Data Science Programming

Data preprocessing, modeling, and reporting



SAIF HADDAD

[21112014]

1. Make use, in your program, of the built-in data types, data structures, conditional programming,

and looping.

I have written a program using the Python language and my IDE was CoLab, and I have used some tools such as Pandas and NumPy libraries and sklearn, so I can be able to handle the data and visualize and evaluate the algorithm

**Pandas library:**

**1-To Load Data:**

so I can load the CSV file and open it to read, just by writing this syntax (pd.read\_filetype()).

**2-To Make DataFrames**

**3-To Clean Data:**

By using some syntaxes such as:

pd. isnull().sum: to see how many null values I have in the data set.

df. dropna: to delete any row that contains a null value.

**4-To Fill Data:**

By using this syntax

Df. fillna(“string or int or ….”): to replace the null values with another value.

**5-To Visualize Data**

**6-To View The Data:**

By using some syntaxes such as:

df. head(): to print the data.

df. shape(): to see how many columns and rows this data is.

**7- statistics:**

By using some syntaxes such as:

df. mean(): to return the mean of all the columns.

df. Max: to return the highest value in each column.

df. Min: to return the lowest value in each column.

df. median: to return the median of each column.

NumPy Library:

Just by using this syntax:

np. random. choice(): To let the computer take random choices.

At the first, I load the CSV file by this syntax (pd.read\_filetype()), then used this syntax (data. isnull().sum()) to see how many nulls I have in each column. After that, I started deleting all the columns that I saw that I don’t need and they will not help me with anything and they are not dependent on other columns or other columns are dependent on these columns such as: CASE\_ID, OPEN\_DATE, CLOSE\_DATE, OPEN\_USER, CLOSE\_USER, RESOLUTION, RESOLUTION\_DESCRIPTION, CASE\_DESC

I deleted the CASE\_ID because I see it as waste data that is taking place in the storage, and it will not help me with anything.

I deleted the OPEN\_DATE and the CLOSE\_DATE because for me I don’t care about the open date and the close date I just care about how many doses this problem takes time, I mean the period between the open date and the close date which I can found it in the AGE\_BRACHET.

I deleted the OPEN\_USER and the CLOSE\_USER because it is not important information for me to know who is the user who opens and closes the problem, and also the open user and the close contain big data and this data is just for one employee, but on the other hand, I have the OPEN\_GROUP and the CLOSE\_GROUP so these as more useful because the group contains more than one employee so I can see the name of the group that contains more than one.

I deleted the RESOLUTION, RESOLUTION\_DESCRIPTION, and CASE\_DESC because they contain a lot of null values over 80%, and they are not dependent on anything, it will be so difficult to handle them.

After I deleted all the columns that I don’t want I started to fill the nulls values in the columns (OFFER\_NAME, CUSTOMER\_GROUP, ESCALATED\_GROUP, OPEN\_GR, CLOSE\_GROUP, CALLBACK\_MECHANISM, AGE\_BRACKET).

OFFER\_NAME column: I filled all the null values with the string UNKNOWN because I can't predict what the name of the customer offer is and if I filled it wrong it might make a problem between the customers and the company.

CUSTOMER\_GROUP column: I filled all the null values with the mode which means, first I saw the count of every customer group value and then fill all the nulls with the same value of the most repeated value before filling the nulls.

ESCALATED\_GROUP column: this column is dependent on the ESCALATED\_FLAG so what I came up with is that when the problem is not escalated this means that there are no groups that did escalate the problem because from the beginning there were not escalated. So I fill all the null values with the string NOT\_ESCALATED.

CALLBACK\_MECHANISM column: in this column, I filled 50% of the nulls with the string SMS and the other 50% of the nulls with the string Phone, I consider this way is the best because maybe not all the customers have an email but all the customers have a telephone that has a number card in it.

AGE\_BRACKET column: I filled all the null values with the string STILL\_ACTIVE because as we can see every cell that is null is also the same cell in the close date is null, and as I said before that the AGE\_BRACHET is the period between the OPEN\_DATE and the CLOSE\_DATE so we can consider that the cell in the AGE\_BRACKET that have a null value it does not have a CLOSE\_DATE which means it is still ACTIVE

OPEN\_GR column: I put a code that will see the number of repetitions for each value and convert it to a percentage, so the null values cells will be filled based on the probability of each percentage, and the percentage will be the probability

CLOSE\_GROUP column: I put a code that will see the number of repetitions for each value and convert it to a percentage, so the null values cells will be filled based on the probability of each percentage, and the percentage will be the probability

Generated models:

I used the sklearn library so I can import the kNN model, Decision Tree model, and Gaussian Naive Bayes model

Evaluate the models using different evaluation measures:

As u can see in my code I used three models which are: The kNN model, The Decision Tree model, and The Gaussian Naive Bayes model and I have applied four measures which were

Accuracy, Precision, Recall, and F1\_score I used them so I can see and evaluate how good my program is. As we can see in the program code on the CoLab the Decision Tree model gives me the highest values, which is very good, and this means that the Decision Tree model is the best model among the used models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| The results that I have got from the three models | | | | |
|  | Accuracy | Precision | Recall | F1\_score |
| KNN Model | 0.9351896303408546 | 0.8824742268041237 | 0.8458498023715415 | 0.8637739656912209 |
| Decision Tree Model | 0.9927988478156505 | 0.9861386138613861 | 0.9841897233201581 | 0.9851632047477744 |
| Gaussian Naive Bayes model | 8233317330772923 | 0.6575342465753424 | 0.5691699604743083 | 0.6101694915254237 |

As I mentioned before, we can see in the table the results for a try and the Decision Tree Model gives me the best result