df.info(): it gives no null count and dtypes of all columns

df.shape: it gives rows and columns number

df.describe(): it displays all numeric columns with column information like count,mean,std,min,25%,50%,75%,max

df.isnull().sum(): it displays null count column wise

df.dtypes: it displays dtypes of all columns

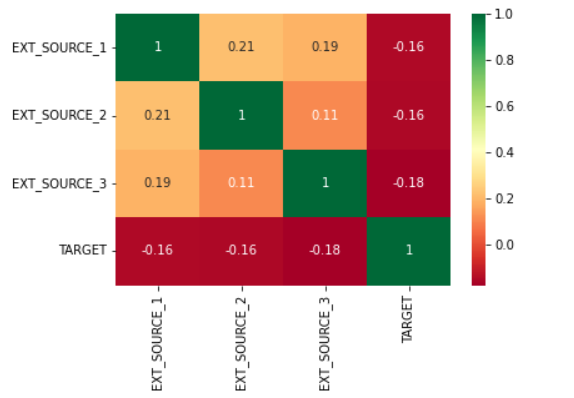
Step 1: round(df.isnull().sum() / df.shape[0] \* 100.00,2)

It will display all columns and percentage of null values present in it.

Plot bar graph : displaying percentage vs columns

eliminate all the columns containing 40% or more than that having null values in it

Step 2: check doubtful columns with main column using heat map graph which shows correlation between them. If no correlation found then drop the columns.



**itertools** is a module in Python that provides a set of fast, memory-efficient tools for handling iterators. Iterators are objects that can be iterated (looped) over, but **itertools** provides more specialized and efficient tools for working with iterators and iterable data.

Seaborn's **countplot** is a categorical plot that can be used to show the counts of observations in each category. It is similar to a bar plot, but the primary focus is on the count of observations in each category rather than aggregating some numerical value.

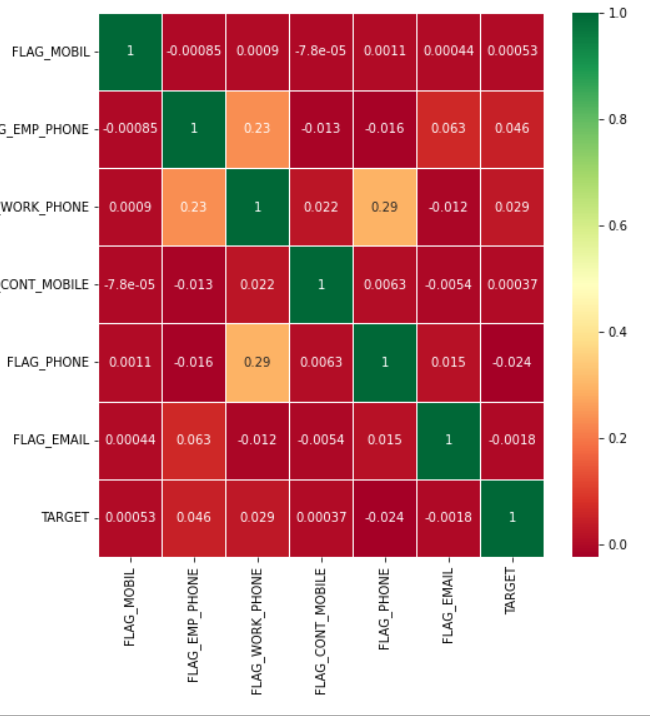
In most of the loan application cases, clients who applied for loans has not submitted FLAG\_DOCUMENT\_X except FLAG\_DOCUMENT\_3. Thus, Except for FLAG\_DOCUMENT\_3, we can delete rest of the columns. Data shows if borrower has submitted FLAG\_DOCUMENT\_3 then there is a less chance of defaulting the loan.

Step 3: # checking is there is any correlation between mobile phone, work phone etc, email, Family members and Region rating

contact\_col = ['FLAG\_MOBIL', 'FLAG\_EMP\_PHONE', 'FLAG\_WORK\_PHONE', 'FLAG\_CONT\_MOBILE', 'FLAG\_PHONE', 'FLAG\_EMAIL','TARGET',]

checked through sns.heatmap correlation between target column and them

Hence no corelation between them.



Step 4: df.drop(labels=unwanted , axis=1,inplace=True)

Dropping unwanted columns labels means columns name and axis 1 inplace =true means the modification must be done in original dataframe.

Step 5: Convert DAYS\_DECISION,DAYS\_EMPLOYED, DAYS\_REGISTRATION,DAYS\_ID\_PUBLISH from negative to positive as days cannot be negative.

Datecol = ['DAYS\_BIRTH','DAYS\_EMPLOYED','DAYS\_REGISTRATION','DAYS\_ID\_PUBLISH']

for col in date\_col:

df\_app[col] = abs(df\_app[col])

by abs() i.e absolute function

Step 6: Binning Numerical Columns to create a categorical column

bins = [0,1,2,3,4,5,6,7,8,9,10,11]

slot = ['0-100K','100K-200K', '200k-300k','300k-400k','400k-500k','500k-600k','600k-700k','700k-800k','800k-900k','900k-1M', '1M Above']

df['AMT\_INCOME\_RANGE']=pd.cut(df['AMT\_INCOME\_TOTAL'],bins,labels=slot)

df['AMT\_INCOME\_RANGE'].value\_counts(normalize=True)\*100

using pd.cut() function and value\_counts() to display

step 7: Checking the number of unique values each column possess to identify categorical columns

df.nunique().sort\_values()

Step 8: Data Type Conversion

df.info(): give no null count and columns name and its dtypes

#Conversion of Object and Numerical columns to Categorical Columns

categorical\_columns = ['NAME\_CONTRACT\_TYPE','CODE\_GENDER','NAME\_TYPE\_SUITE','NAME\_INCOME\_TYPE','NAME\_EDUCATION\_TYPE', 'NAME\_FAMILY\_STATUS', 'NAME\_HOUSING\_TYPE', OCCUPATION\_TYPE', ’WEEKDAY\_APPR\_PROCESS\_START', 'ORGANIZATION\_TYPE', 'FLAG\_OWN\_CAR','FLAG\_OWN\_REALTY', 'LIVE\_CITY\_NOT\_WORK\_CITY', 'REG\_CITY\_NOT\_LIVE\_CITY','REG\_CITY\_NOT\_WORK\_CITY','REG\_REGION\_NOT\_WORK\_REGION', 'LIVE\_REGION\_NOT\_WORK\_REGION', 'REGION\_RATING\_CLIENT', 'WEEKDAY\_APPR\_PROCESS\_START', 'REGION\_RATING\_CLIENT\_W\_CITY' ]

for col in categorical\_columns:

df[col] =pd.Categorical(df[col])

converting columns to category by pd.Categorical

Step 9: # Null Value Data Imputation

Strategy for application data:

To impute null values in categorical variables which has lower null percentage, mode() is used to impute the most frequent items.

To impute null values in categorical variables which has higher null percentage, a new category is created. Eg: unknown column name is added

To impute null values in numerical variables which has lower null percentage, median() is used.

Mean returned decimal values and median returned whole numbers and the columns were number of requests

Used Kdeplot of Sns to see the skeweness of graph if skeweness of graph is one side then we should use median, mean will not be right approach.

And if skewness of graph is one side and have many peaks then mode will be used, mean and median will not be right approach.

Step 10: finding out outliners by sns.boxplot

Step 11: Data Analysis:

Imbalance Analysis of TARGET column having defaulters and repayer

function for plotting repetitive countplots in univariate categorical analysis on application data

This function will create two subplots:

1. Count plot of categorical column w.r.t TARGET;

2. Percentage of defaulters within column

function for plotting repetitive countplots in bivariate categorical analysis

sns.barplot is used

sns.displot

univariate\_categorial=sns.countplot

bivariate\_bar=sns.barplot

bivariate\_rel= sns.relaplot

univariate\_merged=sns.countplot

merged\_pointplot= sns.pairplot