data that has the same 'subject_id' on both data1 and data2 (how='inner')
- Merge all values in data1 and data2, with matching records from both sides where available. (how='outer')

3.6 Time Series

Write a Python Code to manage time series in a dataFrame as is :

- Import the necessary libraries, read apple_1980_2014.csv file. The corresonding dataframe will named apple.
- Check out the type of the columns
- Transform the Date column as a datetime type (use pd.to_datetime)
- Set the date as the index
- Is there any duplicate dates? (use is_unique)
- Make the first entry the oldest date. (use sort_index)
- Get the last business day of each month (use resample('BM'), BM for business month end frequency)
- What is the difference in days between the first day and the oldest (use index.min , index.max and days)
- How many months in the data we have?

3.7 Visualization

Write a Python Code to visualize data:

- Import the necessary libraries (mathplotlib as plt, pandas as pd, …) , read recent-grads.csv file. The datafarme must be named df.
- Using plt, plot df["P75th"] along df["Rank"]
- Plot df["P75th"] along df["Rank"] using df.plot
- plot columns "P25th", "Median", "P75th" along "Rank", using df.plot

- Plot the histogram of “Median” Columns.
- Create a new dataframe using the top 5 of df after sorting by “Median” values (use df.sort_values(...) and .head)
- Create a bar plot using “Major” as x and “Median” as y.
- Create a new dataframe named top_medians, with the “Median” > 60000 and sort them.
- Create a bar plot using “Major” as x and “Median”, P25th and P75th as y.
- Create a scatter plot for "Unemployment_rate" along “Median”
- Add cat_totals = df.groupby("Major_category") ["Total"].sum().sort_values() and Create a barh (horizental bar) plot of cat_totals
- Create a *pie* plot for cat_totals

4. Machine Learning Exercises using standard methods (~5h)

4.1 Regression (~1h)

1) Read linear-regression-dataset.csv and do a scatter plot salary along the experience.

- Using LinearRegression of sklearn to build a linear regression fit.
- Show score and correlation.
- Make a scater plot of data and a linear plot of the predicted values

2) Read multiple-linear-regression-dataset.csv file.

- Make a scatter plot of data (x=experience, y=age, c=salary, cmap="plasma")
- Using LinearRegression of sklearn to build a linear regression fit to predict salary along experience and age.
- Show intercept and coef parameters.
- Compute and show salaries for (3,25), (10,25).

3) Read polynomial-regression.csv file.

- Make a scatter plot of data (x=car_price, y=car_max_speed)
- Using PolynomialFeatures and LinearRegression of sklearn to transform the system in 4 variables.
- Build a linear regression fit to predict car_price along car_max_speed.
- Plot predicted curve (line) and data values (scatter) of car_price along car_max_speed.

- Show the score of the fit.

4) Read salaries.csv file.

- Make a scater plot of data (x=”Education Level”, y=”Salary”)
- Using StandardScaler of sklearn, scale x and y.
- Using SVR, make a fit, show score, and plot data valuers (scatter) and predicted curve (line) on the same figure.

5) Read decision-tree-regression-dataset.csv file (with header=None). Using DecisionTreeRegressor of sklearn, Make a fit, show score, and plot data values (scatter) and predicted curve (line) on the same figure.

6) Read random-forest-regression-dataset.csv file (with header=None). Using RandomForestRegressor of sklearn, make a fit, show score, and plot data values (scatter) and predicted curve (line) on the same figure.

4.2 Classification (~30 mn)

1) Read class_data.csv as a dataFrame named data. class_data.csv is Breast cancer Wisconsin (diagnostic) dataset. - Print its .info().

- Remove “id” and "Unnamed: 32" columns.
- Make a new dataFrame, M (M For Malignant) containing lines which diagnostic ==”M”.
- Make a new dataFrame, B (B For Benign) containing lines which diagnostic ==”B”.
- Do a scatter plot for the 2 dataset (M and B) of texture_mean along radius_mean.
- Create y as a copy of diagnosis column(change M in 0 and B in 1). All others columns will be used (x_data) as variables.
- Normalize x_data.
- Split the data in 2 parts using train_testsplit of sklearn.model_selection (test_size=0.3).
- Use KneighborsClassifier with n_neighbors = 3 to make fit. Show the score.
- Make the same for n_neighbors varying from 2 to 14.
- Draw score (accuracy) along the n_neighbors values. What is the best value of n_neighbors?
- Make a prediction with the best value of n_neighbors.
- Using confusion_matrix of sklearn.metrics to plot confusion matrix. Seaborn.heatmap is recommended.

NB : You can do the same study using others classifiers as : SVC, Decision Tree, Random Forest, ...

4.3 Hyperparameter Tuning (~20 mn)

Read class_data.csv as a dataFrame named data.

Remove “id” and "Unnamed: 32" columns. Create x, y and normalize (see 4.2) .

Import GridSearchCV (from sklearn.model_selection) and LogisticRegression from sklearn linear_model. We will search the best parameter C and the best penality (l1 or l2) using GridSearchCV. Fro this :

- Create a list penalty (= ['l1', 'l2']) and a list np.logspace(0, 4, 10) for C.
- Make a dict with the 2 lists : hyperparameters = dict(C=C, penalty=penalty)
- Create the model clf = GridSearchCV(logistic, hyperparameters, cv=5, verbose=0)
- Fit the parameters to predict y using x : best_model = clf.fit(x, y)
- Print the parameters of best model using best_model.best_estimator_.get_params()
- Plot confusion matrix.

4.4 Clustering (~30 mn)

1) Read cluster_1.csv file.

- Scatter y along x using color columns as colors.
- Using KMeans for n_clusters varying from 1 to 10, fit data to obtain the clusters and corresponding inertia.
- Plot inertia along n_clusters.
- Choose the best value of n_clusters. Using this value of n_clusters, make a fit.
- Scatter y along x using the clusters numbers as color. Show also at the same figure the center of clusters (.cluster_centers_) with red color.

2) Do the same study for cluster_2.csv. Conclusion?

3) Do the same study for cluster_1.csv but using DBSCAN method. Use clusters.labels_ for colors. Try with several values of eps (from 1 to 20). Print the number of clusters and the number of anomaly points. Note : clusters_labels_ = -1 for anomaly points.

4) Do the same study for cluster_2.csv with DBSCAN.

4.5 Dimensional reduction/Visualization (~30 mn)

read data in roboBohr.csv. The data provided was originally downloaded from PubChem. Using atomic charges, a Coulomb Matrix was computed for every molecule according to [Rupp et al. PRL, 2012].

Columns ‘Eat’ contain atomic energies.

- Drop ['Unnamed: 0', 'pubchem_id', 'Eat'] columns to create X. Y will be the Eat column.
  X contains 1274 columns.
- Using StandardScaler and normalize from sklearn.preprocessing, apply a StandScalin and normalize X

We will search the best way to reduce the number of variables as small as possible.

- Using PCA, make a projection to 2D (PCA(n_components=2, random_state=100)). Measure the time to make the fit (you can use time.time() after importing time, or %%time in notebook). Plot X_pca using y to color the scatter.
- Using t-SNE, make a projection to 2D (TSNE(n_components=2, random_state=100)). Measure the time to make the fit Plot X_tsne using y as color for the scatter.
- Using PCA, change n_components from 2 to 15 and show “explain variance” (pca.explained_variance_ratio_.sum()) along n_components. What is the min value of n_components to obtain a variance > 0.95 ?
- Measure the time to make a PCA with this value of n_components.
- Make a fit with pca using n_components=0.95 as parameter., and show pca_n_components_. This is an other way to obtain the best value of number of components.

4.6 Project (~1h30)

You will develop a code to predict atomization energies of molecules. The dataset (roboBohr.csv), contains ground state energies of 16,242 molecules calculated by quantum mechanical simulations.

The data contains 1277 columns. The first 1275 columns are entries in the Coulomb matrix that act as molecular features. The 1276th column is the Pubchem Id where the molecular structures are obtained. The 1277th column is the atomization energy calculated by simulations using the Quantum Espresso package. The data is used for a publication in Journal of Chemical Physics.

You must analyze the data (histogram, variance, NaN values, non significative variables, …).

After that you will search the best model to predict Atomization energies using Coulum matrix.

You can use any (one or several) tools of **scikit-learn** : Regression, Scaling, normalization, train_test_split, GridSearchCV, … However, **you cannot use any other ML libraries (as Tensorflow, XGBoost,..)**. **The project is noted.** You can work alone or in pairs. So you have to provide **jupyter notebook** : code with comments. You will be helped during the practical work by the teacher. The code is to be provided one week later.

5. Deep Learning Using Tensorflow (~ 4h30)

5.1 Regression (~1h30)

5.1.1 Prediction of wine quality

The goal of this exercise is to explore a simple linear regression problem based on Portugese white wine.

The dataset is based on Cortez, A. et. al. Elsevier, 47(4):547-553, 2009.

Read winequality.csv file contains data on 4989 wines. For each wine 11 features are recorded (fixed acidity, volatile acidity, citric acid, residual sugar...). The final column contains the quality of the wine.

Build a simple neural network with at least on hidden layer. Compile it with *mse* as loss function and Adam as optimizer. For training set, use the first 3000 values (First 10 columns for x and last column for y). For testing set, use the rest of the data (from 3000 to end).

Fit it.

Plot train loss, test loss (using history["loss"][1:] and ["val_loss"][1:]).

Test it, using the test set and compare predicted values to true ones.

Try to improve your prediction by scaling features, changing of number of neurons, number of hidden layers, activation functions, BatchNormalization, dropout, number of epochs, …

Can PCA reduction of variables improve the prediction ?

Try RandomForestClassifier and compare its performance to that of DNN.

5.1.2 Prediction of atomization energy

Develop a code to predict atomization energies of molecules (see 4.6 Project) using a Dense neural network (**DNN**).

Compare the performance of **DNN** model to that of **RandomForestRegressor** one.

5.2 Neural Networks Classification (~1h30)

In 2021, G. YIN et al. Published (https://doi.org/10.1002/jrs.6080), in the journal J. Raman Spectroscopy an article claiming that COVID-19 can be detected from the Raman spectrum of serum using machine learning.

You will develop and apply several models for diagnosing Covid 19 based on a database constructed by these authors. Data are provided in RAMAN_DATA.xlsx and RAMAN_DATA.csv. The data is the same in both files. So you can use whichever you like.

Work to be carried out :

- Read the data. Check that there are no missing values, otherwise delete rows with missing data. In the 'diagnosis' column, replace 'Healthy' with 0 and 'SARS-CoV-2' with 1.
- Plot on the same graph a Raman spectrum where the diagnosis is positive and another IR spectrum where the diagnosis is negative.
- Is the problem one of regression or classification (answer on the Jupyter sheet)?
- Do any pre-processing you think is relevant, and divide the data into two parts: one for training and the other for testing.
- Implement a neural network model with, in addition to the input and output layers, at least one hidden layer and a Dropout method to try and avoid overfitting. Perform the fit. Then display the confusion matrix, accuracy score and f1-score for the data in the test section and then for the data in the section used for the fit. You can use SparseCategoricalCrossentropy for your fit, and tf.keras.layers.Softmax to compute the probabilities for making your classification.

  Is there any overfitting? What can we say about the quality of the prediction?

- For each of the following models: DecisionTree, AdaBoost and Bagging (with KNeighbors as estimator): Make a fit. Then display the confusion matrix, score (accuracy) and f1-score for the data from the test part and then those from the part used for training.
  Is there any overfitting? What can we say about the quality of the prediction?
- For the method: RandomForest perform an in-depth study using the "cross-validated grid-search" method and display the scores for the 2 parts of the data (test and training) as well as the optimal model parameters. Is there any overfitting? What can we say about the quality of the prediction?

5.3 Recurrent Neural Networks (~1h30)

You will develop a Recurrent Neural Networks, forecast weather based on input features such as temperature, humidity, Rainfall, Air Pressure and so on.

The dateset donnees-meteo-essentielles-LYS.csv, contains weather features at Lyon–Saint Exupéry Airport, collected every 10 minutes, beginning in 2010.

First, read onnees-synop-essentielles-omm-LYS.csv file. After that :

- Show all line with at least one “Nan” value.
- Using interpolate method of dataFrame to replace all Nan valuers by values interpolated from the other data : df.interpolate(inplace=True).
- Drop ["Date","Pression au niveau mer","Periode de mesure de la rafale"] columns from the data
- Take 80% of the data (dataset_train as a NumPy matrix) for training and 20% for testing (dataset_test as a NumPy matrix). To convert a dataframe to numpy, you can use: data.to_numpy().
- Using TimeseriesGenerator (to be imported from tensorflow.keras.preprocessing.sequence) to make a train_generator and test_generator, with sequency of 16 elements en batch_size = 32 and epochs = 5. For example, for training, make: train_generator = TimeseriesGenerator(dataset_train, dataset_train, length=sequence_len, batch_size=batch_size)
- Build a RNN model with :
  - InputLayer(input_shape=(sequence_len, features_len) where features_len = number of data columns.
  - keras.layers.LSTM(100, activation='relu'))
  - keras.layers.Dropout(0.2)
  - keras.layers.Dense(features_len)
- Define a callbaks as is : tf.keras.callbacks.ModelCheckpoint(filepath=”bet_model.h5”, verbose=0, save_best_only=True)
- Compile the model, using 'adam' as optimizer, ‘mse’ as loss and ‘mae’ as metrics

- Fit you model (add callbacks = [bestmodel_callback] as option)
- Using the datset_test compute 10 random, values for Temperature and compare to true values (given in column 11 of the dataset).

Try to improve the model, by normalization of data :

```
mean = dataset_train.mean()
std  = dataset_train.std()
dataset_train = (dataset_train - mean) / std
dataset_test  = (dataset_test  - mean) / std
```

Warning : After prediction, you must denormalize the data to obtain temperature in °C (s = s*std + mean). For this, you can use this function :

```
def denormalize(mean, std, seq):
    nseq = seq.copy()
    for i,s in enumerate(nseq):
        s = s*std + mean
        nseq[i]=s
    return nseq
```

5.4 Neural Networks Classification : Cancer detection

Skin cancer is the most common human malignancy, is primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by dermoscopic analysis, a biopsy and histopathological examination. Automated classification of skin lesions using images is a challenging task owing to the fine-grained variability in the appearance of skin lesions.

The dataset (See dataCancer directory) is taken from the ISIC (International Skin Image Collaboration) Archive. It consists of 1800 pictures of benign moles and 1497 pictures of malignant classified moles. The pictures have all been resized to low resolution (224x224x3) RGB. The goal of the project is to create a model, which can classify a mole visually into benign and malignant.

The data has 2 different classes of skin cancer which are listed below : Benign and Malignant.

In first step, load in the pictures. To do this, you can use : *image_dataset_from_directory function* to be imported from *tensorflow.keras.utils*. As the pictures have already been resized to 224x224, there's no need to resize them.

image_dataset_from_directory can split data in 2 subsets: training and validation (see subset, validation_split, shuffle seed, image_size and batch_size parameters). Set batch_size to 32 and image_size to (244, 244).

We will use images of dataCancer/train directory for training and validation.

Images of dataCancer/test directory will be used to test the model. To read images from test directory, use shuffle=False. After that execute this instruction:

*test_ds = test_ds.shuffle(buffer_size=8*batch_size, seed=42, reshuffle_each_iteration=**False**).*

test_ds is the name of data returned by image_dataset_from_directory for test directory.

If needed (for confusion matrix and classification_report of sklearn.metrics), you can get NumPy tables from data set returned by *image_dataset_from_directory*, using :

label = np.concatenate([y for x, y in data], axis=0)

x = np.concatenate([x for x, y in data], axis=0)

Show few random images.

5.4.1 Implementation of a DNN code

Building a DNN Model (with at least one hidden layer, eventually with drop ones).

Compile and fit the model. You can SparseCategoricalCrossentropy(from_logits=True) as loss, and "accuracy" as metrics.

Plot loss and accuracy for training and validation sets.

Test your model. Plot confusion matrix. Print classification_report.

5.4.3 Implementation of CNN code

Building a CNN Model, compile and fit it

Plot loss and accuracy for training and validation sets.

Test your model. Plot confusion matrix. Print classification_report.

5.4.4 Implementation of classification using one or severla standrd methods.

Building one or several standard classification Models (as LogisticRegression,

RandomForestClassifier,...).

Fit it (them).

Test your model. Plot confusion matrix. Print classification_report.