Universidad Autónoma de Querétaro
DIVISIÓN DE INVESTIGACIÓN Y POSGRADO

Faculty of Engineering
MCIA

Activity 2

Data Imputation and Normalization

Student:
Juan Manuel
Aviña Muñoz

Professor:
Dr. Marco
Aceves Fernandez

September 22nd, 2023

Contents

1 Introduction

Data imputation and normalization are two fundamental techniques used in data preprocessing and preparation for various data analysis and machine learning tasks. These techniques help improve the quality of data and enhance the performance of models by handling missing values and ensuring that the data is in a suitable format for analysis [1].

1.1 Data Imputation

Data Imputation is the process of filling in missing or incomplete data points in a dataset with estimated or calculated values. Missing data can occur for various reasons, such as measurement errors, data collection issues, or deliberate omissions. Imputation is essential because many machine learning algorithms and statistical methods cannot handle missing data.

Common methods for data imputation include:

- **Mean, Median, or Mode Imputation:** Replace missing values with the mean *(average)*, median *(middle value)*, or mode *(most frequent value)* of the available data in the respective feature.
- **Linear Regression Imputation:** Predict missing values using linear regression models based on other features in the dataset.
- **K-Nearest Neighbors *(KNN)* Imputation:** Estimate missing values by averaging or interpolating from the values of k-nearest neighbors in the feature space.
- **Interpolation:** Use interpolation techniques, such as linear or spline interpolation, to estimate missing values based on adjacent data points.
- **Machine Learning-based Imputation:** Train machine learning models, such as decision trees or random forests, to predict missing values based on other features.

The choice of imputation method depends on the nature of the data and the specific problem you are trying to solve [2].

1.2 Data Normalization

Data normalization, also known as feature scaling or standardization, is the process of rescaling the values of features in a dataset to a common scale without distorting their relative differences. Normalization is crucial because it ensures that all features have equal influence on machine learning models, preventing certain features from dominating others due to their scale [3].

Common methods for data normalization include:

- **Min-Max Scaling:** This method scales the data to a specific range, typically between 0 and 1. It is calculated using the formula:

$$X_{normalized} = \frac{X - X_min}{X_max - X_min} \tag{1}$$

where X is the original value $X_{normalized}$ is the scaled value, X_min is the minimum value in the feature and X_max is the maximum value in the feature

- **Z-Score Standardization:** This method scales data to have a mean (average) of 0 and a standard deviation of 1. It is calculated using the formula:

$$X_{standardized} = \frac{X - \mu}{\sigma} \tag{2}$$

where X is the original feature value, $X_{standardized}$ is the standardized value, μ is the mean of the feature, and σ is the standard deviation of the feature.

- **Robust Scaling:** This method scales data based on the interquartile range *(IQR)* to handle outliers. It is calculated using the formula:

$$X_{scaled} = \frac{X - Q1}{Q3 - Q1} \tag{3}$$

where X is the original feature value, X_{scaled} is the scaled value, $Q1$ is the first quartile and $Q3$ is the third quartile.

Normalization ensures that features with different units and scales can be compared and used effectively in machine learning algorithms, preventing bias toward features with larger values.

2 Theoretical Foundation

Data imputation and normalization are grounded in mathematical and statistical principles. Data imputation methods make use of statistical techniques to estimate missing values, while data normalization involves linear transformations and statistical concepts to rescale features in a way that enhances the performance of various data analysis and machine learning algorithms. Understanding the theoretical foundations of these techniques is essential for selecting the most appropriate methods for your specific data preprocessing needs. Here is a brief overview of these two data preprocessing techniques:

2.1 Data Imputation

- **Statistics:** Data imputation techniques often rely on statistical principles to estimate missing values. Key concepts include:

  - **Central Tendency:** mean, median, and mode imputation are based on measures of central tendency that describe the typical value of a dataset.
  - **Regression Analysis:** employs statistical regression models to predict missing values based on relationships with other variables.
  - **K-Nearest Neighbors:** uses the concept of distance metrics and neighborhood relationships to impute values.

- **Probability and Distribution:** In some cases, imputation methods may involve probability distributions to estimate missing values, particularly when dealing with complex data types. Bayesian statistics can be applied to make probabilistic inferences about missing data.

- **Stochastic Processes:** Time series data imputation may be based on stochastic processes or autoregressive models that account for temporal dependencies.
  - **Roulette data imputation** is a method to impute missing data values using random sampling based on the distribution of the observed data. It's similar to imputing missing data values with random values, but in this case, the randomness is constrained by the distribution of the observed data. This approach ensures that the imputed values follow a distribution similar to the observed data, making it a more realistic imputation method compared to simply filling missing values with fixed values or random values.

2.2 Data Normalization

- **Linear Algebra** Data normalization is rooted in linear algebra concepts. The core idea is to scale or transform data in a way that retains the underlying relationships between data points. For example:
  - **Min-Max Scaling:** Rescaling data to a specific range, often (0, 1) can be seen as a linear transformation applied to each feature.
  - **Z-Score Standardization:** Standardization centers data around the mean and scales it by the standard deviation, which are linear operations.

- **One-hot-encoding:** Generally used when we have a categorical feature with multiple categories and we need to represent each category as a separate binary feature. This transformation allows machine learning algorithms to work with categorical data more effectively, as they typically require numerical inputs.

- **Machine Learning and Optimization:** Normalization is crucial for many machine learning algorithms that rely on optimization techniques, such as gradient descent. Scaling features ensures that the optimization process converges efficiently and prevents certain features from dominating the learning process.

- **Distance Metrics:** In machine learning, distance-based algorithms (e.g., k-means clustering) rely on the Euclidean distance or other distance metrics. Normalizing features ensures that each feature contributes proportionately to the distance calculation.

- **Principal Component Analysis (PCA):** *PCA*, a dimensionality reduction technique, relies on feature scaling to identify the principal components that explain the most variance in the data. Scaling helps ensure that all features contribute fairly to this analysis.

Inconsistently organized data can lead to a range of issues, particularly in the context of a relational database where a logical and efficient design is of utmost importance. A poorly structured database can yield inaccurate information, pose usability challenges, and potentially result in operational failures.

The majority of these issues stem from two design flaws: redundant data and anomalies. Redundant data refers to the presence of unnecessary repetitions or recurring data, often in the form of repeating data groups. Anomalies, on the other hand, encompass any irregularities in data storage that compromise data integrity. Such irregularities can include inconsistent handling of operations like deletion, insertion, and updating, which in turn generate inconsistent data [4].

Data normalization can significantly impact the convergence and stability of optimization algorithms in machine learning. Algorithms like gradient descent perform better on normalized data.

3 Methods and Materials

3.1 Data Set

We have a set of 240 Stars with 5 Types:

- Red Dwarf - 0
- Brown Dwarf - 1
- White Dwarf - 2
- Main Sequence - 3
- Super Giants - 4
- Hyper Giants - 5

Temperature: Temperature in Kelvin.
R: Radius of star relative to the sun.
L: Luminosity of the star relative to the sun.
Absolute Magnitude: Magnitude of the star relative to the sun.

Color: Color of the star.

Spectral Class: An asteroid spectral type is assigned to asteroids based on their emission spectrum, color, and sometimes albedo. These types are thought to correspond to an asteroid's surface composition.

Database obtained from the Kaggle platform [5] sourced from NASA observations.

3.2 Resources

Equipment:

- Intel i7-9750H @ 2.60 GHz processor.
- 16 GB DDR4 @ 2666 MHz.
- 1 Tb HDD + 1 Tb SSD.
- GeForce GTX 1660Ti 6 GB VRAM GDDR6.

3.3 Libraries

The following libraries were utilized for this activity:

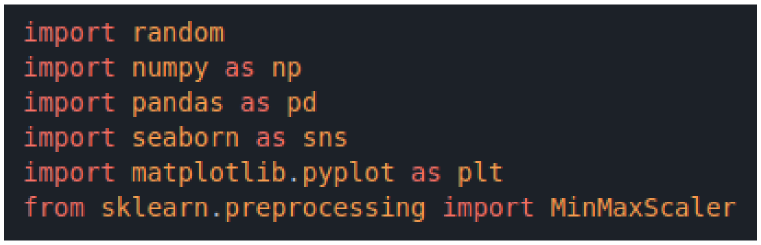
- **Random** is part of the python's standard libraries that provides functions for generating random numbers and perform random operations.
- **NumPy** is a fundamental package for scientific computing in Python as it provides support for large, multi-dimensional arrays and matrices, along with a variety of mathematical functions to operate on these arrays. NumPy is a core library in the data science ecosystem and is used for tasks like numerical computations, linear algebra operations, random number generation, and more.
- **Pandas** is a powerful library for data manipulation and analysis in Python. It provides data structures (primarily Series and DataFrame) to efficiently handle and manipulate structured data. Pandas enables tasks such as loading data from various file formats (CSV, Excel, SQL databases), cleaning and preprocessing data, filtering, merging, transforming, and summarizing data. It's widely used for data wrangling and exploration.
- **Seaborn** is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn simplifies the process of creating complex visualizations like scatter plots, bar plots, histograms, box plots, and more. It also supports color palettes, themes, and additional statistical plotting capabilities.
- **Matplotlib** is a versatile 2D plotting library in Python. It provides a wide range of functions to create static, interactive, and animated visualizations. While it can be used to create basic plots, it can also be customized to create complex visualizations. Matplotlib is the foundation for many other visualization libraries and tools.

- **Scikit-learn** is a machine learning library. It's *preprocessing* module offers various functions for data preprocessing tasks, such as scaling, data imputation, encoding categorical values and much more.

These libraries are essential for various data analysis, data preprocessing, and data visualization tasks in Python.

4 Results

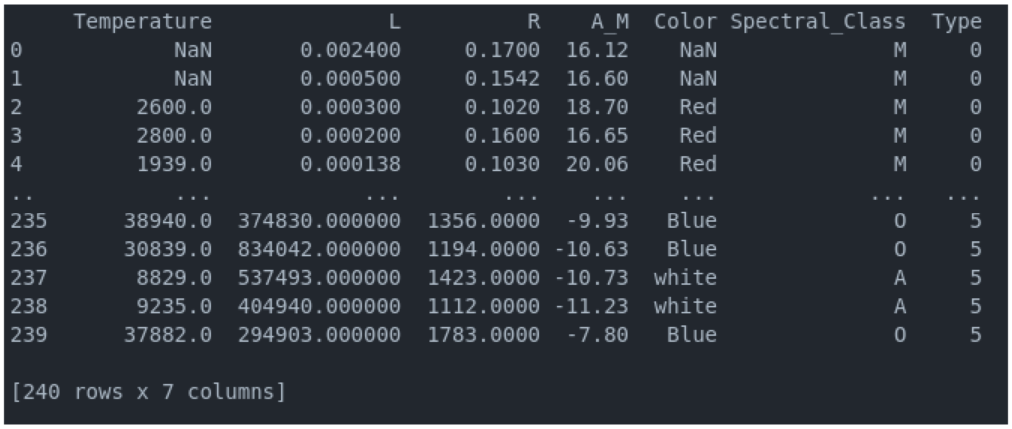
The first step is to import the libraries for this project, as seen on Figure 1.



```
import random
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
```

Figure 1: *Python libraries.*

Next, we import the dataset to be used and display our data, as shown in Figure 2.

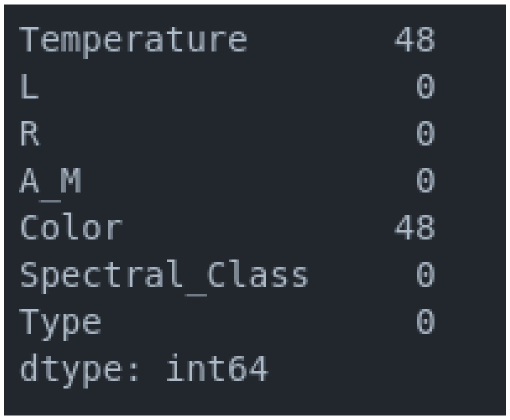


```
    Temperature              L         R    A_M  Color Spectral_Class  Type
0           NaN       0.002400    0.1700  16.12    NaN              M     0
1           NaN       0.000500    0.1542  16.60    NaN              M     0
2        2600.0       0.000300    0.1020  18.70    Red              M     0
3        2800.0       0.000200    0.1600  16.65    Red              M     0
4        1939.0       0.000138    0.1030  20.06    Red              M     0
..          ...            ...       ...    ...    ...            ...   ...
235     38940.0  374830.000000 1356.0000  -9.93   Blue              O     5
236     30839.0  834042.000000 1194.0000 -10.63   Blue              O     5
237      8829.0  537493.000000 1423.0000 -10.73  white              A     5
238      9235.0  404940.000000 1112.0000 -11.23  white              A     5
239     37882.0  294903.000000 1783.0000  -7.80   Blue              O     5

[240 rows x 7 columns]
```

Figure 2: *Values from Data Set.*

The next step involves checking our dataset for any missing values.



```
Temperature       48
L                  0
R                  0
A_M                0
Color             48
Spectral_Class     0
Type               0
dtype: int64
```

Figure 3: `Missing values from Data Set.`

as seen on Figure 3, we observe missing values on the *"Temperature"* and *"Color"* features. To address this issue, we will perform imputation.

However, before proceeding, it's crucial to understand the distribution of our data, as illustrated in Figures 4, 5, 6, and 7.

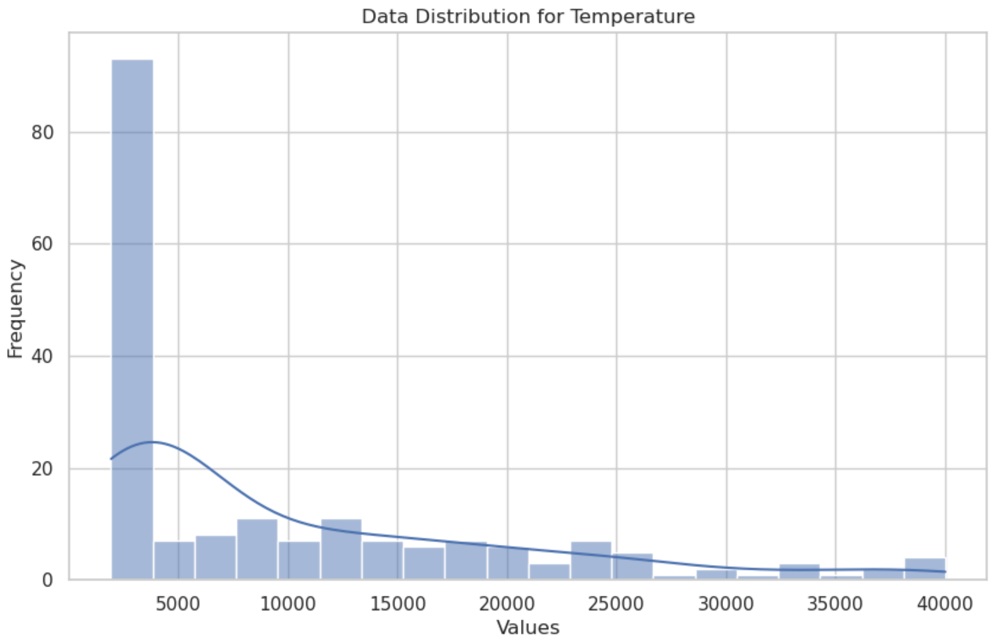
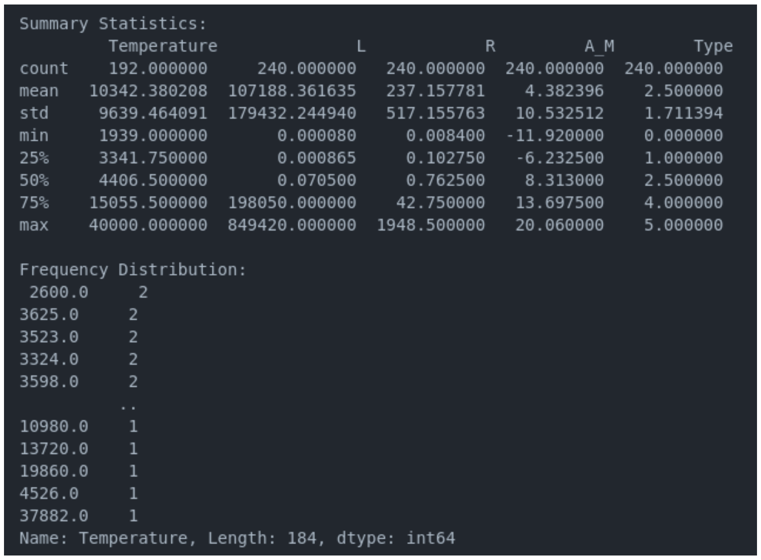


Figure 4: `"Temperature" graphical distribution.`



```
Summary Statistics:
         Temperature              L            R         A_M        Type
count     192.000000     240.000000   240.000000  240.000000  240.000000
mean    10342.380208  107188.361635   237.157781    4.382396    2.500000
std      9639.464091  179432.244940   517.155763   10.532512    1.711394
min      1939.000000       0.000080     0.008400  -11.920000    0.000000
25%      3341.750000       0.000865     0.102750   -6.232500    1.000000
50%      4406.500000       0.070500     0.762500    8.313000    2.500000
75%     15055.500000  198050.000000    42.750000   13.697500    4.000000
max     40000.000000  849420.000000  1948.500000   20.060000    5.000000

Frequency Distribution:
 2600.0     2
3625.0     2
3523.0     2
3324.0     2
3598.0     2
          ..
10980.0    1
13720.0    1
19860.0    1
4526.0     1
37882.0    1
Name: Temperature, Length: 184, dtype: int64
```

Figure 5: *"Temperature" distribution.*

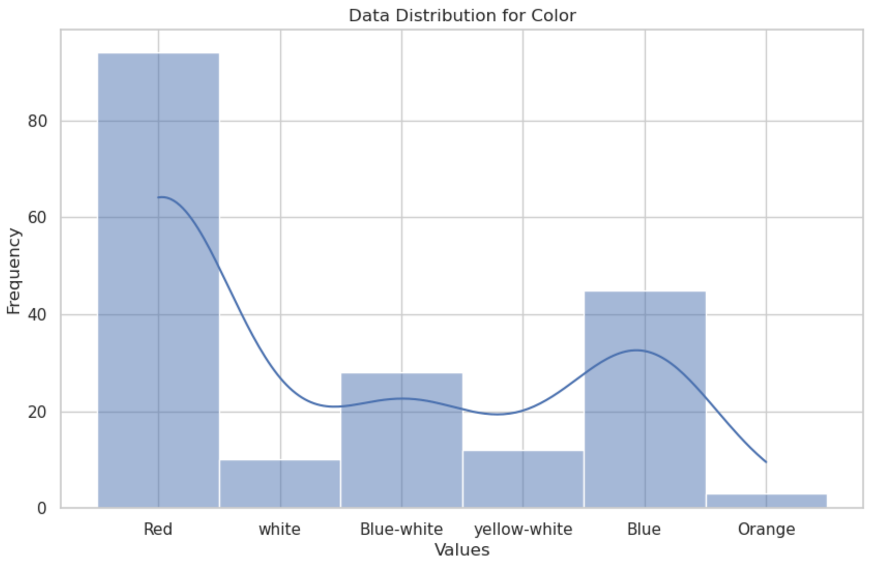
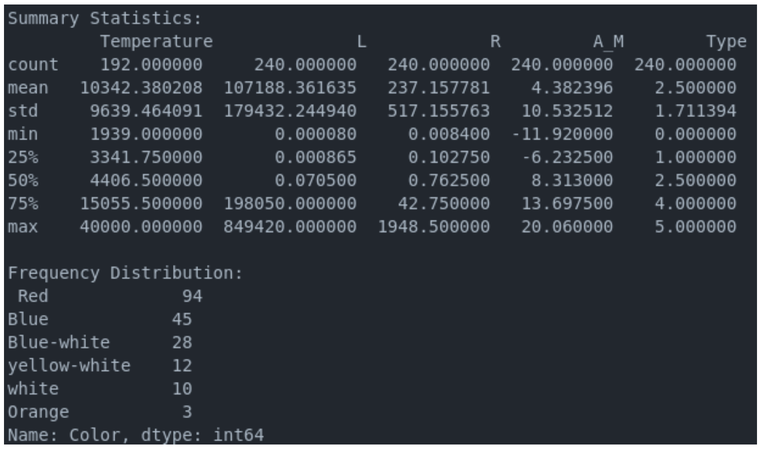


Figure 6: *"Color" graphical distribution.*

4.1 Data Imputation

Now, as we commence the data imputation process, it's essential to bear in mind that our goal is to maintain a distribution that closely mirrors the original one, this is a crucial step to ensure the proper functioning of our future algorithms.

In this project, we opted to utilize the mean values for imputation, as depicted in Figures 8 and 9.



```
Summary Statistics:
         Temperature             L            R         A_M        Type
count     192.000000     240.000000   240.000000  240.000000  240.000000
mean    10342.380208  107188.361635   237.157781    4.382396    2.500000
std      9639.464091  179432.244940   517.155763   10.532512    1.711394
min      1939.000000       0.000080     0.008400  -11.920000    0.000000
25%      3341.750000       0.000865     0.102750   -6.232500    1.000000
50%      4406.500000       0.070500     0.762500    8.313000    2.500000
75%     15055.500000  198050.000000    42.750000   13.697500    4.000000
max     40000.000000  849420.000000  1948.500000   20.060000    5.000000

Frequency Distribution:
 Red              94
Blue             45
Blue-white       28
yellow-white     12
white            10
Orange            3
Name: Color, dtype: int64
```

Figure 7: *"Color" distribution.*

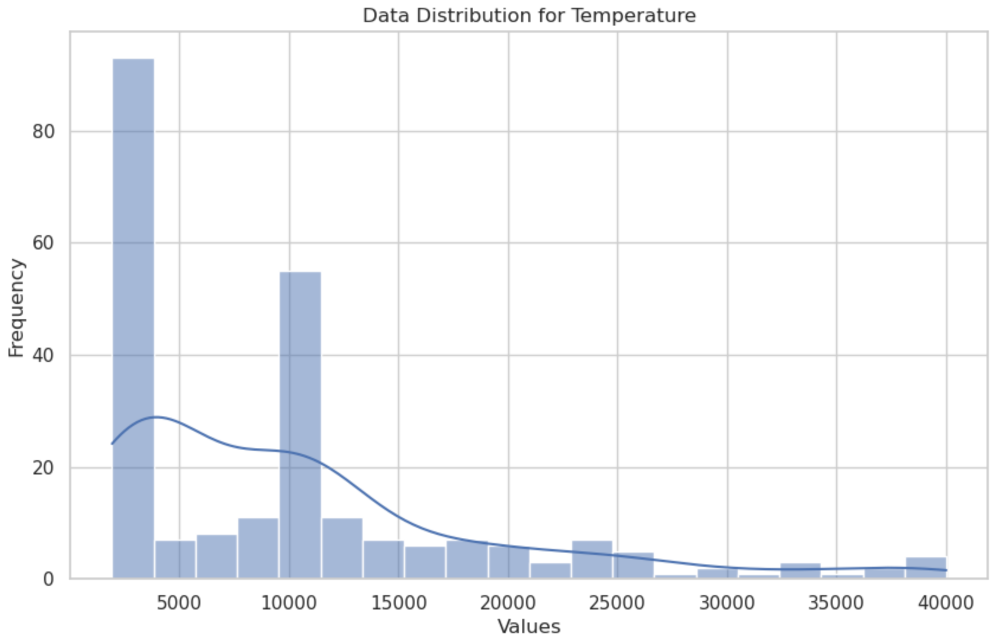
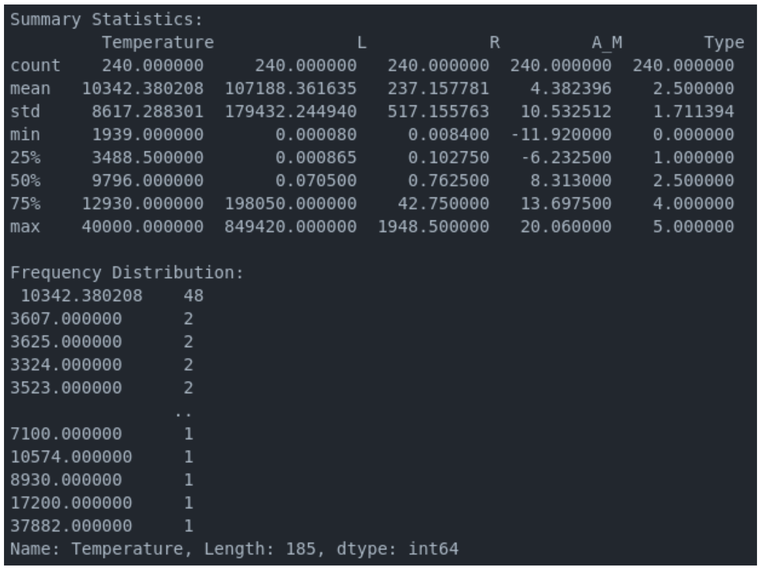


Figure 8: *"Temperature" graphical distribution with mean impute.*



```
Summary Statistics:
        Temperature              L            R         A_M        Type
count    240.000000     240.000000   240.000000  240.000000  240.000000
mean   10342.380208  107188.361635   237.157781    4.382396    2.500000
std     8617.288301  179432.244940   517.155763   10.532512    1.711394
min     1939.000000       0.000080     0.008400  -11.920000    0.000000
25%     3488.500000       0.000865     0.102750   -6.232500    1.000000
50%     9796.000000       0.070500     0.762500    8.313000    2.500000
75%    12930.000000  198050.000000    42.750000   13.697500    4.000000
max    40000.000000  849420.000000  1948.500000   20.060000    5.000000

Frequency Distribution:
 10342.380208    48
3607.000000      2
3625.000000      2
3324.000000      2
3523.000000      2
                ..
7100.000000      1
10574.000000     1
8930.000000      1
17200.000000     1
37882.000000     1
Name: Temperature, Length: 185, dtype: int64
```

Figure 9: *"Temperature" distribution with mean impute.*

Also, we used a random imputation, Figures 10 and 11 just to compare our results.

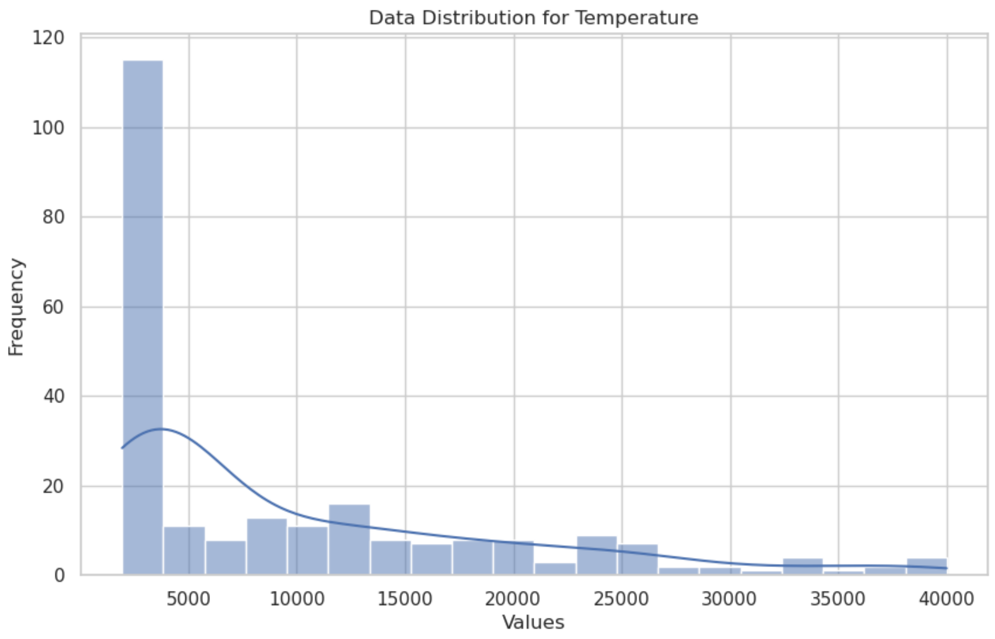
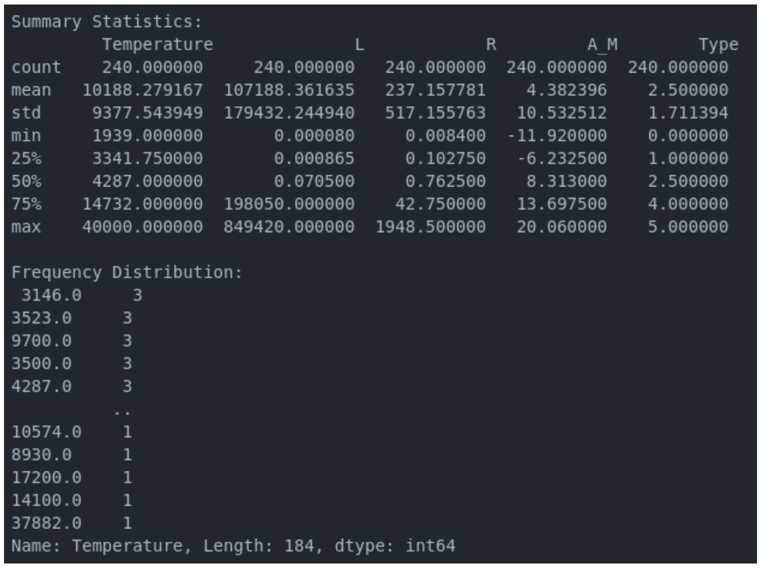


Figure 10: *"Temperature" graphical distribution with random impute.*

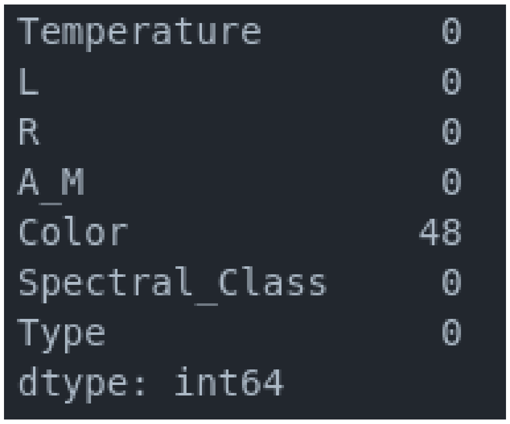


```
Summary Statistics:
         Temperature              L            R         A_M        Type
count    240.000000     240.000000   240.000000  240.000000  240.000000
mean   10188.279167  107188.361635   237.157781    4.382396    2.500000
std     9377.543949  179432.244940   517.155763   10.532512    1.711394
min     1939.000000       0.000080     0.008400  -11.920000    0.000000
25%     3341.750000       0.000865     0.102750   -6.232500    1.000000
50%     4287.000000       0.070500     0.762500    8.313000    2.500000
75%    14732.000000  198050.000000    42.750000   13.697500    4.000000
max    40000.000000  849420.000000  1948.500000   20.060000    5.000000

Frequency Distribution:
 3146.0     3
3523.0     3
9700.0     3
3500.0     3
4287.0     3
          ..
10574.0    1
8930.0     1
17200.0    1
14100.0    1
37882.0    1
Name: Temperature, Length: 184, dtype: int64
```

Figure 11: *"Temperature" distribution with random impute.*

To verify the successful completion of the process, we examine the presence of any missing values within the *"Temperature"* feature, as indicated in Figure 12.



```
Temperature        0
L                  0
R                  0
A_M                0
Color             48
Spectral_Class     0
Type               0
dtype: int64
```

Figure 12: *Missing values after data imputation.*

The next step is to impute our categorical data from the *"Color"* feature, first by using the mode values, Figures 13 and 14, since it still has missing values.

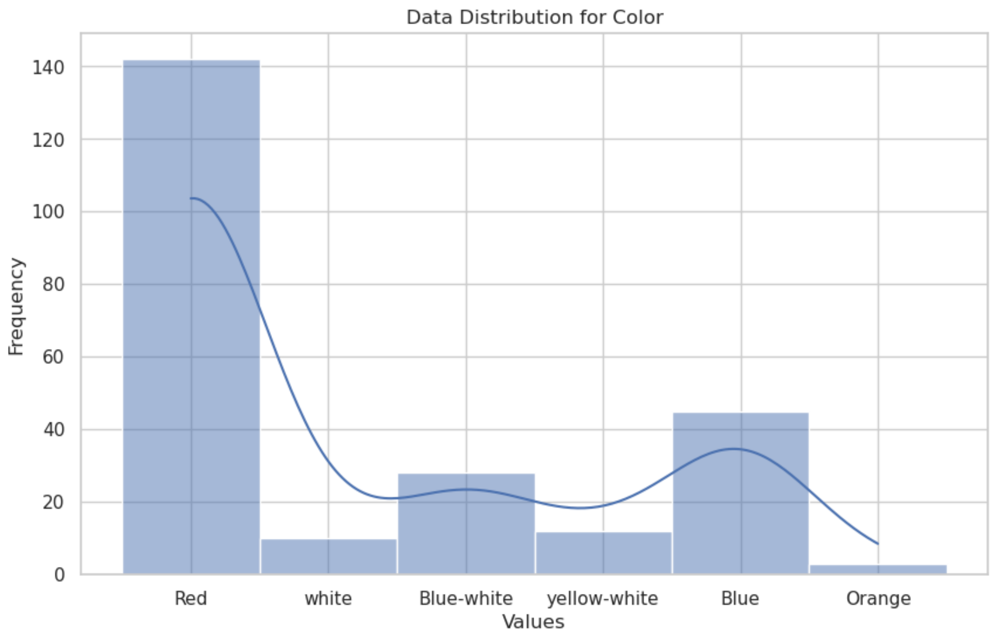
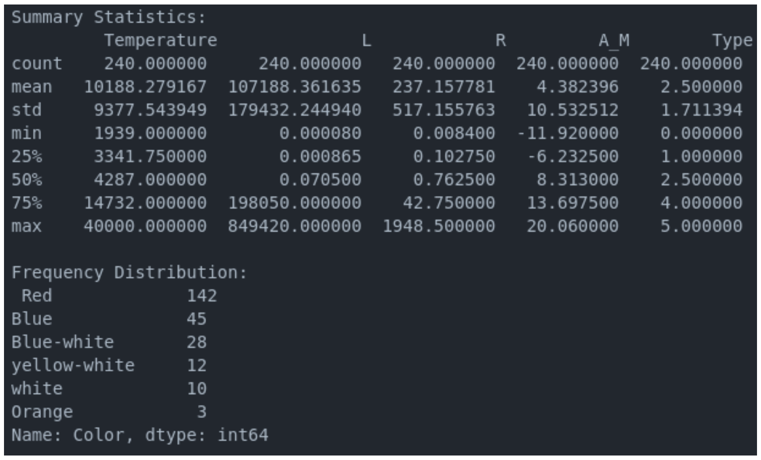


Figure 13: *"Color" graphical distribution with mode impute.*



```
Summary Statistics:
        Temperature              L           R         A_M        Type
count    240.000000     240.000000  240.000000  240.000000  240.000000
mean   10188.279167  107188.361635  237.157781    4.382396    2.500000
std     9377.543949  179432.244940  517.155763   10.532512    1.711394
min     1939.000000       0.000080    0.008400  -11.920000    0.000000
25%     3341.750000       0.000865    0.102750   -6.232500    1.000000
50%     4287.000000       0.070500    0.762500    8.313000    2.500000
75%    14732.000000  198050.000000   42.750000   13.697500    4.000000
max    40000.000000  849420.000000 1948.500000   20.060000    5.000000

Frequency Distribution:
 Red              142
Blue               45
Blue-white         28
yellow-white       12
white              10
Orange              3
Name: Color, dtype: int64
```

Figure 14: *"Color" distribution with mode impute.*

Now, for comparison, we use a random imputation method, seen on Figures 15 and 16.

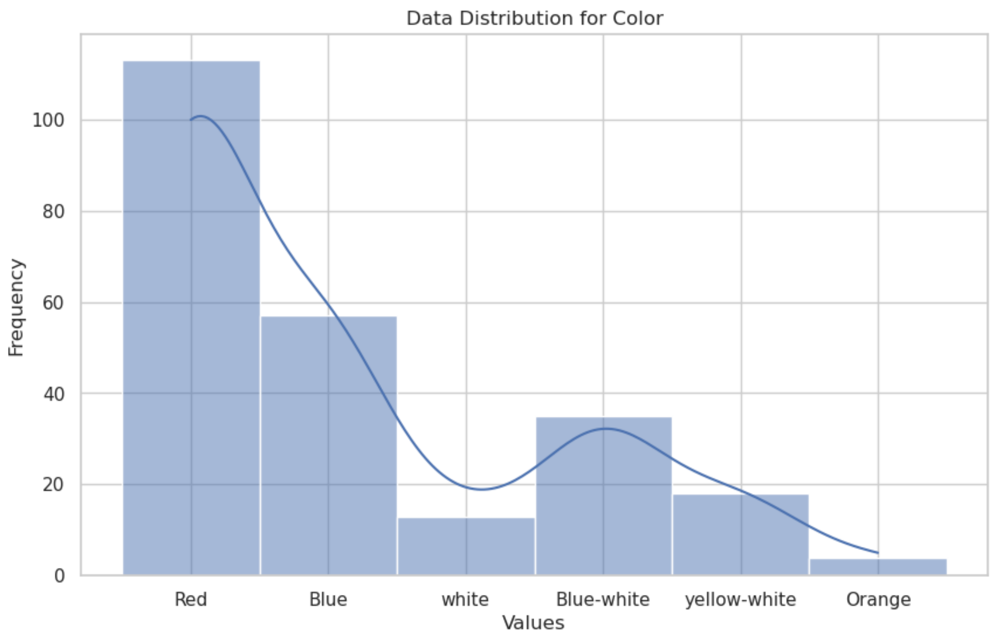
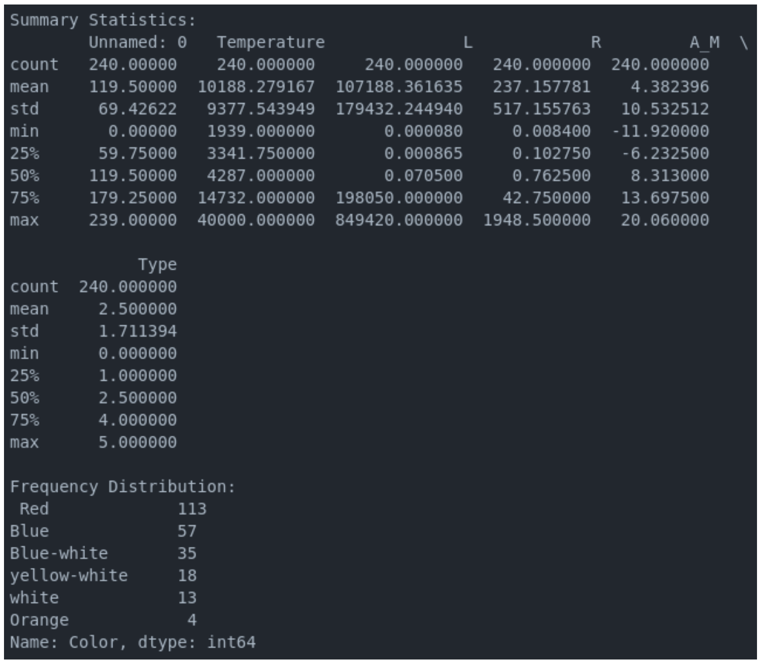


Figure 15: *"Color" graphical distribution with random impute.*



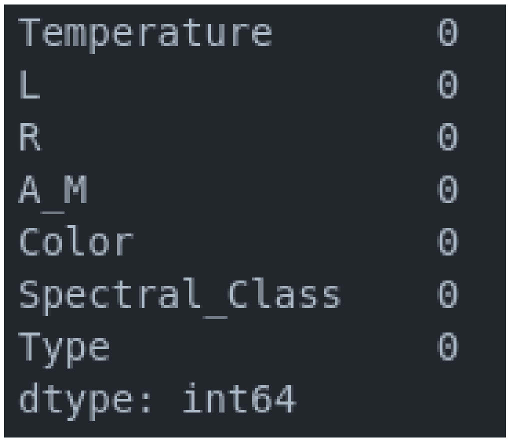
```
Summary Statistics:
        Unnamed: 0   Temperature              L           R         A_M  \
count   240.00000    240.000000     240.000000   240.000000  240.000000
mean    119.50000  10188.279167  107188.361635   237.157781    4.382396
std      69.42622   9377.543949  179432.244940   517.155763   10.532512
min       0.00000   1939.000000       0.000080     0.008400  -11.920000
25%      59.75000   3341.750000       0.000865     0.102750   -6.232500
50%     119.50000   4287.000000       0.070500     0.762500    8.313000
75%     179.25000  14732.000000  198050.000000    42.750000   13.697500
max     239.00000  40000.000000  849420.000000  1948.500000   20.060000

             Type
count  240.000000
mean     2.500000
std      1.711394
min      0.000000
25%      1.000000
50%      2.500000
75%      4.000000
max      5.000000

Frequency Distribution:
 Red             113
Blue             57
Blue-white       35
yellow-white     18
white            13
Orange            4
Name: Color, dtype: int64
```

Figure 16: *"Color" distribution with random impute.*

To validate the correctness of our imputation method, we inspect the dataset for any missing values within the *"Color"* feature, as illustrated in Figure 17.

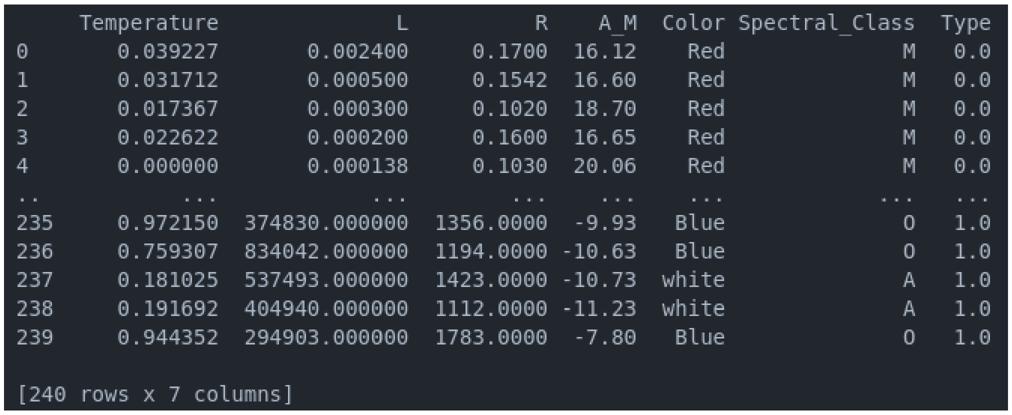


```
Temperature       0
L                 0
R                 0
A_M               0
Color             0
Spectral_Class    0
Type              0
dtype: int64
```

Figure 17: *Missing values after data imputation.*

4.2 Data Normalization

With our data now properly imputed, we proceed to perform data normalization for two of our features, *"Temperature"* and *"Type"*, Figure 18.



```
     Temperature             L          R    A_M  Color Spectral_Class  Type
0       0.039227      0.002400     0.1700  16.12    Red              M   0.0
1       0.031712      0.000500     0.1542  16.60    Red              M   0.0
2       0.017367      0.000300     0.1020  18.70    Red              M   0.0
3       0.022622      0.000200     0.1600  16.65    Red              M   0.0
4       0.000000      0.000138     0.1030  20.06    Red              M   0.0
..           ...           ...        ...    ...    ...            ...   ...
235     0.972150  374830.000000  1356.0000  -9.93   Blue              O   1.0
236     0.759307  834042.000000  1194.0000 -10.63   Blue              O   1.0
237     0.181025  537493.000000  1423.0000 -10.73  white              A   1.0
238     0.191692  404940.000000  1112.0000 -11.23  white              A   1.0
239     0.944352  294903.000000  1783.0000  -7.80   Blue              O   1.0

[240 rows x 7 columns]
```

Figure 18: *Data set after min-max normalization.*

First, the *"Temperature"* feature, as shown on Figures 19 and 20.

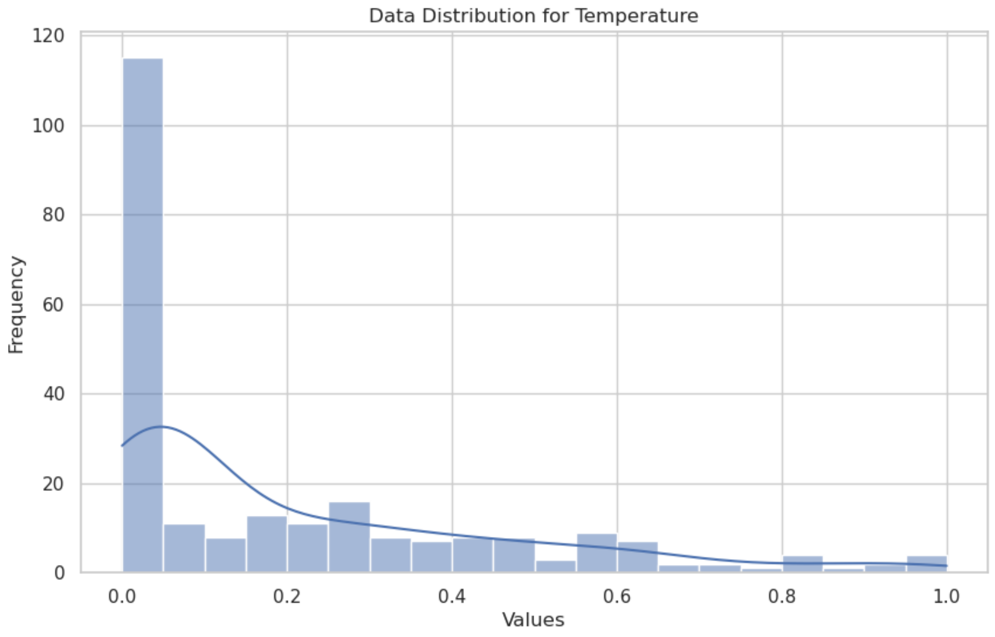
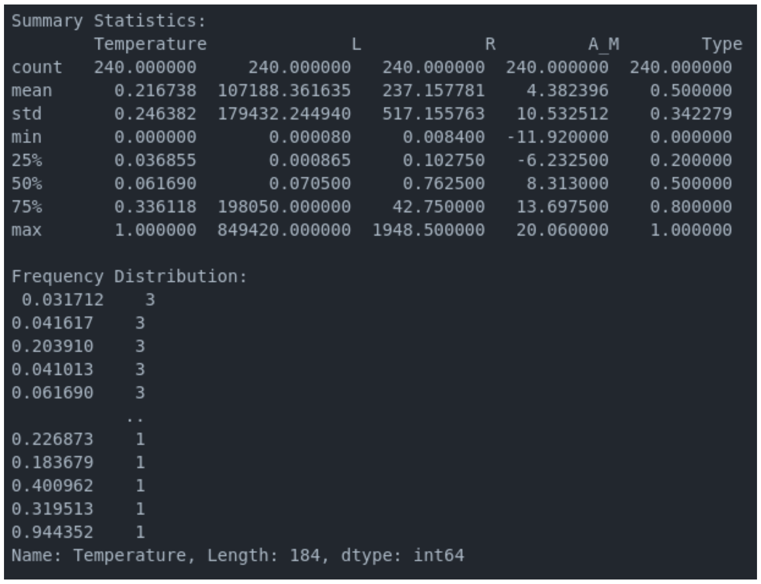


Figure 19: *"Temperature" graphical distribution with min-max normalization.*



```
Summary Statistics:
        Temperature              L            R         A_M        Type
count    240.000000     240.000000   240.000000  240.000000  240.000000
mean       0.216738  107188.361635   237.157781    4.382396    0.500000
std        0.246382  179432.244940   517.155763   10.532512    0.342279
min        0.000000       0.000080     0.008400  -11.920000    0.000000
25%        0.036855       0.000865     0.102750   -6.232500    0.200000
50%        0.061690       0.070500     0.762500    8.313000    0.500000
75%        0.336118  198050.000000    42.750000   13.697500    0.800000
max        1.000000  849420.000000  1948.500000   20.060000    1.000000

Frequency Distribution:
 0.031712    3
0.041617    3
0.203910    3
0.041013    3
0.061690    3
           ..
0.226873    1
0.183679    1
0.400962    1
0.319513    1
0.944352    1
Name: Temperature, Length: 184, dtype: int64
```

Figure 20: *"Temperature" distribution with min-max normalization.*

Lastly, we address the *”Type”* feature, as indicated in Figures 21 and 22. This feature exhibits values ranging from 1 to 5, as illustrated in Figure 2.

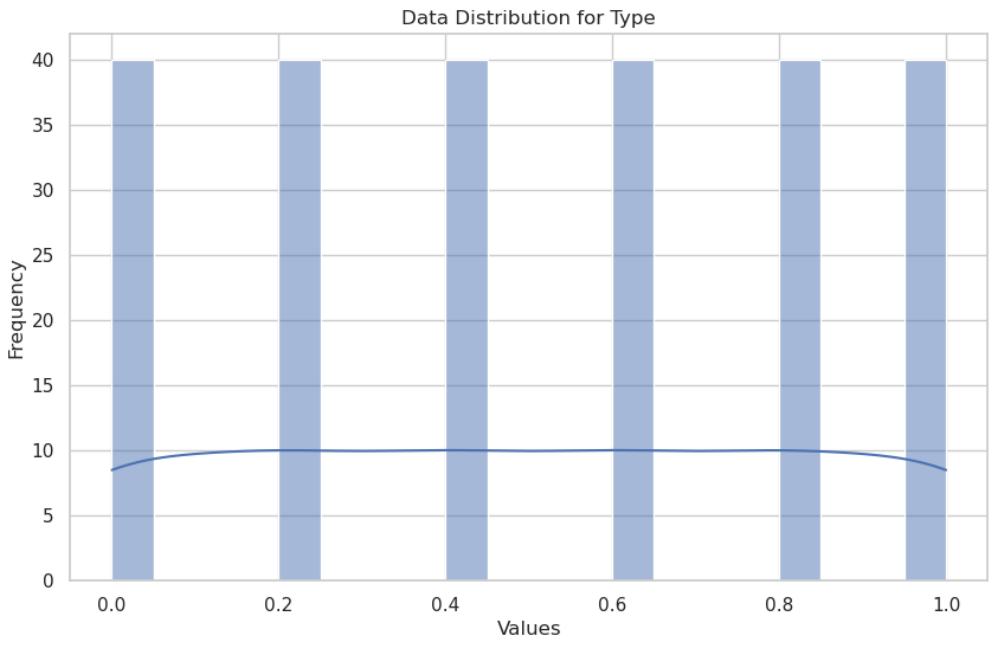
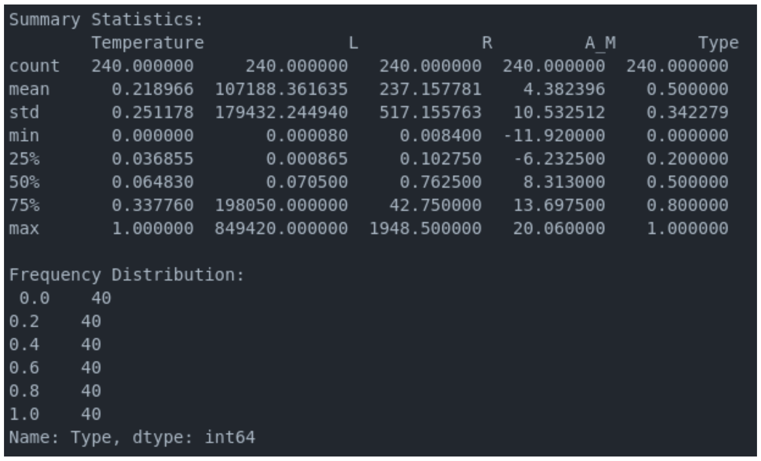


Figure 21: *"Type" graphical distribution with min-max normalization.*



```
Summary Statistics:
       Temperature              L            R          A_M        Type
count   240.000000     240.000000   240.000000  240.000000  240.000000
mean      0.218966  107188.361635   237.157781    4.382396    0.500000
std       0.251178  179432.244940   517.155763   10.532512    0.342279
min       0.000000       0.000080     0.008400  -11.920000    0.000000
25%       0.036855       0.000865     0.102750   -6.232500    0.200000
50%       0.064830       0.070500     0.762500    8.313000    0.500000
75%       0.337760  198050.000000    42.750000   13.697500    0.800000
max       1.000000  849420.000000  1948.500000   20.060000    1.000000

Frequency Distribution:
 0.0    40
0.2    40
0.4    40
0.6    40
0.8    40
1.0    40
Name: Type, dtype: int64
```

Figure 22: `"Type" distribution with min-max normalization.`

5 Discussion

As observed in Figures 8 and 9, utilizing the mean for numerical values in our dataset is not suitable due to the imbalanced distribution. Instead, to preserve a distribution closely resembling the original, we opt for randomized imputation.

In other words, the numerical values are not evenly spread across the range but are skewed or concentrated in certain areas. To address this issue, the chosen approach is randomized imputation. Instead of simply using the mean value, the imputation method introduces randomness to generate values that closely resemble the original distribution, as demonstrated in Figures 10 and 11.

Moreover, when dealing with categorical values, Figures 13 and 14 highlight that relying on the mode for imputation fails to maintain the correct distribution. Consequently, we opt for a random imputation strategy, as exemplified in Figures 15 and 16. This choice is based on the observation that our feature displays an imbalance, with the majority of values concentrated within a single category.

This means that the imputed values would not accurately reflect the diversity of categories present in the original data. To address this issue, a random imputation strategy is chosen. This decision is motivated by the fact that the categorical feature has an imbalance, with the majority of values falling into one category. Random imputation helps preserve diversity in the imputed values.

The choice of imputation method is driven by the desire to preserve the statistical characteristics and distribution of the original data as closely as possible, ensuring that the imputed values are a representative reflection of the missing data, which is a crucial consideration for maintaining the integrity of the dataset for future analysis or modeling [6].

Given the wide range of values observed in the *”Temperature”* feature, as observed in Figure 2, it becomes necessary to normalize our data.

Normalization, as illustrated in Figure 18, is essential for ensuring equal weighting of features, scaling them within the range of 0 to 1. This step is crucial because many machine learning algorithms and mathematical models operate under the assumption that all features carry equal significance. Without normalization, features with larger scales may dominate and lead to biased results.

Furthermore, normalization plays a vital role in ensuring the resilience of future models to outliers and extreme values. Outliers within unscaled features have the potential to exert a disproportionate impact on model performance, by normalizing the data, we avoid this concern, as it serves as a remedy for future issues.

The decision to normalize the data within the *"Type"* feature, as illustrated in Figures 21 and 22, is primarily driven by the same motivation, ensure that all features are on a level playing field, thereby enhancing the robustness and fairness of our modeling process.

By normalizing our data we ensure that our data set is prepared for analysis and modeling in a way that avoids biases, improves the performance of machine learning algorithms, and aids in the interpretation of results. It's a fundamental step in data preprocessing to ensure that the underlying relationships in the data are appropriately represented .

6 Conclusions

Data preprocessing, including imputation and normalization, plays a critical role in preparing data for analysis. It ensures that the dataset is in a suitable state for modeling or further exploration.

When performing data imputation, is necessary to consider methods that maintain the characteristics and relationships in our data, whether through statistical measures or machine learning-based techniques.

Feature scaling through normalization ensures that each feature contributes proportionately to the analysis, preventing issues like feature dominance.

Imputation and normalization can lead to improved data quality, making it more suitable for accurate analysis and modeling. The choice of imputation and normalization methods can impact the performance of machine learning models. It's of great importance to be mindful of how these preprocessing steps may affect model outcomes.

It is essential to document the specific techniques, parameters, and rationale behind our data preprocessing steps. This transparency aids in reproducibility and collaboration. Since data preprocessing is often an iterative process, it's necessary to be prepared to revisit and refine the techniques used as we gain more insights into our data and the analysis goals evolve.

In summary, data imputation and normalization are critical components of the data preprocessing pipeline, they contribute to the quality, consistency, and suitability of our data for analysis and modeling, ultimately impacting the success of our project. Effective handling of missing data and proper scaling of features are fundamental steps in the data analysis workflow.

References

[1] L. Pierson, *Data Science for Dummies*. Wiley, 2015.

[2] SimpliLearn, “Introduction to Data Imputation,” 2023, SimpliLearn. [Online]. Available: https://www.simplilearn.com/data-imputation-article

[3] D. Little, R. Rubin, *Statistical Analysis with Missing Data*, 3rd ed. Wiley, 2019.

[4] IT 240 Course, “Normalization: Relational database normalization process,” 2023, DePaul University. [Online]. Available: https://condor.depaul.edu/gandrus/240IT/accesspages/normalization3.htm

[5] B. Dincer, “Star Type Classification/NASA,” 2021, Kaggle. [Online]. Available: https://www.kaggle.com/datasets/brsdincer/star-type-classification

[6] W. Badr, “6 different ways to compensate for missing values in a dataset (data imputation with examples),” 2019, Towards Data Science. [Online]. Available: https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779

[7] M. Aceves, *Inteligencia Artificial Para Programadores Con Prisa*. Universo de Letras, 2021.

[8] W. Shahzad, Q. Rehman, and E. Ahmed, “Missing Data Imputation using Genetic Algorithm for Supervised Learning,” *International Journal of Advanced Computer Science and Applications*, vol. 8, 2017.

[9] scikit-learn developers, “6.4 imputation of missing values,” 2023, Scikit Learn. [Online]. Available: https://scikit-learn.org/stable/modules/impute.html