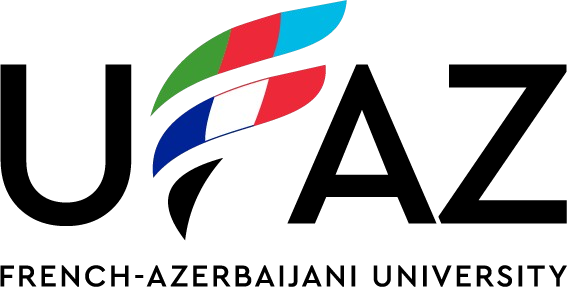
DSAI-22 and PCCE-22 (2022-2024)

**Project Work:** Data Mining

**Professors:** Ahmed Samet, Samer El Zant

**Students:** Elnur Maharramov, Parvin Bayramov, Mirhashim Seyidov



# Pattern mining project

**Introduction**

In Literature review part general informations are given about Data mining what Data mining requires and informations are given about dataset such as which columns dataset has, how many rows it has.

The project is organized into several sections. We will begin by describing the data collection and preparation process, followed by an exploratory data analysis to gain a better understanding of the dataset. And here, we will fill null values this part belongs to Data imputation part.

We will then select and engineer relevant features from the dataset, build a model using data mining techniques, and evaluate its performance. Finally, we will interpret and discuss the results. This part belongs to Classification part.

**Literature review**

Data mining is the process of finding useful information hidden within large amounts of data. It involves using statistical and computational methods to identify patterns and relationships, which can then be used to make predictions or improve decision-making. Data mining is commonly used in business, finance, healthcare, and marketing to uncover insights and improve outcomes. It involves several steps, such as data cleaning, exploration, modeling, and interpretation, and uses techniques like clustering, classification, and regression. With the increasing availability of big data, data mining has become an essential tool for organizations looking to extract insights and make better decisions. Data mining requires a combination of technical skills and domain knowledge. Technical skills include proficiency in statistical and computational methods, database management, data visualization, and programming languages such as Python or R. Domain knowledge is also crucial, as it enables the data miner to understand the context and meaning of the data being analyzed.

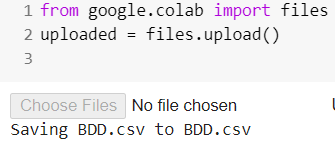
This project focuses on exploring a chemical dataset using data mining methods. The dataset is concerned with calculating the physico-chemical property of an amphiphilic molecule. It consists of 18 attributes and 199 entries, and was created by reviewing several scientific papers with the input of experts. The dataset contains information related to the structural characteristics and physico-chemical properties of the molecules. Our objective is to analyze this dataset using data mining techniques to extract valuable insights and knowledge.

Names of columns are 'Name', 'Formula', 'Family', 'Molar mass', 'Head Family', 'Head', 'Cyclic Head (Y/N)', 'Sugar number in head', 'Carbon number', 'Saturated', 'Total Number of Ramified Carbons', 'Multichain', 'JunTyp', 'Jun Direction', 'CMC (mM)', 'Surface tension (CMC)', 'Amin', 'Efficiency C20', 'Nagg'. And means of these columns:

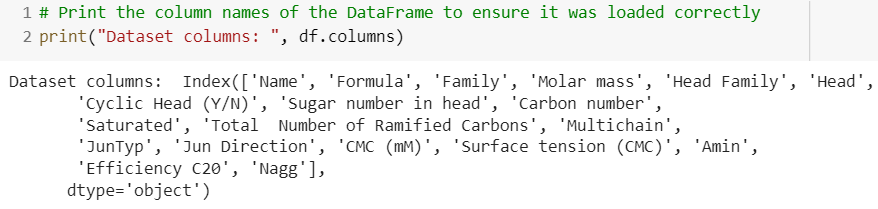
* 'Name': Refers to the name of the molecule.
* 'Formula': Refers to the chemical formula of the molecule.
* 'Family': Refers to the family or class of the molecule.
* 'Molar mass': Refers to the molar mass of the molecule.
* 'Head Family': Refers to the family or class of the head group of the molecule.
* 'Head': Refers to the head group of the molecule.
* 'Cyclic Head (Y/N)': Indicates whether the head group of the molecule is cyclic or not.
* 'Sugar number in head': Refers to the number of sugar units in the head group of the molecule.
* 'Carbon number': Refers to the number of carbon atoms in the molecule.
* 'Saturated': Indicates whether the molecule is saturated or unsaturated.
* 'Total Number of Ramified Carbons': Refers to the total number of ramified (branched) carbon atoms in the molecule.
* 'Multichain': Indicates whether the molecule has multiple chains or not.
* 'JunTyp': Refers to the type of junction between the chains in the molecule.
* 'Jun Direction': Indicates the direction of the junction between the chains in the molecule.
* 'CMC (mM)': Refers to the critical micelle concentration (CMC) of the molecule in millimoles per liter.
* 'Surface tension (CMC)': Refers to the surface tension of the molecule at its CMC.
* 'Amin': Refers to the amin number of the molecule.
* 'Efficiency C20': Refers to the efficiency of the molecule in reducing surface tension at a concentration of 20 mg/L.
* 'Nagg': Refers to the number of molecules in the aggregate.

**Data imputation**

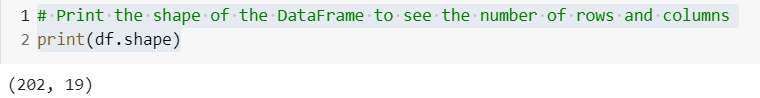
For coding we used google colab and import dataset into google colab by using this code:



Then we import libraries which are needed, and we load dataset by using pandas library, here error occurred and we fixed it by “sep” when import csv file:

To verify that the dataset was loaded correctly, we printed the column names of the “df” using the code "print("Dataset columns: ", df.columns)". This code outputs the column names of the “df” and allows us to check that the columns were loaded correctly. The printed column names can be used as a reference throughout the project to ensure that the correct columns are being used in the analysis.:

We used the code "print(df.shape)" to print the number of rows and columns in the “df”, giving us an idea of the size of the dataset, and here there are 202 rows and we defined three of them are duplicated rows and we removed them:



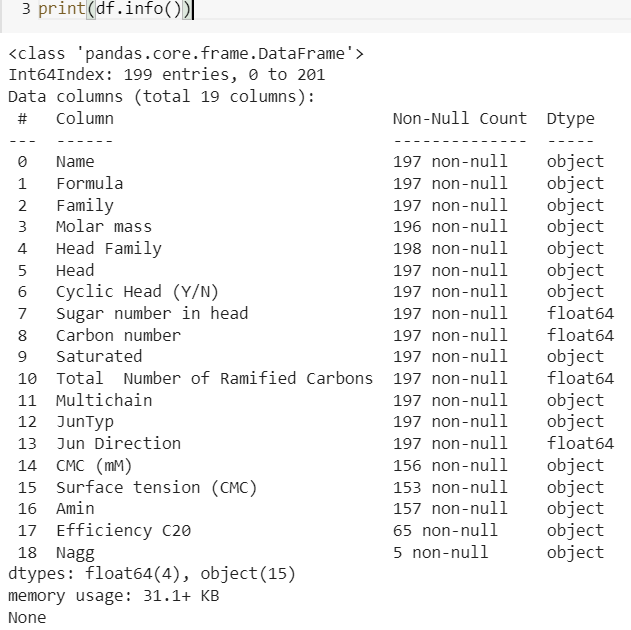
Graphical user interface

Description automatically generated with low confidence

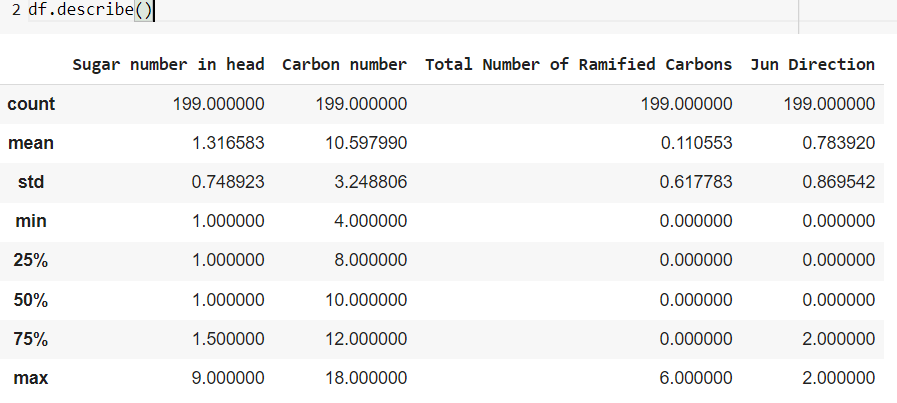
Graphical user interface, application

Description automatically generated

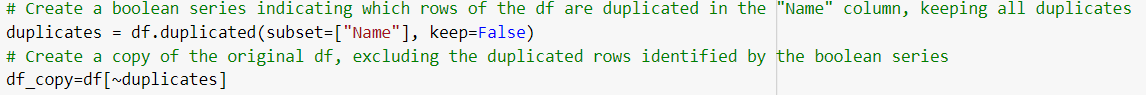
We used the code "print(df.info())" to get a concise summary of “df”, including data types, number of non-null values, and memory usage, which is useful for identifying potential data quality issues and understanding the data types of each column:



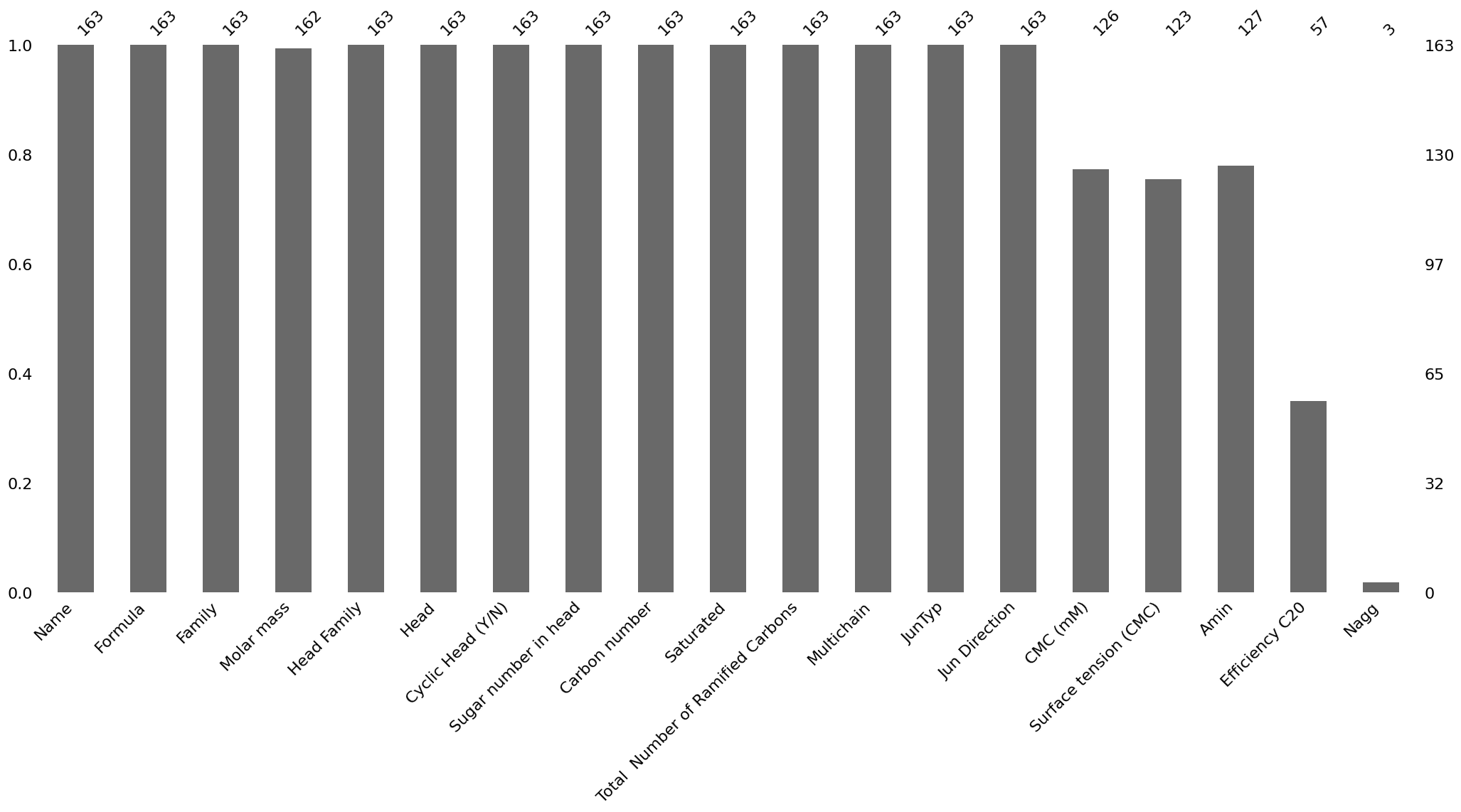
We used the code "df.describe()" to obtain key statistical information about the dataset, including count, mean, standard deviation, minimum, maximum, and quartile values for the numeric columns. This provides insights into the central tendency, variability, and distribution of the data:



We used the code "duplicates = df.duplicated(subset=["Name"], keep=False)" and "df\_copy=df[~duplicates]" to identify and remove duplicated rows in the "Name" column of the DataFrame, ensuring that the resulting DataFrame does not contain any duplicates in the "Name" column:



To visualize the missing values in the “df”, we used the code "msno.bar(df\_copy)", which produces a graphical representation of the missing values in “df”. This helps in identifying potential data quality issues, such as missing or incomplete data, which can impact the accuracy and validity of the data mining results:

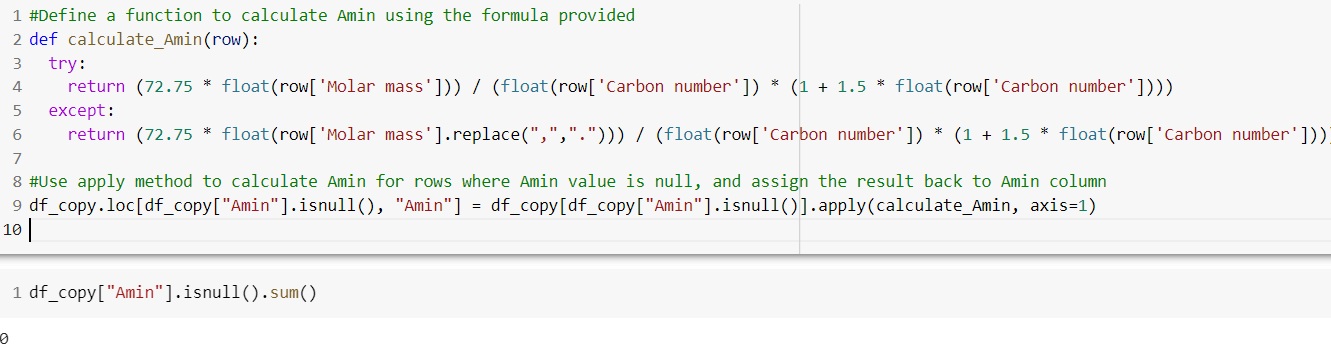


We filled the missing values in the "Molar mass" column of the DataFrame by calculating them using a formula and filling them in using the "fillna" method. This ensures that the resulting DataFrame does not contain any missing values in the "Molar mass" column:

Graphical user interface, text, application

Description automatically generated

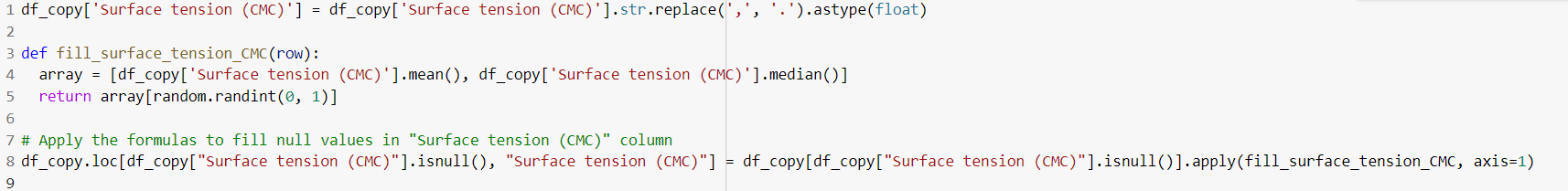
We defined a function "calculate\_Amin" to calculate the "Amin" values using the provided formula. Then, we used the "apply" method to apply the function to rows where the "Amin" value is null, and assigned the result back to the "Amin" column. This ensures that the resulting “df\_copy” does not contain any missing values in the "Amin" column:

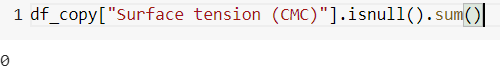


Fill missing values of “CMC” using a function that calculates CMC based on the “Amin” value and other molecular characteristics such as the carbon number, and apply it to the rows where CMC value is null:

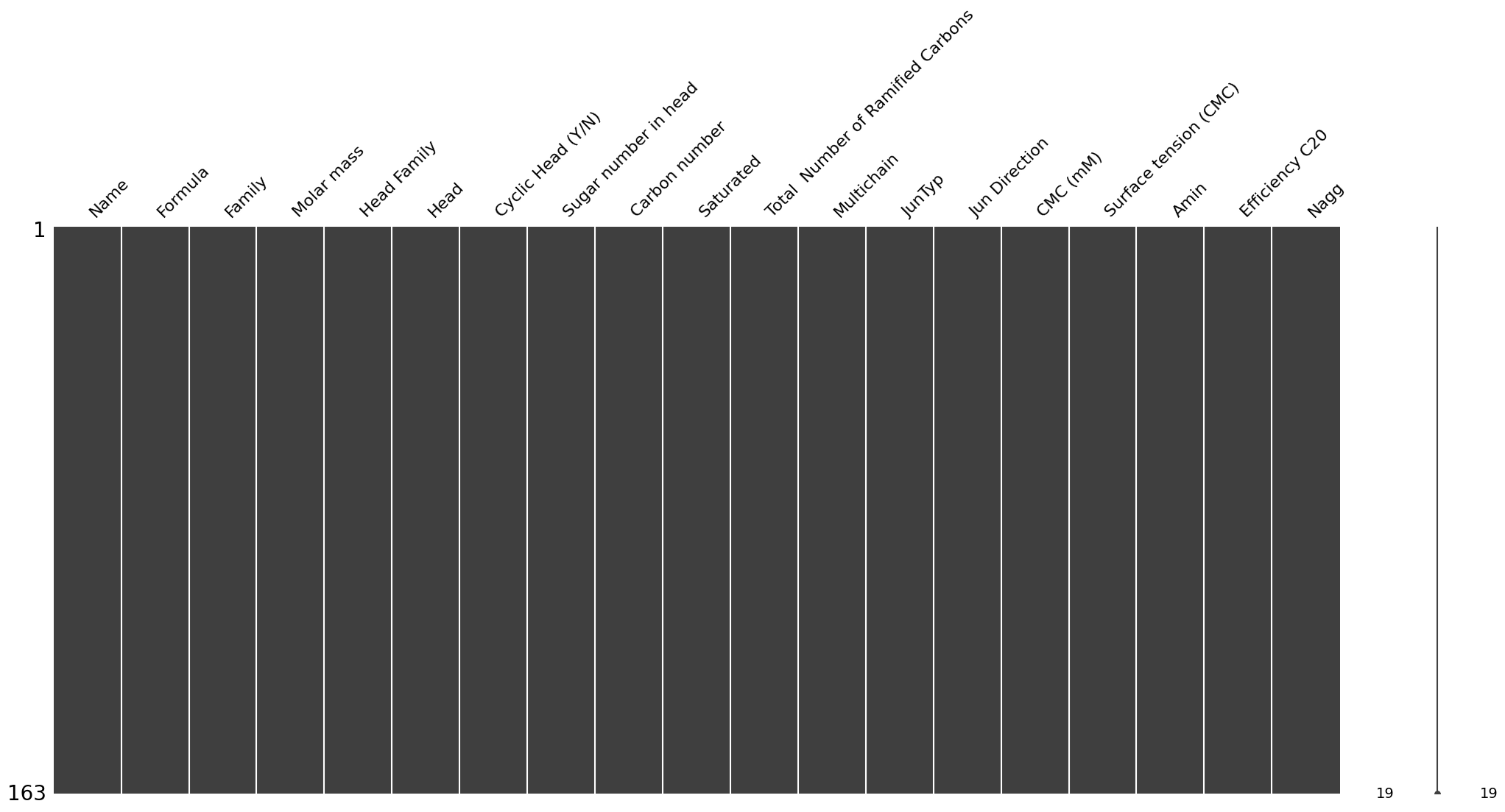


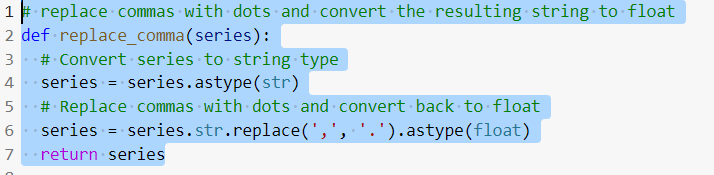
This code fills the null values in the "Surface tension (CMC)" column with either the mean or median value and converts any commas in the data to dots:





And we do some calculations using formulas and using mean, median and we fill null values in the dataset. Then we want to see visually there are any null values in dataset and there are not null values:

This function takes a series as input and replaces any commas in the series with dots, then converts the resulting string to float. We created this function because we need numeric values for model training and testing but in dataset there are only strings and this strings are like this “79,8”(this is an example), this form is not useful.

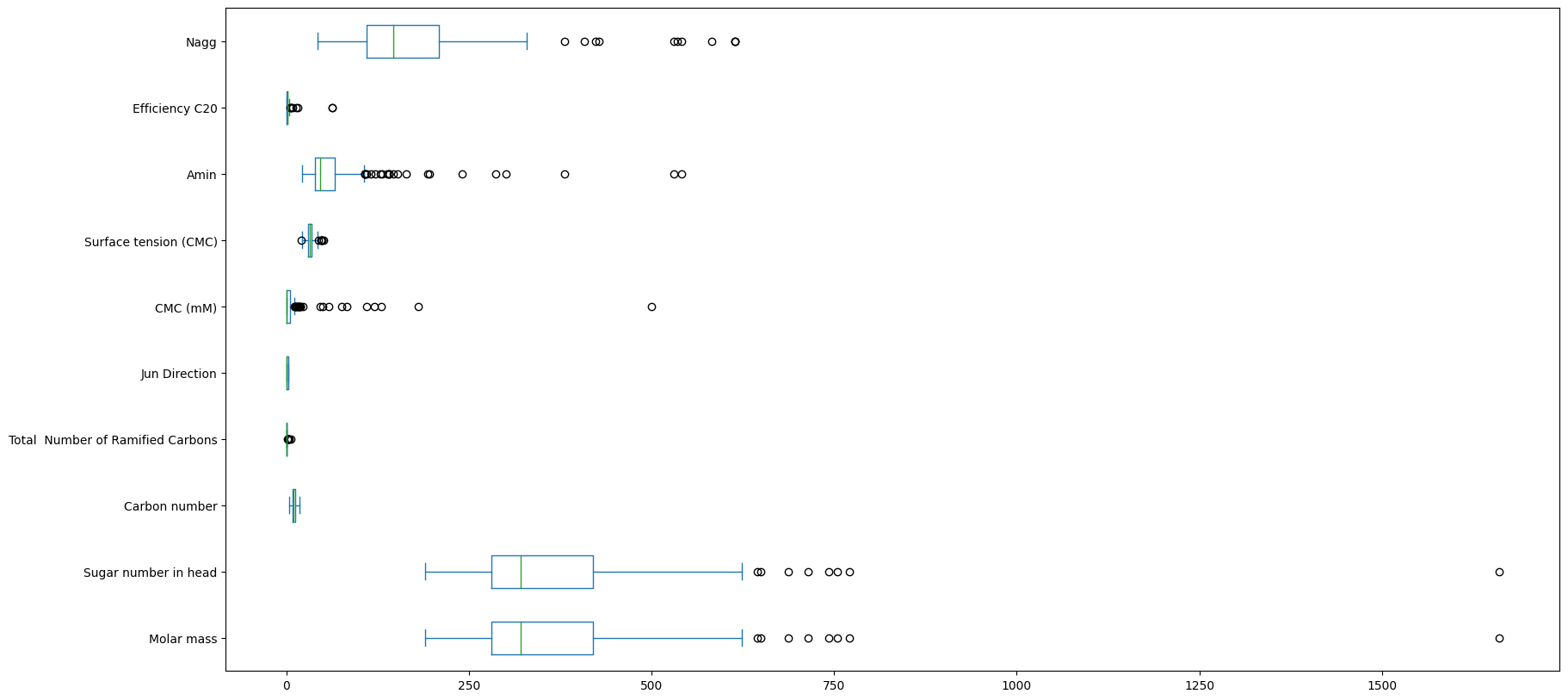


This code detects outliers using the IQR method for each column with numerical data types (int64 and float) in the df\_copy and print the results.

Text

Description automatically generated

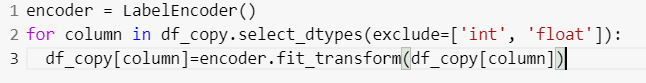
Then we used boxplot to see visually these outliers:

Then we wanted to get informasion about categorical columns and we displayed distributions of these by using histogram and pie chart:

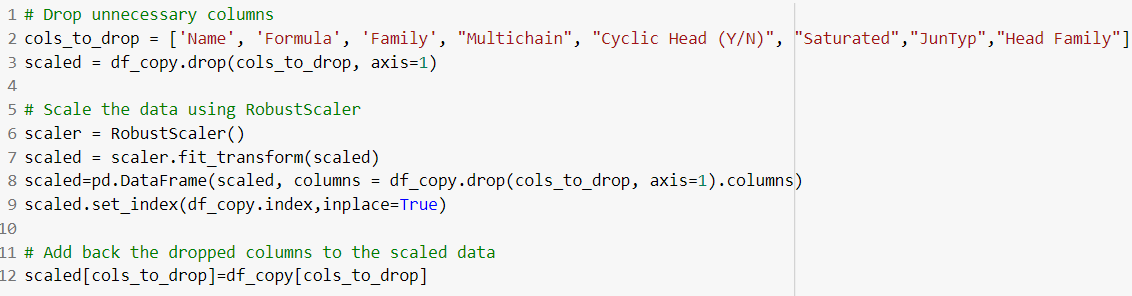
|  |  |
| --- | --- |
| create\_pie(df\_copy["Cyclic Head (Y/N)"]) | create\_barplot(df\_copy["Saturated"]) |
| create\_barplot(df\_copy["Multichain"]) | create\_pie(df\_copy["JunTyp"]) |

**Classification**

Here before creation model we converted categorical data into numerical because model does not understand them if they are useful for model training. And for this process we used LabelEncoder from sklearn library:



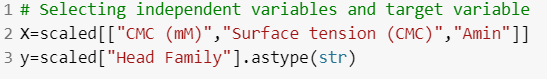
Some columns have large numbers and this reduce models accuracy. In this case we used RobustScaler as scaler on numerical columns to scale data and to get high result:



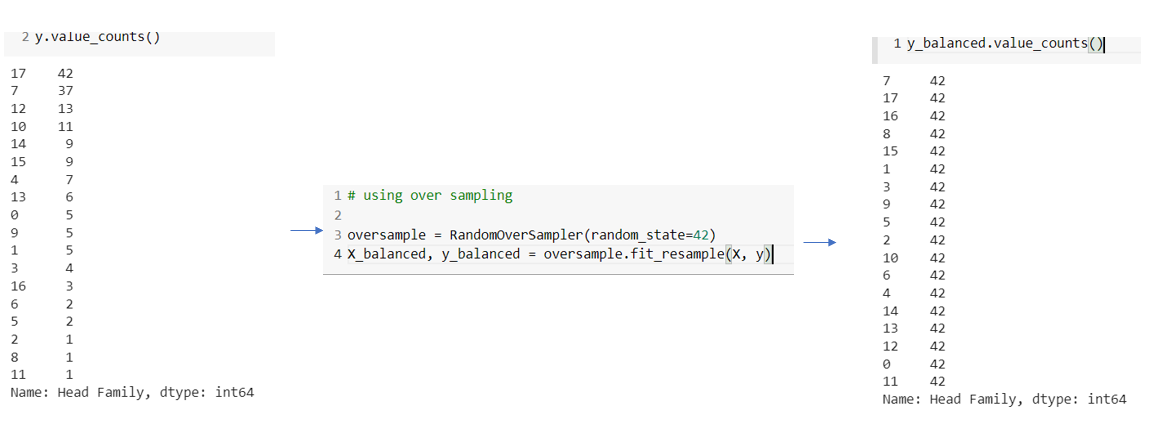
Application

Description automatically generated with low confidence

We selected the independent variables "CMC (mM)", "Surface tension (CMC)", and "Amin", and the target variable "Head Family" from the scaled dataset:



We checked value counts of target and we saw dataset has inbalance problem and we used over sampling to solve this problem and all items have 42 value counts in target:

Then we created 4 models thes models are: DecisionTreeClassifier, RandomForestClassifier, SVC(Sepport Vector Machine Classifier), KNeighborsClassifier. First of all we got results of all models for train and test data and we create table to see comparison between models and which model is good to predict for this dataset from these scores:

|  |  |  |
| --- | --- | --- |
| Model | Train score | Test score |
| DecisionTreeClassifier | 99.6% | 88.5% |
| RandomForestClassifier | 99.6% | 88.1% |
| SVC | 93.1% | 80.6% |
| KNeighborsClassifier | 88.8% | 73.1% |

We also print classification report for all models in the code. And We did hyper parameters tuning to get high and good results for these models and for this purpose we used GridSearchCV. This is code for one model:

Text, letter

Description automatically generated

Text

Description automatically generated

Then we compare all best scores for all models after grid search for this we create table again:

|  |  |
| --- | --- |
| Model | Best score |
| DecisionTreeClassifier | 91.8% |
| RandomForestClassifier | 91.6% |
| SVC | 85.06% |
| KNeighborsClassifier | 90.7% |

Then we also compared prediction results before and after predictions according to GridSearchCV graphically (by using histograms) for all models:

|  |  |  |
| --- | --- | --- |
| Model | before | after |
| DecisionTreeC. |  |  |
| RandomForestC. |  |  |
| SVC |  |  |
| KNeighborsC. |  |  |

**Conclusion**

In conclusion, we have performed a thorough data analysis of the surfactant dataset, which included handling missing values, calculating missing values in columns using provided formulas, detecting outliers, and encoding categorical variables. We have also selected the relevant features and target variable for our machine learning models. With this analysis, we can now proceed to build and train our model to predict the head family of surfactants based on their properties. This analysis has provided valuable insights into the data, which will enable us to build a more accurate and robust model.