Cairo University

Faculty of Computers and Artificial Intelligence

# Machine Learning

## Assignment2

## Project Team

| ID | Name |
| --- | --- |
| 20210348 | Mohamed Fathi Sayed |
| 20210229 | Abdel-Rahman Mohamed Ahmed |
| 20210388 | Mostafa Ahmed Mohamed |

## Perform analysis on the dataset to:

- **check whether there are missing values:**

```
Temperature    25
Humidity       40
Wind_Speed     32
Cloud_Cover    33
Pressure       27
Rain            0
dtype: int64
```

- **Apply the two techniques to handle missing data, dropping missing values and replacing them with the average of the feature:**

  ⇨ **Drop Technique**

```
Check Missing After Drop =>
 Temperature    0
Humidity       0
Wind_Speed     0
Cloud_Cover    0
Pressure       0
Rain           0
dtype: int64
```

```
Dropped Missing =>
       Temperature   Humidity  Wind_Speed  Cloud_Cover     Pressure     Rain
0        19.096119  71.651723   14.782324    48.699257   987.954760  no rain
1        27.112464  84.183705   13.289986    10.375646  1035.430870  no rain
2        20.433329  42.290424    7.216295     6.673307  1033.628086  no rain
3        19.576659  40.679280    4.568833    55.026758  1038.832300  no rain
4        19.828060  93.353211    0.104489    30.687566  1009.423717  no rain
...            ...        ...         ...          ...          ...      ...
2495     14.684023  82.054139    8.751728    58.939058  1003.418337     rain
2496     20.754521  92.099534   17.305508    70.889921  1049.801435     rain
2497     22.087516  71.530065    0.857918    84.162554  1039.664865     rain
2498     18.542453  97.451961    5.429309    54.643893  1014.769130     rain
2499     23.720338  89.592641    7.335604    50.501694  1032.378759     rain

[2347 rows x 6 columns]
```

⇨ **Replace Technique**

```
Check Missing After Replace =>
 Temperature    0
Humidity       0
Wind_Speed     0
Cloud_Cover    0
Pressure       0
Rain           0
dtype: int64
```

⇨

**Does our data have the same scale? If not, you should apply feature scaling on them:**

```
Check Scalling After Drop =>
       Temperature     Humidity   Wind_Speed  Cloud_Cover     Pressure
count  2347.000000  2347.000000  2347.000000  2347.000000  2347.000000
mean     22.586674    64.313486     9.936976    49.826460  1014.362428
std       7.325814    19.969574     5.778717    29.163519    20.157864
min      10.001842    30.005071     0.009819     0.015038   980.014486
25%      16.423651    47.124078     4.786505    24.119752   997.010203
50%      22.533110    64.044753     9.999957    49.735062  1013.591009
75%      28.967040    81.607683    14.955263    75.496921  1031.683526
max      34.995214    99.997481    19.999132    99.997795  1049.985593
Check Scalling After Replace =>
       Temperature     Humidity   Wind_Speed  Cloud_Cover     Pressure
count  2500.000000  2500.000000  2500.000000  2500.000000  2500.000000
mean     22.573777    64.366909     9.911826    49.808770  1014.409327
std       7.295628    19.813325     5.743575    28.869772    20.072933
min      10.001842    30.005071     0.009819     0.015038   980.014486
25%      16.417898    47.493987     4.829795    24.817296   997.190281
50%      22.573777    64.366909     9.911826    49.808770  1014.095390
75%      28.934369    81.445049    14.889660    74.989410  1031.606187
max      34.995214    99.997481    19.999132    99.997795  1049.985593
```

⇨

- **After Scaling**

```
xTrain =>
       Temperature  Humidity  Wind_Speed  Cloud_Cover  Pressure
0        -1.436476  0.273962   -1.504461     1.188434  0.381219
1        -0.053858 -1.595632   -0.752746     1.707058 -1.545006
2        -0.470093 -0.690259   -0.021115    -0.622645  0.917345
3         0.178123 -1.625476    1.294497     0.213524 -1.690423
4         1.526901 -1.134649   -0.797317    -0.348984  0.888535
...            ...       ...         ...          ...       ...
1872      0.078203 -0.893713   -1.450279     0.233644  1.004293
1873      1.486115 -0.847054   -0.945006    -0.839689  0.134456
1874     -0.526243  0.710954    0.148594    -1.238773 -0.187539
1875     -0.523584 -1.090587   -0.661645     1.204614  1.601488
1876      1.514337  1.121178    1.140735    -0.298312  0.084668

[1877 rows x 5 columns]
xTest =>
      Temperature  Humidity  Wind_Speed  Cloud_Cover  Pressure
0       -0.260826 -1.358558    0.829739    -0.692056 -0.070387
1       -0.116059 -0.886083    0.084865    -0.467499  0.984156
2        1.422790  1.776448   -0.451182     0.934342  1.589956
3        1.330837  0.165115   -0.355094     1.401501  0.330636
4        0.941170 -1.280496   -0.492945     0.287127 -0.294605
..            ...       ...         ...          ...       ...
465      0.119888 -1.083123   -1.039789     0.896917  1.221064
```

```
_xTrain =>
       Temperature  Humidity  Wind_Speed  Cloud_Cover  Pressure
0        -1.718125  1.687949    1.697663    -0.229740 -0.142129
1         0.578604 -1.222505    1.195252     0.013527  1.749370
2        -1.611123 -1.677586    0.944283    -0.392969  1.456590
3        -1.293667  0.840139    1.180401     0.752595  0.044229
4        -1.366615  0.086746    0.063829     0.285025 -0.950329
...            ...       ...         ...          ...       ...
1995      0.127745 -0.648388   -1.346847    -0.890397  0.199726
1996     -1.574577  0.002285    0.550592     0.170995  0.614659
1997     -1.645064 -1.671404    1.047639    -1.677150 -1.539260
1998     -1.220415  1.675623    0.209510    -0.446911  0.856001
1999     -0.747327 -1.245507    0.670893    -0.008283 -0.152921

[2000 rows x 5 columns]
_xTest =>
      Temperature  Humidity  Wind_Speed  Cloud_Cover  Pressure
0       -0.439931  0.875070   -0.813364    -0.506291 -0.419847
1       -1.725871 -0.290745   -1.281728    -0.091093 -1.481063
2        1.166779  1.504868    0.490502    -1.364309  0.767471
3       -1.184871  1.141692   -0.207549     0.641584  1.570095
4        1.265119 -1.192291   -0.882951    -1.709711  1.253436
..            ...       ...         ...          ...       ...
495     -1.618694  0.866196   -1.516350     0.251699 -0.048407
```

**Task2:4: and Task 3:1 Provide a detailed report evaluating the performance of scikit learn implementations of the Decision Tree, k-Nearest Neighbors (kNN) and naïve Bayes with respect to the different handling missing data technique:**

- **Replace Technique**

```
Naive Bayes Score {Test Data}:  0.964
Decision Tree Score {Test Data}:  0.996
```

| | Model | Accuracy | Precision | Recall |
|---|---|---|---|---|
| 0 | kNN | 0.966 | 0.928927 | 0.895029 |
| 1 | KNN_Scratch | 0.966 | 0.928927 | 0.895029 |
| 2 | Naive Bayes | 0.964 | 0.980519 | 0.839286 |
| 3 | Decision Tree | 0.996 | 0.997758 | 0.982143 |

  - KNN (both scratch and scikit): Both KNN implementations show high accuracy and precision, but their recall is slightly lower, indicating they miss some true positives.
  - Naive Bayes: Naive Bayes performs well in precision, showing it is very confident when it predicts positive cases, but it misses many true positives due to low recall.
  - Decision Tree: Decision Tree performs exceptionally well across all metrics, with nearly perfect precision and recall, making it the most balanced model for this technique.

- **Drop Technique**

```
Naive Bayes Score {Test Data}:  0.9617021276595744
Decision Tree Score {Test Data}:  0.997872340425532
```

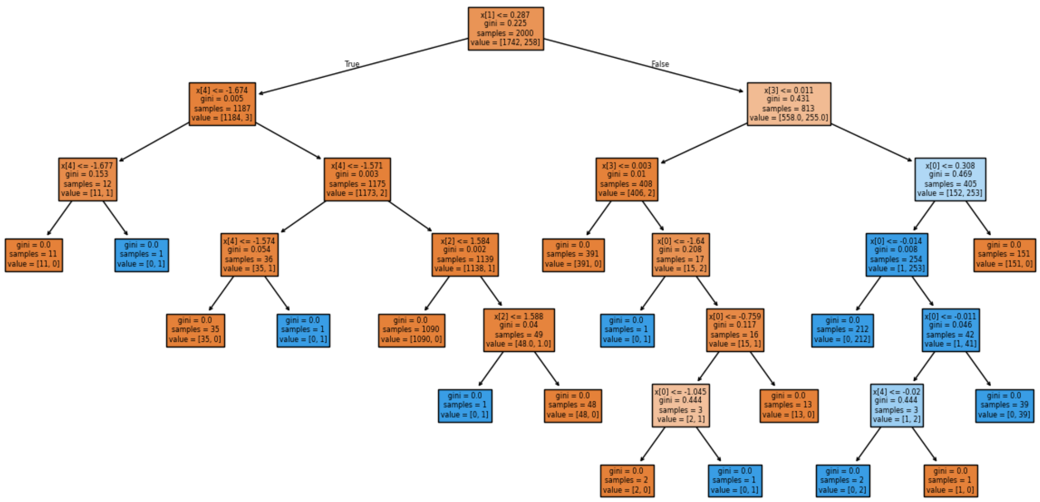
| | Model | Accuracy | Precision | Recall |
|---|---|---|---|---|
| 0 | kNN | 0.961702 | 0.936907 | 0.904302 |
| 1 | KNN_Scratch | 0.961702 | 0.936907 | 0.904302 |
| 2 | Naive Bayes | 0.961702 | 0.978571 | 0.867647 |
| 3 | Decision Tree | 0.997872 | 0.998759 | 0.992647 |

  - KNN (both scratch and scikit): Both KNN implementations show high accuracy and precision, but their recall is slightly lower, indicating they miss some true positives.
  - Naive Bayes: Naive Bayes performs well in precision, showing it is very confident when it predicts positive cases, but it misses many true positives due to low recall.
  - Decision Tree: Decision Tree performs exceptionally well across all metrics, with nearly perfect precision and recall, making it the most balanced model for this technique.

- **Decision Tree Explanation Report**

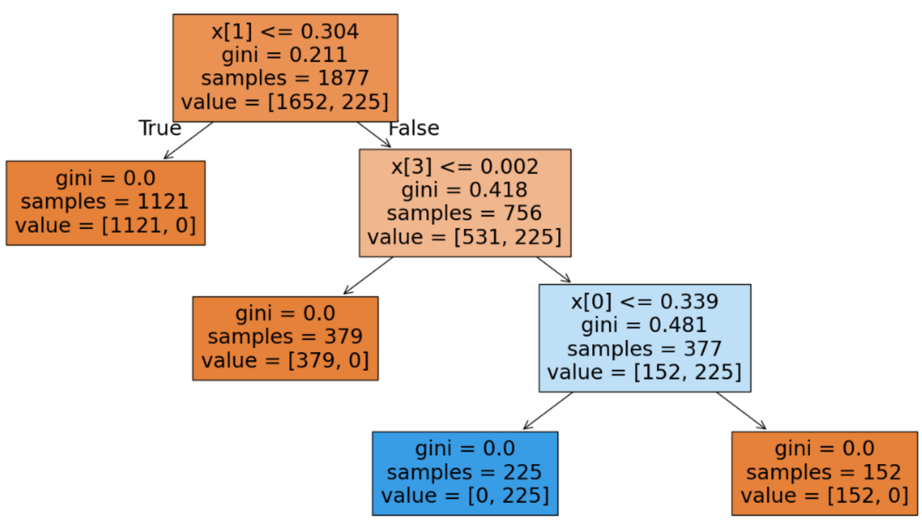
  ⇨ **Create a well-formatted report that includes a plot of the decision tree and a detailed explanation of how the tree makes predictions:**

  - **Replace**



○

  - **Drop**




○

## Discuss the criteria and splitting logic used at each node of the tree:

- **Replace**
  - **Node 1 (Root Node)**
    - Condition: Humidity <= 0.287
    - Criteria: The tree checks if the value of feature Humidity is less than or equal to 0.287.
    - Gini: 0.225 (Measures impurity; the lower, the purer).
    - Samples: 2000 total samples at this node.
    - Values: [1742, 258] (1742 samples for "no rain" and 258 for "rain").
    - Predicted Class: No rain (majority class).
    - Splitting: If the condition is true, follow the left branch; otherwise, follow the right branch.
  - **Node 2 (Left Child of Root Node)**
    - Condition: Pressure <= -1.674
    - Criteria: The tree checks if the value of feature Pressure is less than or equal to -1.674.
    - Gini: 0.005 (Very low impurity).
    - Samples: 1187 total samples at this node.
    - Values: [1184, 3] (1184 samples for "no rain" and 3 for "rain").
    - Predicted Class: No rain (majority class).
    - Splitting: If the condition is true, follow the left branch; otherwise, follow the right branch.
  - **Node 3 (Right Child of Root Node)**
    - Condition: Cloud_Cover <= 0.011
    - Criteria: The tree checks if the value of feature Cloud_Cover is less than or equal to 0.011.
    - Gini: 0.311 (Moderate impurity).
    - Samples: 813 total samples at this node.
    - Values: [558, 255] (558 samples for "no rain" and 255 for "rain").
    - Predicted Class: No rain (majority class).
    - Splitting: If the condition is true, follow the left branch; otherwise, follow the right
  - **Node 4 (Left Child of Node 2)**
    - Condition: Pressure <= -1.677
    - Criteria: The tree checks if the value of feature Pressure is less than or equal to -1.677.
    - Gini: 0.153 (Low impurity).
    - Samples: 12 total samples at this node.
    - Values: [11, 1] (11 samples for "no rain" and 1 for "rain").
    - Predicted Class: No rain (majority class).
    - Splitting: If the condition is true, follow the left branch; otherwise, follow the right branch
  - **Node 5 (right Child of Node2)**
    - Condition: Pressure <= -1.571
    - Criteria: The tree checks if the value of feature Pressure is less than or equal to -1.571.
    - Gini: 0.003 (Very low impurity).
    - Samples: 1175 total samples at this node.
    - Values: [1173, 2] (1173 samples for "no rain" and 2 for "rain").
    - Predicted Class: No rain (majority class).
    - Splitting: If the condition is true, follow the left branch; otherwise, follow the right branch.

- ⇨ **Node 6 (left Child of Node 3)**
  - Condition: Cloud_Cover<= 0.003
  - Criteria: The tree checks if the value of feature Cloud_Cover is less than or equal to 0.003.
  - Gini: 0.01 (Very low impurity).
  - Samples: 408 total samples at this node.
  - Values: [406, 2] (406 samples for "no rain" and 2 for "rain").
  - Predicted Class: No rain (majority class).
  - Splitting: If the condition is true, follow the left branch; otherwise, follow the right branch.
- ⇨ **Node 7 (right Child Of Node 3)**
  - Condition: Temperature <= 0.308
  - Criteria: The tree checks if the value of feature Temperature is less than or equal to 0.308.
  - Gini: 0.469 (Moderate impurity).
  - Samples: 465 total samples at this node.
  - Values: [152, 253] (152 samples for "no rain" and 253 for "rain").
  - Predicted Class: Rain (majority class).
  - Splitting: If the condition is true, follow the left branch; otherwise, follow the right branch.
- ⇨ **Node 8 (left Child of Node 4)**
  - Condition: This is a leaf node, so there is no splitting condition anymore.
  - Gini: 0.0 (Pure node). The Gini index is 0 because this node contains samples of only one class—there's no impurity.
  - Samples: 11 total samples in this node.
  - Values: [11, 0] or [0, 11], depending on the class distribution (all 11 samples belong to either "no rain" or "rain").
  - Predicted Class: The predicted class is the majority class. In this case, since Gini = 0, all samples belong to the same class, so the predicted class is either "no rain" or "rain" (depending on the sample distribution).
  - Splitting: This is a leaf node, so no further splits occur.
- ⇨ **1- Orange Nodes (Decision Nodes):**
  - These are the internal nodes where the data is split based on a specific condition (such as Pressure <= -1.677).
  - Each node represents a feature condition that leads to further branching, dividing the data into subgroups. These nodes contain information like the Gini index (impurity measure), the number of samples passing through this node, and the class distribution (value).
- ⇨ **2- Blue Nodes (Leaf Nodes with a Single Class Label):**
  - These are the leaf nodes (terminal nodes) where the final classification decision is made.
  - A blue leaf node indicates that the classification is pure, meaning that all samples that reach this node belong to a single class. The "value" represents the number of samples for each class, and the Gini index is 0 (pure).
- ⇨ **3- Light Blue Nodes (Leaf Nodes with Multiple Class Labels):**
  - These nodes are also leaf nodes, but they represent a more mixed classification, where multiple classes are present in the final decision.
  - The samples in these nodes belong to different classes, which can be seen in the "value" (e.g., [5, 6] indicating the presence of multiple classes).

⇨ **4- Light Orange Nodes (Intermediate Decision Nodes Leading to Mixed Class Distributions):**

- These nodes are similar to the orange decision nodes but they are on a deeper level of the tree, and they often represent splits that are not as clean or clear-cut as those in higher nodes.
- These nodes can lead to multiple classes being distributed among the child leaf nodes. However, their decision is still based on a feature condition that eventually divides the data into smaller groups.

## Drop

- **1. Criteria for Splitting:**
  - **GINI Impurity:**
  - The splitting logic is based on minimizing the **Gini Impurity** at each node.
  - Gini Impurity measures the likelihood of incorrect classification by randomly selecting a class label based on the class distribution in the node.
  - A lower Gini value indicates purer nodes, which is the goal of the splitting
- **2. Each Node Explanation:**
  - **Root Node (Level 0):**
    - **Feature**: Humidity
    - **Split Condition**: Humidity≤ 0.304
    - **Gini Value**: 0.211
    - **Samples**: 1877
    - **Class Distribution**: [1652 (Class 0) (No Rain), 225 (Class 1) (Rain)]
    - The root node splits based on feature Humidity. It chooses the threshold 0.304 to minimize impurity, separating most of the Class 0 (No Rain) samples to the left and some of Class 1 (Rain) samples to the right.
  - **Left Child (Level 1 - True Path):**
    - **Feature**: None (Pure Node)
    - **Gini Value**: 0.0 (it means that the node is **pure**, meaning all the instances in that node belong to the **same class**. In other words, there is no uncertainty or disorder in the classification for that node.)
    - **Samples**: 1121
    - **Class Distribution**: [1121 (Class 0) (No Rain), 0 (Class 1) (Rain)]
    - This is a pure node where all samples belong to Class 0. No further splitting occurs.

- **Right Child (Level 1 - False Path):**
  - **Feature**: Cloud_Cover
  - **Split Condition**: Cloud_Cover ≤ 0.002
  - **Gini Value**: 0.418
  - **Samples**: 756
  - **Class Distribution**: [531 (Class 0) (No Rain), 225 (Class 1) (Rain)]
  - The right child splits on feature Cloud_Cover with a threshold 0.002, further reducing impurity and separating more of Class 1 (Rain) samples.
- **Left Child of Right Child (Level 2 - True Path):**
  - **Feature**: None (Pure Node)
  - **Gini Value**: 0.0 (it means that the node is **pure**, meaning all the instances in that node belong to the **same class**. In other words, there is no uncertainty or disorder in the classification for that node.)
  - **Samples**: 379
  - **Class Distribution**: [379 (Class 0) (No Rain), 0 (Class 1) (Rain)]
  - This is another pure node where all samples belong to Class 0 (No Rain). No further splitting occurs.
- **Right Child of Right Child (Level 2 - False Path):**
  - **Feature**: Temperature
  - **Split Condition**: Temperature ≤ 0.339
  - **Gini Value**: 0.481
  - **Samples**: 377
  - **Class Distribution**: [152 (Class 0) (No Rain), 225 (Class 1) (Rain)]
  - This node splits on feature Temperature with a threshold 0.339 It attempts to further separate the remaining Class 1 (Rain) samples from Class 0 (No Rain) samples.

- **Left Child of Right-Right Child (Level 3 - True Path):**
    - **Feature**: None (Pure Node)
    - **Gini Value**: 0.0 (it means that the node is **pure**, meaning all the instances in that node belong to the **same class**. In other words, there is no uncertainty or disorder in the classification for that node.)
    - **Samples**: 225
    - **Class Distribution**: [0 (Class 0) (No Rain), 225 (Class 1) (Rain)]
    - This is a pure node where all samples belong to Class 1 (Rain). No further splitting occurs.
- **Right Child of Right-Right Child (Level 3 - False Path):**
    - **Feature**: None (Pure Node)
    - **Gini Value**: 0.0 (it means that the node is **pure**, meaning all the instances in that node belong to the **same class**. In other words, there is no uncertainty or disorder in the classification for that node.)
    - **Samples**: 152
    - **Class Distribution**: [152 (Class 0) (No Rain), 0 (Class 1) (Rain)]
    - This is another pure node where all samples belong to Class 0 (No Rain). No further splitting occurs.

- **Color indicates the predicted class**: Each leaf node is colored according to the majority class that it predicts. This helps visually identify how the tree classifies the samples based on the features.
- **Mixed colors**: At non-leaf nodes (such as internal nodes), where the tree is still making decisions based on the features, you might see mixed colors (like a gradient). These internal nodes decide which path to take based on a feature split.

## Performance Metrics Report

- **Provide a detailed report evaluating the performance of your implementations of the k-Nearest Neighbors (KNN) from scratch with different k values at least 5 values. Include the accuracy, precision, and recall metrics for models:**

  - **K = 1**

    - 

      | | Model | Accuracy | Precision | Recall |
      |---|---|---|---|---|
      | 0 | kNN | 0.965957 | 0.926964 | 0.937335 |
      | 1 | KNN_Scratch | 0.965957 | 0.926964 | 0.937335 |

    - With k=1, the model tends to memorize the training data and is sensitive to noise, as it considers only the nearest neighbor for classification. While accuracy is high, precision and recall might suffer slightly due to **overfitting** to noisy or outlier data points

  - **K = 3**

    - 

      | | Model | Accuracy | Precision | Recall |
      |---|---|---|---|---|
      | 0 | kNN | 0.965957 | 0.94081 | 0.919008 |
      | 1 | KNN_Scratch | 0.965957 | 0.94081 | 0.919008 |

    - With k=3. The classifier becomes less sensitive to noise compared to k=1, leading to a slight improvement in precision but a slight drop in recall. The overall performance remains consistent.

  - **K = 5**

    - 

      | | Model | Accuracy | Precision | Recall |
      |---|---|---|---|---|
      | 0 | kNN | 0.961702 | 0.931766 | 0.910411 |
      | 1 | KNN_Scratch | 0.961702 | 0.931766 | 0.910411 |

    - with k=5, accuracy decreases slightly, and recall decreases marginally, showing that the model becomes more general and **less overfitted**. The decision boundaries are smoother, but some minority classes might be misclassified due to the majority voting principle.

- **K = 7**

  - | | Model | Accuracy | Precision | Recall |
    |---|---|---|---|---|
    | 0 | kNN | 0.961702 | 0.936907 | 0.904302 |
    | 1 | KNN_Scratch | 0.961702 | 0.936907 | 0.904302 |

  - with k = 7, precision and recall are still balanced with a slight decrease in recall. The model's decisions rely on a broader neighborhood, which smooths the decision boundaries even more but could result in missing finer details.

- **K = 9**

  - | | Model | Accuracy | Precision | Recall |
    |---|---|---|---|---|
    | 0 | kNN | 0.965957 | 0.935831 | 0.925117 |
    | 1 | KNN_Scratch | 0.965957 | 0.935831 | 0.925117 |

  - With k =9, the performance metrics improve slightly compared to k=7, with a slight increase in recall. The model appears to find a balance between generalization and sensitivity to individual data points.

- **Compare these results with the performance of the corresponding algorithms implemented using scikit-learn:**

  - | | Model | Accuracy | Precision | Recall |
    |---|---|---|---|---|
    | 0 | kNN | 0.965957 | 0.926964 | 0.937335 |
    | 1 | KNN_Scratch | 0.965957 | 0.926964 | 0.937335 |

    - At k=1, both models achieve high recall (**0.9373**), but slightly lower precision (**0.9269**) compared to larger k-values. This is expected as smaller k-values can lead to **overfitting**, where the model is highly sensitive to noise in the data.

  - Increasing k (e.g., k=5,7,9) improves the balance between precision and recall, reducing sensitivity to outlier