**Cairo University**

**Faculty of Computers and Artificial Intelligence**

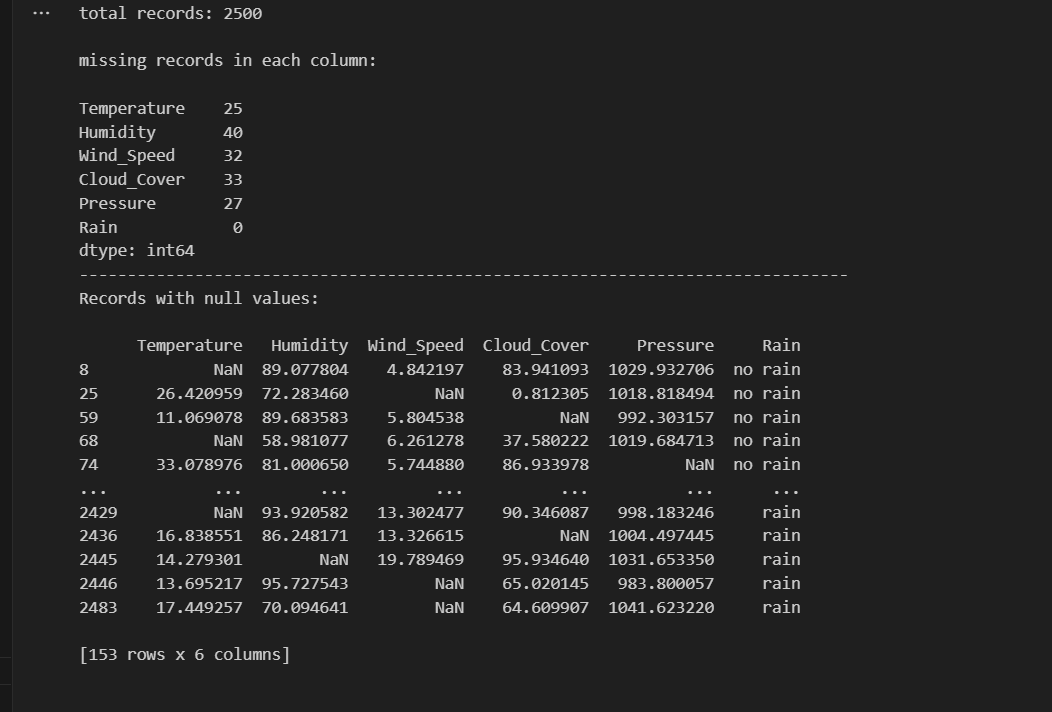
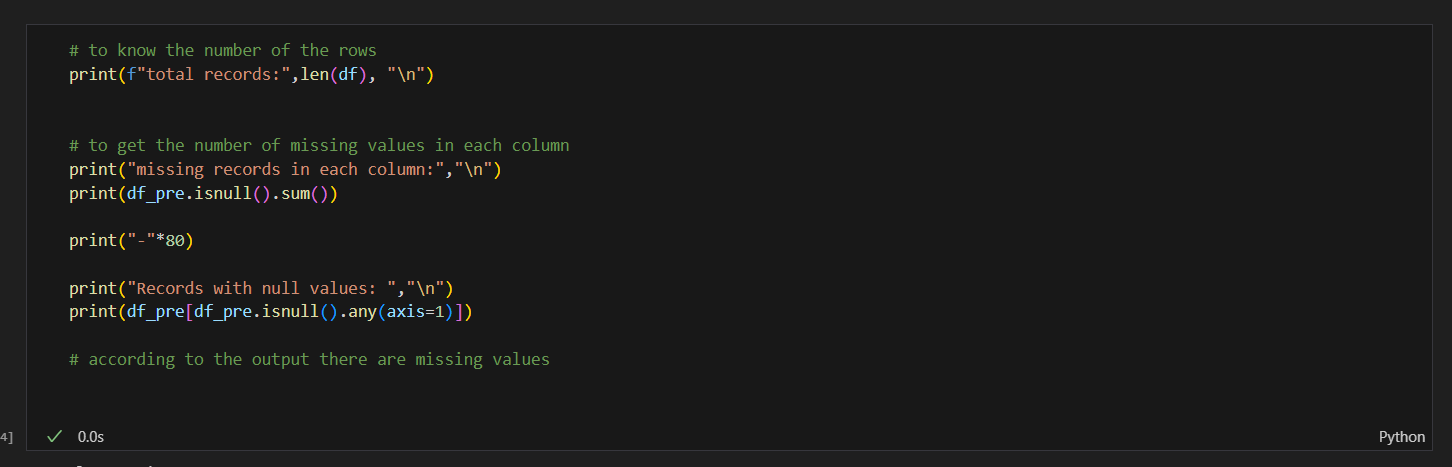
**Machine Learning**

**Assignment 2 Report**

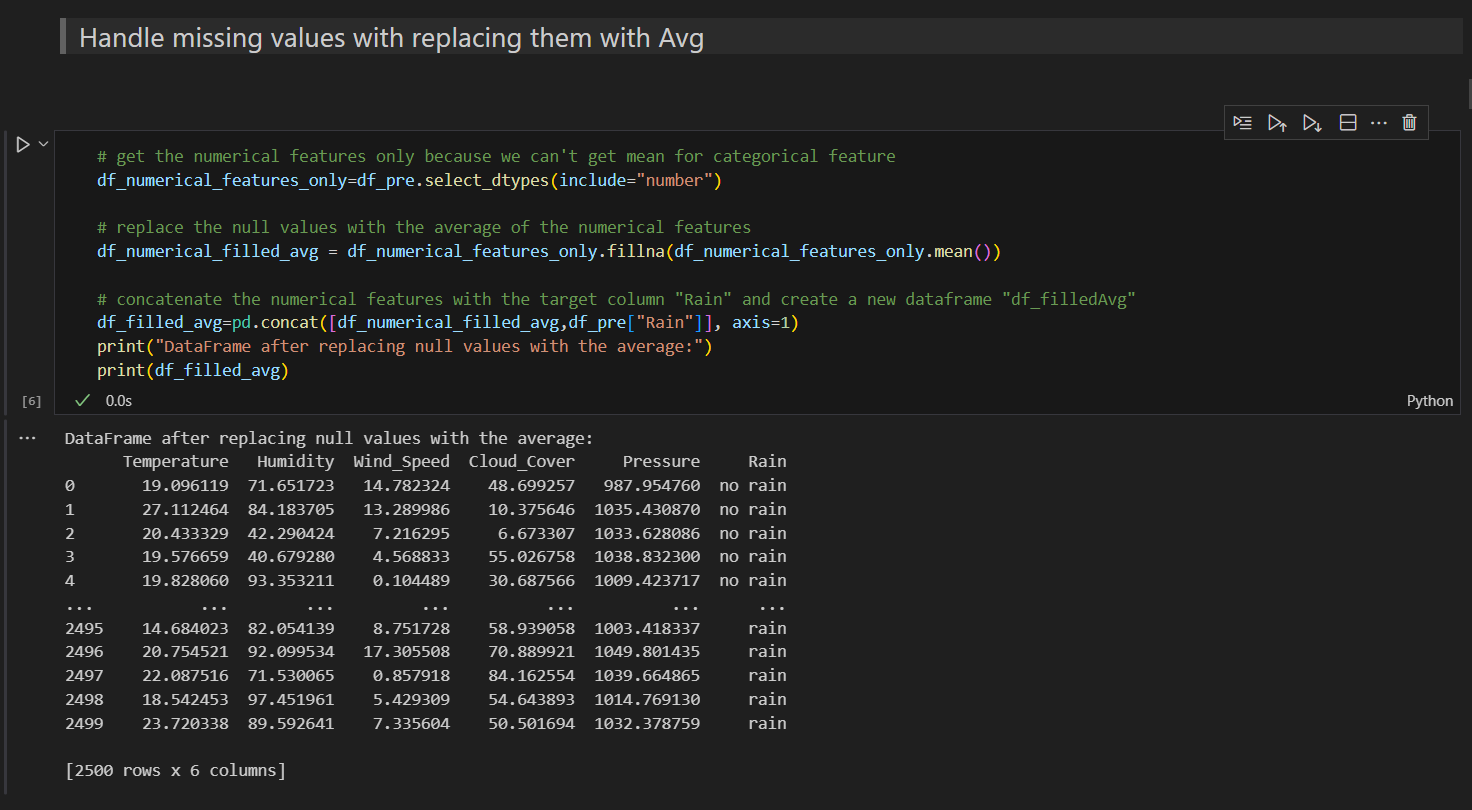
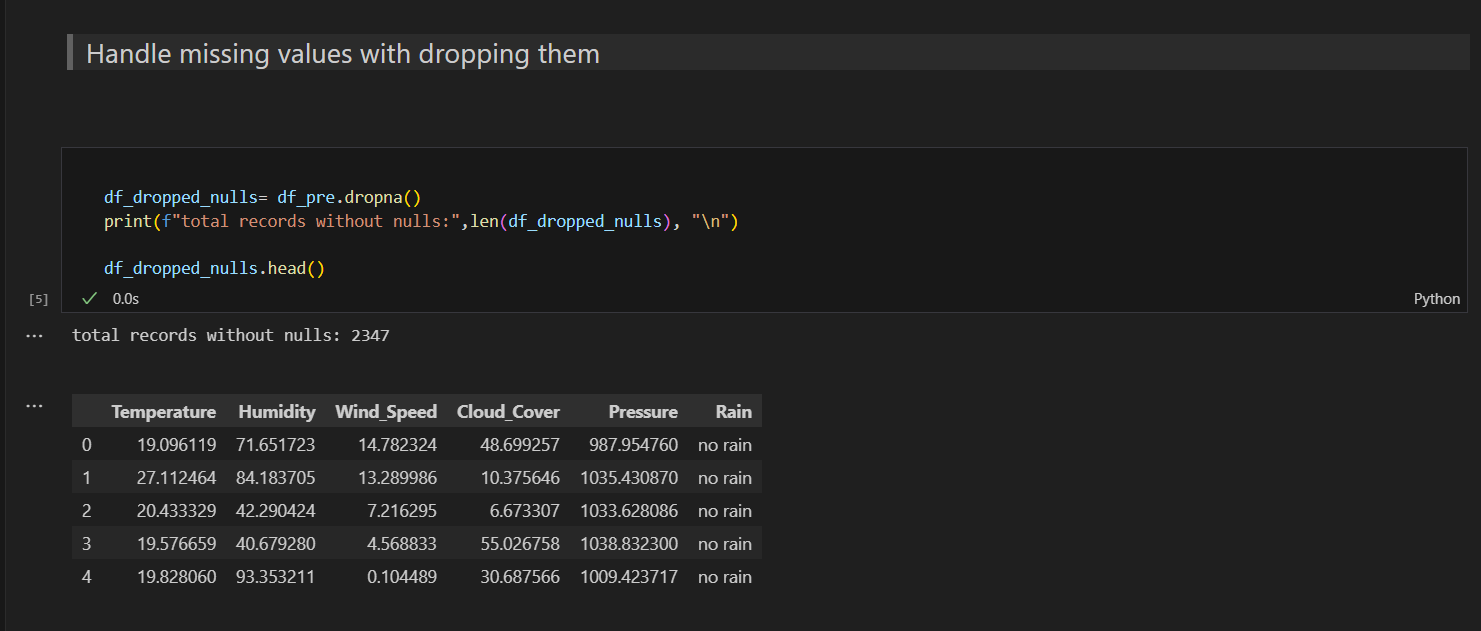
|  |  |
| --- | --- |
| **Team Members** | **IDS** |
| **Ahmed Saad Ahmed** | **20210020** |
| **Maryam Osama Mohamed** | **20211090** |
| **Shahd Mostafa Mahmoud** | **20211054** |
| **Shahd Osama Hamed** | **20211052** |
| **Seif Ibrahim Ramadan** | **20210164** |

Task 1: Preprocessing

1. Does the dataset contain any missing data? Identify them.

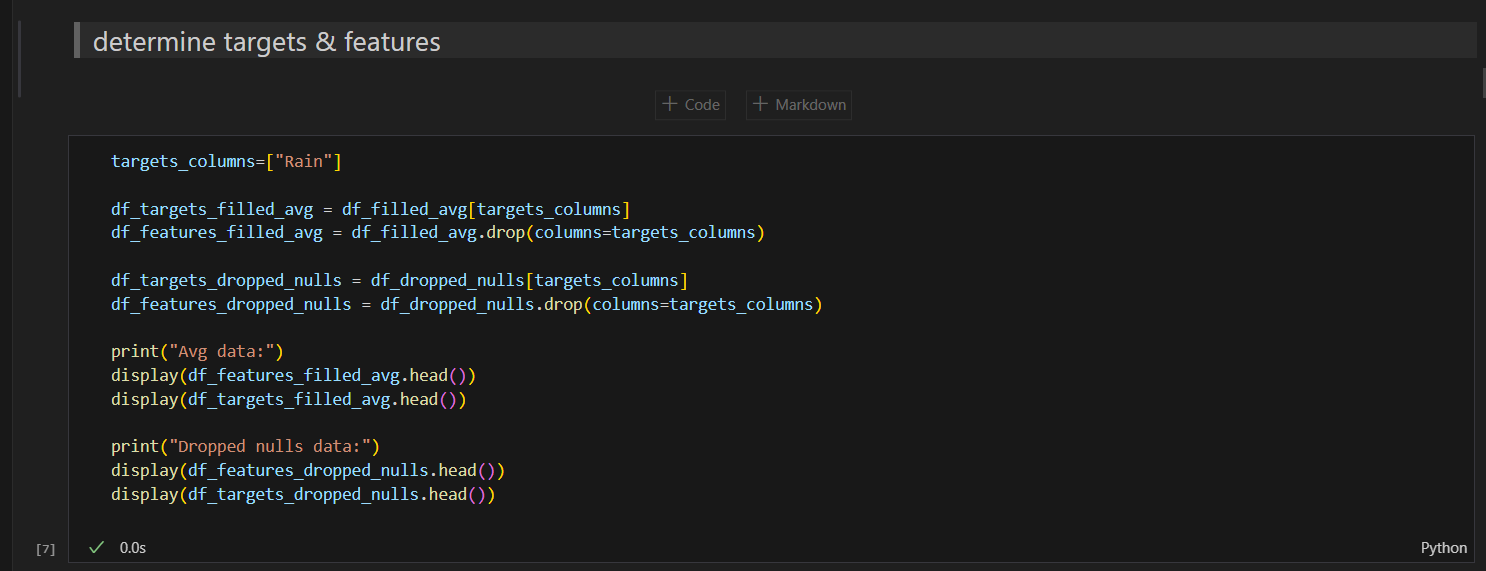


1. Apply the two techniques to handle missing data, dropping missing values and replacing them with the average of the feature.

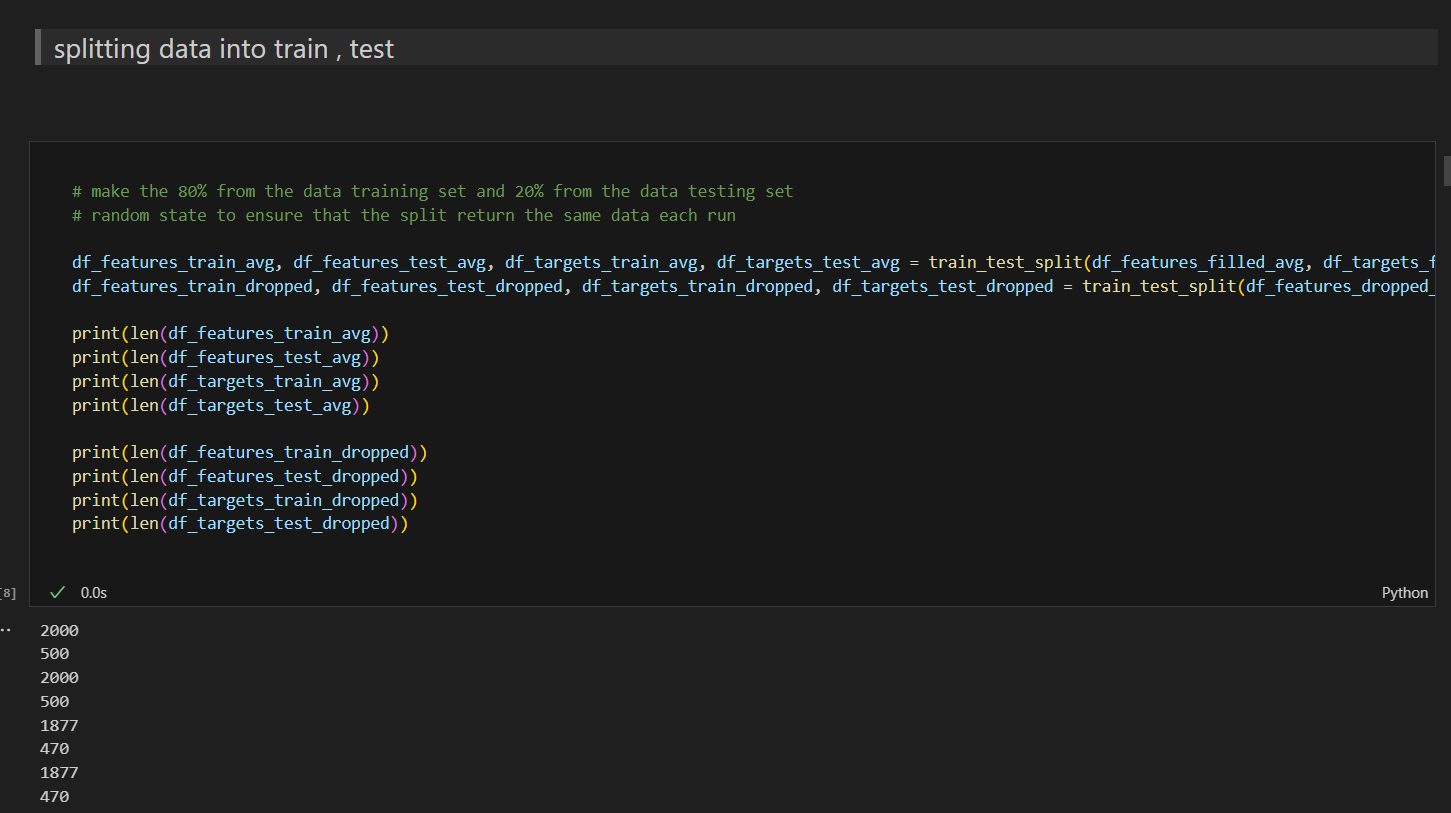


1. Splitting our data to training and testing for training and evaluating our models

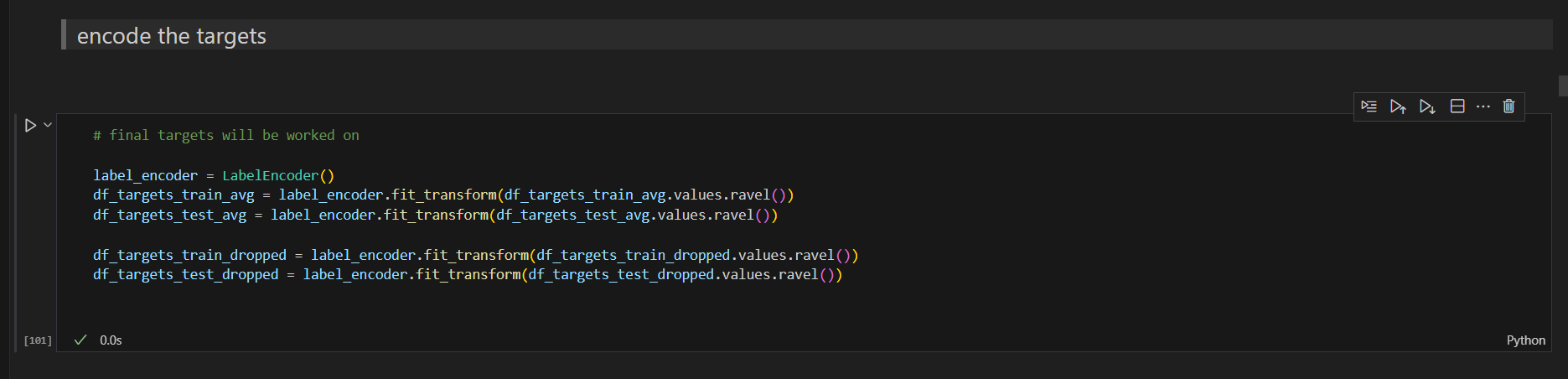
* Get the targets and features first



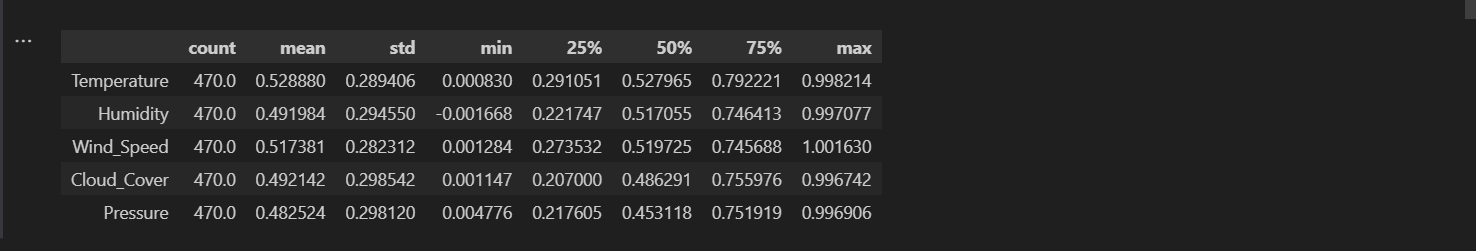
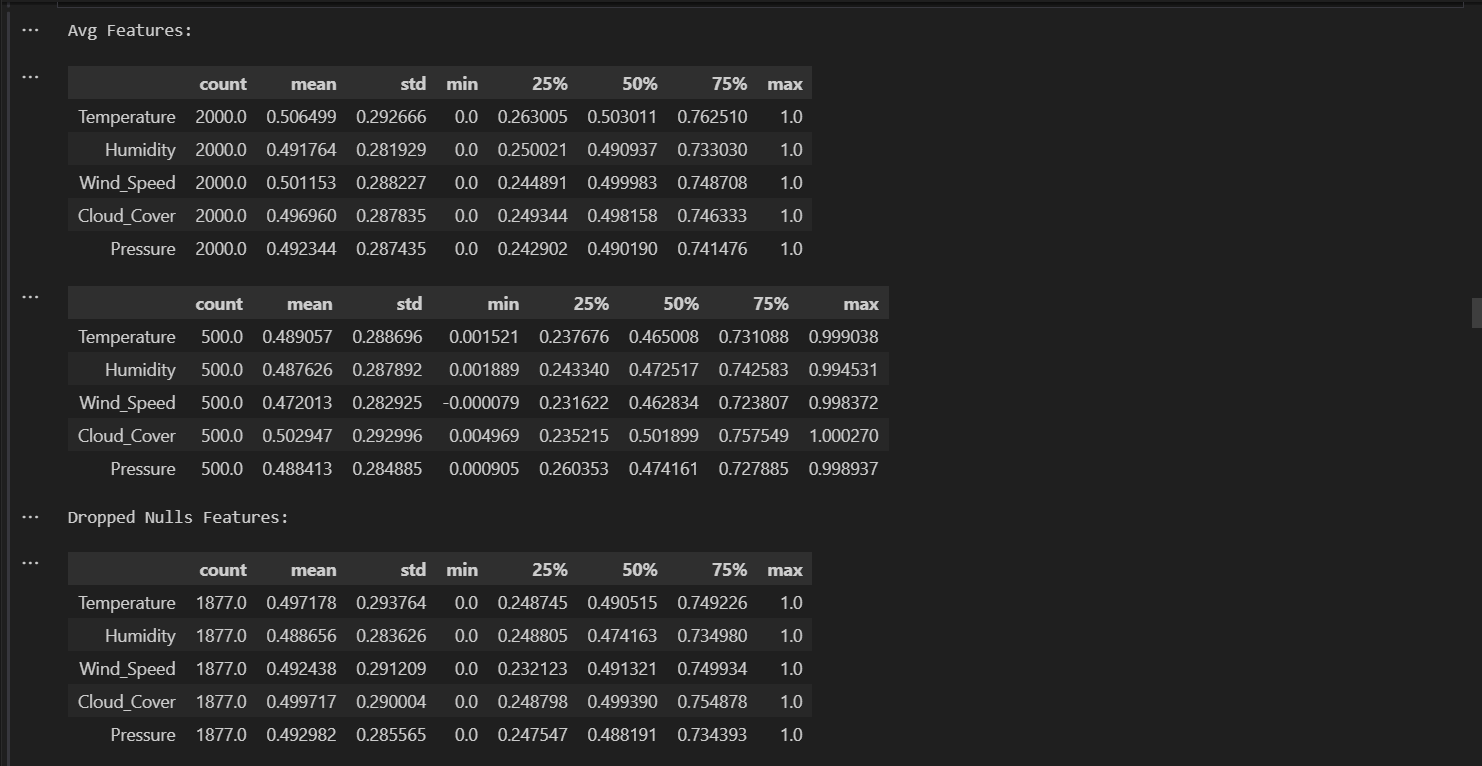
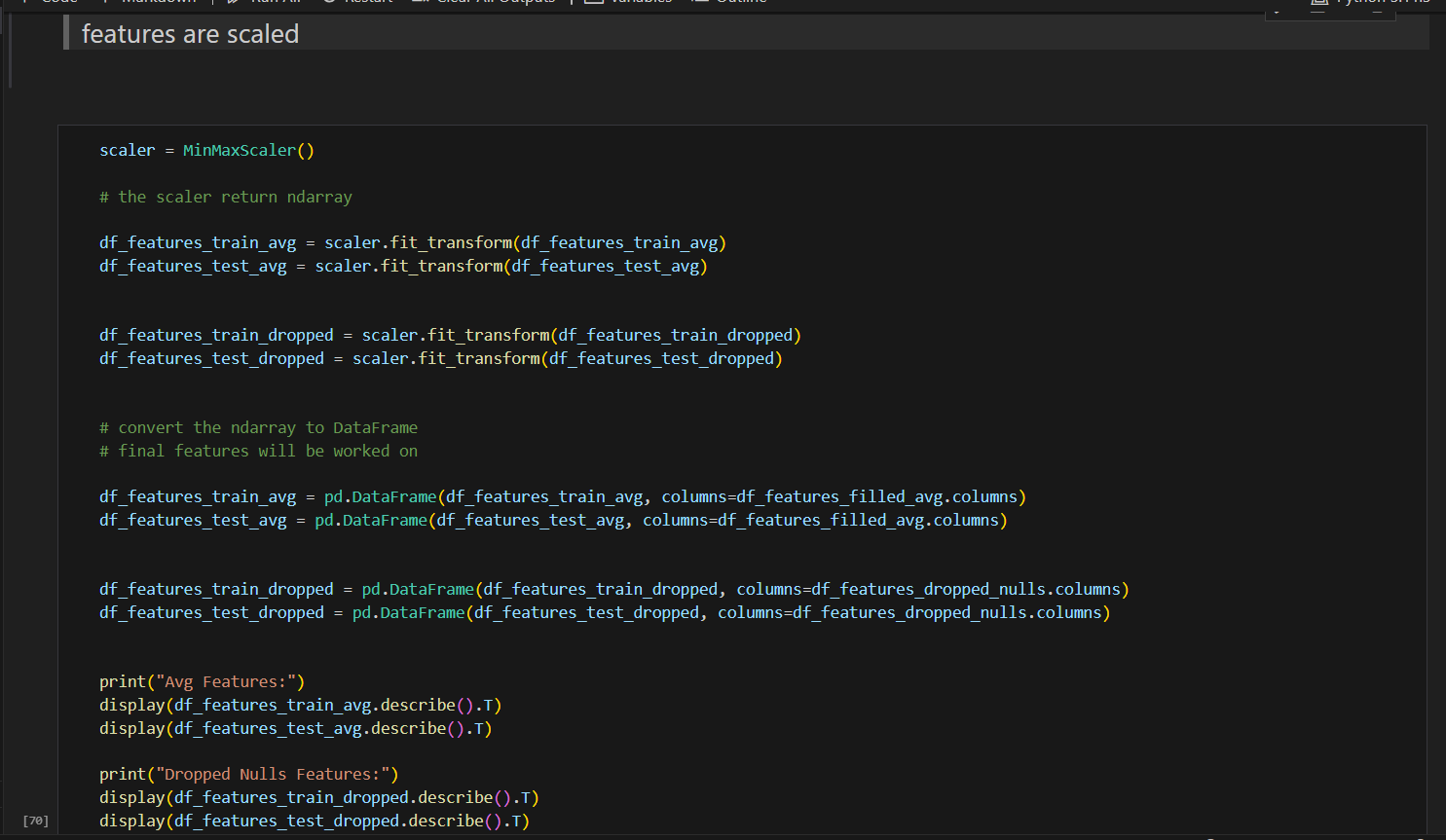
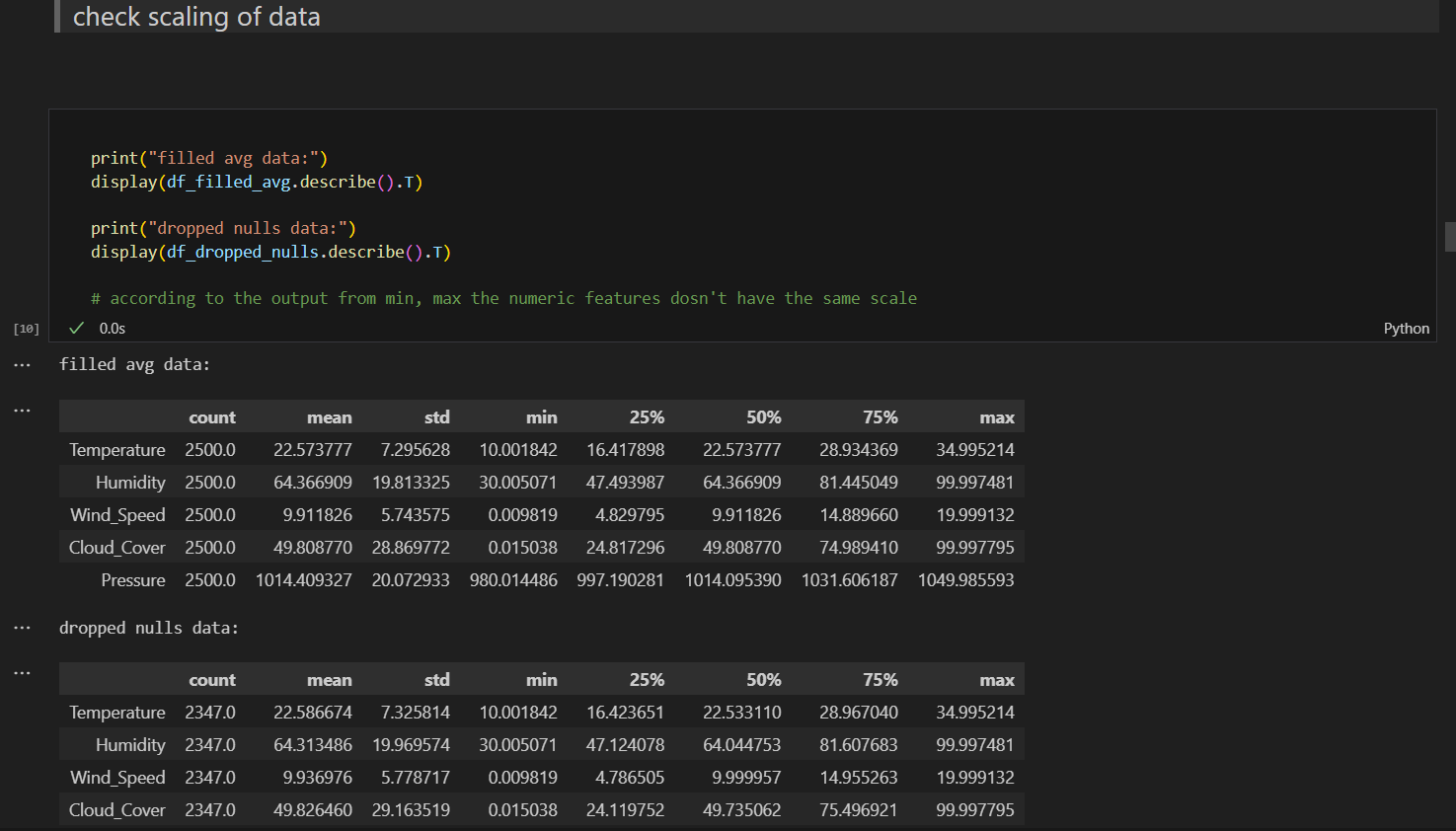
* Splitting the data to training and testing



* Encode the targets using label encoding to make it binary



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

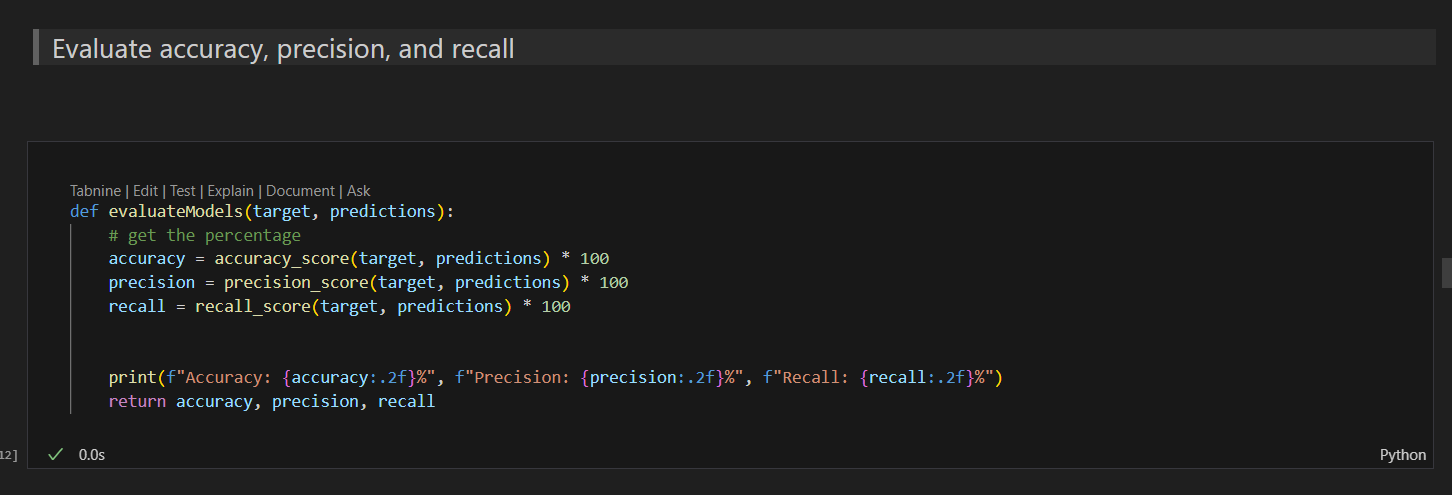


Task 2: Implement Decision Tree, k-Nearest Neighbors (kNN) and naïve Bayes

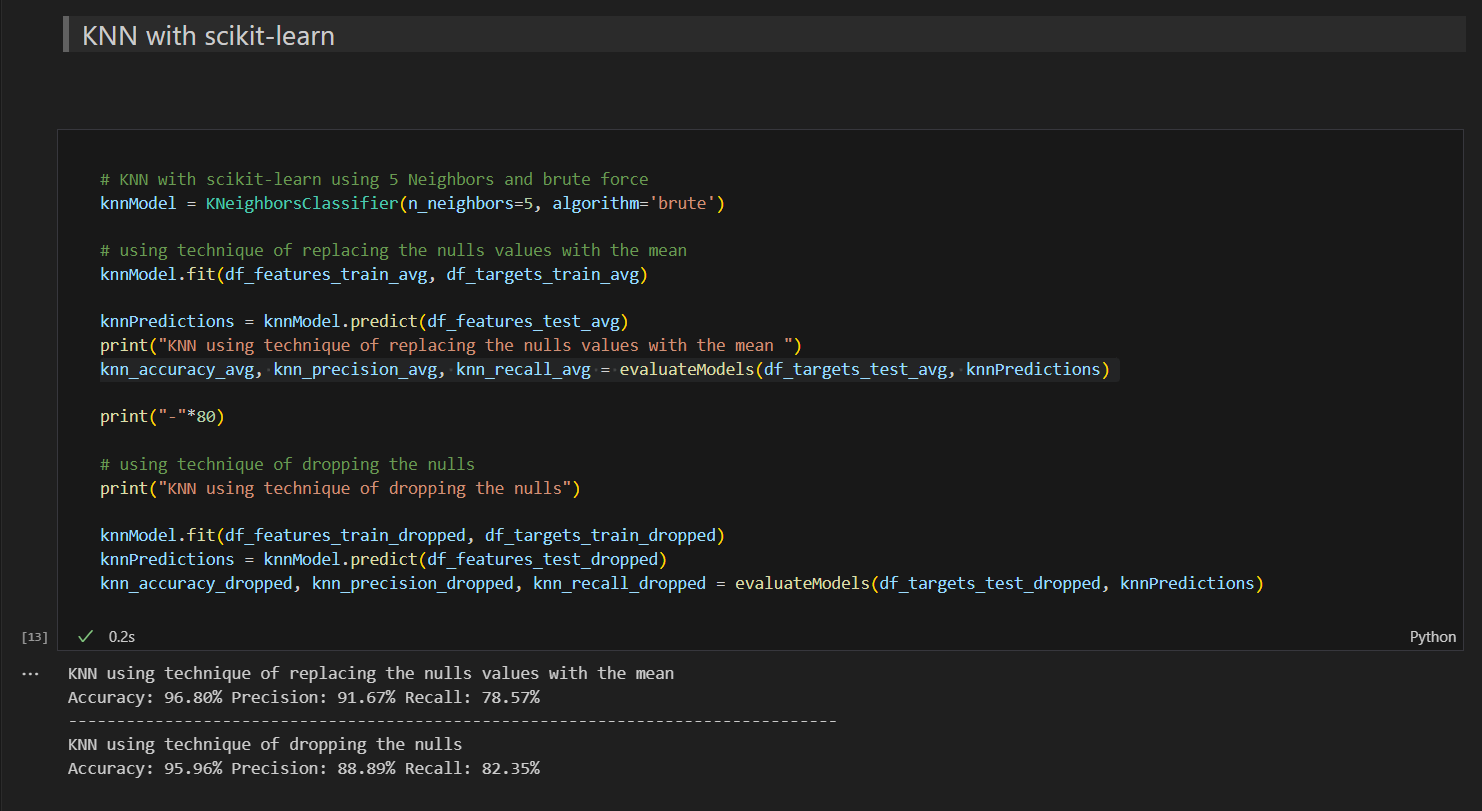
1. Using scikit-learn implement Decision Tree, kNN and Naïve Bayes

* First, we create this function to test the models and

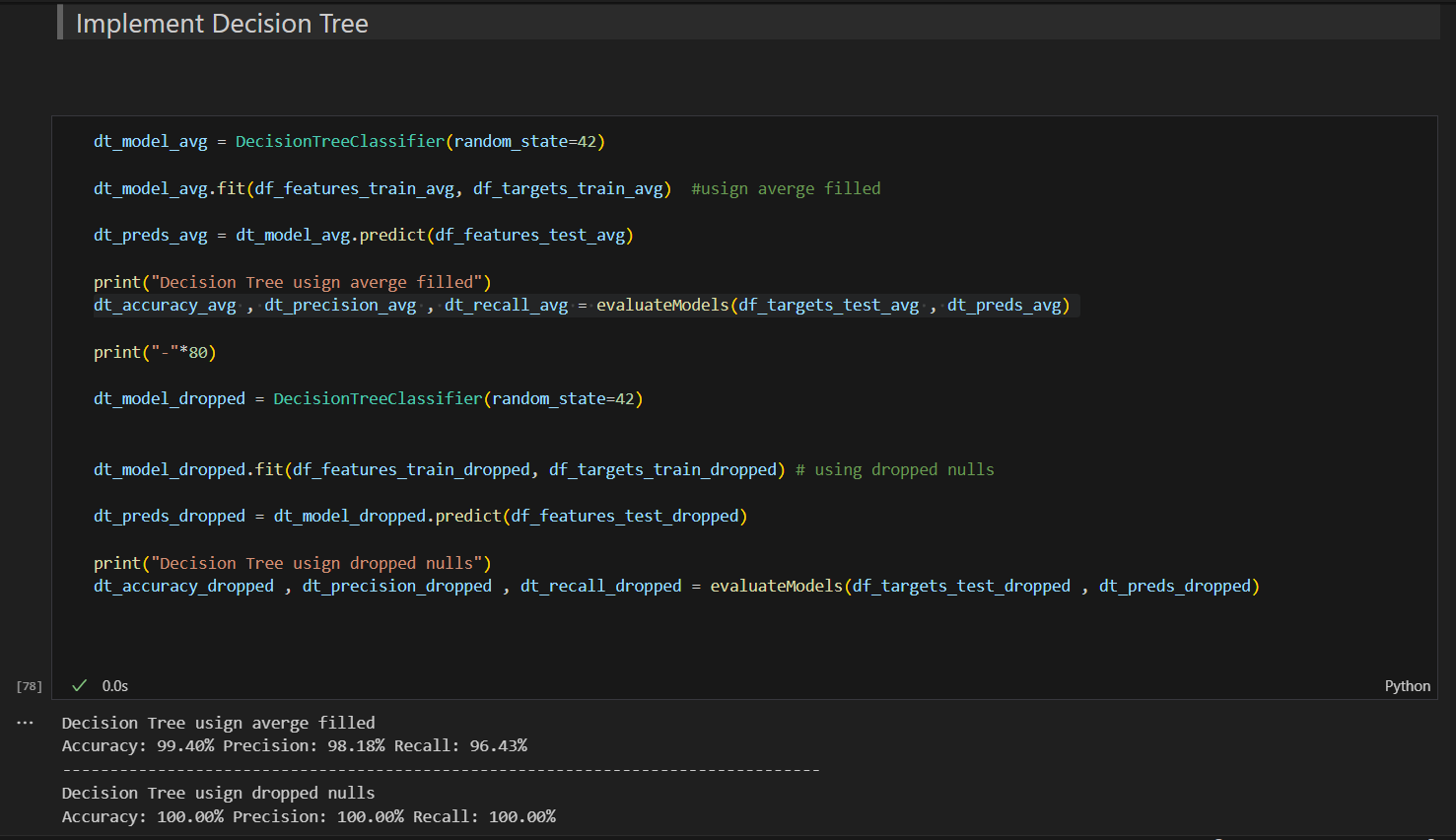
evaluate accuracy, precision, and recall



* Knn:



* Decision Tree:



* Naïve Bayes:

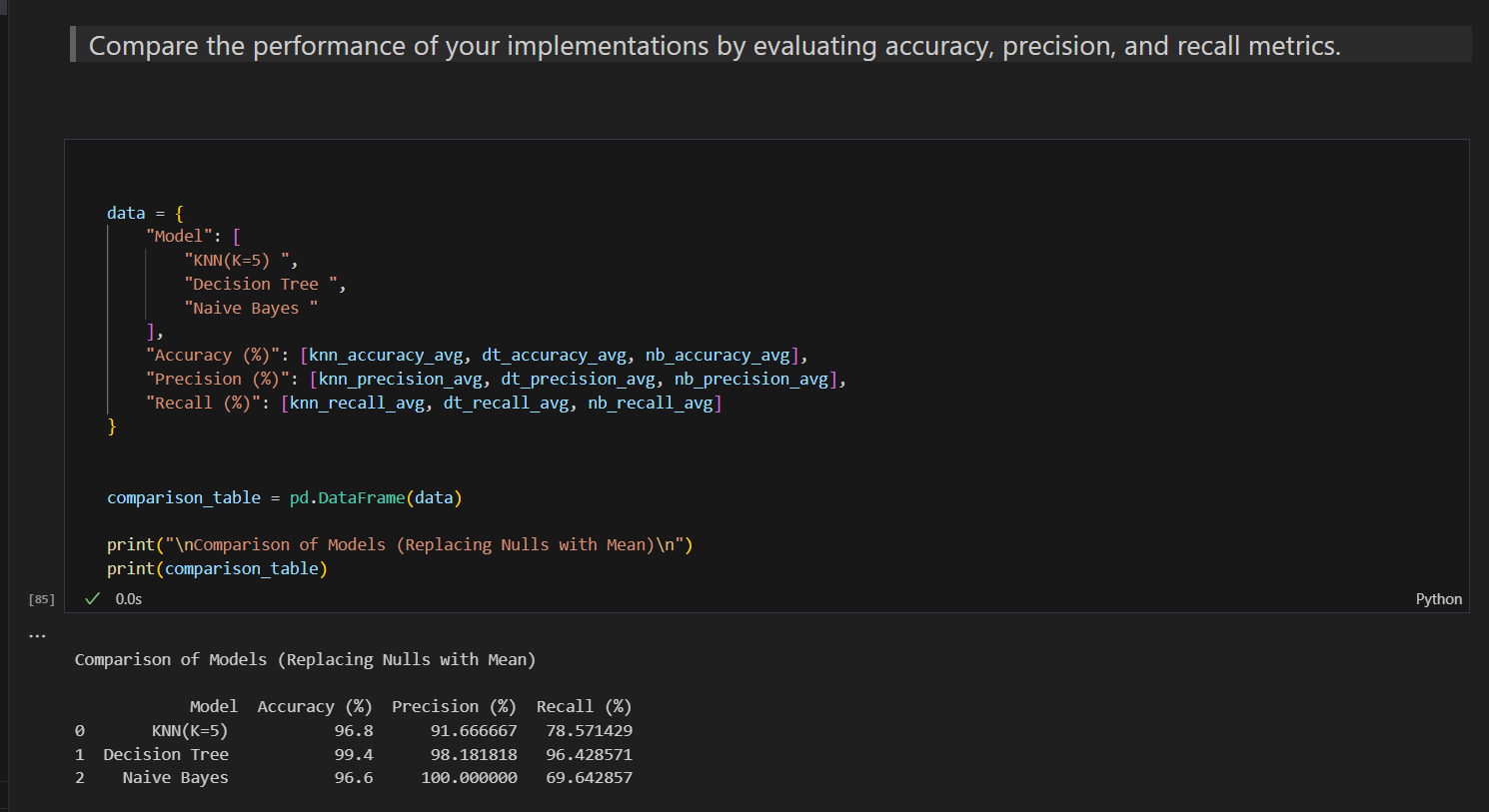
A screenshot of a computer

Description automatically generated

1. Compare the performance of your implementations by evaluating accuracy, precision, and recall metrics.

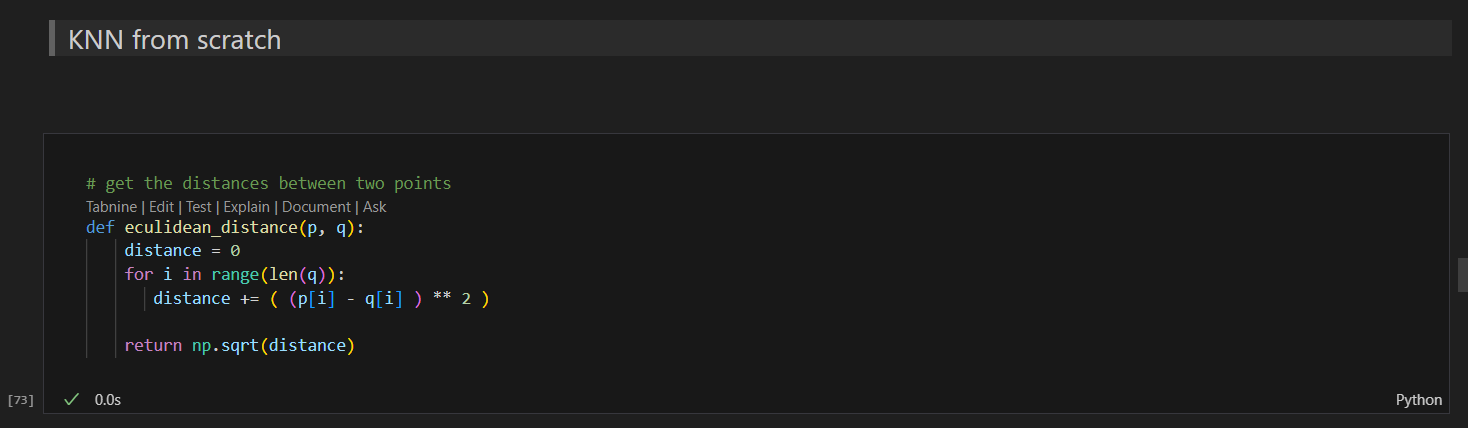
* Using replacing nulls with the average of the feature.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | accuracy | precision | Recall |
| kNN(k=5) | 96.80% | 91.67% | 78.57% |
| Decision Tree | 99.40% | 98.18% | 96.43% |
| Naïve Bayes | 96.60% | 100.00% | 69.64% |

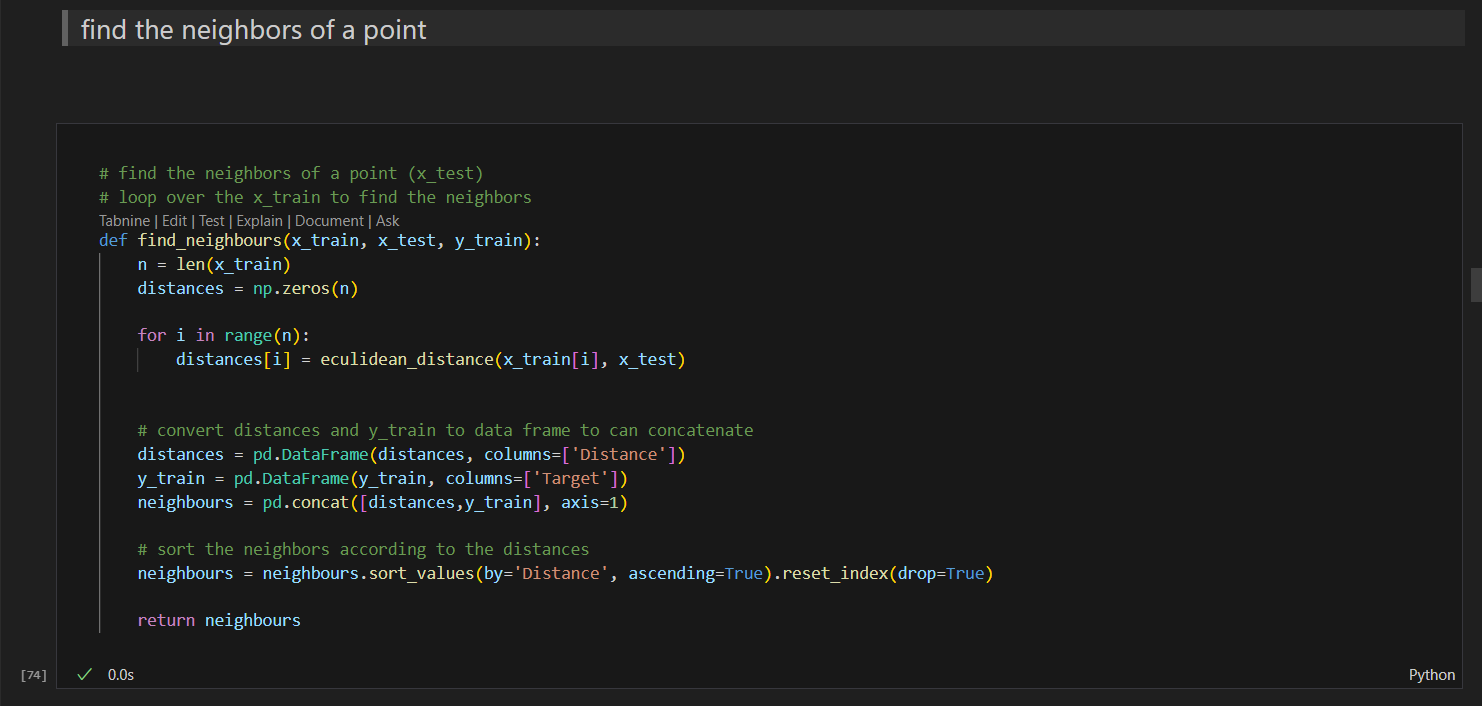


1. Implement k-Nearest Neighbors (kNN) algorithm from scratch.

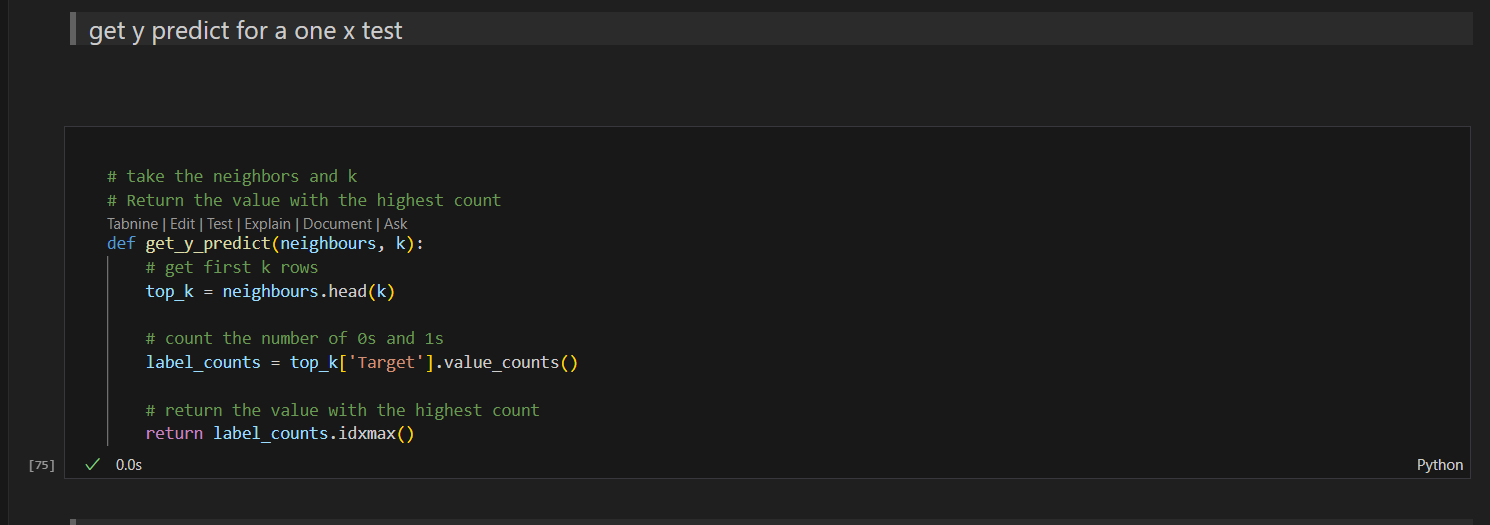
* Function to calculate the Euclidean distance



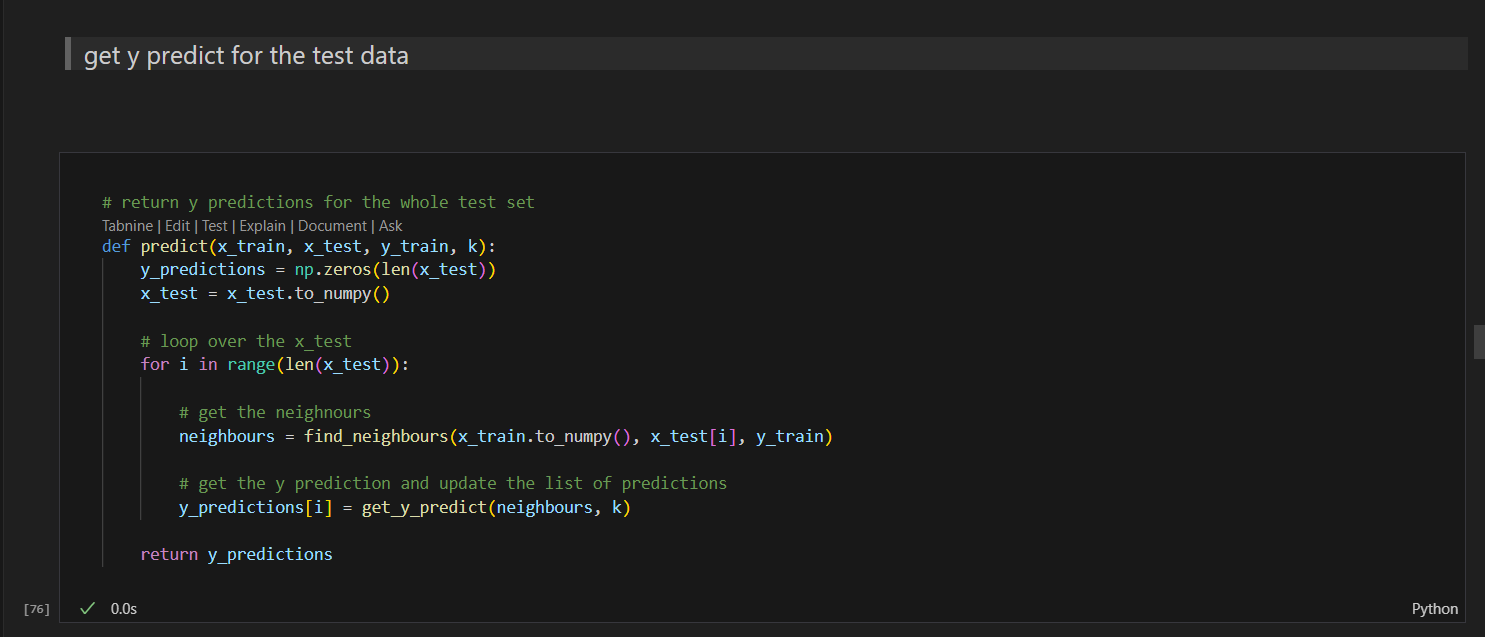
* Function to get the neighbors of one point



* Function to get the y predict for one test value



* The final function that returns the y\_predictions of the x\_test



1. Report the results and compare the performance of your custom kNearest Neighbors (kNN) implementation with the pre-built kNN algorithms in scikit-learn, using the evaluation metrics mentioned in point 2. Using any missing handling techniques, you chose from task 1.2.

* Using replacing nulls with the average of the feature.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | accuracy | precision | recall | |
| kNN(k=5) scikit-learn | 96.80% | 91.67% | | 78.57% |
| kNN(k=5) from scratch | 96.80% | 91.67% | | 78.57% |

A screenshot of a computer

Description automatically generated

Task 3: Interpreting the Decision Tree and Evaluation Metrics Report

1. The effect of different data handling

* Provide a detailed report evaluating the performance of scikitlearn implementations of the Decision Tree, k-Nearest Neighbors (kNN) and naïve Bayes with respect to the different handling missing data technique.
* dropping missing values

|  |  |  |  |
| --- | --- | --- | --- |
| Model | accuracy | precision | recall |
| kNN(k=5) | 95.96% | 88.89% | 82.35% |
| Decision Tree | 100.00% | 100.00% | 100.00% |
| Naïve Bayes | 96.38% | 100.00% | 75.00% |

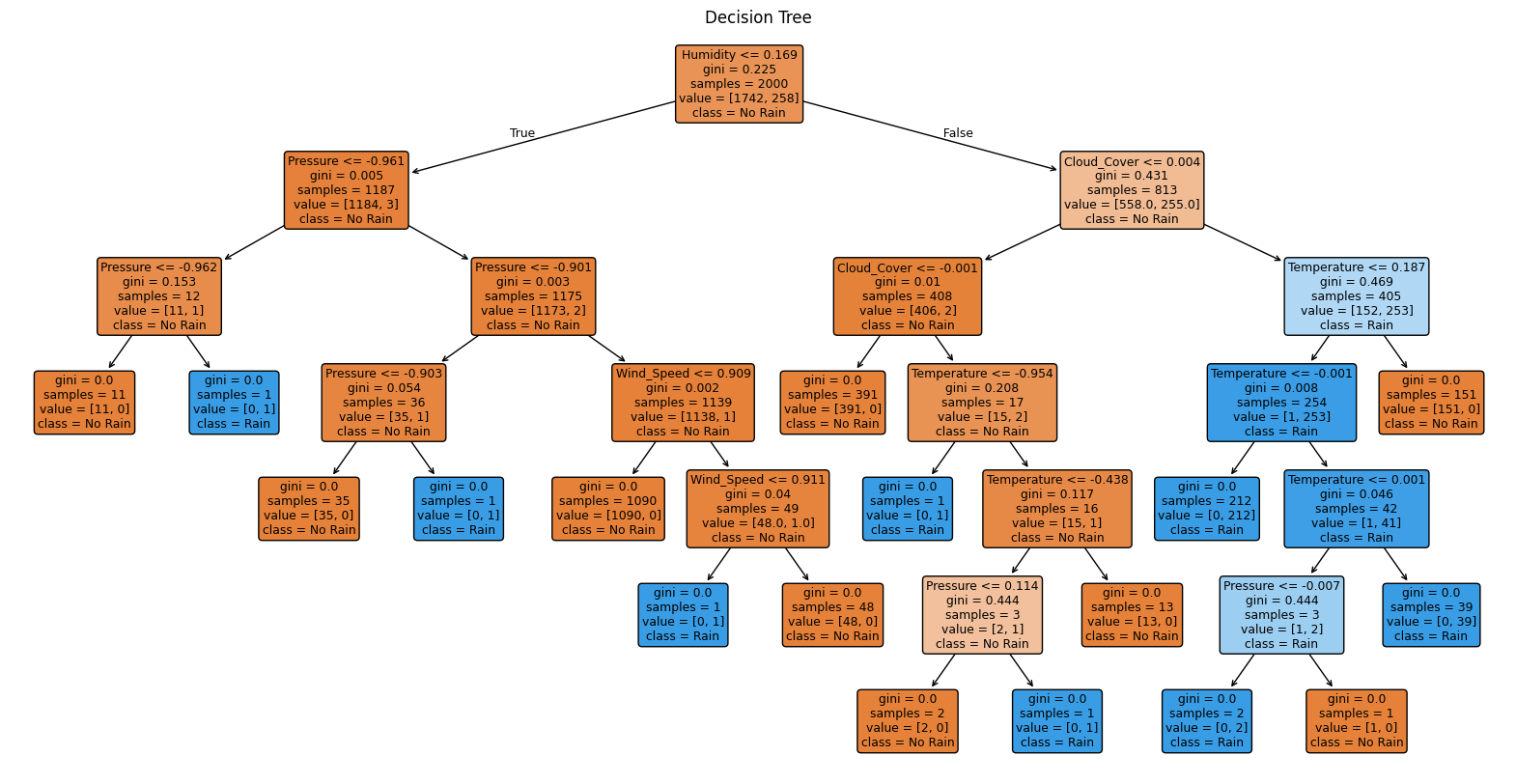
* replacing them with the average

|  |  |  |  |
| --- | --- | --- | --- |
| Model | accuracy | precision | recall |
| kNN(k=5) | 96.80% | 91.67% | 78.57% |
| Decision Tree | 99.40% | 98.18% | 96.43% |
| Naïve Bayes | 96.60% | 100.00% | 69.64% |

* KNN:
* We compare between the two different handling missing data techniques. It seems that the techniques of “replacing them with the average” is better in “accuracy” and “precision” but worst in “recall”.
* So, if you care more about making fewer mistakes when predicting "Rain" choose “replacing them with the average”.
* if you care about never missing a “Rain” choose **“**dropping missing values**”.**
* Decision Tree:
* It seems that the techniques of “dropping missing values” is better at all. So, we must use this technique.
* Naïve Bayes:
* The two techniques are mostly equal in “accuracy” and “precision” but “dropping missing values” is better in “recall” so if you care about never missing a “Rain” choose **“**dropping missing values**”.**

1. 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.



How the Tree Makes Predictions

At each node:

1-Feature Selection: The algorithm chooses the feature fffffffffffthat minimizes the impurity (Gini)

2- Splitting: The data is split into two subsets:

Left branch: Instances satisfying the condition (Feature ≤ ggggggggThreshold)

Right branch: Instances not satisfying the condition gggggggg (Feature > Threshold)

This process repeats until:

A pure leaf node is reached

A maximum depth or minimum samples per node ggggggconstraint is met

* Discuss the criteria and splitting logic used at each node of the tree.

Gini Index: Measures impurity at each node

Gini = 0: Pure node

Gini > 0: Mixed classes

Feature Thresholds: Optimized to split the data such that bbbsubsets have maximum homogeneity

Recursive Splitting: Continues until all nodes are pure or ggggconstraints are met

* Example:

1. Root Node

* The dataset is split into two groups based on whether the humidity is less than or equal to 0.169
  + Left child (True): All samples with Humidity <= 0.169
  + Right child (False): All samples with Humidity > 0.169
* The goal here is to split the data into groups that are more homogeneous in terms of the target variable (Rain/No Rain)

2. Left Child Node (Humidity <= 0.169)

* This node splits further based on the pressure values
  + Left child (True): Samples with Pressure <= -0.961
  + Right child (False): Samples with Pressure > -0.961

3. Right Child Node (Humidity > 0.169)

* The tree splits based on cloud cover
  + Left child (True): Samples with Cloud\_Cover <= 0.004
  + Right child (False): Samples with Cloud\_Cover > 0.004
* This helps the tree further separate the data into more homogeneous subsets.
* The splitting continues down the tree based on various features like Wind Speed, Temperature, and Pressure, with each condition carefully chosen to increase the homogeneity of the resulting child nodes
* Each decision node checks for the best feature (based on Gini Impurity) and the best threshold value that minimizes impurity and improves class separation

1. 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.
* Compare these results with the performance of the corresponding algorithms implemented using scikit-learn.
* replacing them with the average

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | accuracy | precision | | recall |
| KNN(3) from scratch | 97.20% | | 90.38% | 83.93% |
| KNN(5) from scratch | 96.80% | | 91.67% | 78.57% |
| KNN(7) from scratch | 96.40% | | 88.00% | 78.57% |
| KNN(9) from scratch | 96.60% | | 88.24% | 80.36% |
| KNN(11) from scratch | 97.20% | | 90.38% | 83.93% |
| KNN(5) scikit-learn | 96.80% | | 91.67% | 78.57% |
| Decision Tree | 99.40% | | 98.18% | 96.43% |
| Naïve Bayes | 96.60% | | 100.00% | 69.64% |

* By comparing those models, the “accuracy” is close between them. So, it prefers when comparing between them not to include the “accuracy”.
* If you care more about making fewer mistakes when predicting "Rain" choose “Naïve Bayes” because it is the highest in “precision”.
* if you care about never missing a “Rain” choose **“**Decision Tree**”** because it is the highest in “recall”, and any other models will be better than “Naïve Bayes” in this case.
* By looking at the KNN values it is no difference between the “from scratch” and “scikit-learn” with the same “K”.
* KNN with “k = 3” and “k = 11” have the same values and it is better than KNN “scikit-learn” with “k=5”.