**Assignment -2**

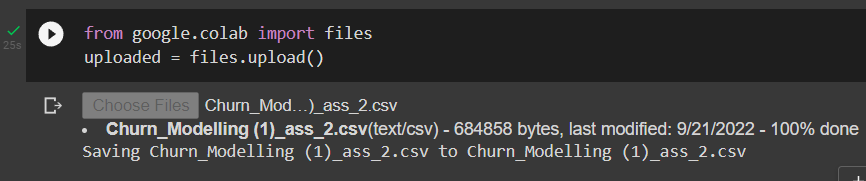
**Data Visualization and Pre-processing**

|  |  |
| --- | --- |
| Assignment Date | 3 October 2022 |
| Student Name | M.HARI PRASAD |
| Student Roll Number | 737819ECR049 |

**Tasks:-**

1. Download the dataset: Dataset

2. Load the dataset.

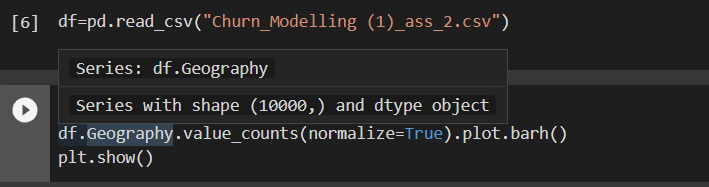


3. Perform Below Visualizations.

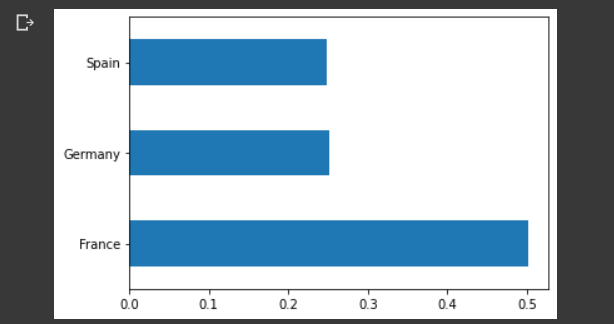
● Univariate Analysis

If we analyse data over a single variable/column from a dataset, it is known as Univariate Analysis.

Now, let’s analyse the geography category by using plots. Since Geography is a category, we will plot the bar plot.



The output looks likes this,

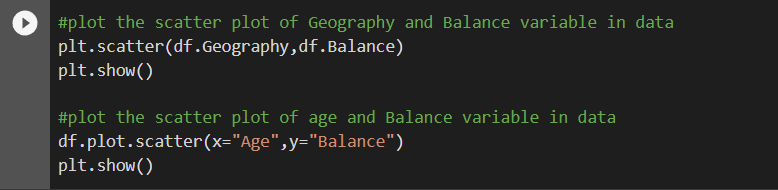


By the above bar plot, we can infer that the data set contains more number of France peoples are there compared to other categories.

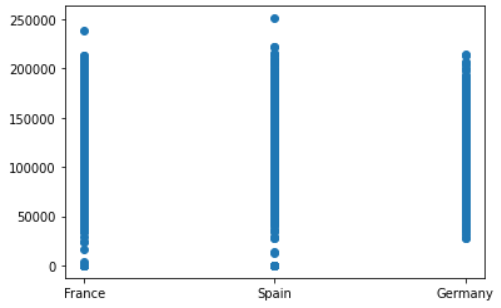
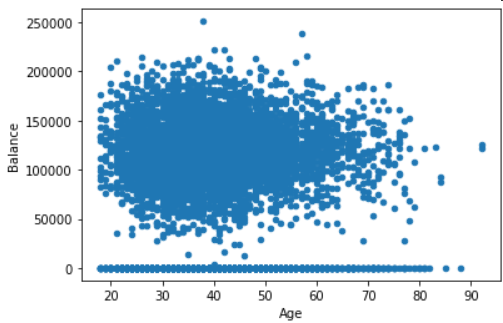
● Bi - Variate Analysis

Analysing the two numeric variables from a dataset is known as numeric-numeric analysis.

Let’s take three columns ‘Balance’, ‘Age’ and ‘Geography’ from our dataset and see what we can infer by plotting to scatter plot between Geography balance and Age balance.



The output looks like this,

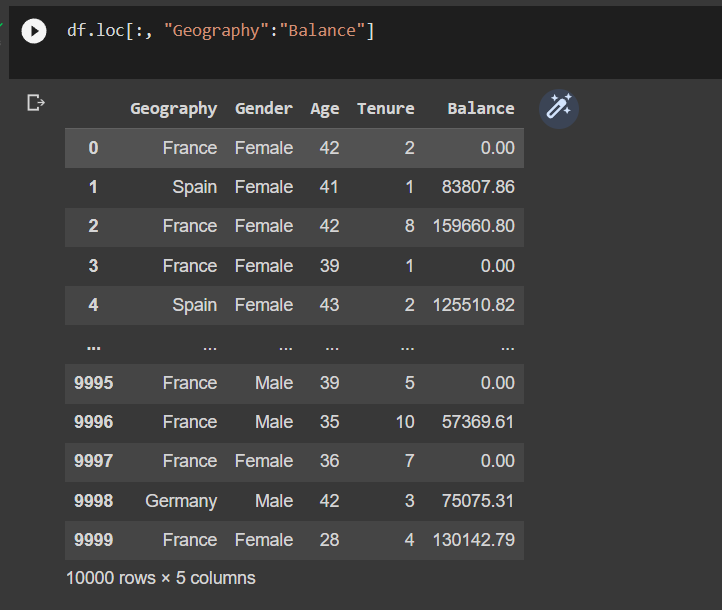
Geography vs balance Age vs balance

● Multi - Variate Analysis

Multivariate analysis is based in observation and analysis of more than one statistical outcome variable at a time.

Let’s take columns from ‘Geography’ to ‘Balance’ from our dataset and see what we can infer by plotting to scatter plot.

Code for taking required columns from our dataset, and output for that code



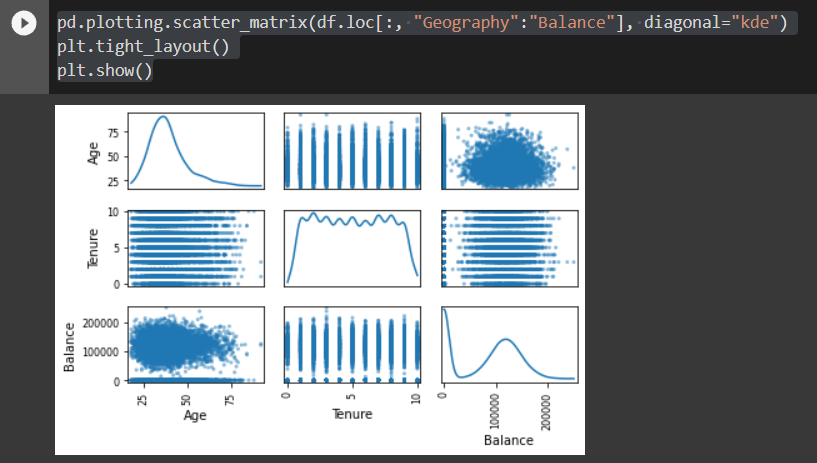
To make a matrix scatterplot of just these 5 variables using the scatter\_matrix() function we type:

pd.plotting.scatter\_matrix(df.loc[:, "Geography":"Balance"], diagonal="kde")

plt.tight\_layout()

plt.show()

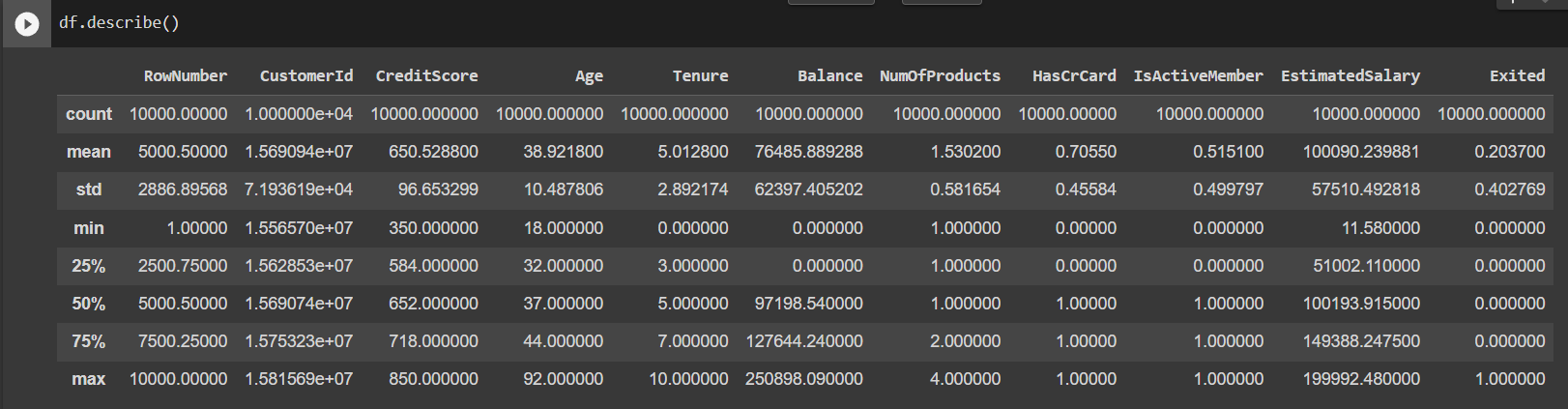
The output looks like this,



4. Perform descriptive statistics on the dataset.

The describe() function computes a summary of statistics pertaining to the Data Frame columns.

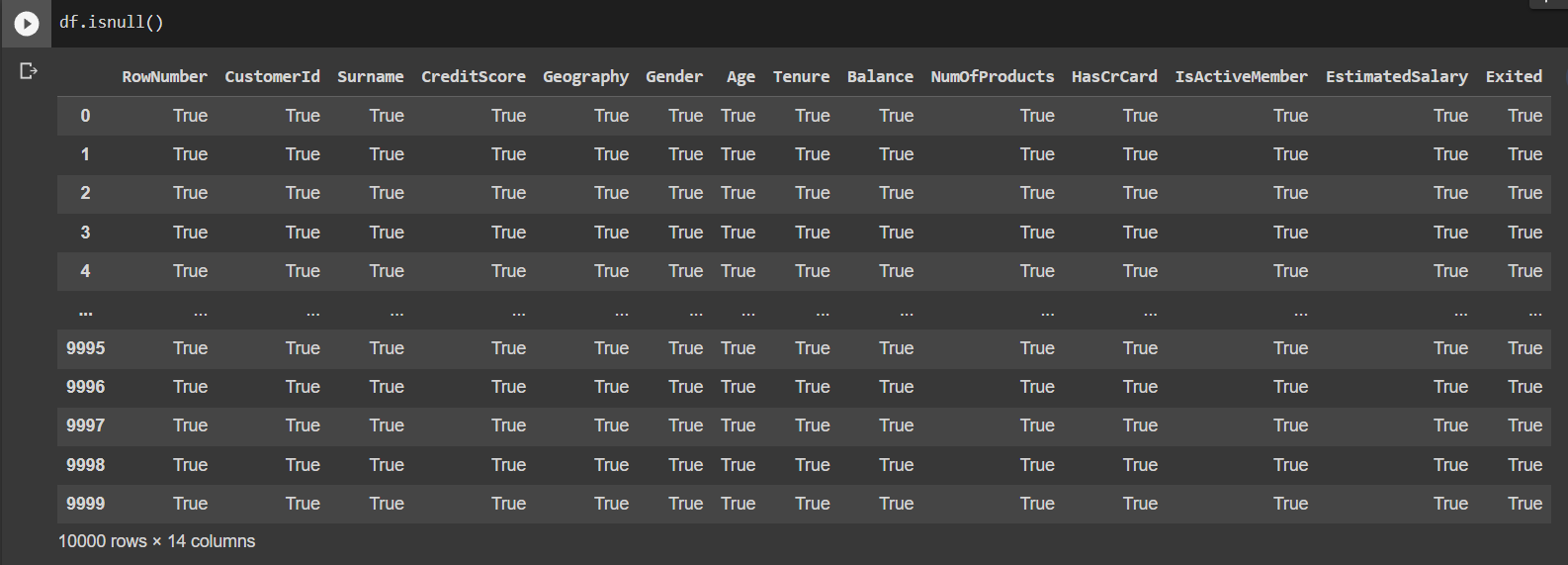
The output of descriptive statistics on the dataset look like,



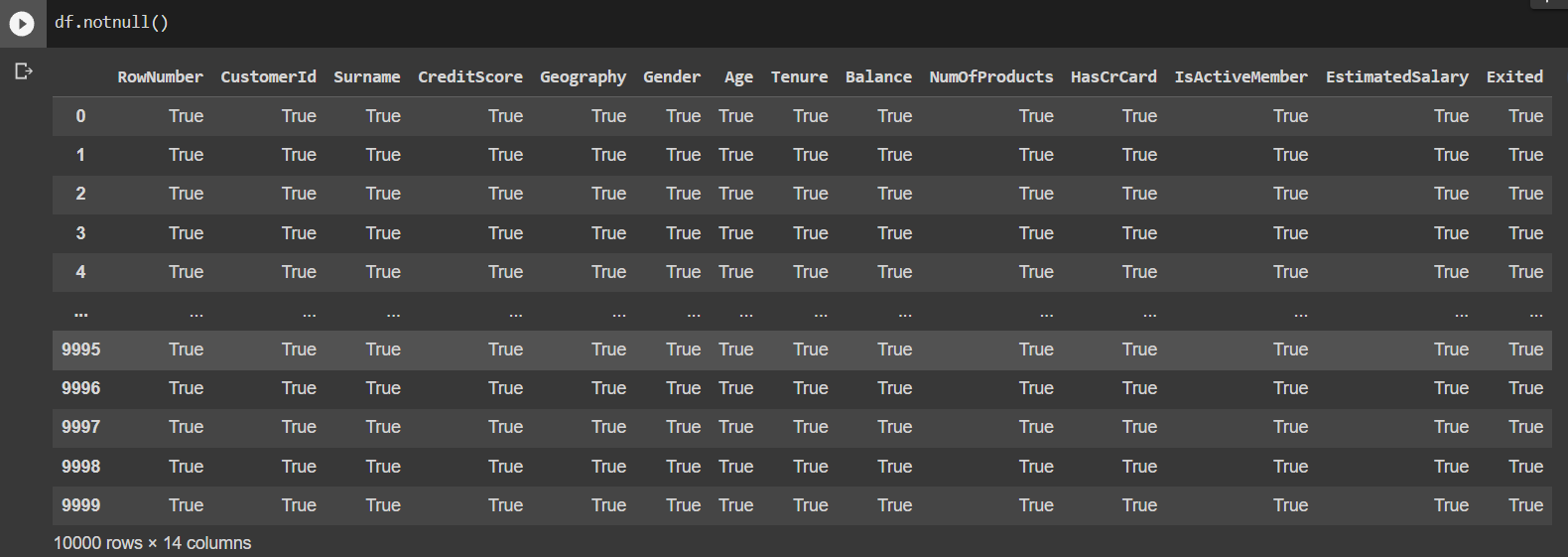
5. Handle the Missing values.

In order to check missing values in Pandas DataFrame, we use a function isnull() and notnull(). Both function help in checking whether a value is NaN or not.

In order to check null values in Pandas DataFrame, we use isnull() function this function return dataframe of Boolean values which are True for NaN values.



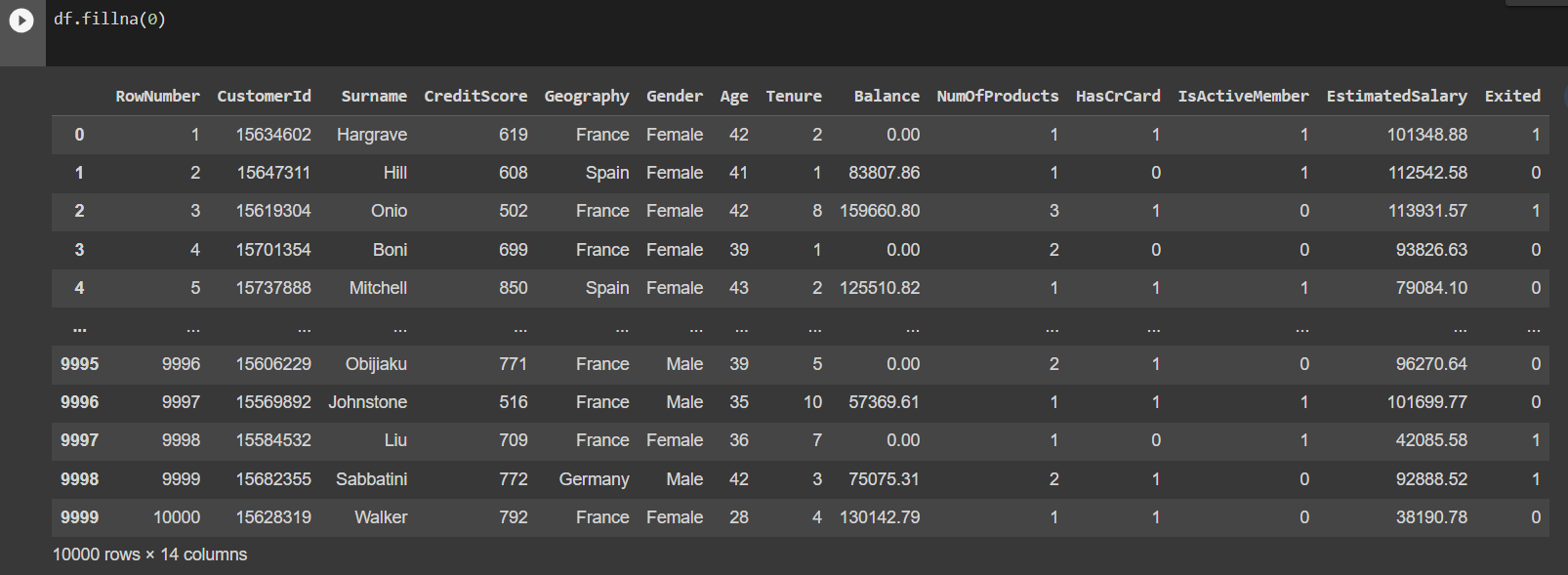
In order to check null values in Pandas Dataframe, we use notnull() function this function return dataframe of Boolean values which are False for NaN values.



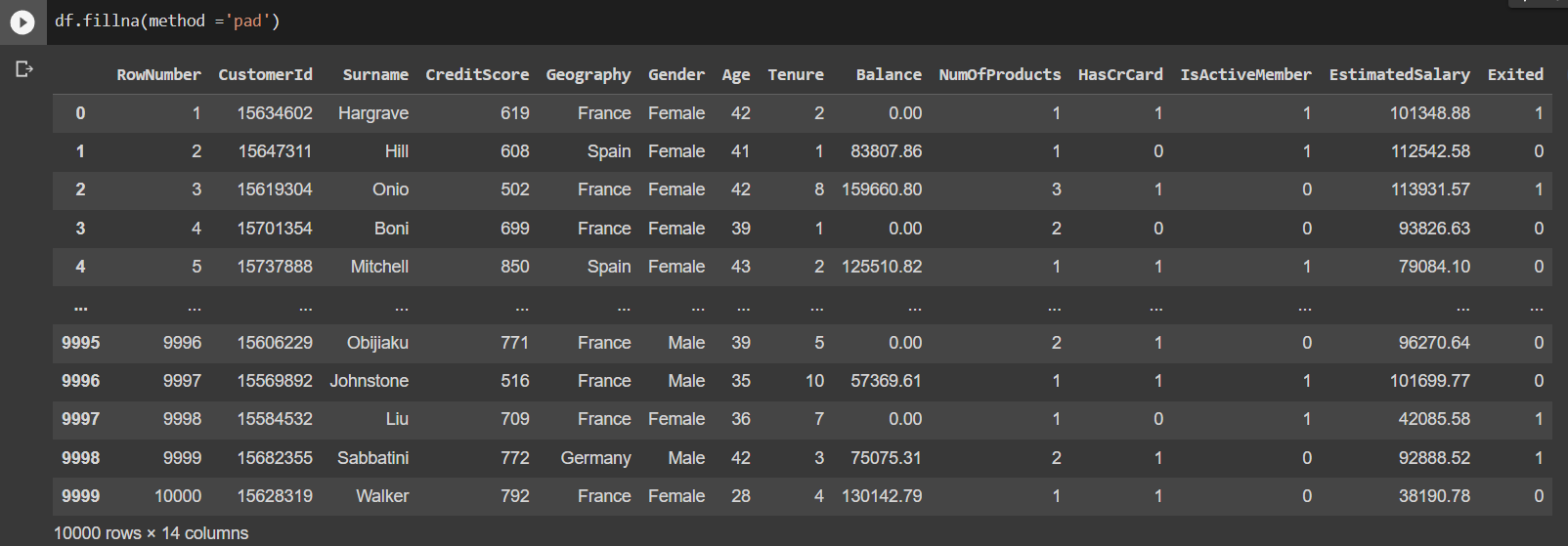
**Filling missing values using fillna(), replace() and interpolate():**

In order to fill null values in a datasets, we use fillna(), replace() and interpolate() function these function replace NaN values with some value of their own.

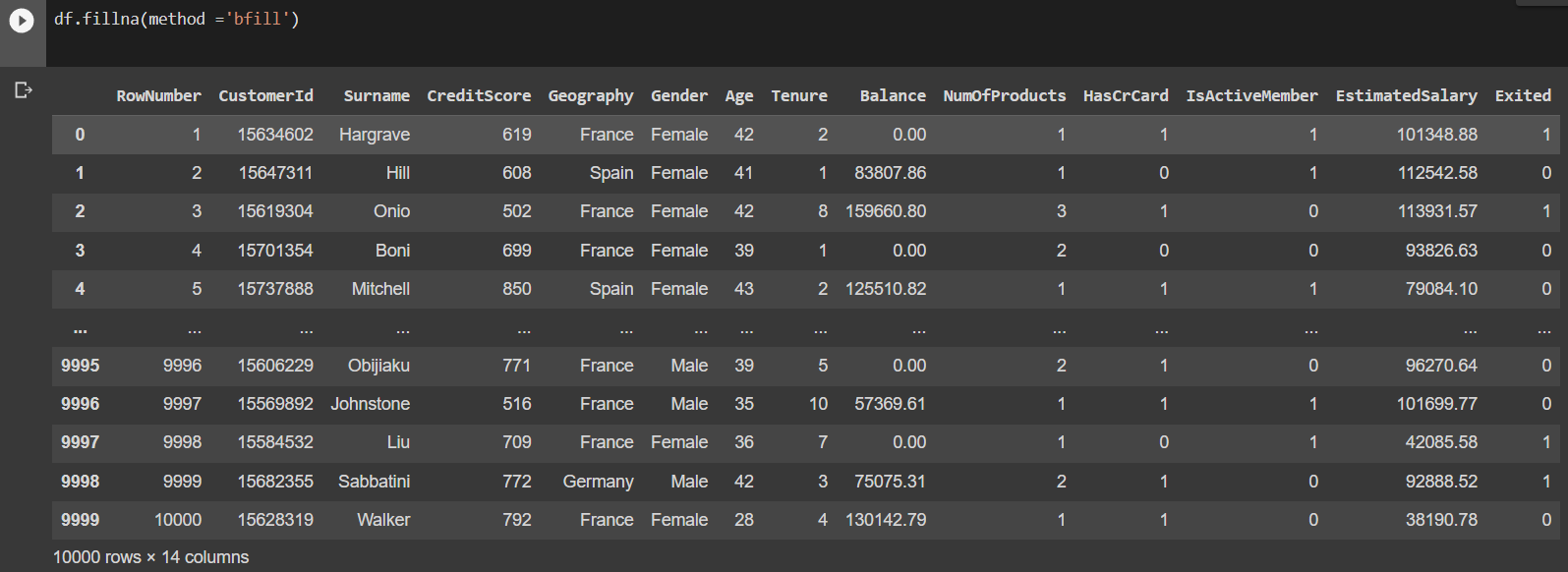
Filling null values with a single value



Filling null values with the previous ones

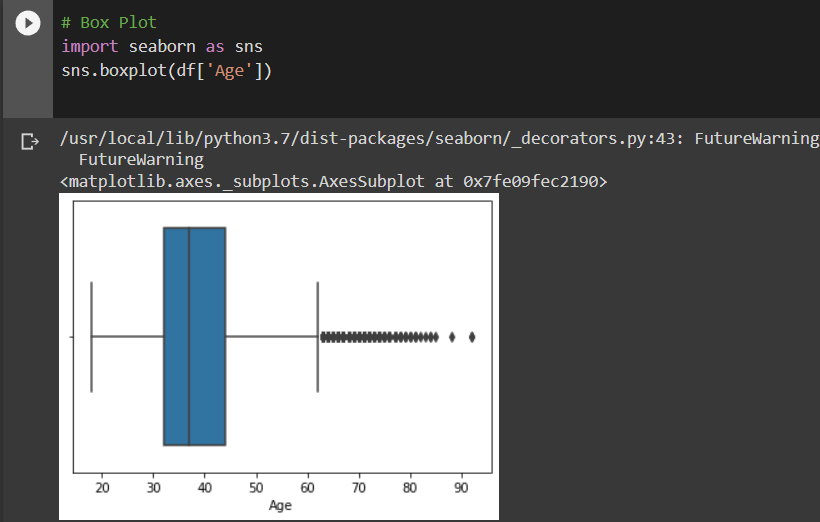


Filling null value with the next ones

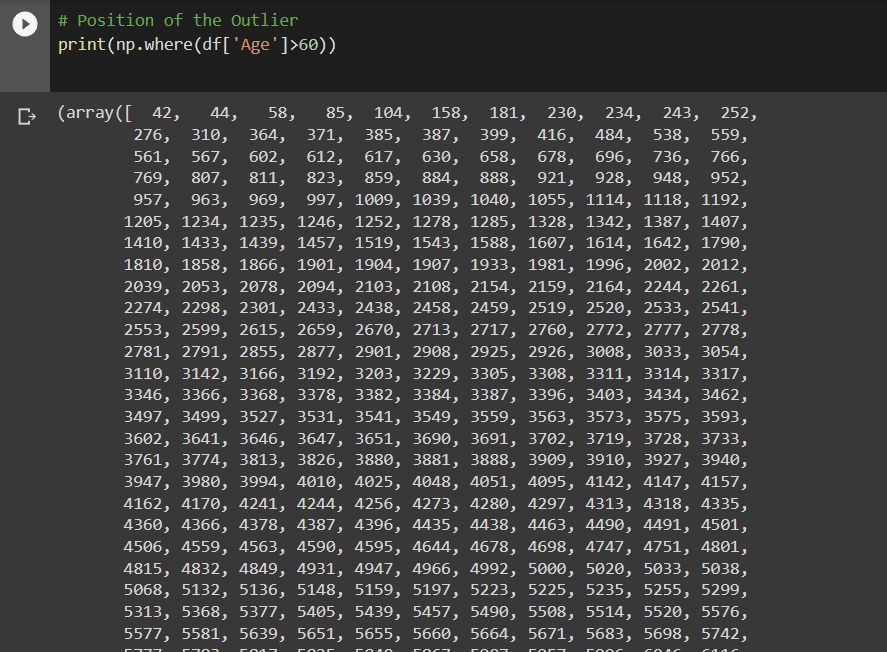


6. Find the outliers and replace the outliers

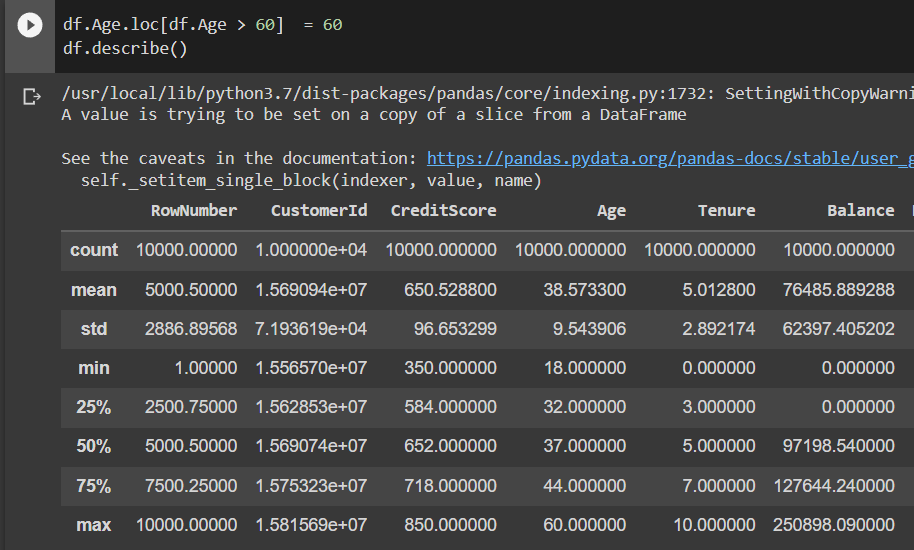
Outliers can be detected using visualization, Using Box Plot - It captures the summary of the data effectively and efficiently with only a simple box and whiskers. Boxplot summarizes sample data using 25th, 50th, and 75th percentiles. One can just get insights(quartiles, median, and outliers) into the dataset by just looking at its boxplot.



In the above graph, can clearly see that values above 60 are acting as the outliers. The outliers indexes are shown below,



Replacing of outliers is done by .loc function, all the outliers are replaced with the max age value as 60. The output looks like,

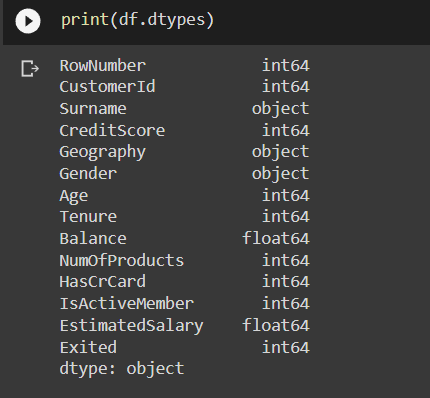


7. Check for Categorical columns and perform encoding.

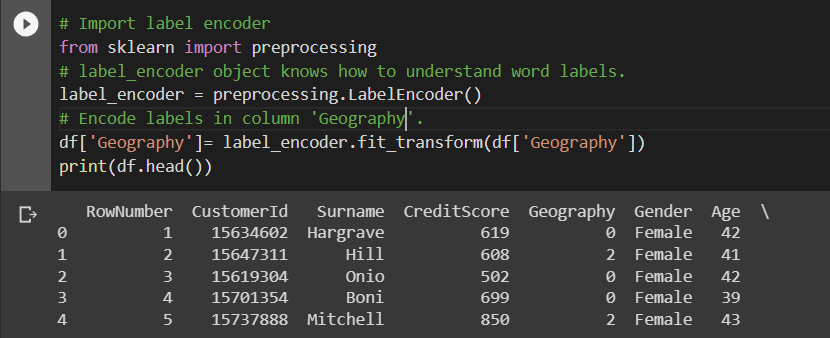
Categorical encoding is a process of converting categories to numbers.

Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering.

Understanding the datatypes of each columns:

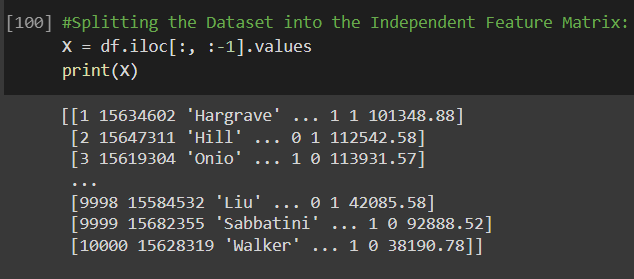


As you can see here, Geography, is the categorical feature as it is represented by the object data type. Now, let us implement label encoding:

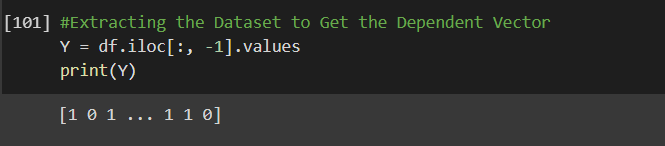
 As you can see here, label encoding uses alphabetical ordering. Hence, France has been encoded with 0, the Germany with 1, and Spain with 2.

8. Split the data into dependent and independent variables.

For dependent variable X, it takes all the rows in the dataset and it takes all the columns up to the one before the last column.



For independent variable Y, it takes all the rows, but only column 4 from the dataset.

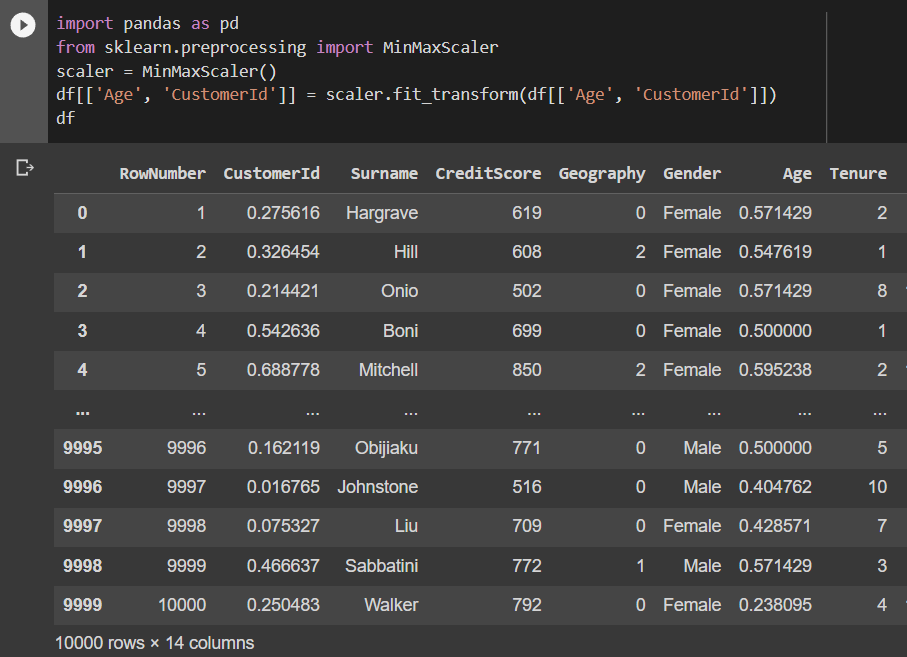


9. Scale the independent variables

When a dataset has values of different columns at drastically different scales, it gets tough to analyze the trends and patterns and comparison of the features or columns. So, in cases where all the columns have a significant difference in their scales, are needed to be modified in such a way that all those values fall into the same scale. This process is called Scaling.

Min-Max Normalization

Here, all the values are scaled in between the range of [0,1] where 0 is the minimum value and 1 is the maximum value. The age and customer ID columns are scaled in below figure,



10. Split the data into training and testing

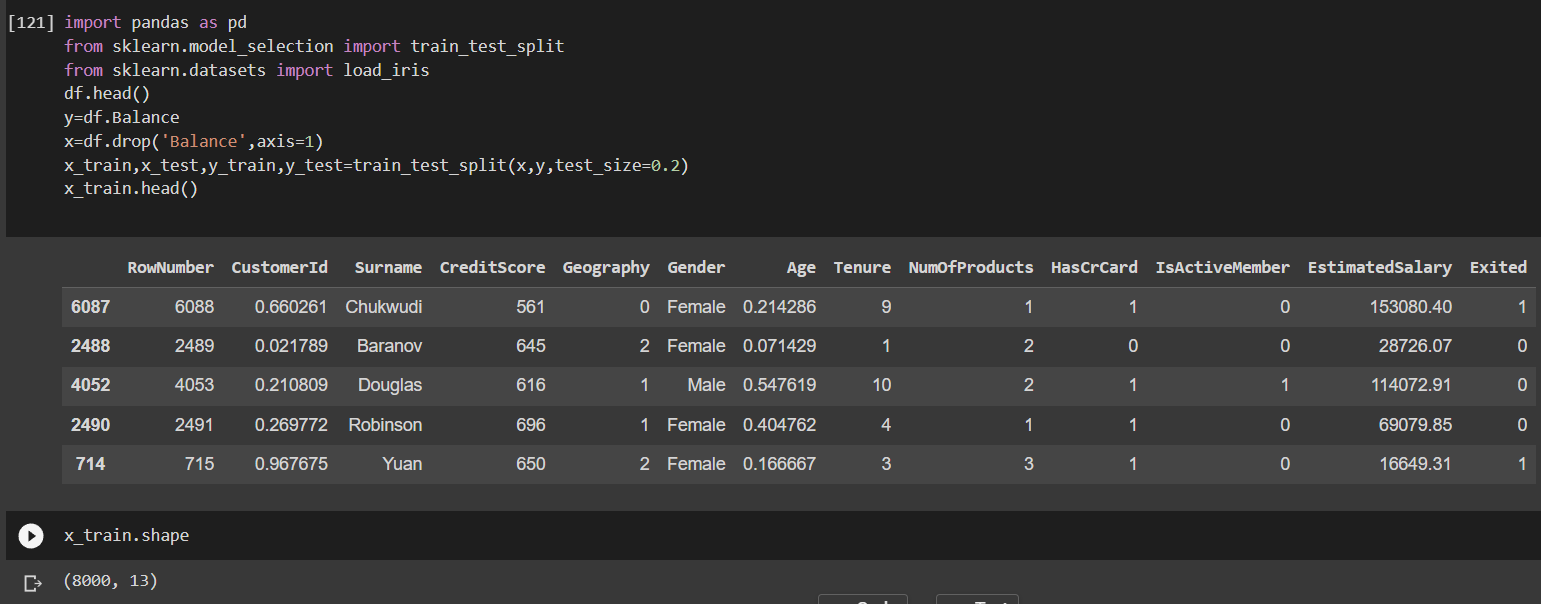
training set—a subset to train a model

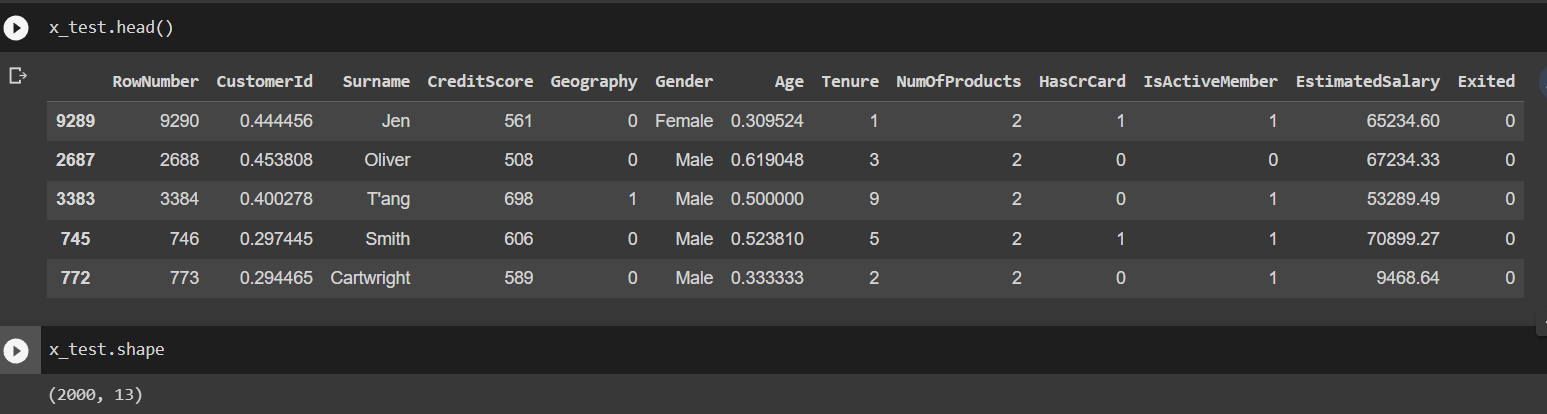
test set—a subset to test the trained model

1. Loading the dataset
2. Splitting

Let’s split this data into labels and features. Now, what’s that? Using features, we predict labels. I mean using features (the data we use to predict labels), we predict labels (the data we want to predict).

Balance is a label to predict balance in y; we use the drop() function to take all other data in x. Then, we split the data.





The line test\_size=0.2 suggests that the test data should be 20% of the dataset and the rest should be train data. With the outputs of the shape() functions, you can see that we have 2000 rows in the test data and 8000 in the training data.