Term: M1-EE

Semester: 2

AY: 2025-2026

Abdelbacet Mhamdi

Dr.-Ing. in Electrical Engineering

Senior Lecturer at ISET Bizerte

abdelbacet.mhamdi@bizerte.r-iset.tn

Artificial Intelligence - Part 2

LAB MANUAL

INSTITUT SUPÉRIEUR DES ÉTUDES TECHNOLOGIQUES

Institute of Technological Studies of Bizerte

Honor code

The University of North Carolina at Chapel Hill
Department of Physics and Astronomy
`http://physics.unc.edu/undergraduate-program/labs/general-info/`

"During this course, you will be working with one or more partners with whom you may discuss any points concerning laboratory work. However, you must write your lab report, in your own words.

Lab reports that contain identical language are not acceptable, so do not copy your lab partner's writing.

If there is a problem with your data, include an explanation in your report. Recognition of a mistake and a well-reasoned explanation is more important than having high-quality data, and will be rewarded accordingly by your instructor. A lab report containing data that is inconsistent with the original data sheet will be considered a violation of the Honor Code.

Falsification of data or plagiarism of a report will result in prosecution of the offender(s) under the University Honor Code.

On your first lab report you must write out the entire honor pledge:

The work presented in this report is my own, and the data was obtained by my lab partner and me during the lab period.

On future reports, you may simply write *"Laboratory Honor Pledge"* and sign your name."

Contents

In order to activate the virtual environment and launch **Jupyter Notebook**, you need to proceed as follow

① Press simultaneously the keys CTRL ALT and T on the keyboard;

② Type `jlai2` in the console prompt line;

Terminal

```
student@isetbz:~$ jlai2
```

③ Finally hit the ↵ key.

KEEP THE SYSTEM CONSOLE OPEN.

▼ Remark 1

You should be able to utilize Julia from within the notebook through:

Jupyter Lab *at http://localhost:2468*

Pluto *at http://localhost:1234*

Submit your lab reports in **PDF** format. Be sure to review your files with your instructor before the lab session ends.

https://forms.gle/sHQgBigF6hFFYqD16

Please use one of the provided templates when preparing your lab assessments:

LaTeX `https://www.overleaf.com/read/pwgpyvcxcvym#9e34eb`

Typst `https://typst.app/universe/package/ailab-isetbz`

A. Mhamdi

1 Regression

Student's name	. . . .	. . . .	. . . .
Score **/20**	. . . .	. . . .	. . . .

Detailed Credits

Anticipation ***(4 points)***	. . . .	. . . .	. . . .
Management ***(2 points)***	. . . .	. . . .	. . . .
Testing ***(7 points)***	. . . .	. . . .	. . . .
Data Logging ***(3 points)***	. . . .	. . . .	. . . .
Interpretation ***(4 points)***	. . . .	. . . .	. . . .

In order to conduct the labs effectively, it is highly recommended to check the codes available at `https://github.com/a-mhamdi/jlai/` $\longrightarrow$ *Codes* $\longrightarrow$ *Julia* $\longrightarrow$ *Part-2* $\longrightarrow$ *Jupyter* $\longrightarrow$ *.*regression.*ipynb*

Linear regression is a simple and popular method for modeling relationships in data and is often used as a starting point for more complex ML algorithms. It is a type of regression analysis that models the relationship between a continuous outcome variable y, aka as dependent variable or target, and one or more predictor variables x, aka independent variable, or feature, by fitting a straight line to the data. This line can then be used to make predictions about the value y based on the values of x.

Hereafter is an example of how we might implement linear regression in **Julia**:

Julia Code

```julia
# Define the input data
X = [0 2; 1 1; -1 0.5; 1 5] # matrix of input data
y = [2; 3; 4; 5] # vector of output values
```

```julia

          #= NORMAL EQUATION =#

# Compute the coefficients using the normal equation
coefficients = (X' * X) \ X' * y
# Print the coefficients
println("Coefficients are $coefficients")

# Define some test input
x_test = [1 6]
# Compute the predicted output
y_pred = x_test * coefficients
# Print the predicted output
println("Predicted output is $y_pred")

          #= LOADING `LinearRegressor` FROM `MLJLinearModels` =#

# Import the required library
using MLJ

LR = @load LinearRegressor pkg=MLJLinearModels
lr_ = LR(fit_intercept=false)
# Bind an instance of `lr_` to training data
lr = machine(lr_, table(X), y) |> fit!
# Display the fitted parameters
println("Fitted parameters are $(fitted_params(lr))")

# Recall the same previously defined test input
x_test = [1 6]
# Compute the predicted output
y_hat = predict(lr, x_test)
# Print the predicted output
println("Predicted output is $y_hat")
```

▼ Remark 2

*In **MLJ** a model and training/validation data are typically bound together in a machine:*

```
julia> lr = machine(model, X, y)
```

'`lr`' stores learned parameters, among other things.

*If you want to avoid the warning that pops up at the REPL, it is more convenient to coerce to continuous the scitypes of data being fed to model in machine when using **MLJ** package. You can further check the* *documentation by typing:*

```
julia> @doc MLJLinearModels.LinearRegression
```

Task № 1:

x_1 and x_2 both are vectors of 10000 random values. x_1 is an array of float values sampled from a normal distribution with mean equal to $\mu = -3$ and a standard deviation fixed to $\sigma = 2.7$. The elements of x_2 are however integer values depicted from a uniform distribution ranging from -8 to 4.

a) Write **Julia** code to generate and plot histograms of x_1 and x_2

b) Standardize x_1 using the package **Distributions** at first, and then **MLJ**.

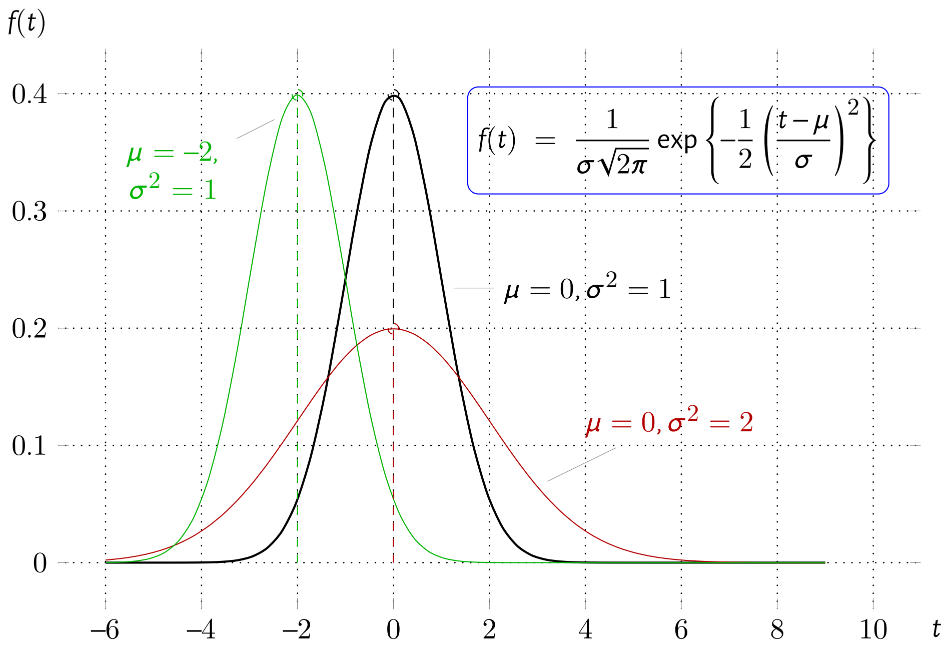
c) Normalize x_2

d) Say that we have a target $y = -x_1 + 3.5x_2$

- Generate the vector y;
- By applying normal equation, do you get $\hat{\theta} = \begin{bmatrix} -1 & 3.5 \end{bmatrix}^T$
- Using **MLJ** package, load a linear regression model, bind an instance of it to the data in $X = \text{hcat}(x_1, x_2)$. Compute $\hat{\theta}$ again.

▼ Remark 3

The graphs of some continuous univariate normal distributions are shown below:



KEY TERMS

Model refers to the mathematical formula or equation that is used to make predictions based on the data.

Coefficient represent the strength and direction of the relationship between a particular predictor variable and the predicted variable.

Residual is the difference between the actual and predicted outputs. It is used to measure the goodness of fit of the model.

2 | Classification

Student's name	. . . .	. . . .	. . . .
Score ***/20***	. . . .	. . . .	. . . .

Detailed Credits

Anticipation ***(4 points)***	. . . .	. . . .	. . . .
Management ***(2 points)***	. . . .	. . . .	. . . .
Testing ***(7 points)***	. . . .	. . . .	. . . .
Data Logging ***(3 points)***	. . . .	. . . .	. . . .
Interpretation ***(4 points)***	. . . .	. . . .	. . . .

In order to conduct the labs effectively, it is highly recommended to check the codes available at `https://github.com/a-mhamdi/jlai/` ⟶ *Codes* ⟶ *Julia* ⟶ *Part-2* ⟶ *Jupyter* ⟶ *{logistic-regression.ipynb, knn.ipynb & svc.ipynb}*

Logistic regression is a popular type of statistical model that is used to predict the likelihood of an event occurring. It is a type of regression analysis that is generally used when the dependent variable is binary, meaning it can only take on one of two values, such as 0 or 1.

The output of a k-nearest neighbors (k-**NN**) model is a class membership (*e.g.* "cat" or "dog"). To make a prediction for a new data point, the algorithm finds the closest data points in the training set (*i.e.* the "nearest neighbors") and assigns, through the majority vote of their outputs mechanism, as the prediction for the new data point. The number of nearest neighbors (k) is a hyperparameter that must be specified in advance.

k-**NN** is a simple and effective algorithm, but it can be computationally expensive and is not suitable for large datasets. It is also sensitive to the scale and distribution of the data.

Support vector machines (**SVM**s) are a powerful and flexible tool for solving a wide range of machine learning problems, and have been widely used in many different fields, including text classification, image classification, and bioinformatics.

Hereafter is an example of how we might implement these types of classifiers in **Julia**:

JULIA CODE

```julia
using MLJ

# Load the data
X, y = make_blobs(100, 2; centers=2, cluster_std=[2., -1.])

# Transform data into dataframe
using DataFrames
df = DataFrame(X)
df.y = y
first(df, 5)

# Draw the scatter plot of features.
using Plots
scatter(df.x1, df.x2; group=df.y)

# Load the classifiers `LR`, `KNN`, and `SVC`
LR  = @load LogisticClassifier pkg=MLJLinearModels
KNN = @load KNNClassifier pkg=NearestNeighborModels
SVC = @load SVC pkg=LIBSVM

# Construct the pipelines
lr_pipe_  = Standardizer() |> LR()
knn_pipe_ = Standardizer() |> KNN()
svc_pipe_ = Standardizer() |> SVC()

# Feed and fit training data to pipelines
lr_pipe  = machine(lr_pipe_, X, y)  |> fit!
knn_pipe = machine(knn_pipe_, X, y) |> fit!
svc_pipe = machine(svc_pipe_, X, y) |> fit!

# Make evaluations
evaluate!(lr_pipe, operation=predict_mode, measures=f1score)
evaluate!(knn_pipe, operation=predict_mode, measures=f1score)
evaluate!(svc_pipe, operation=predict, measures=f1score)
```

In this example, *X* is a table of 100 observations. It is formed by two predictors *(aka features)* and *y* is a vector of target labels *(aka class labels)*.

Task № 2:

Consider the Fisher's classic iris dataset. The measurements are from 3 different species of iris: *setosa*, *versicolor* and *virginica*. There are 50 examples of each species. There are 4 measurements for each example: *sepal length*, *sepal width*, *petal length* and *petal width*. The measurements are in centimeters.

a) Write a code that prepares a pipeline, a composite model which allows, at first, to standardize the features and then load the logistic classifier.

- Evaluate the performances of your pipeline using a cross-validation set. Provide the measures you know as input argument to the '`evaluate!`' function.
- Write a code that draws the confusion matrix as shown by the builtin function '`confusion_matrix`'.
- Given the results you got, write a code to compute the accuracy, precision, recall and f1-score.

b) Repeat the same steps for k-**NN** classifier

c) Apply the same process as before to a **SVM** classifier.

▼ Remark 4

*If you want to load the iris dataset using **MLJ**, you can simply type:*

```
julia> X, y = @load_iris
```

Key Terms

Model refers to the mathematical formula or equation that is used to make the most plausible predictions based on the input data.

Class designates a group or a category to which a data point belongs. For example, in a classification task to predict whether an email is spam or not, the classes would be "spam" and "not spam".

Probability of a data point belonging to a particular class is often used to make predictions. For example, if a model predicts that a given email has a 90% probability of being spam, it is likely to be classified as spam.

3 | Clustering

Student's name	. . . .	. . . .	. . . .
Score */20*	. . . .	. . . .	. . . .

Detailed Credits

Anticipation ***(4 points)***	. . . .	. . . .	. . . .
Management ***(2 points)***	. . . .	. . . .	. . . .
Testing ***(7 points)***	. . . .	. . . .	. . . .
Data Logging ***(3 points)***	. . . .	. . . .	. . . .
Interpretation ***(4 points)***	. . . .	. . . .	. . . .

In order to conduct the labs effectively, it is highly recommended to check the code available at `https://github.com/a-mhamdi/jlai/` ⟶ *Codes* ⟶ *Julia* ⟶ *Part-2* ⟶ *Jupyter* ⟶ *kmeans.ipynb*

K-**Means** clustering is a method of unsupervised learning in machine learning. It is used to divide a dataset into a specified number (K) of clusters, with each cluster containing data points that are similar to each other. The goal of the algorithm is to minimize the within-cluster sum of squares (*wcss*), which measures the similarity of the data points within each cluster.

To perform *K*-**Means** clustering, the algorithm first randomly selects K data points from the dataset and assigns them to be the centroids of the K clusters. It then computes the distance between each data point and each centroid, and assigns each data point to the cluster whose centroid is closest to it. The algorithm then updates the centroids of each cluster by taking the mean of all of the data points in the cluster. This process is repeated until the centroids of the clusters no longer change, or until a maximum number of iterations is reached.

Hereafter is an example of how we might implement the *K*-Means clustering algorithm in **Julia**. This implementation uses the `Clustering` package to calculate distances. It takes as input the features in X and the number of clusters K, and returns the cluster centers and the cluster assignments of each point.

JULIA CODE

```julia
# Import librairies
using CSV, DataFrames

# Load the dataset from CSV file
df = CSV.read("./Datasets/Mall_Customers.csv", DataFrame);

# Take a look @ data
first(df, 5)
income = df[!, 4];
ss = df[!, 5];

# Plots pkg
using Plots
scatter(income, ss, legend=false)

# Clustering pkg
using Clustering

# Features construction
X = hcat(ss, income);
typeof(X)
hat_clusters = kmeans(X', 5; display=:iter)

# Scatter plots
scatter(ss, income, marker_z=hat_clusters.assignments,
    color=:winter,
    legend=false)

scatter!(hat_clusters.centers[1,:]', hat_clusters.centers[2,:]',
    color=:black,
    labels=["#1" "#2" "#3" "#4" "#5"],
    legend=true)
```

Task № 3:

a) Generate Gaussian blobs with 1000 random 7-dimensional points.

b) We denote by X the matrix of features. Cluster X into 5 clusters using K-Means.

c) Verify the number of clusters.

d) Get the assignments of points to clusters.

e) Get the cluster sizes.

f) Get the clusters centers.

g) Use a dimensionality reduction technique like PCA, to transform your problem into a 3-dimensional space.

h) Plot your results with each point color mapped to the assigned cluster index.

▼ Remark 5

*It is possible to load the PCA object using **MLJ** this way:*

```
julia> PCA = @load PCA pkg=MultivariateStats
```

Key Terms

Cluster refers to a group of data points that are similar to one another.

Centroid of a cluster is the mean of all the data points in that cluster.

Distance measures are used to determine how similar or dissimilar two data points are. The distance between two points is often used to determine which points belong in the same cluster.

4 Project Assessment

You will have the opportunity to go into greater detail on a topic that was covered in class and that you are interested in learning more about in the final project. Providing you with a demanding yet attainable test that enables you to exhibit your machine learning expertise is the ultimate objective.

You have to provide all necessary resources, such as sample code, relevant datasets, as well as creating a set of slides to present your work. You are expected to demonstrate your understanding of the material covered throughout this course, as well as familiarizing yourselves with relevant programming languages and libraries. The final project is comprised of:

1. proposal;
2. report documenting your work, results and conclusions;
3. presentation;
4. source code *(You should share your project on* ***GitHub****.)*

Project proposal

It is about two pages long. It includes:

- Title
- Datasets *(If needed!)*
- Idea
- Software *(Not limited to what you have seen in class)*
- Related papers *(Include at least one relevant paper)*
- Teammate *(Teams of three to four students. You should highlight each partner's contribution)*

Project report

It is about ten pages long. It revolves around the following key takeaways:

- Context *(Input(s) and output(s))*
- Motivation *(Why?)*
- Previous work *(Literature review)*
- Flowchart of code, results and analysis
- Contribution parts *(Who did what?)*

Typesetting using LaTeX is a bonus. You can use **LyX** (https://www.lyx.org/) editor. A template is available at https://github.com/a-mhamdi/ailab-isetbz/tree/main/LyX. Here what your report might contain:

1. Provide a summary which gives a brief overview of the main points and conclusions of the report.
2. Use headings and subheadings to organize the main points and the relationships between the different sections.
3. Provide an outline or a list of topics that the report will cover. Including a table of contents can help to quickly and easily find specific sections of your report.
4. Use visuals: Including visual elements such as graphs, charts, and tables can help to communicate the content of a report more effectively. Visuals can help to convey complex information in a more accessible and intuitive way.

If you are using `Julia`, you can generate the documentation using the package **Documenter.jl**. It is a great way to create professional-looking material. It allows to easily write and organize documentation using a variety of markup languages, including **Markdown** and LaTeX, and provides a number of features to help create a polished and user-friendly documentation website.

I will assess your work based on the quality of your code and slides, as well as your ability to effectively explain and demonstrate your understanding of the topic. I will also consider the creativity and originality of your projects, and your ability to apply what you have learned to real-world situations. I also make myself available to answer any questions or provide feedback as you work on your projects.

The overall scope of this manual is to introduce **Artificial Intelligence (AI)** , through either some numerical simulations or hands-on training, to the students at **ISET Bizerte**.

The topics discussed in this manuscript are as follow:

① Data Preprocessing

cleaning; transformation; normalization

② Regression

model; coefficient; residual

③ Classification

model; class; probability

④ Clustering

cluster; centroid; distance

Julia; REPL; *Pluto*; *Fuzzy*; *MLJ*; *DataFrames*; regression; classification; clustering.