AY: 2024-2025
MIDTERM | **AI-ECUE221**
Apr. 2025

M1-S2: Dept. of Electrical Engineering
Teacher: A. Mhamdi
Time Limit: 1h

This document contains **7** pages numbered from **1/7** to **7/7**. As soon as it is handed over to you, make sure it is complete. The **3** tasks are independent and can be treated in the order that suits you.

The following rules apply:

❶ **A handwritten double-sided A4** sheet is permitted.

❷ **Any electronic material**, except basic calculator, is prohibited.

❸ **Mysterious or unsupported answers** will not receive full credit.

❹ **Round results** to the nearest thousandth *(i.e., third digit after the decimal point).*

❺ **Task N°3:** Each correct answer will grant a mark with no negative scoring.

Task N°1

⌛ 30mn | (8 points)

We consider the following dataset ($x_1 = 1$):

Table 1: Housing energy efficiency prediction.

House Area (x_2)	Insulation Thickness (x_3)	Window Quality (x_4)	Energy Consumption (y)
100	5	3	200
120	6	4	180
150	4	2	220
200	7	5	150
180	5	4	170

The output y can be expressed as a linear function of house characteristics *(i.e.,* $y = x^T \cdot \theta$*).* An estimate of the parameter vector θ is given by:

$$\hat{\theta} \approx [266.667, \text{-}0.076, 3.806, \text{-}25.520]^T$$

(a) (2 points) Predict energy consumption for a house with: Area=170, Window Quality=4, and Insulation Thickness=6.

$$
\begin{aligned}
\hat{y} &= \underbrace{[1,\ 170,\ 6,\ 4]}_{x^{T}} \cdot \underbrace{[266.667,\ \text{-}0.076,\ 3.806,\ \text{-}25.520]^{T}}_{\hat{\theta}} \\
&\approx 174.503
\end{aligned}
$$

(b) (6 points) Compute each of the following metrics:

$\boxed{\textbf{MAE}}$ $\boxed{\textbf{RMSE}}$ $\boxed{\textbf{MAPE}}$ $\boxed{\textbf{R-squared}}$

Given the value of $\hat{\theta}$, we can compute the predicted output:

$$\hat{y} = X \cdot \hat{\theta} \approx [201.537,\ 178.303,\ 219.451,\ 150.509,\ 169.937]^{T}$$

The error vector ε is:

$$\varepsilon = y - \hat{y} \approx [\text{-}1.537,\ 1.697,\ 0.549,\ \text{-}0.509,\ 0.063]^{T}$$

Mean Absolute Error (MAE)

$$
\begin{aligned}
\text{MAE} &= \frac{1}{5}\sum_{i=1}^{5} |\varepsilon_i| \\
&\approx 0.871
\end{aligned}
$$

Root Mean Squared Error (RMSE)

$$
\begin{aligned}
\text{RMSE} &= \sqrt{\frac{1}{5}\sum_{i=1}^{5} \varepsilon_i^2} \\
&\approx 1.078
\end{aligned}
$$

Mean Absolute Percentage Error (MAPE)

$$
\begin{aligned}
\text{MAPE} &= \frac{1}{5}\sum_{i=1}^{5} \left|\frac{\varepsilon_i}{y_i}\right| \cdot 100\% \\
&\approx 0.467\%
\end{aligned}
$$

R-squared

$$
\begin{aligned}
R^2 &= 1 - \frac{\sum_{i=1}^{5} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{5} (y_i - \bar{y})^2} \\
&\approx 0.998 \\
\bar{y} &= \frac{200 + 180 + 220 + 150 + 170}{5} = 184.0
\end{aligned}
$$

Institute of Technological Studies of Bizerte

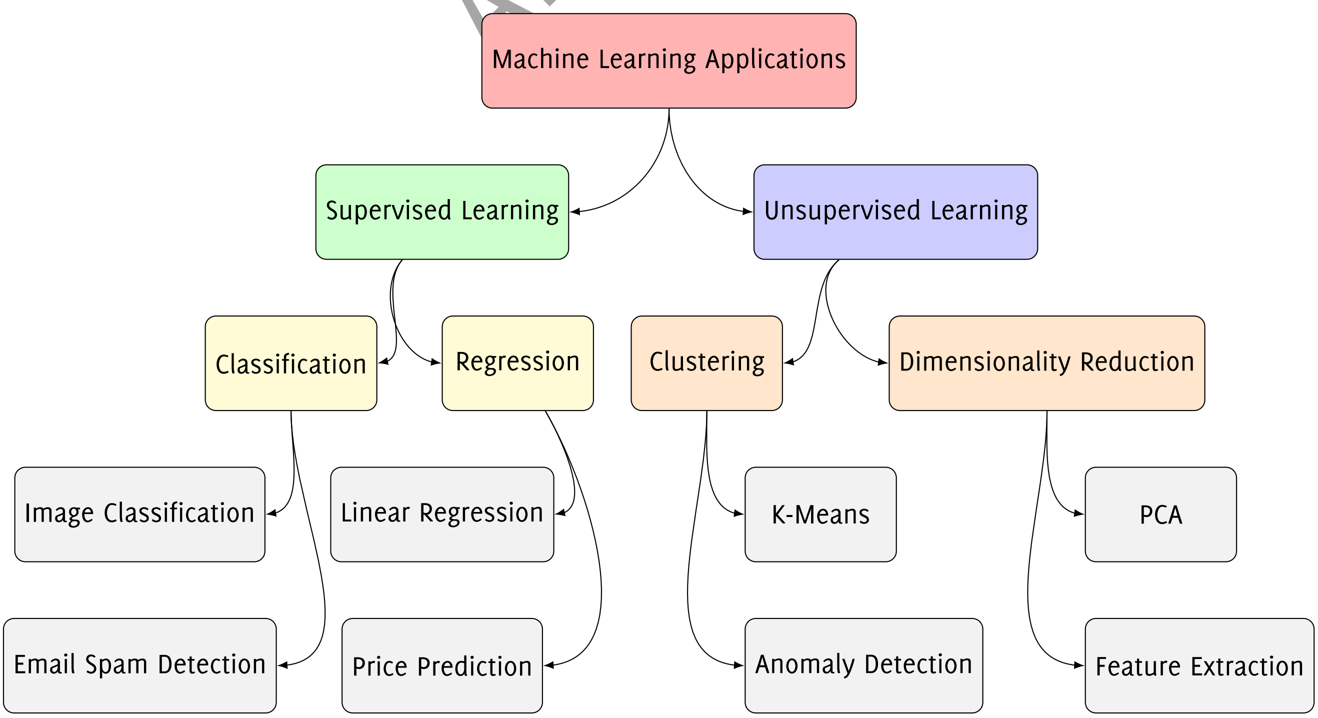
AY: 2024-2025		**Full Name:**	....
M1-S2: Dept. of Electrical Engineering		**ID:**	....
Midterm | **AI-ECUE221**		**Class:**	**RAIA1** ....
Apr. 2025		**Room:**	....
Teacher: A. Mhamdi		**Time Limit:**	**1h**

Answer Sheet

Task №2 ⌛ 15mn | ($3\frac{1}{2}$ points)

Categorize the following machine learning applications into their appropriate learning paradigms by placing each one in the correct branch:

Anomaly Detection	Feature Extraction	PCA
Classification	Image Classification	Email Spam Detection
Unsupervised Learning	Linear Regression	Price Prediction
Supervised Learning	Clustering	Dimensionality Reduction
K-Means	Regression	



Do not write anything here

✂- -

Task N°3

⌛ 15mn | ($8\frac{1}{2}$ points)

(a) ($\frac{1}{2}$ point) Which technique helps reduce overfitting in linear regression?

- ◯ Increasing model complexity
- ◯ Adding polynomial features
- √ Using regularization (L1/L2)
- ◯ Decreasing training data

(b) ($\frac{1}{2}$ point) The cost function for linear regression is typically:

- ◯ Mean Absolute Error (MAE)
- √ Mean Squared Error (MSE)
- ◯ Cross-Entropy Loss
- ◯ Hinge Loss

(c) ($\frac{1}{2}$ point) Gradient descent is primarily used for:

- ◯ Data visualization
- ◯ Feature engineering
- ◯ Data cleaning
- √ Optimizing model parameters

(d) ($\frac{1}{2}$ point) What is the purpose of splitting data into training and test sets?

- ◯ To increase model complexity
- √ To evaluate generalization performance
- ◯ To reduce feature dimensions
- ◯ To compute gradients

(e) ($\frac{1}{2}$ point) What is MLJ's primary approach to handling different data types in machine learning workflows?

- √ Uses Julia's native `Tables.jl` interface for data compatibility
- ◯ Only works with CSV files
- ◯ Requires manual type conversion
- ◯ Requires conversion to NumPy arrays

(f) ($\frac{1}{2}$ point) What is the recommended way to handle missing values in MLJ?

Do not write anything here

- ○ Only works with complete datasets
- ○ Automatically drops rows with missing values
- ○ Requires manual imputation before model fitting
- √ Provides built-in transformers like `FillImputer`

(g) (½ point) What is MLJ's recommended method for feature scaling?

- ○ No scaling support
- ○ Manual z-score calculation
- √ Standardizer transformer from MLJModels
- ○ External Python preprocessing

(h) (1 point) Which feature makes MLJ particularly suitable for reproducible data preprocessing?

- ○ Automatic hyperparameter optimization
- √ Composable, persistent pipeline blueprints
- ○ Exclusive use of deep learning models
- ○ Integration with JavaScript visualizations

(i) (1 point) What does this code do?

```julia
using DataFrames
df = DataFrame(A=1:5, B=11:15)
df = transform(df, :A => (x -> x.^2) => :A_squared)
```

- ○ Filters rows where A > 3
- ○ Replaces column A with its squared values
- √ Creates a new column A_squared containing squares of column A
- ○ Groups data by column A

(j) (1 point) What is the purpose of this code?

```julia
using DataFrames, MLJ
df = DataFrame(X=[1, missing, 3], Y=[5, 6, missing])
df = coalesce.(df, 0)
```

- ○ Drops all rows with missing values

DO NOT WRITE ANYTHING HERE

√ Replaces missing values with 0 in all columns
◯ Imputes missing values using column means
◯ Creates dummy variables for missing entries

(k) (1 point) What does this pipeline accomplish?

```julia
using MLJ
LR = @load LinearRegressor pkg=GLM
pipe = FillImputer() |> Standardizer() |> LR()
```

◯ One-hot encodes features, then fits a linear model
◯ Imputes missing values with 0, skips scaling, then fits a GLM
√ Fills missing values with column means, standardizes features, then fits a linear regression
◯ Removes rows with missing values, normalizes features, then fits a decision tree

(l) (1 point) Which code snippet will run without errors?

```julia
# Option 1
mutable struct Sensor
    id::Int
    readings::Vector{Float64}
end
s = Sensor(1, [1.0, 2.0])
push!(s.readings, 3.0)

# Option 2
struct Sensor
    id::Int
    readings::Vector{Float64}
end
s = Sensor(1, [1.0, 2.0])
push!(s.readings, 3.0)
```

◯ Neither works ◯ Only Option 1 works ◯ Only Option 2 works √ Both work