

# **SECB3203 (01) – PROGRAMMING FOR BIOINFORMATICS**

# **FACULTY OF COMPUTING**

**TITLE:**

**"UNRAVELING ALZHEIMER’S DISEASE MYSTERIES**

**THROUGH PROTEIN ANALYSIS”**

**Progress 2**

**NAME:**

# **MAATHUREE A/P VEERABALAN (A21EC0051)**

**QAISARA BT BADRUL HISHAM (A21EC0125)**

# **LECTURER’S NAME:**

**DR. NIES HUI WEN**

# **TABLE OF CONTENTS**

| **CONTENT** | **PAGE** |
| --- | --- |
| **IMPORTING DATA** | 3 - 4 |
| **DATA WRANGLING** | |
| **IDENTIFYING MISSING VALUES** | 5 |
| **HANDLING MISSING VALUES** | 6 |
| **DATA FORMATTING** | 7 |
| **DATA BINNING** | 8 |
| **INDICATOR VARIABLES** | 9 |

**Importing Dataset**

The dataset used for this project was accumulated from Kaggle.com titled Alzheimer's Disease Classifications. This extensive dataset is noted as oasis.csv which covers over 400 specimen Alzheimer diagnosis.

To execute data analytics we first begin by importing the dataset which we have acquired from Kaggle.com prior. This will then lay out a foundation to which will aid in any subsequent analysis and model development that will be done. The dataset imported will be presented through Python.



Based on the code above we were able to import the library named ‘pandas’ as this library is more suited for the task at hand which is data manipulation.

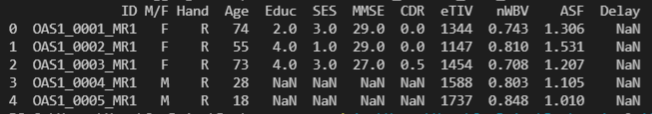
****

The dataset oasis.csv is then loaded with the specification of the file path as well as assigning it to a Data Frame (which is labeled as df).



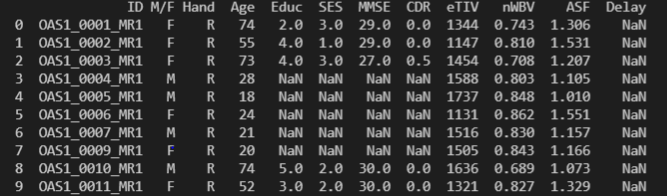
Breaking down the functionality of the code above;

* < **print(df.head()**) > is used to display the first few rows within the dataset which in this case is derived from the oasis.csv file. It prints out 5 rows as that is the default setting.

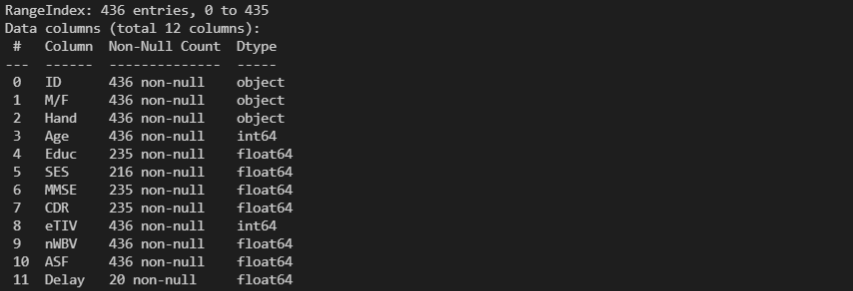


To specify the number of rows wanted to display, simply add an numerator in the bracket.

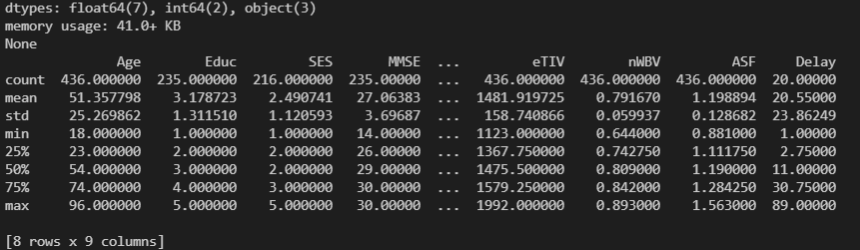
< **print(df.head(10)**) >



* < **print(df.info())** > this command produces a more detailed summary of the Data Frame which includes information regarding the number of non-null values, data types of each column, and the memory usage.

****

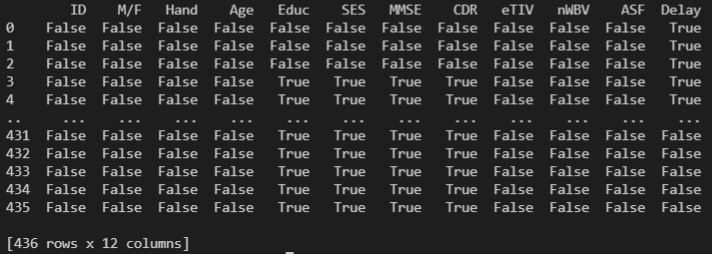
* **< print(df.describe()) >** this command generates the count, mean, standard deviation, minimum, 25th percentile, median (50th percentile), 75th percentile, and maximum. It provides insights into the central tendency and spread of the numerical columns in the dataset.



**Data Wrangling**

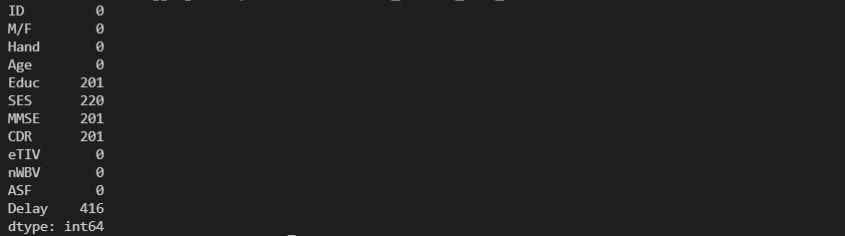
**Identifying Missing Values**

< print(df.isnull()) > this syntax is used to identify any null values within the dataset.



The false reading indicates that there are no null values and true indicates null values are present in the Data Frame.

To identify how many null values are there as a whole in the wide range of values you can use the syntax < **print(df.isnull().sum())** >



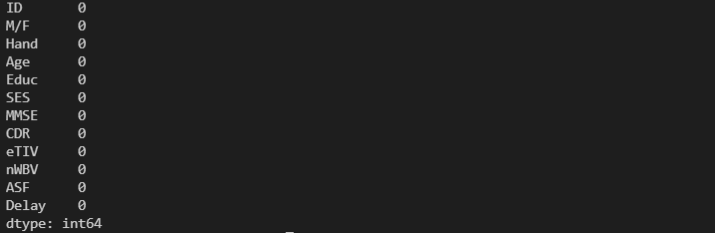
You can now see which columns these null values are residing.

**Handling Missing Values**

Now there are several ways to handle null or missing values, the way that we have chosen for our project is by filling the null values. This entails that wherever a null value is present it will be filled by a value of our choosing. In this case we will be replacing it with the value 0 using the syntax < df.fillna(value = 0) >



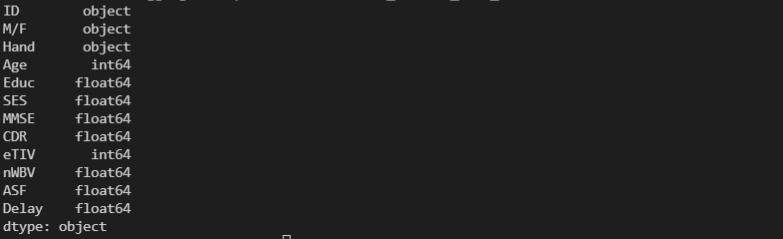
Figure above shows the changing the null values to 0 as well as creating a new Data frame(df2). The results are as shown below.



As the figure illustrates there are no missing values left in the Data Frames.

**Data Formatting**

The need for data formatting is to ensure that the data is being presented in a unified manner as to make certain that the data is easily readable. Actions regarding data formatting involves converting data types, adjusting decimals, or representing dates uniformly. The syntax for displaying data types is as such < **print(df.dtypes)** >



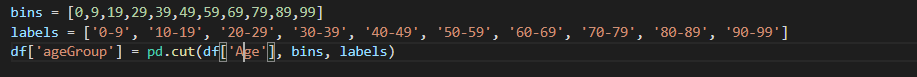
As represented in the figure above the Data Frame consist of 3 main data types;

* Object - consist of multiple data types existing in the same column.
* Int64 - consist of integer data type
* Float64 - consist of floating point data type

Due to the action of filling in the null values done prior we refrain from doing any changes to the current format.

**Data Binning**

Data binning or data bucketing is an action of grouping bins of data or buckets of data that share somewhat similar characteristics. This will in turn simplify the data as a whole because values in specific intervals can now be represented by a singular representative value. Binning also may improve accuracy in a predictive model.



The figure above shows the process of binning the column Age into several intervals which is '0-9', '10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80-89', '90-99'. The name ‘ageGroup’ will now refer to the new column that has been produced as a result of the binning process. This column will now indicate the age range in which the specimen falls under.

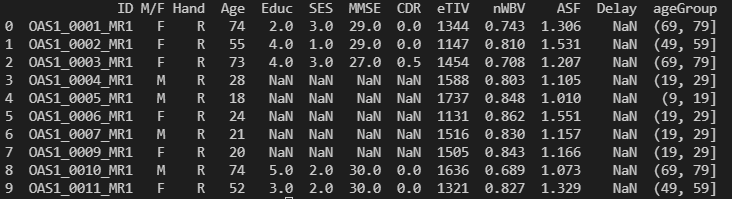


Figure above shows the new column has been created.

**Indicator Variables**

We believe our projects require no indicator variable steps during the operation of data processing due to not having categorical values within our dataset. Hence there is no need to represent the presence or absence of data of a particular category.