Import need pacages

In [113…
```
import warnings
warnings.filterwarnings('ignore')
from operator import itemgetter
import pandas as pd #dataframe
import numpy as np #mathematical computations
import matplotlib.pyplot as plt #visualization
import matplotlib
import seaborn as sns #visualization
import json #exporting columns
import pickle #saving the model
from sklearn.linear_model import LinearRegression #Linear Regression
from sklearn.linear_model import Lasso #Lasso Regression
from sklearn.tree import DecisionTreeRegressor #Decision Tree Regression
from sklearn.ensemble import RandomForestRegressor #Random Forest Regression
from sklearn.model_selection import train_test_split #Splitting the dataset into training and testing
from sklearn.model_selection import ShuffleSplit #Random shuffling
from sklearn.model_selection import cross_val_score #Score cross validation
from sklearn.model_selection import GridSearchCV #Hyper parameter tuning
from warnings import simplefilter #Filtering warnings
import seaborn as sns
import missingno as msno
import statsmodels.api as sm
from datetime import datetime
from scipy import stats
from matplotlib.pyplot import MultipleLocator
```

1. Observe the data

Import the data set and show the title

part3_result.csv is the file without deleting the Non registered organizations It's used to compare the result All the file need be modified manumally before importing

In [113…
```
#Orginal_data = pd.read_csv('./Original_dataset.csv',encoding = "ISO-8859-1")
#Causes_data = pd.read_csv('./Original_Causes.csv',encoding = "ISO-8859-1")
```

In [113…
```
#combined_data = pd.merge(Orginal_data, Causes_data, how='left', on=['Campaign Id']) #Combined data set
#combined_data.to_csv('New_original_combined_data.csv', index=None)
```

In [113…
```
# Here need change the csv file from part3_result a little bit
# Delete the space front and end of Campaign_Goal manually
# Change the number into number type instead of accounting type
# Or it can be changed into numeric number
```

```
combined_data = pd.read_csv('./Part3_result.csv',encoding = "ISO-8859-1")
#combined_data = pd.read_csv('./Final_data.csv',encoding = "ISO-8859-1")
```

In [113…
```
combined_data.shape
```

Out[1138]: (10139, 79)

In [113…
```
columns_name = combined_data.columns
columns_name
```

Out[1139]:
```
Index(['Campaign Id*', 'Campaign Title_x', 'Receiving NPO Name*',
       'Receiving NPO Id*', 'NPO Status*',
       'Number of campaigns from the same NPO that started within the same Year Month',
       'Public Campaign Access*', 'Creator Type*', 'Creator Id*',
       'Campaign Status*', 'Actual Donation Amount', 'Distinct Donors',
       'Campaign Goal', 'Campaign Completion Rate', 'Days Left for Campaign',
       'Campaign Start Date', 'Campaign End Date',
       'NPO Ipc Status For Tax Deductibility', 'Campaign Image1 Id',
       'Campaign Image2 Id', 'Campaign Image3 Id', 'Campaign Image4 Id',
       'Campaign Image5 Id', 'Campaign Video', 'Impact Message 1',
       'Impact Message 2', 'Impact Message 3', 'Impact Message 4',
       'Impact Message 5', 'Custom Amount 1', 'Custom Amount 2',
       'Custom Amount 3', 'Custom Amount 4', 'Description of Campaign',
       'Description of NPO', 'Campaign Title_y', 'Org Cause Animal Welfare',
       'Org Cause Arts & Heritage', 'Org Cause Children & Youth',
       'Org Cause Community', 'Org Cause Disability', 'Org Cause Education',
       'Org Cause Elderly', 'Org Cause Environment', 'Org Cause Families',
       'Org Cause Health', 'Org Cause Humanitarian',
       'Org Cause Social Service', 'Org Cause Sports',
       'Org Cause Women & Girls', 'Cam Cause Animal Welfare',
       'Cam Cause Arts & Heritage', 'Cam Cause Children & Youth',
       'Cam Cause Community', 'Cam Cause Disability', 'Cam Cause Education',
       'Cam Cause Elderly', 'Cam Cause Environment', 'Cam Cause Families',
       'Cam Cause Health', 'Cam Cause Humanitarian',
       'Cam Cause Social Service', 'Cam Cause Sports',
       'Cam Cause Women & Girls', 'Pub Enquiry Person', 'Pub Enquiry Contact',
       'Pub Enquiry Email', 'Web URL', 'Facebook Link', 'Org_causes',
       'Cam_causes', 'S/N', 'Type', 'UEN', 'IPC Period', 'Sector',
       'Classification', 'Activities', 'Scale_type'],
      dtype='object')
```

In [114…
```
# change the columns name
for i in range(len(columns_name)):
    combined_data = combined_data.rename(columns={columns_name[i] : columns_name[i].replace(" ","_")})
```

In [114…
```
columns_name = combined_data.columns
```

In [114…
```
for i in range(len(columns_name)):
    combined_data = combined_data.rename(columns={columns_name[i] : columns_name[i].replace('*','')})
combined_data.columns
```

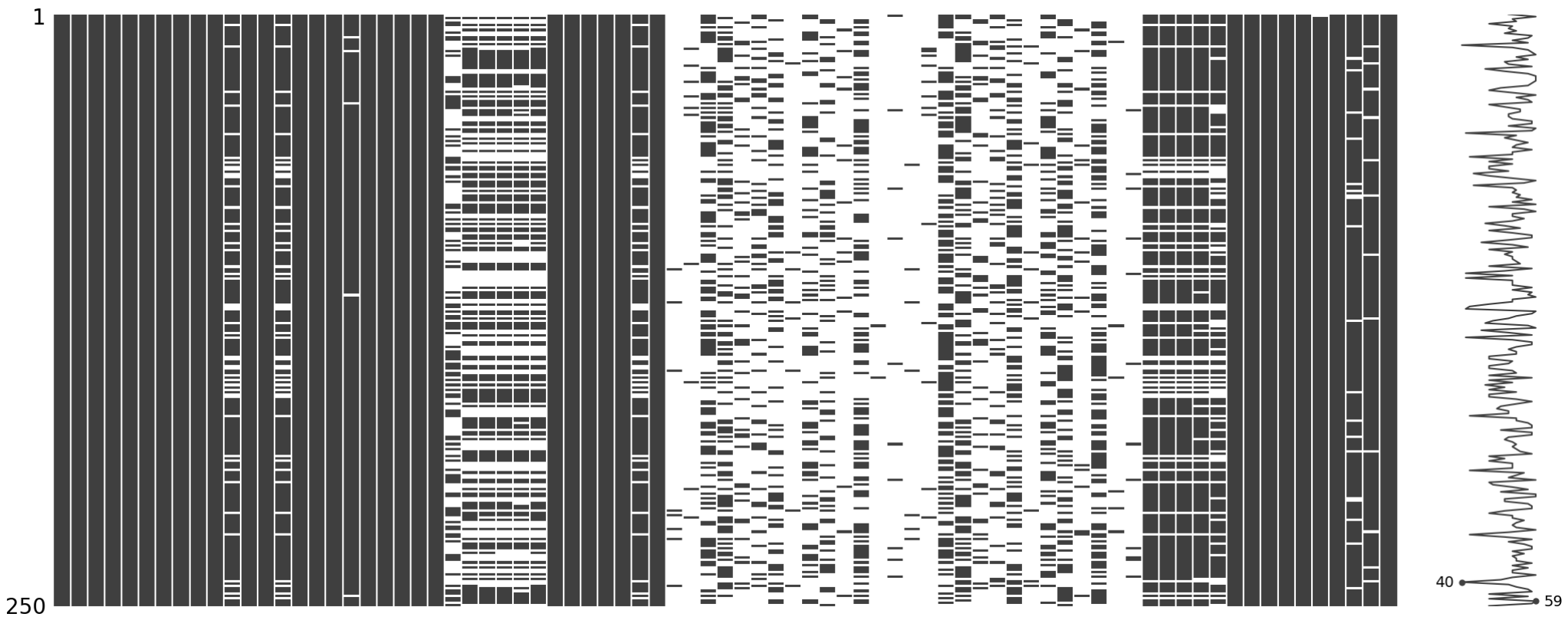
Out[1142]:
```
Index(['Campaign_Id', 'Campaign_Title_x', 'Receiving_NPO_Name',
       'Receiving_NPO_Id', 'NPO_Status',
       'Number_of_campaigns_from_the_same_NPO_that_started_within_the_same_Year_Month',
       'Public_Campaign_Access', 'Creator_Type', 'Creator_Id',
       'Campaign_Status', 'Actual_Donation_Amount', 'Distinct_Donors',
       'Campaign_Goal', 'Campaign_Completion_Rate', 'Days_Left_for_Campaign',
       'Campaign_Start_Date', 'Campaign_End_Date',
       'NPO_Ipc_Status_For_Tax_Deductibility', 'Campaign_Image1_Id',
       'Campaign_Image2_Id', 'Campaign_Image3_Id', 'Campaign_Image4_Id',
       'Campaign_Image5_Id', 'Campaign_Video', 'Impact_Message_1',
       'Impact_Message_2', 'Impact_Message_3', 'Impact_Message_4',
       'Impact_Message_5', 'Custom_Amount_1', 'Custom_Amount_2',
       'Custom_Amount_3', 'Custom_Amount_4', 'Description_of_Campaign',
       'Description_of_NPO', 'Campaign_Title_y', 'Org_Cause_Animal_Welfare',
       'Org_Cause_Arts_&_Heritage', 'Org_Cause_Children_&_Youth',
       'Org_Cause_Community', 'Org_Cause_Disability', 'Org_Cause_Education',
       'Org_Cause_Elderly', 'Org_Cause_Environment', 'Org_Cause_Families',
       'Org_Cause_Health', 'Org_Cause_Humanitarian',
       'Org_Cause_Social_Service', 'Org_Cause_Sports',
       'Org_Cause_Women_&_Girls', 'Cam_Cause_Animal_Welfare',
       'Cam_Cause_Arts_&_Heritage', 'Cam_Cause_Children_&_Youth',
       'Cam_Cause_Community', 'Cam_Cause_Disability', 'Cam_Cause_Education',
       'Cam_Cause_Elderly', 'Cam_Cause_Environment', 'Cam_Cause_Families',
       'Cam_Cause_Health', 'Cam_Cause_Humanitarian',
       'Cam_Cause_Social_Service', 'Cam_Cause_Sports',
       'Cam_Cause_Women_&_Girls', 'Pub_Enquiry_Person', 'Pub_Enquiry_Contact',
       'Pub_Enquiry_Email', 'Web_URL', 'Facebook_Link', 'Org_causes',
       'Cam_causes', 'S/N', 'Type', 'UEN', 'IPC_Period', 'Sector',
       'Classification', 'Activities', 'Scale_type'],
      dtype='object')
```

In [114…
```
combined_data = combined_data.rename(columns={'Actual_Donation_Amount' : 'Amount_raised'} )
combined_data = combined_data.rename(columns={'NPO_Ipc_Status_For_Tax_Deductibility' : 'Tax_duction_status'} )
combined_data = combined_data.rename(columns={'Scale_type' : 'Financial_Size'} )
combined_data.columns
```

Out[1143]:
```
Index(['Campaign_Id', 'Campaign_Title_x', 'Receiving_NPO_Name',
       'Receiving_NPO_Id', 'NPO_Status',
       'Number_of_campaigns_from_the_same_NPO_that_started_within_the_same_Year_Month',
       'Public_Campaign_Access', 'Creator_Type', 'Creator_Id',
       'Campaign_Status', 'Amount_raised', 'Distinct_Donors', 'Campaign_Goal',
       'Campaign_Completion_Rate', 'Days_Left_for_Campaign',
       'Campaign_Start_Date', 'Campaign_End_Date', 'Tax_duction_status',
       'Campaign_Image1_Id', 'Campaign_Image2_Id', 'Campaign_Image3_Id',
       'Campaign_Image4_Id', 'Campaign_Image5_Id', 'Campaign_Video',
       'Impact_Message_1', 'Impact_Message_2', 'Impact_Message_3',
       'Impact_Message_4', 'Impact_Message_5', 'Custom_Amount_1',
       'Custom_Amount_2', 'Custom_Amount_3', 'Custom_Amount_4',
       'Description_of_Campaign', 'Description_of_NPO', 'Campaign_Title_y',
       'Org_Cause_Animal_Welfare', 'Org_Cause_Arts_&_Heritage',
       'Org_Cause_Children_&_Youth', 'Org_Cause_Community',
       'Org_Cause_Disability', 'Org_Cause_Education', 'Org_Cause_Elderly',
       'Org_Cause_Environment', 'Org_Cause_Families', 'Org_Cause_Health',
       'Org_Cause_Humanitarian', 'Org_Cause_Social_Service',
       'Org_Cause_Sports', 'Org_Cause_Women_&_Girls',
       'Cam_Cause_Animal_Welfare', 'Cam_Cause_Arts_&_Heritage',
       'Cam_Cause_Children_&_Youth', 'Cam_Cause_Community',
       'Cam_Cause_Disability', 'Cam_Cause_Education', 'Cam_Cause_Elderly',
       'Cam_Cause_Environment', 'Cam_Cause_Families', 'Cam_Cause_Health',
       'Cam_Cause_Humanitarian', 'Cam_Cause_Social_Service',
       'Cam_Cause_Sports', 'Cam_Cause_Women_&_Girls', 'Pub_Enquiry_Person',
       'Pub_Enquiry_Contact', 'Pub_Enquiry_Email', 'Web_URL', 'Facebook_Link',
       'Org_causes', 'Cam_causes', 'S/N', 'Type', 'UEN', 'IPC_Period',
       'Sector', 'Classification', 'Activities', 'Financial_Size'],
      dtype='object')
```

In [114…
```
msno.matrix(combined_data.sample(250))
```

Out[1144]:
```
<AxesSubplot: >
```



In [114…

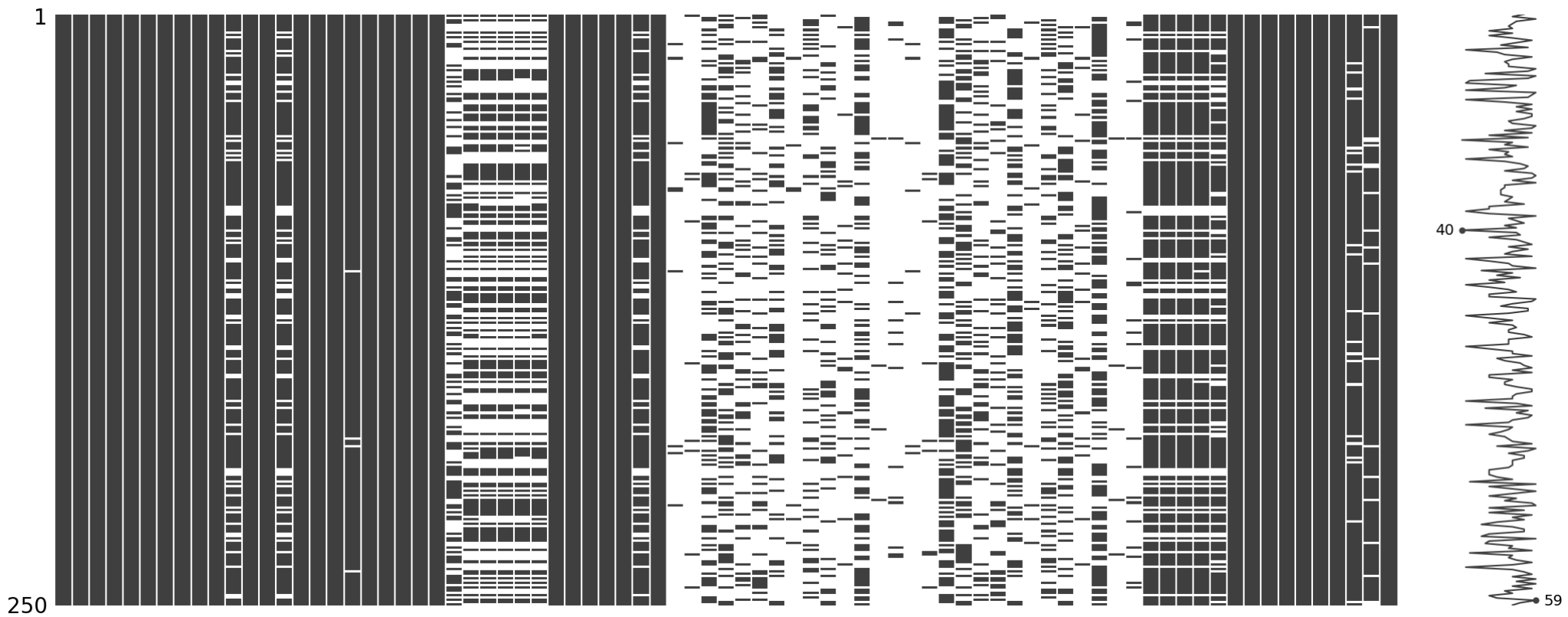
```
combined_data.columns #Show the columns of data set
```

Out[1145]:
```
Index(['Campaign_Id', 'Campaign_Title_x', 'Receiving_NPO_Name',
       'Receiving_NPO_Id', 'NPO_Status',
       'Number_of_campaigns_from_the_same_NPO_that_started_within_the_same_Year_Month',
       'Public_Campaign_Access', 'Creator_Type', 'Creator_Id',
       'Campaign_Status', 'Amount_raised', 'Distinct_Donors', 'Campaign_Goal',
       'Campaign_Completion_Rate', 'Days_Left_for_Campaign',
       'Campaign_Start_Date', 'Campaign_End_Date', 'Tax_duction_status',
       'Campaign_Image1_Id', 'Campaign_Image2_Id', 'Campaign_Image3_Id',
       'Campaign_Image4_Id', 'Campaign_Image5_Id', 'Campaign_Video',
       'Impact_Message_1', 'Impact_Message_2', 'Impact_Message_3',
       'Impact_Message_4', 'Impact_Message_5', 'Custom_Amount_1',
       'Custom_Amount_2', 'Custom_Amount_3', 'Custom_Amount_4',
       'Description_of_Campaign', 'Description_of_NPO', 'Campaign_Title_y',
       'Org_Cause_Animal_Welfare', 'Org_Cause_Arts_&_Heritage',
       'Org_Cause_Children_&_Youth', 'Org_Cause_Community',
       'Org_Cause_Disability', 'Org_Cause_Education', 'Org_Cause_Elderly',
       'Org_Cause_Environment', 'Org_Cause_Families', 'Org_Cause_Health',
       'Org_Cause_Humanitarian', 'Org_Cause_Social_Service',
       'Org_Cause_Sports', 'Org_Cause_Women_&_Girls',
       'Cam_Cause_Animal_Welfare', 'Cam_Cause_Arts_&_Heritage',
       'Cam_Cause_Children_&_Youth', 'Cam_Cause_Community',
       'Cam_Cause_Disability', 'Cam_Cause_Education', 'Cam_Cause_Elderly',
       'Cam_Cause_Environment', 'Cam_Cause_Families', 'Cam_Cause_Health',
       'Cam_Cause_Humanitarian', 'Cam_Cause_Social_Service',
       'Cam_Cause_Sports', 'Cam_Cause_Women_&_Girls', 'Pub_Enquiry_Person',
       'Pub_Enquiry_Contact', 'Pub_Enquiry_Email', 'Web_URL', 'Facebook_Link',
       'Org_causes', 'Cam_causes', 'S/N', 'Type', 'UEN', 'IPC_Period',
       'Sector', 'Classification', 'Activities', 'Financial_Size'],
      dtype='object')
```

In [114…
```
extract_data = combined_data #[Need_variable]
```

In [114…
```
msno.matrix(extract_data.sample(250))
```

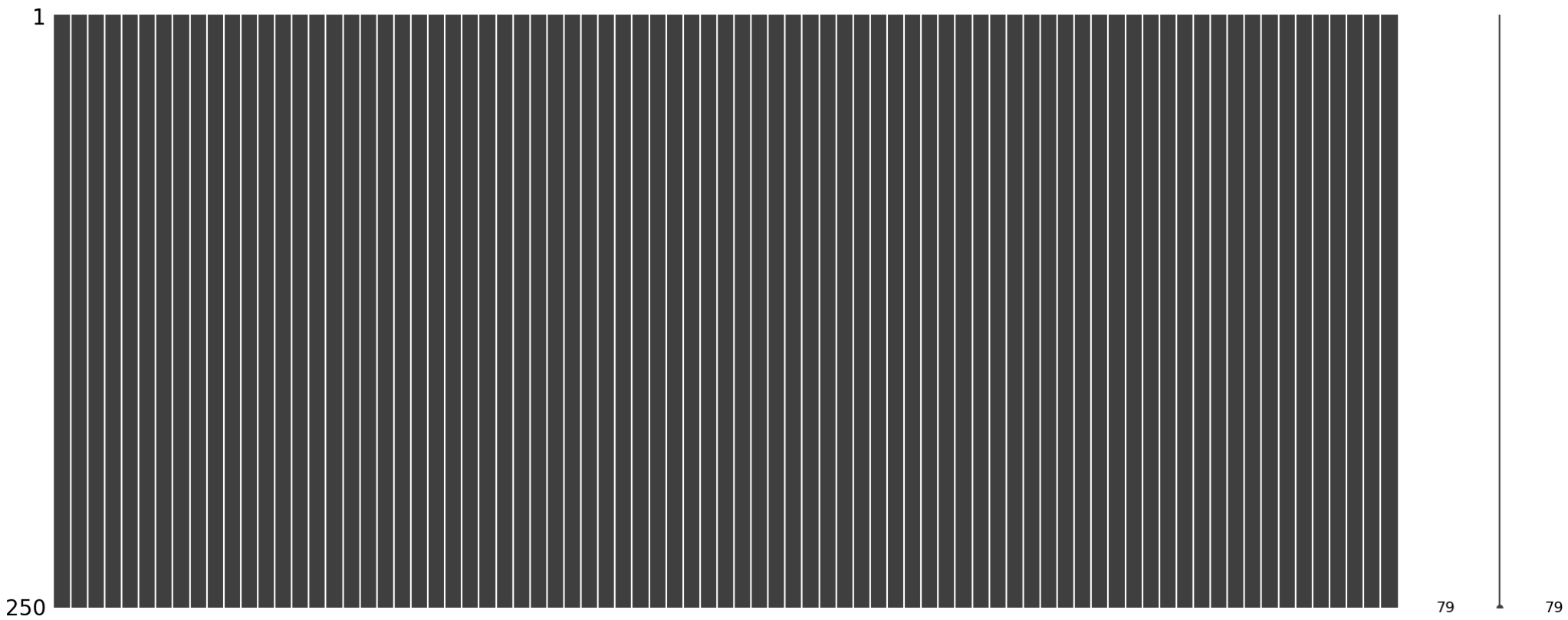
Out[1147]:
```
<AxesSubplot: >
```



In [114…
```
extract_data = extract_data.fillna('0')  #fill some missing causes data with 0
```

In [114…
```
msno.matrix(extract_data.sample(250))
```

Out[1149]: <AxesSubplot: >



In [115…
```
Total_Rows = combined_data.shape[0]  #Get the rows number
print(Total_Rows)  #Print out total rows number
```

```
10139
```

In [115…
```
extract_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10139 entries, 0 to 10138
Data columns (total 79 columns):
 #   Column                                                                    Non-Null Count  Dtype
---  ------                                                                    --------------  -----
 0   Campaign_Id                                                               10139 non-null  int64
 1   Campaign_Title_x                                                          10139 non-null  object
 2   Receiving_NPO_Name                                                        10139 non-null  object
 3   Receiving_NPO_Id                                                          10139 non-null  int64
 4   NPO_Status                                                                10139 non-null  bool
 5   Number_of_campaigns_from_the_same_NPO_that_started_within_the_same_Year_Month  10139 non-null  int64
 6   Public_Campaign_Access                                                    10139 non-null  int64
 7   Creator_Type                                                              10139 non-null  object
 8   Creator_Id                                                                10139 non-null  int64
 9   Campaign_Status                                                           10139 non-null  object
 10  Amount_raised                                                             10139 non-null  object
 11  Distinct_Donors                                                           10139 non-null  int64
 12  Campaign_Goal                                                             10139 non-null  int64
 13  Campaign_Completion_Rate                                                  10139 non-null  object
 14  Days_Left_for_Campaign                                                    10139 non-null  object
 15  Campaign_Start_Date                                                       10139 non-null  object
 16  Campaign_End_Date                                                         10139 non-null  object
 17  Tax_duction_status                                                        10139 non-null  object
 18  Campaign_Image1_Id                                                        10139 non-null  int64
 19  Campaign_Image2_Id                                                        10139 non-null  int64
 20  Campaign_Image3_Id                                                        10139 non-null  int64
 21  Campaign_Image4_Id                                                        10139 non-null  int64
 22  Campaign_Image5_Id                                                        10139 non-null  int64
 23  Campaign_Video                                                            10139 non-null  object
 24  Impact_Message_1                                                          10139 non-null  object
 25  Impact_Message_2                                                          10139 non-null  object
 26  Impact_Message_3                                                          10139 non-null  object
 27  Impact_Message_4                                                          10139 non-null  object
 28  Impact_Message_5                                                          10139 non-null  object
 29  Custom_Amount_1                                                           10139 non-null  int64
 30  Custom_Amount_2                                                           10139 non-null  int64
 31  Custom_Amount_3                                                           10139 non-null  int64
 32  Custom_Amount_4                                                           10139 non-null  int64
 33  Description_of_Campaign                                                   10139 non-null  object
 34  Description_of_NPO                                                        10139 non-null  object
 35  Campaign_Title_y                                                          10139 non-null  object
 36  Org_Cause_Animal_Welfare                                                  10139 non-null  object
 37  Org_Cause_Arts_&_Heritage                                                 10139 non-null  object
 38  Org_Cause_Children_&_Youth                                                10139 non-null  object
 39  Org_Cause_Community                                                       10139 non-null  object
 40  Org_Cause_Disability                                                      10139 non-null  object
 41  Org_Cause_Education                                                       10139 non-null  object
 42  Org_Cause_Elderly                                                         10139 non-null  object
 43  Org_Cause_Environment                                                     10139 non-null  object
 44  Org_Cause_Families                                                        10139 non-null  object
 45  Org_Cause_Health                                                          10139 non-null  object
 46  Org_Cause_Humanitarian                                                    10139 non-null  object
 47  Org_Cause_Social_Service                                                  10139 non-null  object
 48  Org_Cause_Sports                                                          10139 non-null  object
 49  Org_Cause_Women_&_Girls                                                   10139 non-null  object
 50  Cam_Cause_Animal_Welfare                                                  10139 non-null  object
 51  Cam_Cause_Arts_&_Heritage                                                 10139 non-null  object
 52  Cam_Cause_Children_&_Youth                                                10139 non-null  object
```

```
 53  Cam_Cause_Community                                10139 non-null  object
 54  Cam_Cause_Disability                               10139 non-null  object
 55  Cam_Cause_Education                                10139 non-null  object
 56  Cam_Cause_Elderly                                  10139 non-null  object
 57  Cam_Cause_Environment                              10139 non-null  object
 58  Cam_Cause_Families                                 10139 non-null  object
 59  Cam_Cause_Health                                   10139 non-null  object
 60  Cam_Cause_Humanitarian                             10139 non-null  object
 61  Cam_Cause_Social_Service                           10139 non-null  object
 62  Cam_Cause_Sports                                   10139 non-null  object
 63  Cam_Cause_Women_&_Girls                            10139 non-null  object
 64  Pub_Enquiry_Person                                 10139 non-null  object
 65  Pub_Enquiry_Contact                                10139 non-null  object
 66  Pub_Enquiry_Email                                  10139 non-null  object
 67  Web_URL                                            10139 non-null  object
 68  Facebook_Link                                      10139 non-null  object
 69  Org_causes                                         10139 non-null  int64
 70  Cam_causes                                         10139 non-null  int64
 71  S/N                                                10139 non-null  int64
 72  Type                                               10139 non-null  object
 73  UEN                                                10139 non-null  object
 74  IPC_Period                                         10139 non-null  object
 75  Sector                                             10139 non-null  object
 76  Classification                                     10139 non-null  object
 77  Activities                                         10139 non-null  object
 78  Financial_Size                                     10139 non-null  int64
dtypes: bool(1), int64(20), object(58)
memory usage: 6.0+ MB
```

There is no donations per donor, So add a columns of donations per donor

In [115…
```
extract_data['Distinct_Donors'] = pd.to_numeric( extract_data['Distinct_Donors'])
extract_data['Amount_raised'] = pd.to_numeric( extract_data['Amount_raised'])
```

In [115…
```
extract_data.columns
```

Out[1153]:
```
Index(['Campaign_Id', 'Campaign_Title_x', 'Receiving_NPO_Name',
       'Receiving_NPO_Id', 'NPO_Status',
       'Number_of_campaigns_from_the_same_NPO_that_started_within_the_same_Year_Month',
       'Public_Campaign_Access', 'Creator_Type', 'Creator_Id',
       'Campaign_Status', 'Amount_raised', 'Distinct_Donors', 'Campaign_Goal',
       'Campaign_Completion_Rate', 'Days_Left_for_Campaign',
       'Campaign_Start_Date', 'Campaign_End_Date', 'Tax_duction_status',
       'Campaign_Image1_Id', 'Campaign_Image2_Id', 'Campaign_Image3_Id',
       'Campaign_Image4_Id', 'Campaign_Image5_Id', 'Campaign_Video',
       'Impact_Message_1', 'Impact_Message_2', 'Impact_Message_3',
       'Impact_Message_4', 'Impact_Message_5', 'Custom_Amount_1',
       'Custom_Amount_2', 'Custom_Amount_3', 'Custom_Amount_4',
       'Description_of_Campaign', 'Description_of_NPO', 'Campaign_Title_y',
       'Org_Cause_Animal_Welfare', 'Org_Cause_Arts_&_Heritage',
       'Org_Cause_Children_&_Youth', 'Org_Cause_Community',
       'Org_Cause_Disability', 'Org_Cause_Education', 'Org_Cause_Elderly',
       'Org_Cause_Environment', 'Org_Cause_Families', 'Org_Cause_Health',
       'Org_Cause_Humanitarian', 'Org_Cause_Social_Service',
       'Org_Cause_Sports', 'Org_Cause_Women_&_Girls',
       'Cam_Cause_Animal_Welfare', 'Cam_Cause_Arts_&_Heritage',
       'Cam_Cause_Children_&_Youth', 'Cam_Cause_Community',
       'Cam_Cause_Disability', 'Cam_Cause_Education', 'Cam_Cause_Elderly',
       'Cam_Cause_Environment', 'Cam_Cause_Families', 'Cam_Cause_Health',
       'Cam_Cause_Humanitarian', 'Cam_Cause_Social_Service',
       'Cam_Cause_Sports', 'Cam_Cause_Women_&_Girls', 'Pub_Enquiry_Person',
       'Pub_Enquiry_Contact', 'Pub_Enquiry_Email', 'Web_URL', 'Facebook_Link',
       'Org_causes', 'Cam_causes', 'S/N', 'Type', 'UEN', 'IPC_Period',
       'Sector', 'Classification', 'Activities', 'Financial_Size'],
      dtype='object')
```

In [115…
```
extract_data['Creator_Type'].unique()
```

Out[1154]:
```
array(['NPO', 'INDIVIDUAL', 'Corporate', 'OTHER', 'COMMUNITY',
       'CORPORATE', 'GOVERNMENT', 'SCHOOL'], dtype=object)
```

2. Data processing

The format of the date needs to be modified and the duration will be calculated below

In [115…
```
month_dictionary = {'Jan':'1',
 'Feb':'2',
 'Mar':'3',
 'Apr':'4',
 'May':'5',
 'Jun':'6',
 'Jul':'7',
 'Aug':'8',
 'Sep':'9',
 'Oct':'10',
 'Nov':'11',
 'Dec':'12'}
extract_data['Campaign_Start_Day'] = '0'
```

```
extract_data['Campaign_Start_Month'] = '0'
extract_data['Campaign_Start_Year'] = '0'
extract_data['Campaign_End_Day'] = '0'
extract_data['Campaign_End_Month'] = '0'
extract_data['Campaign_End_Year'] = '0'
extract_data['Campaign_Start'] = '0'
extract_data['Campaign_End'] = '0'
extract_data['Campaign_Start_Year'] = '0'
extract_data['Campaign_Duration'] = '0'
i = 0

# Get the detail infomation of data
for row in extract_data['Campaign_Start_Date']:
    extract_data.loc[i, 'Campaign_Start_Day'] = extract_data.loc[i, 'Campaign_Start_Date'].split('-', 3 )[0]
    extract_data.loc[i, 'Campaign_Start_Month'] = month_dictionary[ extract_data.loc[i, 'Campaign_Start_Date'].split('-', 3 )[1] ]
    extract_data.loc[i, 'Campaign_Start_Year'] = '20'+ extract_data.loc[i, 'Campaign_Start_Date'].split('-', 3 )[2]
    extract_data.loc[i, 'Campaign_End_Day'] = extract_data.loc[i, 'Campaign_End_Date'].split('-', 3 )[0]
    extract_data.loc[i, 'Campaign_End_Month'] = month_dictionary[extract_data.loc[i, 'Campaign_End_Date'].split('-', 3 )[1]]
    extract_data.loc[i, 'Campaign_End_Year'] = '20' + extract_data.loc[i, 'Campaign_End_Date'].split('-', 3 )[2]
    extract_data.loc[i, 'Campaign_Start'] = extract_data.loc[i, 'Campaign_Start_Year'] + '-' + extract_data['Campaign_Start_Month'].iloc[i] + '-' + extract_data['C
    extract_data.loc[i, 'Campaign_End'] = extract_data['Campaign_End_Year'].iloc[i] + '-' + extract_data['Campaign_End_Month'].iloc[i] + '-' + extract_data['Campai
    extract_data.loc[i, 'Campaign_Duration'] = (datetime.strptime(extract_data.loc[i, 'Campaign_End'], '%Y-%m-%d') - datetime.strptime(extract_data.loc[i,'Campaign
#   if extract_data.loc[i, 'Campaign_Duration'] < 0:
#     extract_data.loc[i, 'Campaign_Duration'] = 0
    i += 1
```

Change variables to categorial

Classfy video into “0” and ”1“ two categories

In [115…

```
Video_or_not = lambda x0: (x0 != '0').astype(np.int)

extract_data["Campaign_Video"] = Video_or_not(extract_data["Campaign_Video"])
extract_data["Campaign_Video"].unique()
```

Out[1156]: array([1, 0])

Classfy Creator_type Sector Campaign_Start_Year

In [115…

```
num_deductibility = 0
extract_data['Donation_per_donor'] = 0
Creator_type = ['NPO', 'INDIVIDUAL', 'Corporate', 'OTHER', 'COMMUNITY', 'CORPORATE', 'GOVERNMENT', 'SCHOOL']
Sector = ['Social and Welfare', 'Others', 'Health', 'Arts and Heritage', 'Sports', 'Education', 'Religious', 'Community']
extract_data['Sector_type'] = 0
Campaign_Start_Year = ['2017', '2018', '2019', '2020', '2021', '2022']
for j in range(len(extract_data["Amount_raised"])):
    if extract_data["Distinct_Donors"].iloc[j] != 0:
        extract_data['Donation_per_donor'].iloc[j] = extract_data['Amount_raised'].iloc[j]/extract_data['Distinct_Donors'].iloc[j]
    else:
        extract_data['Donation_per_donor'].iloc[j] = 0

    if extract_data['Tax_duction_status'].iloc[j] == True:
```

```
        extract_data.loc[j, 'Tax_duction_status'] = 1
        num_deductibility += 1
    else:
        extract_data.loc[j, 'Tax_duction_status'] = 0
    if extract_data['Creator_Type'].iloc[j] == Creator_type[0]:
        extract_data['Creator_Type'].iloc[j] = 0
    if extract_data['Creator_Type'].iloc[j] == Creator_type[1]:
        extract_data['Creator_Type'].iloc[j] = 1
    if extract_data['Creator_Type'].iloc[j] == Creator_type[2]:
        extract_data['Creator_Type'].iloc[j] = 2
    if extract_data['Creator_Type'].iloc[j] == Creator_type[3]:
        extract_data['Creator_Type'].iloc[j] = 3
    if extract_data['Creator_Type'].iloc[j] == Creator_type[4]:
        extract_data['Creator_Type'].iloc[j] = 4
    if extract_data['Creator_Type'].iloc[j] == Creator_type[5]:
        extract_data['Creator_Type'].iloc[j] = 5
    if extract_data['Creator_Type'].iloc[j] == Creator_type[6]:
        extract_data['Creator_Type'].iloc[j] = 6
    if extract_data['Creator_Type'].iloc[j] == Creator_type[7]:
        extract_data['Creator_Type'].iloc[j] = 7

    if extract_data['Sector'].iloc[j] == Sector[0]:
        extract_data['Sector_type'].iloc[j] = 0
    if extract_data['Sector'].iloc[j] == Sector[1]:
        extract_data['Sector_type'].iloc[j] = 1
    if extract_data['Sector'].iloc[j] == Sector[2]:
        extract_data['Sector_type'].iloc[j] = 2
    if extract_data['Sector'].iloc[j] == Sector[3]:
        extract_data['Sector_type'].iloc[j] = 3
    if extract_data['Sector'].iloc[j] == Sector[4]:
        extract_data['Sector_type'].iloc[j] = 4
    if extract_data['Sector'].iloc[j] == Sector[5]:
        extract_data['Sector_type'].iloc[j] = 5
    if extract_data['Sector'].iloc[j] == Sector[6]:
        extract_data['Sector_type'].iloc[j] = 6
    if extract_data['Sector'].iloc[j] == Sector[7]:
        extract_data['Sector_type'].iloc[j] = 7


print("Number of deductibility:", num_deductibility)
```

```
Number of deductibility: 9633
```

In [115…

```
Campaign_Start_Year = ['2017', '2018', '2019', '2020', '2021', '2022']

extract_data['Campaign_Start_Year_category'] = 0
for j in range(len(extract_data["Amount_raised"])):
    if extract_data['Campaign_Start_Year'].iloc[j] == Campaign_Start_Year[0]:
        extract_data['Campaign_Start_Year_category'].iloc[j] = 0
    if extract_data['Campaign_Start_Year'].iloc[j] == Campaign_Start_Year[1]:
        extract_data['Campaign_Start_Year_category'].iloc[j] = 1
    if extract_data['Campaign_Start_Year'].iloc[j] == Campaign_Start_Year[2]:
        extract_data['Campaign_Start_Year_category'].iloc[j] = 2
    if extract_data['Campaign_Start_Year'].iloc[j] == Campaign_Start_Year[3]:
        extract_data['Campaign_Start_Year_category'].iloc[j] = 3
    if extract_data['Campaign_Start_Year'].iloc[j] == Campaign_Start_Year[4]:
        extract_data['Campaign_Start_Year_category'].iloc[j] = 4
```

```
    if extract_data['Campaign_Start_Year'].iloc[j] == Campaign_Start_Year[5]:
        extract_data['Campaign_Start_Year_category'].iloc[j] = 5
```

In [115…
```
extract_data['Campaign_Funds_Raised'] = 0
for index, row in extract_data.iterrows():
    if extract_data.loc[index,'Amount_raised']>0 :
        extract_data.loc[index,'Campaign_Funds_Raised'] = 1
    if extract_data.loc[index,'Financial_Size'] == 50000:
        extract_data.loc[index,'Financial_Size'] = 0
    if extract_data.loc[index,'Financial_Size'] == 200000:
        extract_data.loc[index,'Financial_Size'] = 1
    if extract_data.loc[index,'Financial_Size'] == 500000:
        extract_data.loc[index,'Financial_Size'] = 2
    if extract_data.loc[index,'Financial_Size'] == 250000:
        extract_data.loc[index,'Financial_Size'] = 3
    if extract_data.loc[index,'Financial_Size'] == 1000000:
        extract_data.loc[index,'Financial_Size'] = 4
    if extract_data.loc[index,'Financial_Size'] == 5000000:
        extract_data.loc[index,'Financial_Size'] = 5
    if extract_data.loc[index,'Financial_Size'] == 20000000:
        extract_data.loc[index,'Financial_Size'] = 6
    if extract_data.loc[index,'Financial_Size'] == 10000000:
        extract_data.loc[index,'Financial_Size'] = 7
```

Calculate the numbers of "org_causes" and "camp_causes"

In [116…
```
Org_causes = ['Org_Cause_Animal_Welfare', 'Org_Cause_Arts_&_Heritage',
       'Org_Cause_Children_&_Youth', 'Org_Cause_Community',
       'Org_Cause_Disability', 'Org_Cause_Education', 'Org_Cause_Elderly',
       'Org_Cause_Environment', 'Org_Cause_Families', 'Org_Cause_Health',
       'Org_Cause_Humanitarian', 'Org_Cause_Social_Service',
       'Org_Cause_Sports', 'Org_Cause_Women_&_Girls',
       ]
Cam_causes = ['Cam_Cause_Animal_Welfare', 'Cam_Cause_Arts_&_Heritage',
       'Cam_Cause_Children_&_Youth', 'Cam_Cause_Community',
       'Cam_Cause_Disability', 'Cam_Cause_Education', 'Cam_Cause_Elderly',
       'Cam_Cause_Environment', 'Cam_Cause_Families', 'Cam_Cause_Health',
       'Cam_Cause_Humanitarian', 'Cam_Cause_Social_Service',
       'Cam_Cause_Sports', 'Cam_Cause_Women_&_Girls']
Length_Org_causes = len(Org_causes)
Length_Cam_causes = len(Cam_causes)
extract_data['Org_causes'] = 0
extract_data['Cam_causes'] = 0

for j in range(Total_Rows):
    num_Org_causes = 0
    num_Cam_causes = 0
    for position1 in range(Length_Org_causes):
        num_Org_causes += 1 if extract_data[Org_causes[position1]].iloc[j] != '0' else 0
    extract_data['Org_causes'].iloc[j] = num_Org_causes
    for position2 in range(Length_Cam_causes):
        num_Cam_causes += 1 if extract_data[Cam_causes[position2]].iloc[j] != '0' else 0
    extract_data['Org_causes'].iloc[j] = num_Org_causes
    extract_data['Cam_causes'].iloc[j] = num_Cam_causes
```

Add a columns of numbers of images

In [116…
```
Image_Number = lambda x0,x1,x2,x3,x4: (x0 != 0).astype(np.int) +(x1 != 0).astype(np.int) + (x2 != 0).astype(np.int) + (x3 != 0).astype(np.int) + (x4 != 0).astype(n
extract_data["Number_of_images"] =  Image_Number(extract_data["Campaign_Image1_Id"],extract_data["Campaign_Image2_Id"],extract_data["Campaign_Image3_Id"],extract_d
```

In [116…
```
# extract_data.to_csv('before_clean.csv')
```

3. Clean data

Here delete the rows where Campaign_Duration less than 1

In [116…
```
num = 0
for index, row in extract_data.iterrows():
    if extract_data.loc[index,'Campaign_Duration'] <= 1:
        extract_data.drop(index, inplace=True)
        num += 1
print("Total delete numbers where duraiton less than 0:", num)
Total_Rows = extract_data.shape[0]
```

```
Total delete numbers where duraiton less than 0: 97
```

In [116…
```
Impact_msg_list = ['Impact_Message_1','Impact_Message_2','Impact_Message_3','Impact_Message_4','Impact_Message_5']
Msg_category_list = ['Msg1_category','Msg2_category','Msg3_category','Msg4_category','Msg5_category']
def sentence_length(s):
    return len([i for i in s.split(' ') if i])

#for j in range(len(Impact_msg_list)):
#    cnt=0
#    for s in extract_data[Impact_msg_list[j]]:
#        extract_data[Msg_category_list[j]].iloc[cnt] = 0 if sentence_length(s) <= 2 else (1 if sentence_length(s) <=7 else (2 if sentence_length(s) <=20 else  3))
#        cnt += 1
cnt=0
```

In [116…
```
# Nunber of description words
extract_data['Words_of_campaigns'] = 0
extract_data['Num_desc_NPO'] = 0
for index, row in extract_data.iterrows():

    extract_data.loc[index, 'Words_of_campaigns'] = sentence_length( str(extract_data.loc[index, 'Description_of_Campaign']))
    extract_data.loc[index, 'Num_desc_NPO'] = sentence_length( str(extract_data.loc[index, 'Description_of_NPO']))
```

Delete the lines where NPO is blank

In [116…
```
num1 = 0
for index, row in extract_data.iterrows():
    if extract_data.loc[index, 'Num_desc_NPO'] <= 1 and   extract_data.loc[index, 'Amount_raised']<=0 :
        extract_data.drop(index, inplace=True)
```

```
        num1 += 1
print("Total delete numbers where Num_desc_NPO less than 1:", num1)
Total_Rows = extract_data.shape[0]
```

```
Total delete numbers where Num_desc_NPO less than 1: 1987
```

Re-consider description contents again

In [116…
```
num2 = 0
num3 = 0
for index, row in extract_data.iterrows():
    if ("TEST" in str(extract_data.loc[index, 'Description_of_Campaign']).upper() or "TEST" in str(extract_data.loc[index, 'Campaign_Title_x']).upper()) and extrac
        extract_data.drop(index, inplace=True)
        num2 += 1
'''
for index, row in extract_data.iterrows():
    if extract_data.loc[index, 'Words_of_campaigns']< 5:
        extract_data.drop(index, inplace=True)
        num2 += 1
'''
print("Total delete numbers:", num2)
```

```
Total delete numbers: 0
```

In [116…
```
# extract_data['Cam_Org_causes'] = extract_data['Org_causes']-extract_data['Cam_causes']
```

Average donation per donator

In [116…
```
# extract_data['Avg_donation_amount'] =  extract_data['Amount_raised']/extract_data['Distinct_Donors']
```

Here delete the rows where Amount_raised is 0 but Distinct_Donors bigger than 0

In [117…
```
num = 0
for index, row in extract_data.iterrows():
    if extract_data.loc[index,'Amount_raised'] <= 0 and extract_data.loc[index,'Distinct_Donors'] > 0 :
        extract_data.drop(index, inplace=True)
        num += 1
print("Total delete numbers where Amount_raised less than 0 but Distinct_Donors bigger than 0:", num)
Total_Rows = extract_data.shape[0]
```

```
Total delete numbers where Amount_raised less than 0 but Distinct_Donors bigger than 0: 58
```

Remove campaigns where Number of campaigns from the same NPO that started within the same Year Month = 984 or = 106

In [117…
```
num = 0
for index, row in extract_data.iterrows():
    if extract_data.loc[index,'Number_of_campaigns_from_the_same_NPO_that_started_within_the_same_Year_Month'] == 984 or extract_data.loc[index,'Number_of_campaign
    #if extract_data.loc[index,'Number_of_campaigns_from_the_same_NPO_that_started_within_the_same_Year_Month'] > 16:
        extract_data.drop(index, inplace=True)
        num += 1
```

```
print("Total delete numbers where Remove campaigns where Number of campaigns from the same NPO that started within the same Year Month = 984 or = 106:", num)
Total_Rows = extract_data.shape[0]
```

```
Total delete numbers where Remove campaigns where Number of campaigns from the same NPO that started within the same Year Month = 984 or = 106: 37
```

Remove all active campaigns. The last campaign should be as of END OCT 2022

In [117…
```
num = 0
for index, row in extract_data.iterrows():
    if extract_data.loc[index,'Campaign_Status'].upper() != "ENDED" :
        extract_data.drop(index, inplace=True)
        num += 1
print("Total delete numbers where Remove all active campaigns. The last campaign should be as of END OCT 2022:", num)
Total_Rows = extract_data.shape[0]
```

```
Total delete numbers where Remove all active campaigns. The last campaign should be as of END OCT 2022: 612
```

Remove 40,000, 56,000, 80,000, and 100,000

In [117…
```
num = 0
for index, row in extract_data.iterrows():
    if extract_data.loc[index,'Custom_Amount_4'] >=40000 :
        extract_data.drop(index, inplace=True)
        num += 1
print("Remove 40,000, 56,000, 80,000, and 100,000:", num)
Total_Rows = extract_data.shape[0]
```

```
Remove 40,000, 56,000, 80,000, and 100,000: 8
```

In [117…
```
extract_data.shape
```

Out[1174]: (7340, 95)

Add new column call Campaign Funds Raised

In [117…
```
extract_data.shape
```

Out[1175]: (7340, 95)

In [117…
```
#extract_data.to_csv("Cleaned_data.csv")
```

In [117…
```
extract_data['Default_Contribution'] = (extract_data['Custom_Amount_1']+extract_data['Custom_Amount_2'] +extract_data['Custom_Amount_3'] + extract_data['Custom_Amo
```

4. Other Processing

Polarity

In [117…

```
'''
extract_data['Msg1_polarity'] = 0
extract_data['Msg1_subjectivity'] = 0
extract_data['Msg2_polarity'] = 0
extract_data['Msg2_subjectivity'] = 0
extract_data['Msg3_polarity'] = 0
extract_data['Msg3_subjectivity'] = 0
extract_data['Msg4_polarity'] = 0
extract_data['Msg4_subjectivity'] = 0
extract_data['Msg5_polarity'] = 0
extract_data['Msg5_subjectivity'] = 0


extract_data['Description_Campaign_polarity'] = 0
#extract_data['Description_Campaign_subjectivity'] = 0

from textblob import TextBlob
# polarity项为文本积极性，是在[-1.0，1.0]范围内的浮点数
# subjectivity项为主观评分，是在[0.0，1.0]范围内的浮点数，其中0.0是非常客观的，而1.0是非常主观的
#Impact_msg_list = ['Impact Message 1','Impact Message 2','Impact Message 3','Impact Message 4','Impact Message 5','Description of Campaign']
Impact_msg_list = ['Description_of_Campaign']
#Msg_polarity_list = ['Msg1_polarity','Msg2_polarity','Msg3_polarity','Msg4_polarity','Msg5_polarity','Description_Campaign_polarity']
Msg_polarity_list = ['Description_Campaign_polarity']
#Msg1_subjectivity_list = ['Msg1_subjectivity','Msg2_subjectivity','Msg3_subjectivity','Msg4_subjectivity','Msg5_subjectivity','Description_Campaign_subjectivity']
for j in range(len(Impact_msg_list)):
    t=0
    for i in extract_data[Impact_msg_list[j]]:
        blob = TextBlob(str(i))
        sentiment = blob.sentiment
        extract_data[Msg_polarity_list[j]].iloc[t] = sentiment.polarity
        #extract_data[Msg1_subjectivity_list[j]].iloc[t] = sentiment.subjectivity
        t+=1
# sum the total five messages polarity and subjectivity
#extract_data["Total_Msg_polarity"] = extract_data["Msg1_polarity"]+extract_data["Msg2_polarity"]+extract_data["Msg3_polarity"]+extract_data["Msg4_polarity"]+extra
#extract_data["Total_Msg_subjectivity"] = extract_data["Msg1_subjectivity"]+extract_data["Msg2_subjectivity"]+extract_data["Msg3_subjectivity"]+extract_data["Msg4_
extract_data.iloc[0:30,28:]
'''
```

Out[1178]:

```
'\nextract_data[\'Msg1_polarity\'] = 0\nextract_data[\'Msg1_subjectivity\'] = 0\nextract_data[\'Msg2_polarity\'] = 0\nextract_data[\'Msg2_subjectivity\'] = 0\nex
tract_data[\'Msg3_polarity\'] = 0\nextract_data[\'Msg3_subjectivity\'] = 0\nextract_data[\'Msg4_polarity\'] = 0\nextract_data[\'Msg4_subjectivity\'] = 0\nextract
_data[\'Msg5_polarity\'] = 0\nextract_data[\'Msg5_subjectivity\'] = 0\n\n\nextract_data[\'Description_Campaign_polarity\'] = 0\n#extract_data[\'Description_Campa
ign_subjectivity\'] = 0\n\nfrom textblob import TextBlob\n# polarity项为文本积极性，是在[-1.0，1.0]范围内的浮点数\n# subjectivity项为主观评分，是在[0.0，1.0]范围内的浮点数，
其中0.0是非常客观的，而1.0是非常主观的\n#Impact_msg_list = [\'Impact Message 1\',\'Impact Message 2\',\'Impact Message 3\',\'Impact Message 4\',\'Impact Message 5
\',\'Description of Campaign\']\nImpact_msg_list = [\'Description_of_Campaign\']\n#Msg_polarity_list = [\'Msg1_polarity\',\'Msg2_polarity\',\'Msg3_polarity\',\'M
sg4_polarity\',\'Msg5_polarity\',\'Description_Campaign_polarity\']\nMsg_polarity_list = [\'Description_Campaign_polarity\']\n#Msg1_subjectivity_list = [\'Msg1_s
ubjectivity\',\'Msg2_subjectivity\',\'Msg3_subjectivity\',\'Msg4_subjectivity\',\'Msg5_subjectivity\',\'Description_Campaign_subjectivity\']\nfor j in range(len
(Impact_msg_list)):\n    t=0\n    for i in extract_data[Impact_msg_list[j]]:\n        blob = TextBlob(str(i))\n        sentiment = blob.sentiment\n        extrac
t_data[Msg_polarity_list[j]].iloc[t] = sentiment.polarity\n        #extract_data[Msg1_subjectivity_list[j]].iloc[t] = sentiment.subjectivity\n        t+=1\n# sum
the total five messages polarity and subjectivity \n#extract_data["Total_Msg_polarity"] = extract_data["Msg1_polarity"]+extract_data["Msg2_polarity"]+extract_dat
a["Msg3_polarity"]+extract_data["Msg4_polarity"]+extract_data["Msg5_polarity"]\n#extract_data["Total_Msg_subjectivity"] = extract_data["Msg1_subjectivity"]+extra
ct_data["Msg2_subjectivity"]+extract_data["Msg3_subjectivity"]+extract_data["Msg4_subjectivity"]+extract_data["Msg5_subjectivity"]\nextract_data.iloc[0:30,28:]
\n'
```

Future tense

In [117…
```
# It is accomplished mainly by using word_tokenize package.
# import the releated package
from nltk import word_tokenize, pos_tag
# This package is used for spliting the sentence
# Here is how it works
sentence = "Your donation will testament that suicide prevention is everyone's business."
tokens = word_tokenize(sentence)
print(tokens)
# ['Your', 'donation', 'will', 'testament', 'that', 'suicide', 'prevention', 'is', 'everyone', "'s", 'business', '.']
# Tag different compoments in the sentence
print(pos_tag(tokens))
'''
Here is the result after taging
[('Your', 'PRP$'),
 ('donation', 'NN'),
 ('will', 'MD'),
 ('testament', 'VB'),
 ('that', 'IN'),
 ('suicide', 'JJ'),
 ('prevention', 'NN'),
 ('is', 'VBZ'),
 ('everyone', 'NN'),
 ("'s", 'POS'),
 ('business', 'NN'),
 ('.', '.')]
'''
# MD modal could, will
# So we can use the number of MD tags to judge the future tense.
# Another example   can.may.might.could.should.would.will.must.
sentence2 = 'Your donation will testament that suicide prevention is everyone business.Your $500 donation (5 tins) can contribute towards \
providing a month of meals for a child in HomeSweetHome@Admiralty (HSH).'
text = word_tokenize(sentence2)
tagged = pos_tag(text)

print( ([word for word in tagged if word[1] == "MD"]))
# The result is 2
```

```
['Your', 'donation', 'will', 'testament', 'that', 'suicide', 'prevention', 'is', 'everyone', "'s", 'business', '.']
[('Your', 'PRP$'), ('donation', 'NN'), ('will', 'MD'), ('testament', 'VB'), ('that', 'IN'), ('suicide', 'JJ'), ('prevention', 'NN'), ('is', 'VBZ'), ('everyone',
'NN'), ("'s", 'POS'), ('business', 'NN'), ('.', '.')]
[('will', 'MD'), ('can', 'MD')]
```

In [118…
```
len([word for word in tagged if word[1] == "VB"])
```

Out[1180]: 2

In [118…
```
MD_list = []
test = [word for word in tagged if word[1] == "MD"]
for ite in test:
    if ite[0] not in MD_list:
        MD_list.append(ite[0])
print(MD_list)
```

```
['will', 'can']
```

In [118…
```
from nltk import word_tokenize, pos_tag
MD_list = []
def determine_tense_input(sentence):
```

```
    text = word_tokenize(sentence)
    tagged = pos_tag(text)
    tense = {}
    tense["future"] = len([word for word in tagged if word[1] == "MD"])
    test = [word for word in tagged if word[1] == "MD"]
    for ite in test:
        if ite[0] not in MD_list:
            MD_list.append(ite[0])
    return(tense)
```

In [118…
```
print(MD_list)
```

```
[]
```

In [118…
```
# Nunber of description words
extract_data['Campaign_promises'] = 0
extract_data['Campaign_promises_percentage'] = 0
#extract_data['Num_desc_NPO'] = 0

for index, row in extract_data.iterrows():
    Impact_msg_str = ''
    Impact_msg_str += extract_data.loc[index,'Impact_Message_1']
    Impact_msg_str += extract_data.loc[index,'Impact_Message_2']
    Impact_msg_str += extract_data.loc[index,'Impact_Message_3']
    Impact_msg_str += extract_data.loc[index,'Impact_Message_4']
    Impact_msg_str += extract_data.loc[index,'Impact_Message_5']
    extract_data.loc[index,'Campaign_promises'] = determine_tense_input(Impact_msg_str)['future']
    extract_data.loc[index,'Campaign_promises_percentage'] = determine_tense_input(Impact_msg_str)['future']/sentence_length(Impact_msg_str)
```

In [118…
```
extract_data['Campaign_promises_percentage'].max()
```

Out[1185]:
```
0.12903225806451613
```

In [118…
```
extract_data['Campaign_promises_percentage'].min()
```

Out[1186]:
```
0.0
```

Convert to numeric

In [118…
```
numeric_features1 = ['Amount_raised', 'Campaign_Goal', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',  'Cam_causes', 'Description_Campaign_polarity',"Creator_Type","Campaign_Start_Year
extract_data['Amount_raised'] = pd.to_numeric( extract_data['Amount_raised'])
extract_data['Campaign_Goal'] = pd.to_numeric( extract_data['Campaign_Goal'])
extract_data['Tax_duction_status'] = pd.to_numeric( extract_data['Tax_duction_status'])
extract_data['Campaign_Duration'] = pd.to_numeric( extract_data['Campaign_Duration'])
extract_data['Campaign_Video'] = pd.to_numeric( extract_data['Campaign_Video'])
extract_data['Number_of_images'] = pd.to_numeric( extract_data['Number_of_images'])
extract_data['Words_of_campaigns'] = pd.to_numeric( extract_data['Words_of_campaigns'])
extract_data['Cam_causes'] = pd.to_numeric( extract_data['Cam_causes'])
extract_data['Creator_Type'] = pd.to_numeric(extract_data['Creator_Type'])
#extract_data['Total_Msg_polarity'] = pd.to_numeric( extract_data['Total_Msg_polarity'])
extract_data['Tax_duction_status'] = pd.to_numeric(extract_data['Tax_duction_status'])
extract_data['Campaign_Video'] = pd.to_numeric(extract_data['Campaign_Video'])
extract_data['Campaign_Start_Year'] = pd.to_numeric(extract_data['Campaign_Start_Year'])
#extract_data['Campaign Promise'] = pd.to_numeric(extract_data['Campaign Promise'])
```

```
extract_data["Distinct_Donors"] = pd.to_numeric(extract_data["Distinct_Donors"])
#extract_data['Sector'] = pd.to_numeric( extract_data['Sector'])
extract_data['Sector_type'] = pd.to_numeric(extract_data['Sector_type'] )
extract_data['Campaign_Video'].unique()
```

Out[1187]: array([1, 0])

Log Transformation

In [118…
```
extract_data = extract_data.rename(columns={'Number_of_campaigns_from_the_same_NPO_that_started_within_the_same_Year_Month':'Campaign_frequency'})
```

In [118…
```
extract_data['Campaign_frequency'] =  pd.to_numeric(extract_data['Campaign_frequency'])
extract_data['Campaign_Start_Year'] =  pd.to_numeric(extract_data['Campaign_Start_Year'])
```

In [119…
```
extract_data['Amount_raised_Log']=0
extract_data['Distinct_Donors_Log'] = 0
extract_data['Campaign_Goal_Log'] = 0
extract_data['Campaign_Duration_Log'] = 0
extract_data['Default_Contribution_Log'] = 0
extract_data.loc[index,'Campaign_promises_X_Campaign_frequency'] = 0
extract_data.loc[index,'Campaign_frequency_X_Default_Contribution_Log'] = 0
extract_data['Campaign_success'] = 0
extract_data['Campaign_success_Log'] = 0
eps = 1
for index, row in extract_data.iterrows():
    if extract_data.loc[index,'Amount_raised'] > 0:
        extract_data.loc[index,'Amount_raised_Log'] = np.log(extract_data.loc[index,'Amount_raised'])
    else:
        extract_data.loc[index,'Amount_raised_Log'] = np.log(eps)

    if extract_data.loc[index,'Distinct_Donors'] > 0:
        extract_data.loc[index,'Distinct_Donors_Log'] = np.log(extract_data.loc[index,'Distinct_Donors'])
    else:
        extract_data.loc[index,'Distinct_Donors_Log'] = np.log(eps)

    if extract_data.loc[index,'Campaign_Goal'] > 0:
        extract_data.loc[index,'Campaign_Goal_Log'] = np.log(extract_data.loc[index,'Campaign_Goal'])
    else:
        extract_data.loc[index,'Campaign_Goal_Log'] = np.log(eps)

    if extract_data.loc[index,'Campaign_Duration'] > 0:
        extract_data.loc[index,'Campaign_Duration_Log'] = np.log(extract_data.loc[index,'Campaign_Duration'])
    else:
        extract_data.loc[index,'Campaign_Duration_Log'] = np.log(eps)

    if extract_data.loc[index,'Default_Contribution'] > 0:
        extract_data.loc[index,'Default_Contribution_Log'] = np.log(extract_data.loc[index,'Default_Contribution'])
    else:
        extract_data.loc[index,'Default_Contribution_Log'] = np.log(eps)

    if extract_data.loc[index,'Campaign_Goal'] > 0 and extract_data.loc[index,'Amount_raised'] > 0:
        if extract_data.loc[index,'Campaign_Goal'] <= extract_data.loc[index,'Amount_raised']:
            #extract_data.loc[index,'Campaign_success'] =1
            #extract_data.loc[index,'Campaign_success_Log'] = 1
            extract_data.loc[index,'Campaign_success'] = extract_data.loc[index,'Amount_raised']/extract_data.loc[index,'Campaign_Goal']
```

```
            extract_data.loc[index,'Campaign_success_Log'] = np.log( extract_data.loc[index,'Amount_raised']/extract_data.loc[index,'Campaign_Goal'])
        else:
            extract_data.loc[index,'Campaign_success'] = extract_data.loc[index,'Amount_raised']/extract_data.loc[index,'Campaign_Goal']
            extract_data.loc[index,'Campaign_success_Log'] = np.log( extract_data.loc[index,'Amount_raised']/extract_data.loc[index,'Campaign_Goal'])

    else:
        extract_data.loc[index,'Campaign_success_Log'] = np.log(eps)

    if extract_data.loc[index,'Org_causes'] > 0:
        extract_data.loc[index,'Log_Org_causes'] = np.log(extract_data.loc[index,'Org_causes'])
    else:
        extract_data.loc[index,'Log_Org_causes'] = np.log(eps)

    extract_data.loc[index,'Campaign_promises_X_Campaign_frequency'] = extract_data.loc[index, 'Campaign_promises'] * extract_data.loc[index,'Campaign_frequency']
    extract_data.loc[index,'Campaign_promises_percentage_X_Campaign_frequency'] = extract_data.loc[index, 'Campaign_promises_percentage'] * extract_data.loc[index,
    extract_data.loc[index,'Campaign_frequency_X_Default_Contribution_Log'] = extract_data.loc[index, 'Campaign_frequency'] * extract_data.loc[index,'Default_Contr
    extract_data.loc[index,'Campaign_promises_percentage_X_Default_Contribution_Log'] = extract_data.loc[index, 'Campaign_promises_percentage'] * extract_data.loc[
    extract_data.loc[index,'Campaign_promises_percentage_X_Default_Contribution'] = extract_data.loc[index, 'Campaign_promises_percentage'] * extract_data.loc[inde
```

In [119…

```
extract_data["Campaign_success"] = pd.to_numeric(extract_data["Campaign_success"])
extract_data["Distinct_Donors_Log"] = pd.to_numeric(extract_data["Distinct_Donors_Log"])
extract_data['Campaign_success'].unique()
```

Out[1191]:

```
array([0.11122   , 0.47916   , 1.175     , ..., 0.29099333, 0.03846154,
       0.64464062])
```

In [119…

```
extract_data['Amount_raised_Log_round'] = round(extract_data['Amount_raised_Log'])
extract_data['Amount_raised_Log_round'].value_counts()
```

Out[1192]:

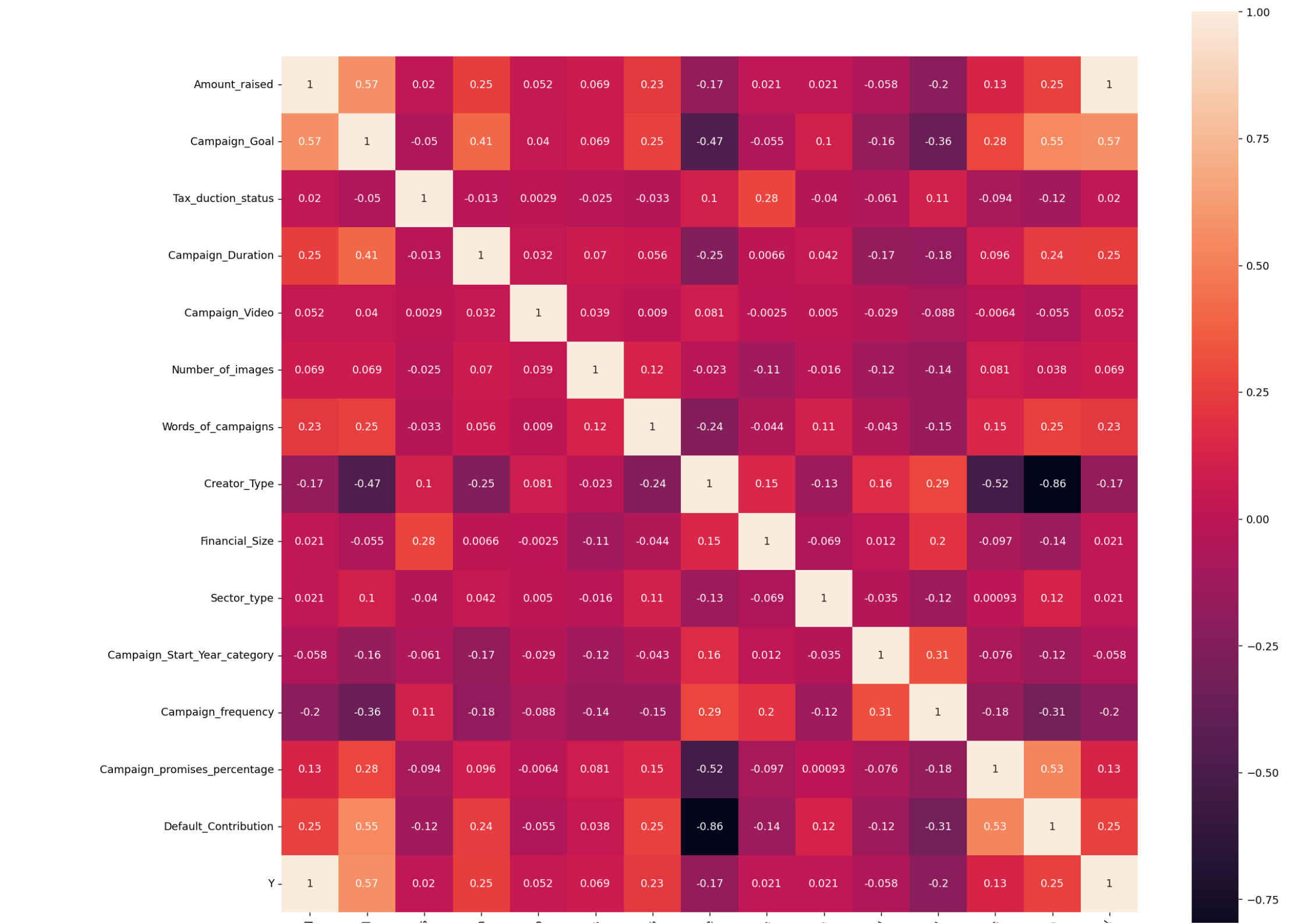
```
8.0     1456
9.0     1375
7.0     1359
6.0      929
10.0     762
5.0      562
11.0     318
4.0      251
3.0      124
12.0      91
2.0       87
13.0      21
14.0       5
Name: Amount_raised_Log_round, dtype: int64
```

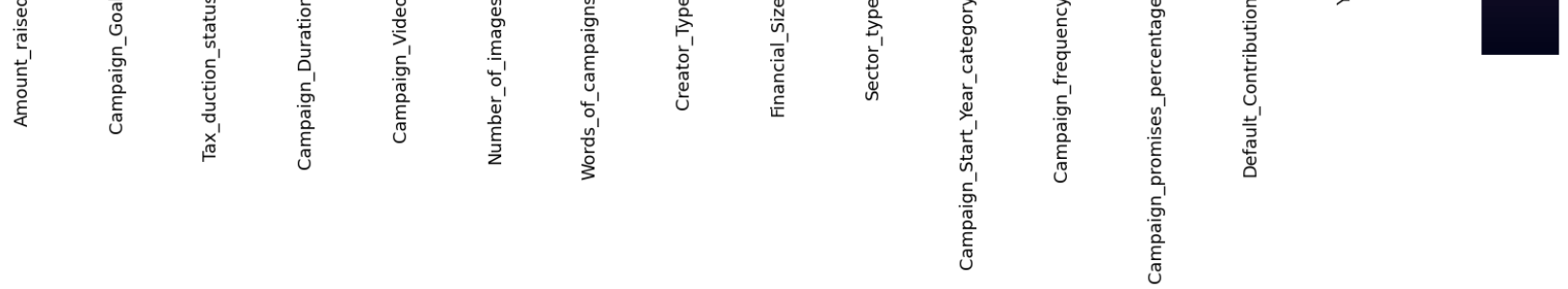
5. Correlation analysis

In [119…

```
# Numeric_features Store the following variables that need to draw correlations
numeric_features1 = ['Amount_raised', 'Campaign_Goal', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                     "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category", "Campaign_frequency",
                     "Campaign_promises_percentage", "Default_Contribution"]

# Correlation analysis
price_numeric = extract_data[numeric_features1]
```

```
correlation = extract_data.corr()
y_train = extract_data['Amount_raised']
corr = plt.subplots(figsize = (18,16), dpi=128)
corr= sns.heatmap(price_numeric.assign(Y=y_train).corr(method='spearman'), annot=True,square=True)
```


Amount_raised 1 0.57 0.02 0.25 0.052 0.069 0.23 -0.17 0.021 0.021 -0.058 -0.2 0.13 0.25 1
Campaign_Goal 0.57 1 -0.05 0.41 0.04 0.069 0.25 -0.47 -0.055 0.1 -0.16 -0.36 0.28 0.55 0.57
Tax_duction_status 0.02 -0.05 1 -0.013 0.0029 -0.025 -0.033 0.1 0.28 -0.04 -0.061 0.11 -0.094 -0.12 0.02
Campaign_Duration 0.25 0.41 -0.013 1 0.032 0.07 0.056 -0.25 0.0066 0.042 -0.17 -0.18 0.096 0.24 0.25
Campaign_Video 0.052 0.04 0.0029 0.032 1 0.039 0.009 0.081 -0.0025 0.005 -0.029 -0.088 -0.0064 -0.055 0.052
Number_of_images 0.069 0.069 -0.025 0.07 0.039 1 0.12 -0.023 -0.11 -0.016 -0.12 -0.14 0.081 0.038 0.069
Words_of_campaigns 0.23 0.25 -0.033 0.056 0.009 0.12 1 -0.24 -0.044 0.11 -0.043 -0.15 0.15 0.25 0.23
Creator_Type -0.17 -0.47 0.1 -0.25 0.081 -0.023 -0.24 1 0.15 -0.13 0.16 0.29 -0.52 -0.86 -0.17
Financial_Size 0.021 -0.055 0.28 0.0066 -0.0025 -0.11 -0.044 0.15 1 -0.069 0.012 0.2 -0.097 -0.14 0.021
Sector_type 0.021 0.1 -0.04 0.042 0.005 -0.016 0.11 -0.13 -0.069 1 -0.035 -0.12 0.00093 0.12 0.021
Campaign_Start_Year_category -0.058 -0.16 -0.061 -0.17 -0.029 -0.12 -0.043 0.16 0.012 -0.035 1 0.31 -0.076 -0.12 -0.058
Campaign_frequency -0.2 -0.36 0.11 -0.18 -0.088 -0.14 -0.15 0.29 0.2 -0.12 0.31 1 -0.18 -0.31 -0.2
Campaign_promises_percentage 0.13 0.28 -0.094 0.096 -0.0064 0.081 0.15 -0.52 -0.097 0.00093 -0.076 -0.18 1 0.53 0.13
Default_Contribution 0.25 0.55 -0.12 0.24 -0.055 0.038 0.25 -0.86 -0.14 0.12 -0.12 -0.31 0.53 1 0.25
Y 1 0.57 0.02 0.25 0.052 0.069 0.23 -0.17 0.021 0.021 -0.058 -0.2 0.13 0.25 1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75



In [119…

```
# pip install heatmapz

from heatmap import heatmap, corrplot
corrplot(price_numeric.assign(Y=y_train).corr(method='spearman').corr(), size_scale=200)
```

Do a scatter plot, using IV with a DV

Scatter plot 1 Amount_raised + Campaign Promise

Scatter plot 2 Amount_raised + Campaign Promise

Scatter plot 3 Number of Distinct_Donors + Campaign Promise

Scatter plot 4 Number of Distinct_Donors + Campaign Promise

In [119…
```
'''
plt.figure(figsize=(15,15))

plt.subplot(221)
plt.scatter(extract_data["Amount_raised"], extract_data["Campaign Promise"])
plt.xlabel('Sepal Length'); plt.ylabel('Sepal Width')

plt.subplot(222)
plt.scatter(extract_data["Distinct_Donors"], extract_data["Campaign Promise"])
plt.xlabel('Sepal Length'); plt.ylabel('Sepal Width')

plt.subplot(223)
plt.scatter(extract_data["Amount_raised"], extract_data["Campaign Promise"])
plt.xlabel('Sepal Length'); plt.ylabel('Sepal Width')

plt.subplot(224)
plt.scatter(extract_data["Amount_raised"], extract_data["Campaign Promise"])
plt.xlabel('Sepal Length'); plt.ylabel('Sepal Width')

plt.show()
'''
```

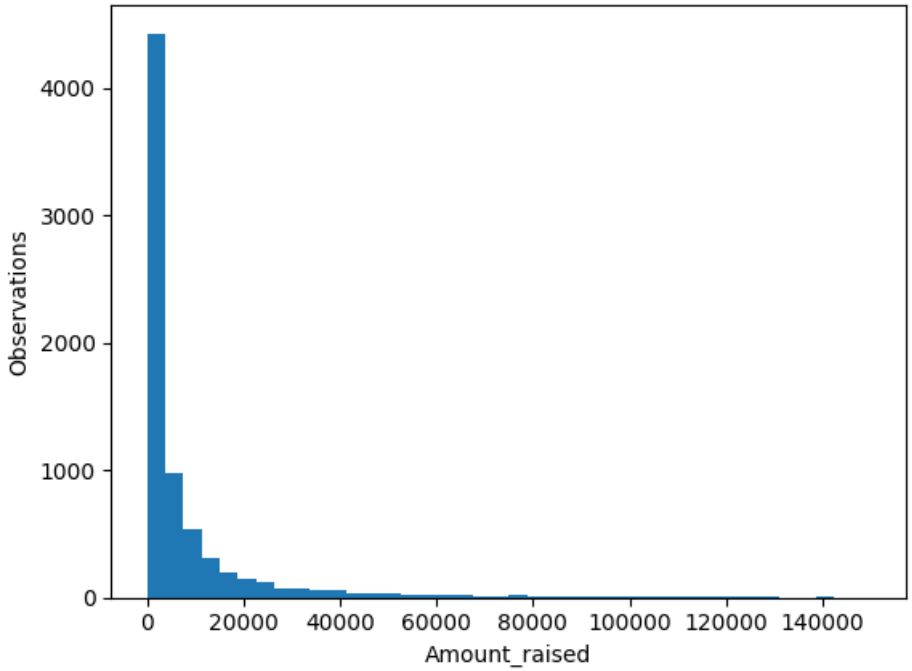
Out[1195]:
```
'\nplt.figure(figsize=(15,15))\n\nplt.subplot(221)\nplt.scatter(extract_data["Amount_raised"], extract_data["Campaign Promise"])\nplt.xlabel(\'Sepal Length\'); p
lt.ylabel(\'Sepal Width\')\n\nplt.subplot(222)\nplt.scatter(extract_data["Distinct_Donors"], extract_data["Campaign Promise"])\nplt.xlabel(\'Sepal Length\'); pl
t.ylabel(\'Sepal Width\')\n\nplt.subplot(223)\nplt.scatter(extract_data["Amount_raised"], extract_data["Campaign Promise"])\nplt.xlabel(\'Sepal Length\'); plt.yl
abel(\'Sepal Width\')\n\nplt.subplot(224)\nplt.scatter(extract_data["Amount_raised"], extract_data["Campaign Promise"])\nplt.xlabel(\'Sepal Length\'); plt.ylabel
(\'Sepal Width\')\n\nplt.show()\n'
```

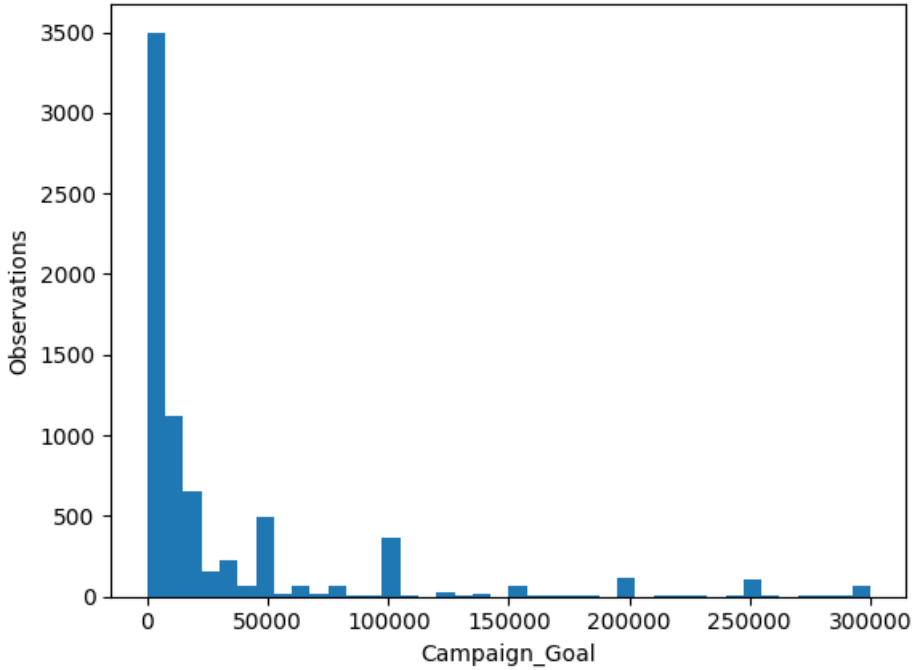
Historgram

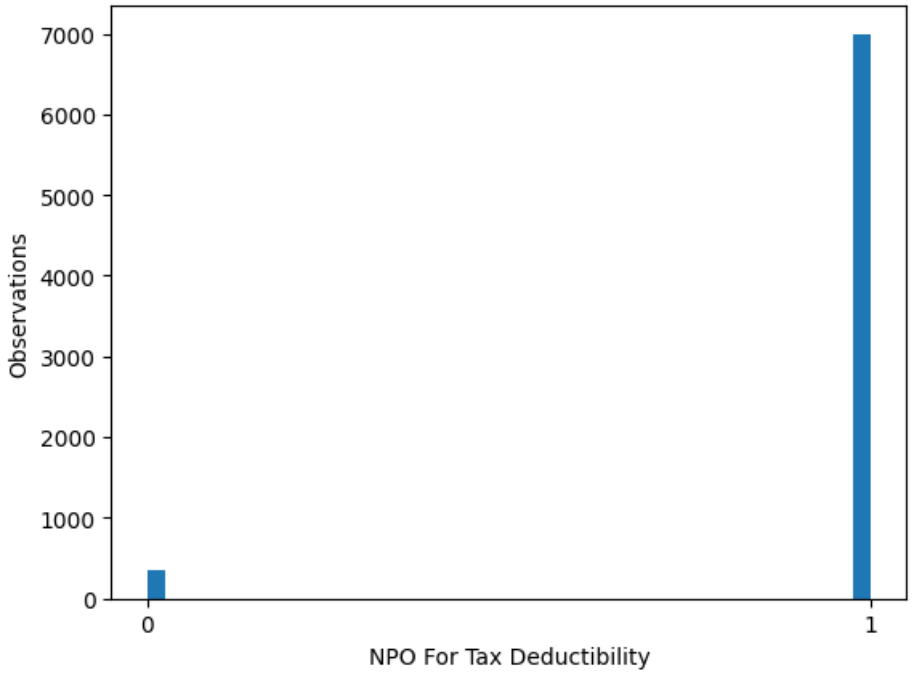
In [119…
```
#plt.figure(figsize=(20,20))
#plt.subplot(441)

'''
for i in numeric_features1:
    plt.hist(extract_data[i], bins=40)
    plt.xlabel(str(i)); plt.ylabel('Observations')
    plt.show()
'''
plt.hist(extract_data['Amount_raised'], bins=40, range=(0,150000))
plt.xlabel('Amount_raised'); plt.ylabel('Observations')
plt.show()

#plt.subplot(442)
plt.hist(extract_data['Campaign_Goal'], bins=40, range=(0,300000))
plt.xlabel('Campaign_Goal'); plt.ylabel('Observations')
plt.show()
```

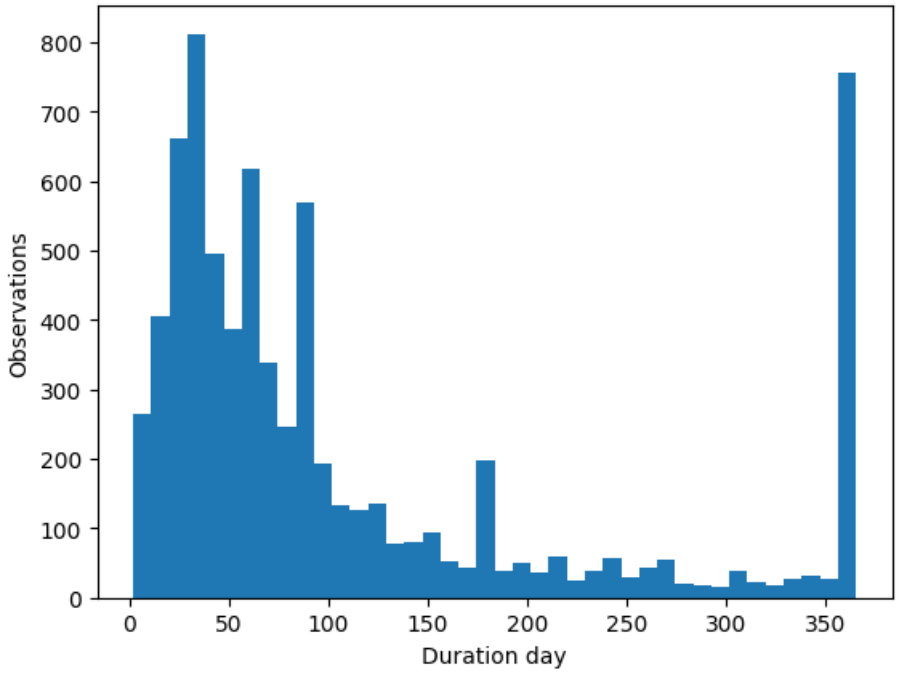
```
#plt.subplot(443)
plt.hist(extract_data['Tax_duction_status'], bins=40)
plt.xlabel('NPO For Tax Deductibility'); plt.ylabel('Observations')
x_major_locator=MultipleLocator(1)
ax = plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.show()
#plt.subplot(444)
plt.hist(extract_data['Campaign_Duration'], bins=40)
plt.xlabel('Duration day'); plt.ylabel('Observations')
plt.show()
#plt.subplot(445)
plt.hist(extract_data['Campaign_Video'], bins=40)
plt.xlabel('Campaign_Video'); plt.ylabel('Observations')
x_major_locator=MultipleLocator(1)
ax = plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.show()

#plt.subplot(446)
plt.hist(extract_data['Number_of_images'], bins=40)
plt.xlabel('Number_of_images'); plt.ylabel('Observations')
plt.show()

#plt.subplot(447)
plt.hist(extract_data['Words_of_campaigns'], bins=40)
plt.xlabel('Words_of_campaigns'); plt.ylabel('Observations')
plt.show()


#plt.subplot(4,4,10)
plt.hist(extract_data['Creator_Type'], bins=40)
plt.xlabel('Creator_Type'); plt.ylabel('Observations')
plt.show()

#plt.subplot(4,4,10)
plt.hist(extract_data['Financial_Size'], bins=40)
plt.xlabel('Financial_Size'); plt.ylabel('Observations')
plt.show()


#plt.subplot(4,4,10)
plt.hist(extract_data['Sector_type'], bins=40)
plt.xlabel('Sector_type'); plt.ylabel('Observations')
plt.show()

#plt.subplot(4,4,11)
plt.hist(extract_data['Campaign_Start_Year_category'], bins=40)
plt.xlabel('Campaign_Start_Year_category'); plt.ylabel('Observations')
plt.show()

#plt.subplot(4,4,12)
plt.hist(extract_data['Campaign_frequency'], bins=40)
plt.xlabel('Campaign_frequency'); plt.ylabel('Observations')
plt.show()

#plt.subplot(4,4,13)
plt.hist(extract_data['Campaign_promises_percentage'], bins=40)
```

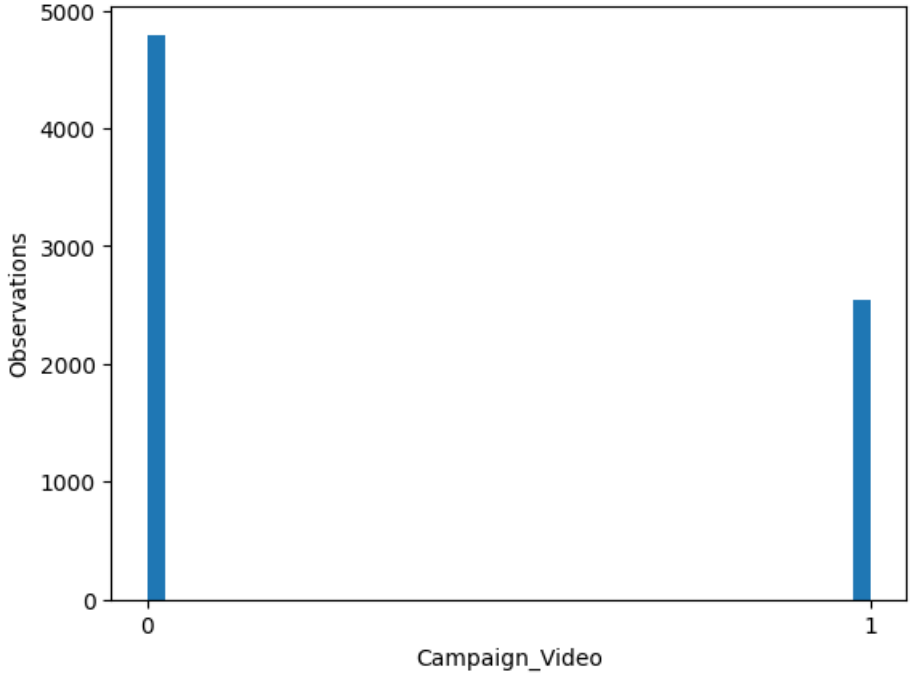
```
plt.xlabel('Future tense percentage'); plt.ylabel('Observations')
plt.show()

#plt.subplot(4,4,14)
plt.hist(extract_data['Default_Contribution'], bins=40, range=(0,1000))
plt.xlabel('Avg custom amount'); plt.ylabel('Observations')
plt.show()
```

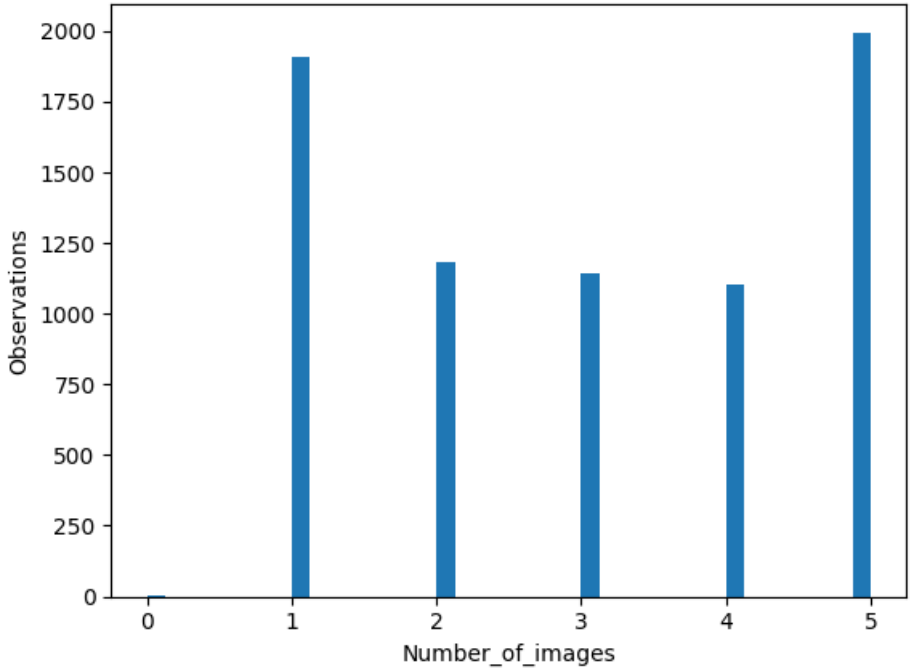


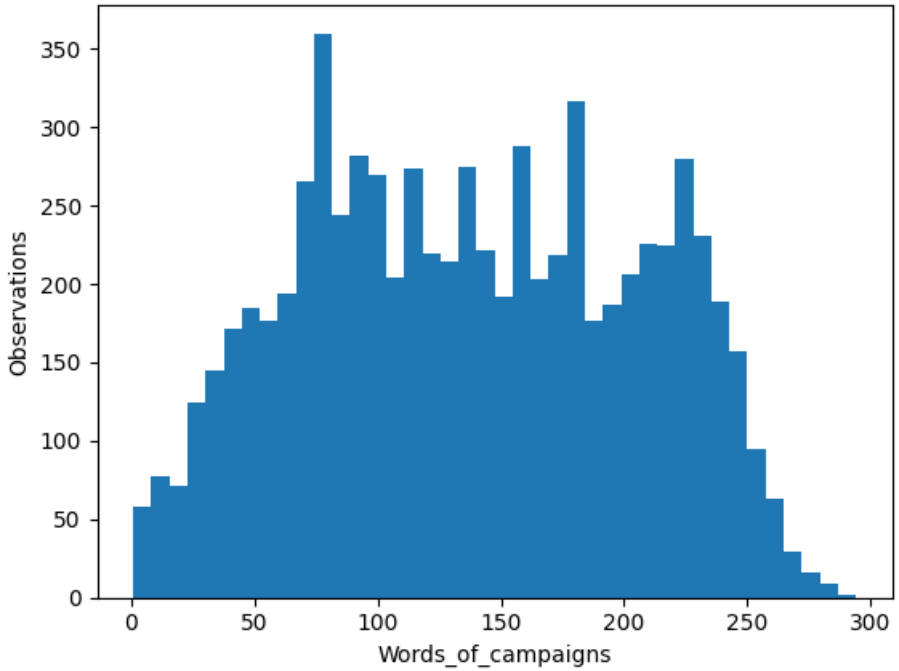

3500
3000
2500
2000
1500
1000
500
0
Observations
0
50000
100000
150000
200000
250000
300000
Campaign_Goal

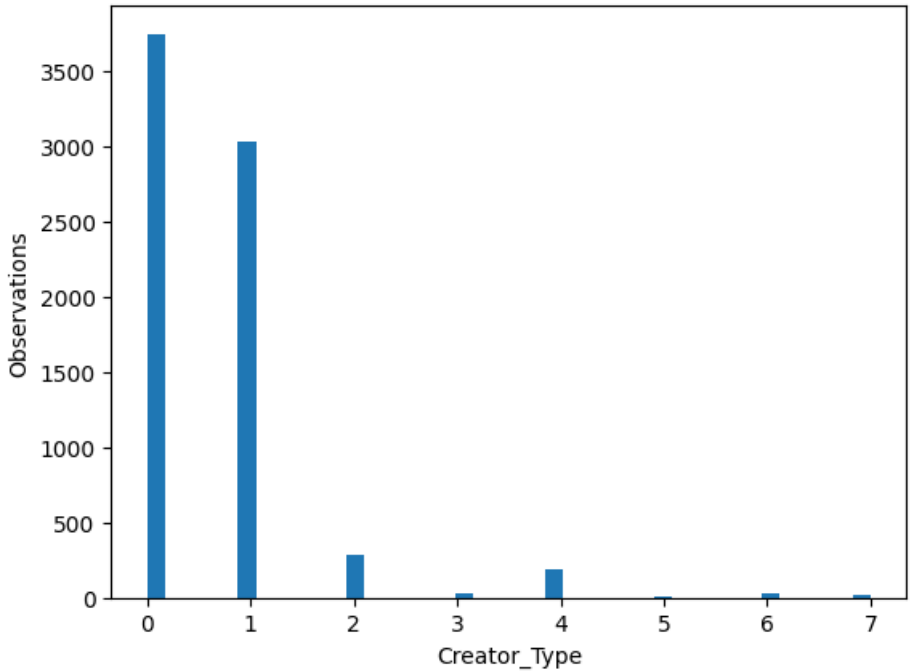

7000
6000
5000
4000
3000
2000
1000
0
Observations
0
1
NPO For Tax Deductibility

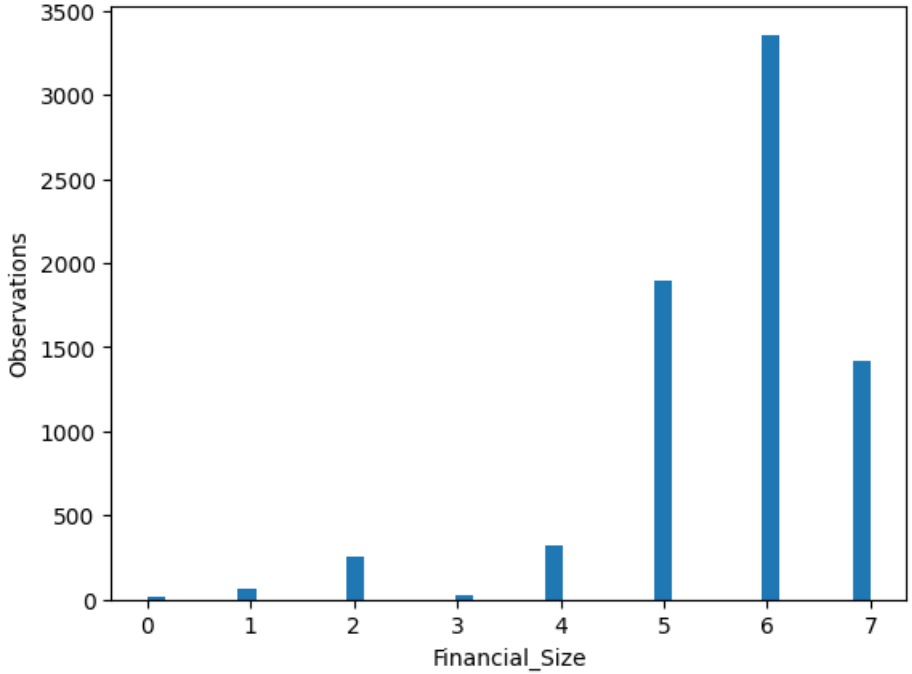

Observations
800
700
600
500
400
300
200
100
0
0
50
100
150
200
250
300
350
Duration day

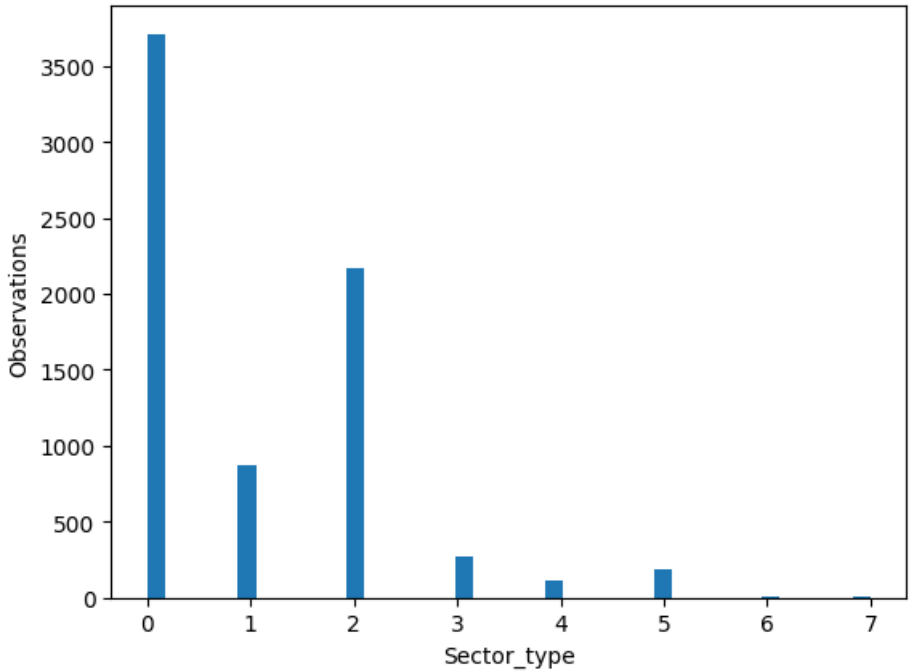

Observations
5000
4000
3000
2000
1000
0
0
1
Campaign_Video

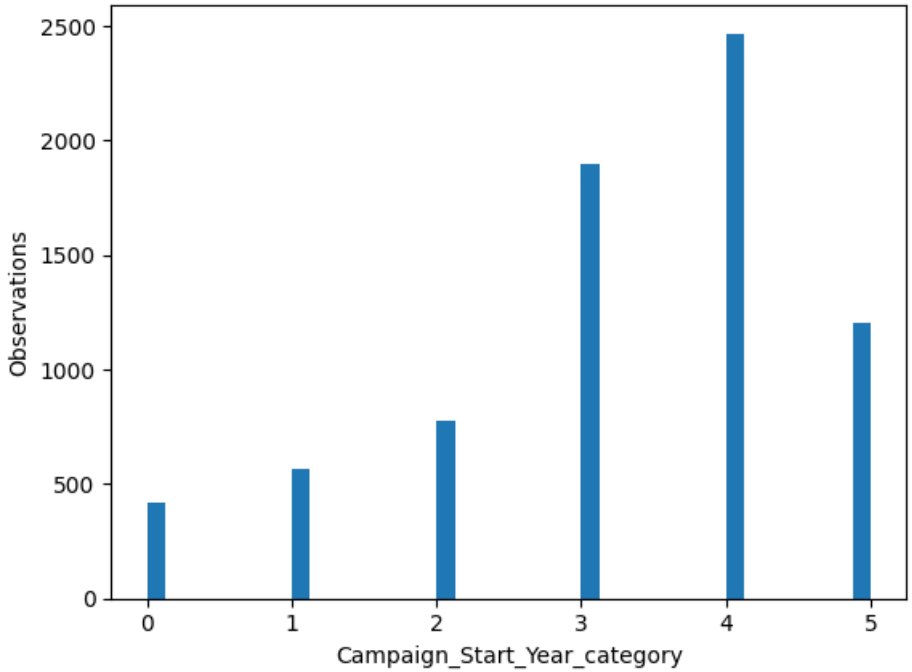

Observations
2000
1750
1500
1250
1000
750
500
250
0
0
1
2
3
4
5
Number_of_images

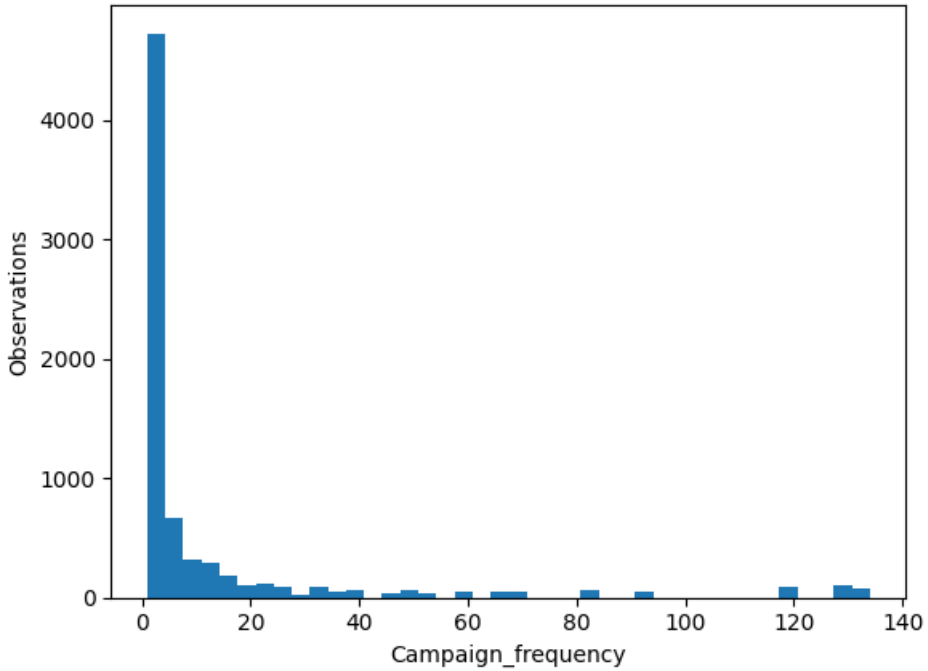

Observations
350
300
250
200
150
100
50
0
0
50
100
150
200
250
300
Words_of_campaigns

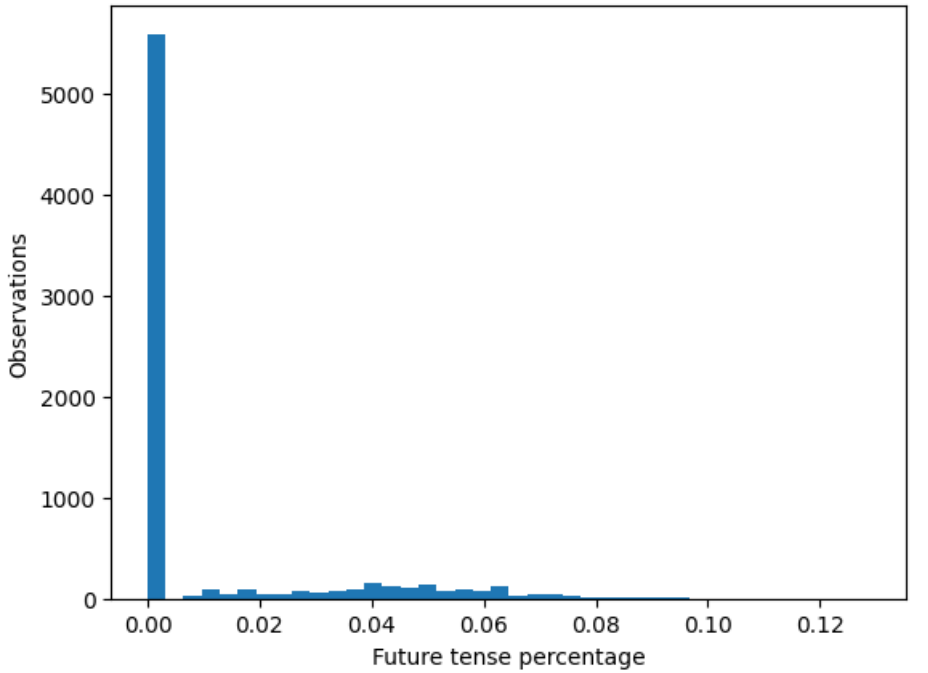

Observations
3500
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
7
Creator_Type

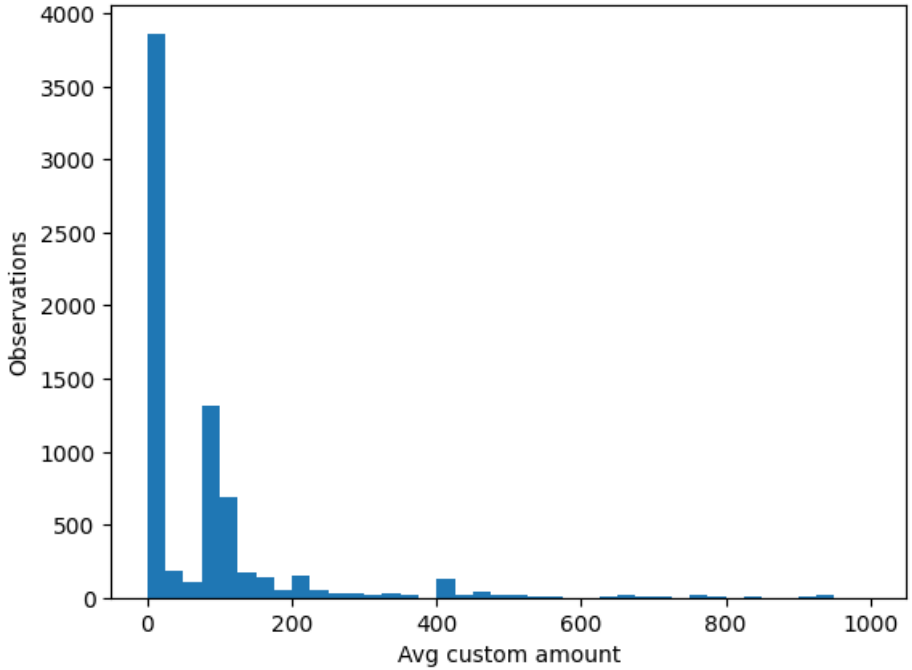

Observations
3500
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
7
Financial_Size


Observations
3500
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
7
Sector_type


Observations
2500
2000
1500
1000
500
0
0
1
2
3
4
5
Campaign_Start_Year_category

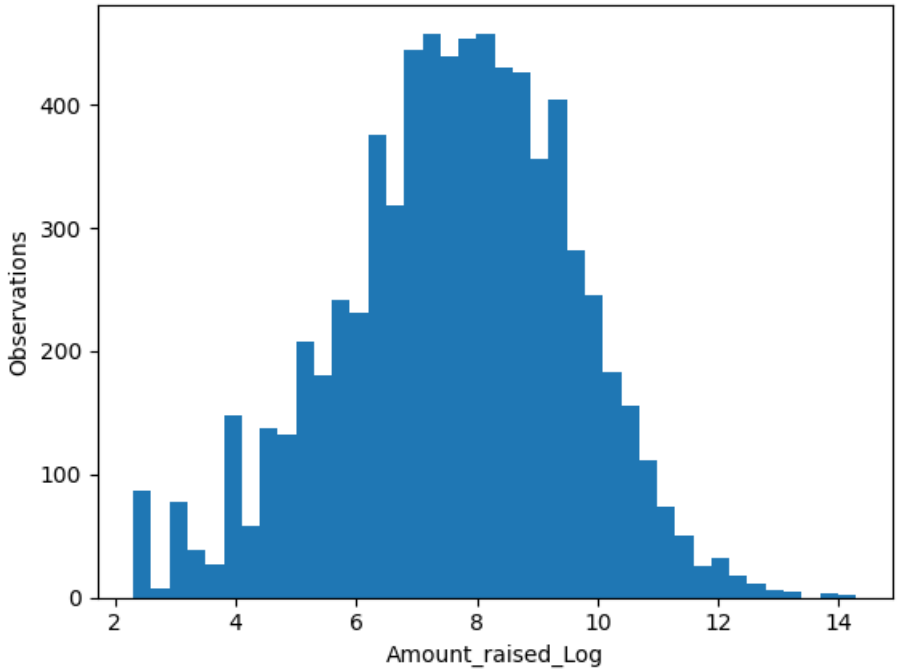

Observations
4000
3000
2000
1000
0
0
20
40
60
80
100
120
140
Campaign_frequency

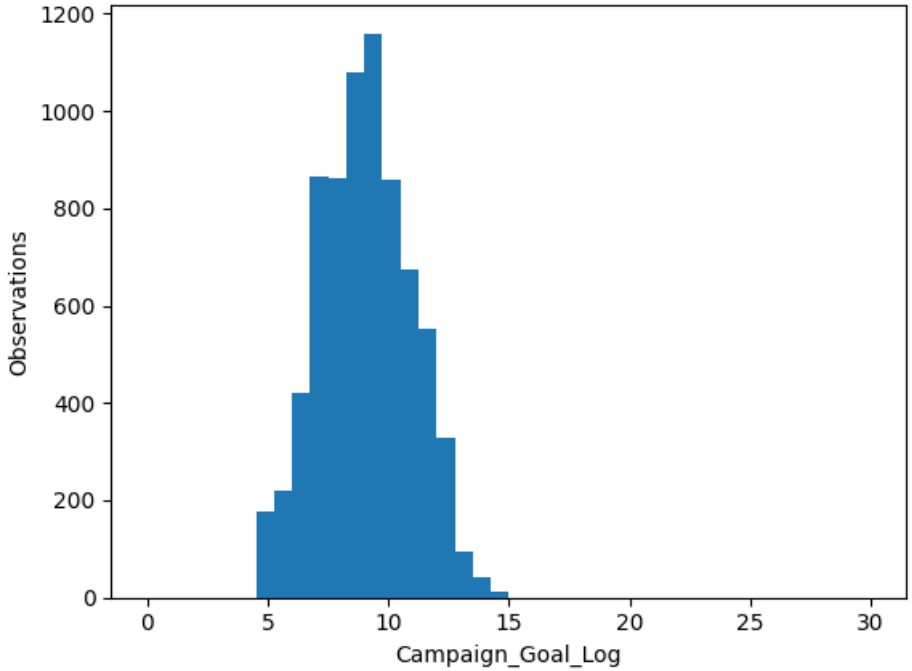

Observations
5000
4000
3000
2000
1000
0
0.00
0.02
0.04
0.06
0.08
0.10
0.12
Future tense percentage

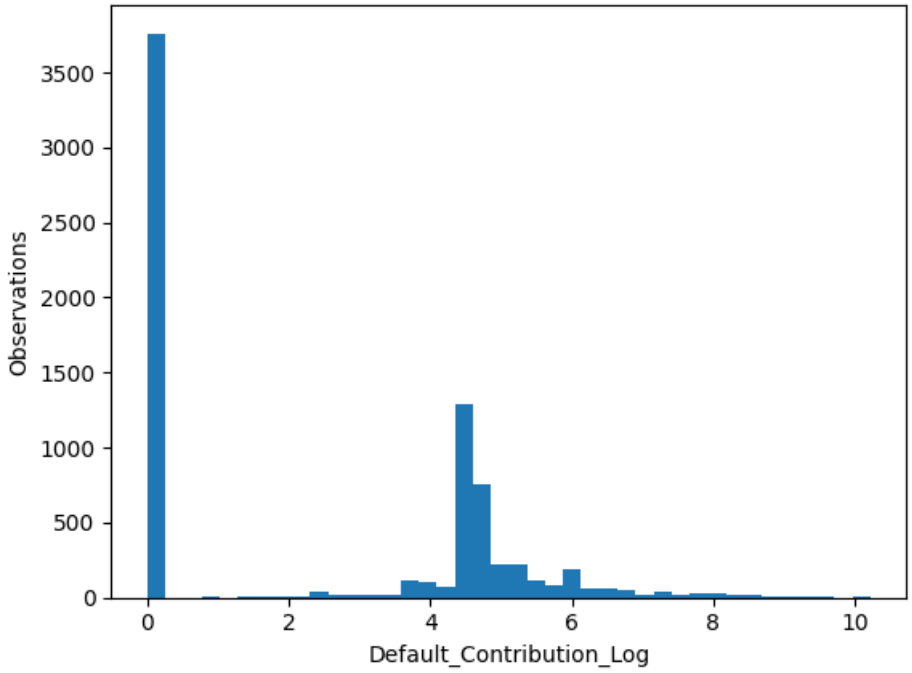


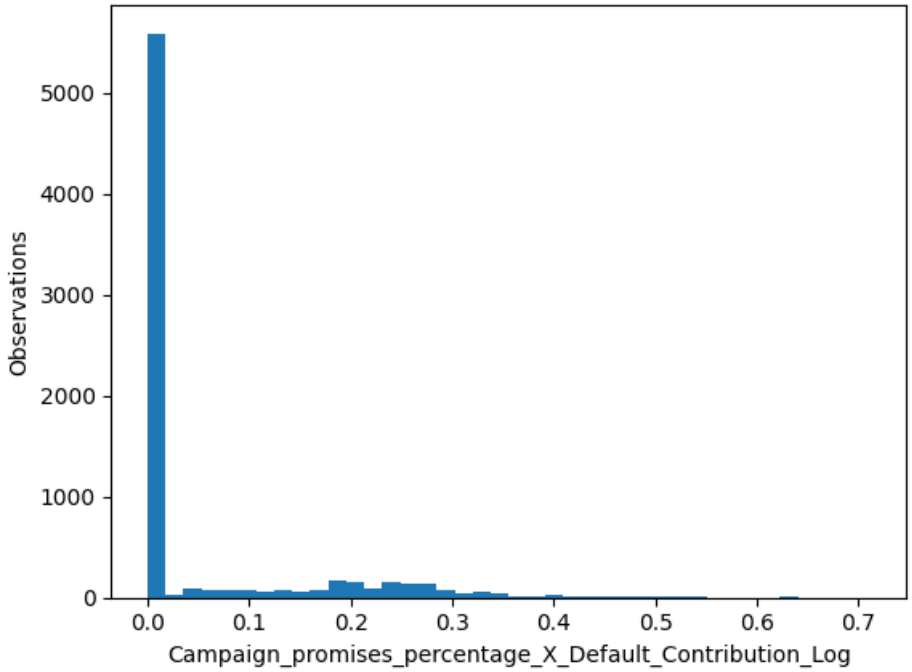
In [119…

```
#plt.figure(figsize=(20,20))

#plt.subplot(441)
plt.hist(extract_data['Amount_raised_Log'], bins=40)
plt.xlabel('Amount_raised_Log'); plt.ylabel('Observations')
plt.show()

#plt.subplot(442)
plt.hist(extract_data['Campaign_Goal_Log'], bins=40, range=(0,30))
plt.xlabel('Campaign_Goal_Log'); plt.ylabel('Observations')
plt.show()

#plt.subplot(445)
plt.hist(extract_data['Default_Contribution_Log'], bins=40)
plt.xlabel('Default_Contribution_Log'); plt.ylabel('Observations')
plt.show()


#plt.subplot(449)
plt.hist(extract_data['Campaign_promises_percentage_X_Default_Contribution_Log'], bins=40)
plt.xlabel('Campaign_promises_percentage_X_Default_Contribution_Log'); plt.ylabel('Observations')
plt.show()
```


Observations
0
100
200
300
400
2
4
6
8
10
12
14
Amount_raised_Log


Observations
0
200
400
600
800
1000
1200
0
5
10
15
20
25
30
Campaign_Goal_Log





Variance inflation factor (Two methods to test make sure they are right)

In [119…
```
'''
Clean_variables = ['Amount_raised','Campaign_Goal', 'Description_Campaign_polarity','Description_Campaign_subjectivity' ]
for item in Clean_variables:
     iqr = extract_data[item].quantile(0.9) - extract_data[item].quantile(0.1)
     q_abnormal_L = extract_data[item] < extract_data[item].quantile(0.1) - 1.5  iqr
     q_abnormal_U = extract_data[item] > extract_data[item].quantile(0.9) + 1.5  iqr
     extract_data = extract_data.drop(extract_data[extract_data[item].quantile(0.1) - 1.5  iqr> extract_data[item]].index)
     extract_data = extract_data.drop(extract_data[extract_data[item].quantile(0.9) + 1.5  iqr< extract_data[item]].index)
     print(item + '中有' + str(q_abnormal_L.sum() + q_abnormal_U.sum()) + '个异常值')
'''
```

Out[1198]:
```
"\nClean_variables = ['Amount_raised','Campaign_Goal', 'Description_Campaign_polarity','Description_Campaign_subjectivity' ]\nfor item in Clean_variables:\n
iqr = extract_data[item].quantile(0.9) - extract_data[item].quantile(0.1)\n     q_abnormal_L = extract_data[item] < extract_data[item].quantile(0.1) - 1.5  iqr\n
q_abnormal_U = extract_data[item] > extract_data[item].quantile(0.9) + 1.5  iqr\n     extract_data = extract_data.drop(extract_data[extract_data[item].quantile
(0.1) - 1.5  iqr> extract_data[item]].index)\n     extract_data = extract_data.drop(extract_data[extract_data[item].quantile(0.9) + 1.5  iqr< extract_data[ite
m]].index)\n     print(item + '中有' + str(q_abnormal_L.sum() + q_abnormal_U.sum()) + '个异常值')\n"
```

Modeling verfication

Variance, Average, Max, Min, Median calculation

In [119…
```
'''
i = 0
plt.figure(figsize=(13, 14))
plt.xticks([])
for title in numeric_features1:
    plt.subplot(4,3,i+1)
    plt.title(title)
    sns.kdeplot(extract_data[title], shade=True)
    plt.xlabel(" ")
    i += 1
'''
#plt.hist(extract_data['Campaign_Goal'], bins=80, histtype="stepfilled", alpha=.8)
```

Out[1199]:
```
'\ni = 0\nplt.figure(figsize=(13, 14))\nplt.xticks([])\nfor title in numeric_features1:\n    plt.subplot(4,3,i+1)\n    plt.title(title)\n    sns.kdeplot(extract_
data[title], shade=True) \n    plt.xlabel(" ") \n    i += 1\n'
```

In [120…
```
s = extract_data['Amount_raised']
print(s.skew())
print(s.kurt())
```

```
18.76533114511192
515.0851788073364
```

To see how log Actual funds raised changes for different levels of future tense percentage and log avg custom amount

In [120…
```
extract_data['Campaign_promises_percentage_Log_level'] = 0
for index, row in extract_data.iterrows():
    if extract_data.loc[index,'Campaign_promises_percentage'] < 0.08043 and extract_data.loc[index,'Campaign_promises_percentage'] > 0:
        extract_data.loc[index,'Campaign_promises_percentage_Log_level'] = 0.04
    if extract_data.loc[index,'Campaign_promises_percentage'] >= 0.08043:
        extract_data.loc[index,'Campaign_promises_percentage_Log_level'] = 0.1
```

In [120…
```
extract_data['Campaign_promises_percentage_Log'] = np.log(extract_data['Campaign_promises_percentage'] + 1)
extract_data['Default_Contribution_Log'] = np.log(extract_data['Default_Contribution'] + 1)
extract_data['Campaign_promises_percentage_Log_X_Default_Contribution_Log'] = extract_data['Campaign_promises_percentage_Log'] * extract_data['Default_Contribution
```

In [120…
```
(extract_data['Campaign_promises_percentage_Log'].value_counts())
```

Out[1203]:

0.000000	5594
0.048790	76
0.059898	55
0.038221	47
0.054067	43
0.068993	34
0.042560	25
0.040822	23
0.028573	20
0.044060	19
0.035718	18
0.059719	18
0.060625	17
0.018019	17
0.058269	16
0.057820	16
0.047628	16
0.045462	16
0.041243	15
0.044452	15
0.057158	15
0.035091	15
0.033902	15
0.046520	14
0.040410	13
0.061558	13
0.039221	13
0.025975	13
0.054808	12
0.043803	12
0.016807	12
0.031749	12
0.065597	12
0.046957	11
0.017094	11
0.065958	11
0.009756	11
0.036368	11
0.064539	10
0.073025	10
0.044951	10
0.044851	9
0.032261	9
0.061369	9
0.053489	9
0.029853	9
0.053110	9
0.040005	9
0.066691	9
0.038466	9
0.010152	9
0.021506	9
0.053346	9
0.062914	8
0.016261	8
0.058496	8
0.050010	8

0.051293	8
0.030772	8
0.051825	8
0.042925	8
0.024098	8
0.041158	8
0.068053	7
0.027151	7
0.022473	7
0.074108	7
0.061694	7
0.051960	7
0.040491	7
0.055570	7
0.076961	7
0.036701	7
0.022990	7
0.013245	6
0.046091	6
0.063716	6
0.052644	6
0.052368	6
0.017858	6
0.036105	6
0.028171	6
0.041673	6
0.039740	6
0.027399	6
0.014599	6
0.058841	6
0.026668	6
0.028710	6
0.018349	6
0.010582	6
0.065383	5
0.047402	5
0.013793	5
0.011173	5
0.016000	5
0.078472	5
0.090972	5
0.011429	5
0.012739	5
0.057987	5
0.012903	5
0.039531	5
0.020619	5
0.056512	5
0.015504	5
0.062132	5
0.043963	5
0.045670	5
0.049393	5
0.054658	5
0.024693	5
0.035846	5
0.066249	5
0.020203	5

0.033673	5
0.010929	5
0.034486	5
0.038915	4
0.051736	4
0.071973	4
0.048202	4
0.038840	4
0.050262	4
0.015038	4
0.050772	4
0.032576	4
0.031499	4
0.043485	4
0.037458	4
0.018692	4
0.024898	4
0.048319	4
0.014815	4
0.032790	4
0.046884	4
0.037041	4
0.043017	4
0.087011	4
0.045985	4
0.037388	4
0.056353	4
0.042200	4
0.041964	4
0.037740	4
0.059423	4
0.028988	4
0.030397	4
0.030305	4
0.049271	4
0.045810	4
0.011300	4
0.011696	4
0.017700	4
0.039846	3
0.036040	3
0.029559	3
0.049597	3
0.012423	3
0.077558	3
0.012270	3
0.022223	3
0.049762	3
0.042111	3
0.033336	3
0.067823	3
0.070204	3
0.020001	3
0.008889	3
0.007547	3
0.032365	3
0.016529	3
0.006473	3

0.011050	3
0.047068	3
0.043297	3
0.071459	3
0.055263	3
0.010363	3
0.052922	3
0.025752	3
0.010471	3
0.050431	3
0.072759	3
0.021053	2
0.035339	2
0.027029	2
0.070068	2
0.045257	2
0.033617	2
0.041847	2
0.019048	2
0.026907	2
0.048397	2
0.019803	2
0.015152	2
0.009569	2
0.007117	2
0.032003	2
0.108214	2
0.082692	2
0.067139	2
0.024391	2
0.033006	2
0.008734	2
0.040166	2
0.072496	2
0.008969	2
0.009302	2
0.014389	2
0.073203	2
0.007843	2
0.056240	2
0.060018	2
0.035591	2
0.052186	2
0.088553	2
0.033226	2
0.047253	2
0.011976	2
0.034289	2
0.050644	2
0.067441	2
0.075508	2
0.021979	2
0.054559	2
0.070952	2
0.076373	2
0.031253	2
0.057570	2
0.017392	2

0.037179	2
0.035507	2
0.080689	2
0.010257	2
0.055880	2
0.008032	2
0.013423	2
0.093090	2
0.034606	2
0.026317	2
0.061875	2
0.025533	2
0.013606	2
0.021819	2
0.036905	2
0.063179	2
0.025318	2
0.059189	2
0.063513	2
0.010695	2
0.031416	1
0.077292	1
0.083382	1
0.016173	1
0.007905	1
0.009390	1
0.042864	1
0.073563	1
0.026202	1
0.006390	1
0.056753	1
0.012121	1
0.030077	1
0.032088	1
0.009950	1
0.027652	1
0.049480	1
0.105361	1
0.034133	1
0.034368	1
0.084083	1
0.079407	1
0.088947	1
0.025001	1
0.078988	1
0.036634	1
0.007605	1
0.040546	1
0.085360	1
0.050858	1
0.061036	1
0.045120	1
0.097638	1
0.008230	1
0.030153	1
0.013072	1
0.007380	1
0.014185	1

0.008439	1
0.015873	1
0.022642	1
0.024244	1
0.070452	1
0.026433	1
0.032157	1
0.092373	1
0.027780	1
0.081346	1
0.069959	1
0.018519	1
0.009662	1
0.031548	1
0.023530	1
0.031952	1
0.010811	1
0.007491	1
0.012821	1
0.017544	1
0.055060	1
0.111226	1
0.070618	1
0.041500	1
0.054488	1
0.009479	1
0.031010	1
0.029414	1
0.027909	1
0.085522	1
0.018868	1
0.034847	1
0.056695	1
0.019418	1
0.023811	1
0.085158	1
0.121361	1
0.018928	1
0.009852	1
0.043675	1
0.009050	1
0.010050	1
0.023347	1
0.007663	1
0.072103	1
0.115069	1
0.023257	1
0.053584	1
0.015385	1
0.031351	1
0.012346	1
0.018238	1
0.031155	1
0.020762	1
0.022306	1
0.029199	1
0.031091	1
0.006557	1

```
0.038319       1
0.035932       1
0.045910       1
0.034191       1
0.032435       1
0.006969       1
0.080043       1
0.033152       1
0.015748       1
0.022728       1
0.040274       1
0.011834       1
0.012579       1
0.021740       1
0.020479       1
0.029632       1
0.061244       1
0.052798       1
Name: Campaign_promises_percentage_Log, dtype: int64
```

In [120…

```
numeric_features2 = ['Amount_raised_Log', 'Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    "Campaign_frequency", "Campaign_promises_percentage_Log", "Default_Contribution_Log",
                    'Campaign_promises_percentage_Log_X_Default_Contribution_Log']
```

In [120…

```
for title in numeric_features2:
    extract_data[title] = pd.to_numeric( extract_data[title])
    print( title, "Median:",np.mean(extract_data[title]))
    print( title, "Std:",np.std(extract_data[title]))

    print( title, "skew:" ,extract_data[title].skew())
    print( title, "kurt:" ,extract_data[title].kurt())

    print( title, "Min:" ,np.min(extract_data[title]))
    print( title, "Max:" ,np.max(extract_data[title]))
```

Amount_raised_Log Median: 7.629439344937267
Amount_raised_Log Std: 1.9559319987280748
Amount_raised_Log skew: -0.2695885136210425
Amount_raised_Log kurt: 0.026774017772850822
Amount_raised_Log Min: 2.302585092994046
Amount_raised_Log Max: 14.287354991888984
Campaign_Goal_Log Median: 8.96277462340678
Campaign_Goal_Log Std: 1.943153656723777
Campaign_Goal_Log skew: 0.042169814867144516
Campaign_Goal_Log kurt: -0.4164323123458229
Campaign_Goal_Log Min: 4.605170185988092
Campaign_Goal_Log Max: 14.508657738524219
Tax_duction_status Median: 0.953133514986376
Tax_duction_status Std: 0.21135282727252958
Tax_duction_status skew: -4.288810774282283
Tax_duction_status kurt: 16.398366003129844
Tax_duction_status Min: 0
Tax_duction_status Max: 1
Campaign_Duration Median: 111.71362397820164
Campaign_Duration Std: 109.63738724026686
Campaign_Duration skew: 1.3605804712058975
Campaign_Duration kurt: 0.5329380034615174
Campaign_Duration Min: 2
Campaign_Duration Max: 365
Campaign_Video Median: 0.3463215258855586
Campaign_Video Std: 0.4757971485768457
Campaign_Video skew: 0.64611521300614
Campaign_Video kurt: -1.5829665318237465
Campaign_Video Min: 0
Campaign_Video Max: 1
Number_of_images Median: 3.011716621253406
Number_of_images Std: 1.5633311901704328
Number_of_images skew: -0.004014605614086031
Number_of_images kurt: -1.516134923828246
Number_of_images Min: 0
Number_of_images Max: 5
Words_of_campaigns Median: 138.02615803814714
Words_of_campaigns Std: 67.14969549138237
Words_of_campaigns skew: 0.023689210896896053
Words_of_campaigns kurt: -1.0433239947584616
Words_of_campaigns Min: 1
Words_of_campaigns Max: 294
Creator_Type Median: 0.6534059945504087
Creator_Type Std: 0.9394665459859312
Creator_Type skew: 2.7832294561074993
Creator_Type kurt: 11.3156534761256
Creator_Type Min: 0
Creator_Type Max: 7
Financial_Size Median: 5.644822888283379
Financial_Size Std: 1.168212803352858
Financial_Size skew: -1.6871125833085105
Financial_Size kurt: 4.036922656822575
Financial_Size Min: 0
Financial_Size Max: 7
Sector_type Median: 1.022615803814714
Sector_type Std: 1.2477907330272626
Sector_type skew: 1.2260896843281928
Sector_type kurt: 1.5250394853019502

```
Sector_type Min: 0
Sector_type Max: 7
Campaign_Start_Year_category Median: 3.2333787465940054
Campaign_Start_Year_category Std: 1.3672513750188997
Campaign_Start_Year_category skew: -0.7585106471530464
Campaign_Start_Year_category kurt: -0.09772936906939256
Campaign_Start_Year_category Min: 0
Campaign_Start_Year_category Max: 5
Campaign_frequency Median: 12.983106267029973
Campaign_frequency Std: 26.97607655733872
Campaign_frequency skew: 3.1822751415246433
Campaign_frequency kurt: 9.790349782167404
Campaign_frequency Min: 1
Campaign_frequency Max: 134
Campaign_promises_percentage_Log Median: 0.010170470208412549
Campaign_promises_percentage_Log Std: 0.020312851254082692
Campaign_promises_percentage_Log skew: 1.8568889713000019
Campaign_promises_percentage_Log kurt: 2.2959955705009776
Campaign_promises_percentage_Log Min: 0.0
Campaign_promises_percentage_Log Max: 0.12136085700426734
Default_Contribution_Log Median: 2.3822113502186117
Default_Contribution_Log Std: 2.536514418145909
Default_Contribution_Log skew: 0.2910714899621106
Default_Contribution_Log kurt: -1.5107784510729427
Default_Contribution_Log Min: 0.0
Default_Contribution_Log Max: 10.221977646629885
Campaign_promises_percentage_Log_X_Default_Contribution_Log Median: 0.049681843460973686
Campaign_promises_percentage_Log_X_Default_Contribution_Log Std: 0.10131909414832245
Campaign_promises_percentage_Log_X_Default_Contribution_Log skew: 2.0346542216522137
Campaign_promises_percentage_Log_X_Default_Contribution_Log kurt: 3.5358872000786663
Campaign_promises_percentage_Log_X_Default_Contribution_Log Min: 0.0
Campaign_promises_percentage_Log_X_Default_Contribution_Log Max: 0.6734088615251426
```

In [120…

```
extract_data['Campaign_promises_percentage_Log'].unique()
```

Out[1206]:
```
array([0.01197619, 0.        , 0.0618754 , 0.08535985, 0.03704127,
       0.02298952, 0.09763847, 0.05635294, 0.02817088, 0.0529224 ,
       0.05236799, 0.05182507, 0.01015237, 0.05406722, 0.0104713 ,
       0.04301739, 0.04879016, 0.01307208, 0.08894749, 0.05971923,
       0.08701138, 0.06538276, 0.04762805, 0.0419642 , 0.05001042,
       0.04380262, 0.05195974, 0.05942342, 0.04959694, 0.06169357,
       0.05781957, 0.02739897, 0.03509132, 0.04396312, 0.02222314,
       0.02424361, 0.06453852, 0.02666825, 0.02197891, 0.02643326,
       0.00975617, 0.02777956, 0.01129956, 0.01834914, 0.03077166,
       0.06062462, 0.03953084, 0.05588046, 0.02871011, 0.02597549,
       0.0588405 , 0.0416727 , 0.03984591, 0.02898754, 0.01273903,
       0.03571808, 0.01418463, 0.06899287, 0.07755823, 0.0129034 ,
       0.05798726, 0.04652002, 0.05129329, 0.04598511, 0.05918887,
       0.02985296, 0.05849621, 0.04255961, 0.04445176, 0.07729167,
       0.04740224, 0.05715841, 0.05455898, 0.02715099, 0.01769958,
       0.0295588 , 0.01092907, 0.01617286, 0.07637298, 0.04688359,
       0.03226086, 0.01212136, 0.03007746, 0.00995033, 0.01036279,
       0.02531781, 0.01342302, 0.01324523, 0.03413301, 0.06136895,
       0.02631731, 0.03154836, 0.04220035, 0.08134564, 0.036105  ,
       0.03584613, 0.03109059, 0.0317487 , 0.03831886, 0.04292504,
       0.03593201, 0.0459097 , 0.037179  , 0.03390155, 0.03428907,
       0.07095174, 0.01169604, 0.0145988 , 0.01626052, 0.03690456,
       0.0165293 , 0.01574836, 0.01600034, 0.01680712, 0.01709443,
       0.04695698, 0.05826891, 0.0102565 , 0.04580954, 0.04082199,
       0.06155789, 0.04040954, 0.03670137, 0.05310983, 0.02409755,
       0.0402739 , 0.03822121, 0.01379332, 0.03636764, 0.01360565,
       0.02173999, 0.02000067, 0.03974033, 0.02047853, 0.0296318 ,
       0.0719735 , 0.02181905, 0.04546237, 0.04495139, 0.05348868,
       0.05077233, 0.05989814, 0.04405999, 0.05358425, 0.04348511,
       0.05651221, 0.06371581, 0.01869213, 0.06351341, 0.04329681,
       0.05064373, 0.05264373, 0.05526268, 0.03367322, 0.03922071,
       0.01081092, 0.03257617, 0.06291383, 0.04485057, 0.04567004,
       0.0257525 , 0.08068891, 0.02469261, 0.04831858, 0.05756985,
       0.01282069, 0.01754431, 0.02489755, 0.05505978, 0.06744128,
       0.07550755, 0.06559728, 0.05448819, 0.00947874, 0.03101024,
       0.01227009, 0.07696104, 0.02941389, 0.03125254, 0.01739174,
       0.08552217, 0.04211149, 0.01481509, 0.03550669, 0.05669534,
       0.01058211, 0.0482021 , 0.01941809, 0.07302514, 0.06805346,
       0.0678226 , 0.00803217, 0.07410797, 0.06213178, 0.07275935,
       0.09309042, 0.01242252, 0.05334598, 0.01069529, 0.02247286,
       0.08515781, 0.07020426, 0.05480824, 0.09097178, 0.02061929,
       0.12136086, 0.0631789 , 0.07145896, 0.05556985, 0.06624939,
       0.00904984, 0.01005034, 0.0255333 , 0.04706751, 0.05173567,
       0.03236528, 0.02334736, 0.03448618, 0.03460553, 0.0065574 ,
       0.00790518, 0.00647251, 0.03278982, 0.00766287, 0.0114287 ,
       0.02857337, 0.02105341, 0.06595797, 0.04367506, 0.01503788,
       0.0098523 , 0.01892801, 0.03333642, 0.04609111, 0.02702867,
       0.02150621, 0.02381065, 0.01980263, 0.03149867, 0.03891542,
       0.03484673, 0.01886848, 0.05218575, 0.06001801, 0.05623972,
       0.02790879, 0.00784318, 0.02690745, 0.05026183, 0.04976151,
       0.03883983, 0.04149973, 0.00930239, 0.00896867, 0.07061757,
       0.11122564, 0.03745756, 0.01550419, 0.01801851, 0.04016604,
       0.00873368, 0.00749067, 0.07210329, 0.04049136, 0.08269172,
       0.10821358, 0.11506933, 0.04124296, 0.04000533, 0.06124363,
       0.0111733 , 0.03846628, 0.03738753, 0.02020271, 0.01257878,
       0.01183446, 0.01515181, 0.00956945, 0.02272825, 0.00711747,
       0.04725288, 0.03315221, 0.08004271, 0.00696867, 0.04927105,
```

```
0.03243528, 0.03419136, 0.01104984, 0.03030535, 0.03200273,
0.03039748, 0.03774033, 0.0235305 , 0.01785762, 0.02919915,
0.02230576, 0.02076199, 0.03115517, 0.01823759, 0.01234584,
0.03135053, 0.01538492, 0.02325686, 0.0319516 , 0.0671393 ,
0.04939276, 0.04115807, 0.02439145, 0.10536052, 0.07847162,
0.04948006, 0.00754721, 0.02765153, 0.0330063 , 0.03208831,
0.05465841, 0.05043085, 0.05675282, 0.0063898 , 0.0724955 ,
0.07356257, 0.0428637 , 0.00938974, 0.00888895, 0.01438874,
0.06669137, 0.08338161, 0.0732034 , 0.0314162 , 0.03436764,
0.02620237, 0.07940678, 0.03559095, 0.00966191, 0.01851905,
0.03603994, 0.06995859, 0.08408312, 0.0885534 , 0.09237332,
0.03215711, 0.03322565, 0.07045166, 0.04839654, 0.07006756,
0.02264248, 0.01587335, 0.00843887, 0.00738011, 0.04525659,
0.03361661, 0.03015304, 0.0082305 , 0.03533937, 0.04512044,
0.06103589, 0.05085842, 0.04184711, 0.04054609, 0.0076046 ,
0.03663413, 0.07898841, 0.01904819, 0.0250013 , 0.05279819])
```

In [120…
```
np.mean(extract_data['Campaign_promises_percentage_Log'])
```

Out[1207]: 0.010170470208412549

In [120…
```
np.mean(extract_data['Campaign_promises_percentage_Log'].unique())
```

Out[1208]: 0.039563003250765096

In [120…
```
np.max(extract_data['Campaign_promises_percentage_Log'])
```

Out[1209]: 0.12136085700426734

In [121…
```
(0.03956+0.1213)/2
```

Out[1210]: 0.08043

In [121…
```
interaction_data1 = extract_data[['Campaign_promises_percentage_Log', 'Default_Contribution_Log', 'Amount_raised_Log']]
```

In [121…
```
extract_data.shape
```

Out[1212]: (7340, 115)

Interaction variables classification

In [121…
```
num = 0
for index, row in interaction_data1.iterrows():
    if interaction_data1.loc[index,'Campaign_promises_percentage_Log'] != 0:
        if interaction_data1.loc[index,'Campaign_promises_percentage_Log'] < 0.038 or interaction_data1.loc[index,'Campaign_promises_percentage_Log'] > 0.039:
            if interaction_data1.loc[index,'Campaign_promises_percentage_Log'] < 0.059:
                interaction_data1.drop(index, inplace=True)
```

In [121…
```
for index, row in interaction_data1.iterrows():
    if interaction_data1.loc[index,'Campaign_promises_percentage_Log'] >= 0.038 and interaction_data1.loc[index,'Campaign_promises_percentage_Log'] <= 0.039:
        interaction_data1.loc[index,'Campaign_promises_percentage_Log'] = 0.038
    if interaction_data1.loc[index,'Campaign_promises_percentage_Log'] >= 0.055 :
        interaction_data1.loc[index,'Campaign_promises_percentage_Log'] = 0.055
```

In [121…

```
extract_data['Campaign_promises_percentage_Log'].unique()
```

Out[1215]:
```
array([0.01197619, 0.        , 0.0618754 , 0.08535985, 0.03704127,
       0.02298952, 0.09763847, 0.05635294, 0.02817088, 0.0529224 ,
       0.05236799, 0.05182507, 0.01015237, 0.05406722, 0.0104713 ,
       0.04301739, 0.04879016, 0.01307208, 0.08894749, 0.05971923,
       0.08701138, 0.06538276, 0.04762805, 0.0419642 , 0.05001042,
       0.04380262, 0.05195974, 0.05942342, 0.04959694, 0.06169357,
       0.05781957, 0.02739897, 0.03509132, 0.04396312, 0.02222314,
       0.02424361, 0.06453852, 0.02666825, 0.02197891, 0.02643326,
       0.00975617, 0.02777956, 0.01129956, 0.01834914, 0.03077166,
       0.06062462, 0.03953084, 0.05588046, 0.02871011, 0.02597549,
       0.0588405 , 0.0416727 , 0.03984591, 0.02898754, 0.01273903,
       0.03571808, 0.01418463, 0.06899287, 0.07755823, 0.0129034 ,
       0.05798726, 0.04652002, 0.05129329, 0.04598511, 0.05918887,
       0.02985296, 0.05849621, 0.04255961, 0.04445176, 0.07729167,
       0.04740224, 0.05715841, 0.05455898, 0.02715099, 0.01769958,
       0.0295588 , 0.01092907, 0.01617286, 0.07637298, 0.04688359,
       0.03226086, 0.01212136, 0.03007746, 0.00995033, 0.01036279,
       0.02531781, 0.01342302, 0.01324523, 0.03413301, 0.06136895,
       0.02631731, 0.03154836, 0.04220035, 0.08134564, 0.036105  ,
       0.03584613, 0.03109059, 0.0317487 , 0.03831886, 0.04292504,
       0.03593201, 0.0459097 , 0.037179  , 0.03390155, 0.03428907,
       0.07095174, 0.01169604, 0.0145988 , 0.01626052, 0.03690456,
       0.0165293 , 0.01574836, 0.01600034, 0.01680712, 0.01709443,
       0.04695698, 0.05826891, 0.0102565 , 0.04580954, 0.04082199,
       0.06155789, 0.04040954, 0.03670137, 0.05310983, 0.02409755,
       0.0402739 , 0.03822121, 0.01379332, 0.03636764, 0.01360565,
       0.02173999, 0.02000067, 0.03974033, 0.02047853, 0.0296318 ,
       0.0719735 , 0.02181905, 0.04546237, 0.04495139, 0.05348868,
       0.05077233, 0.05989814, 0.04405999, 0.05358425, 0.04348511,
       0.05651221, 0.06371581, 0.01869213, 0.06351341, 0.04329681,
       0.05064373, 0.05264373, 0.05526268, 0.03367322, 0.03922071,
       0.01081092, 0.03257617, 0.06291383, 0.04485057, 0.04567004,
       0.0257525 , 0.08068891, 0.02469261, 0.04831858, 0.05756985,
       0.01282069, 0.01754431, 0.02489755, 0.05505978, 0.06744128,
       0.07550755, 0.06559728, 0.05448819, 0.00947874, 0.03101024,
       0.01227009, 0.07696104, 0.02941389, 0.03125254, 0.01739174,
       0.08552217, 0.04211149, 0.01481509, 0.03550669, 0.05669534,
       0.01058211, 0.0482021 , 0.01941809, 0.07302514, 0.06805346,
       0.0678226 , 0.00803217, 0.07410797, 0.06213178, 0.07275935,
       0.09309042, 0.01242252, 0.05334598, 0.01069529, 0.02247286,
       0.08515781, 0.07020426, 0.05480824, 0.09097178, 0.02061929,
       0.12136086, 0.0631789 , 0.07145896, 0.05556985, 0.06624939,
       0.00904984, 0.01005034, 0.0255333 , 0.04706751, 0.05173567,
       0.03236528, 0.02334736, 0.03448618, 0.03460553, 0.0065574 ,
       0.00790518, 0.00647251, 0.03278982, 0.00766287, 0.0114287 ,
       0.02857337, 0.02105341, 0.06595797, 0.04367506, 0.01503788,
       0.0098523 , 0.01892801, 0.03333642, 0.04609111, 0.02702867,
       0.02150621, 0.02381065, 0.01980263, 0.03149867, 0.03891542,
       0.03484673, 0.01886848, 0.05218575, 0.06001801, 0.05623972,
       0.02790879, 0.00784318, 0.02690745, 0.05026183, 0.04976151,
       0.03883983, 0.04149973, 0.00930239, 0.00896867, 0.07061757,
       0.11122564, 0.03745756, 0.01550419, 0.01801851, 0.04016604,
       0.00873368, 0.00749067, 0.07210329, 0.04049136, 0.08269172,
       0.10821358, 0.11506933, 0.04124296, 0.04000533, 0.06124363,
       0.0111733 , 0.03846628, 0.03738753, 0.02020271, 0.01257878,
       0.01183446, 0.01515181, 0.00956945, 0.02272825, 0.00711747,
       0.04725288, 0.03315221, 0.08004271, 0.00696867, 0.04927105,
```

```
0.03243528, 0.03419136, 0.01104984, 0.03030535, 0.03200273,
0.03039748, 0.03774033, 0.0235305 , 0.01785762, 0.02919915,
0.02230576, 0.02076199, 0.03115517, 0.01823759, 0.01234584,
0.03135053, 0.01538492, 0.02325686, 0.0319516 , 0.0671393 ,
0.04939276, 0.04115807, 0.02439145, 0.10536052, 0.07847162,
0.04948006, 0.00754721, 0.02765153, 0.0330063 , 0.03208831,
0.05465841, 0.05043085, 0.05675282, 0.0063898 , 0.0724955 ,
0.07356257, 0.0428637 , 0.00938974, 0.00888895, 0.01438874,
0.06669137, 0.08338161, 0.0732034 , 0.0314162 , 0.03436764,
0.02620237, 0.07940678, 0.03559095, 0.00966191, 0.01851905,
0.03603994, 0.06995859, 0.08408312, 0.0885534 , 0.09237332,
0.03215711, 0.03322565, 0.07045166, 0.04839654, 0.07006756,
0.02264248, 0.01587335, 0.00843887, 0.00738011, 0.04525659,
0.03361661, 0.03015304, 0.0082305 , 0.03533937, 0.04512044,
0.06103589, 0.05085842, 0.04184711, 0.04054609, 0.0076046 ,
0.03663413, 0.07898841, 0.01904819, 0.0250013 , 0.05279819])
```

6. Packaged codes

In [121…

```python
def vif(df, col_i):
        from statsmodels.formula.api import ols
        cols = list(df.columns)
        cols.remove(col_i)
        cols_noti = cols
        formula = col_i + '~' + '+'.join(cols_noti)
        r2 = ols(formula, df).fit().rsquared
        return 1.0 / (1.0 - r2)
```

In [121…

```python
def press_statistic(y_true, y_pred, xs):
    """
    Calculation of the `Press Statistics <https://www.otexts.org/1580>`_
    """
    res = y_pred - y_true
    hat = xs.dot(np.linalg.pinv(xs))
    den = (1 - np.diagonal(hat))
    sqr = np.square(res/den)
    return sqr.sum()

def predicted_r2(y_true, y_pred, xs):
    """
    Calculation of the `Predicted R-squared <https://rpubs.com/RatherBit/102428>`_
    """
    press = press_statistic(y_true=y_true,
                            y_pred=y_pred,
                            xs=xs
    )

    sst  = np.square( y_true - y_true.mean() ).sum()
    return 1 - press / sst

def r2(y_true, y_pred):
    """
```

```
    Calculation of the unadjusted r-squared, goodness of fit metric
    """
    sse  = np.square( y_pred - y_true ).sum()
    sst  = np.square( y_true - y_true.mean() ).sum()
    return 1 - sse/sst
```

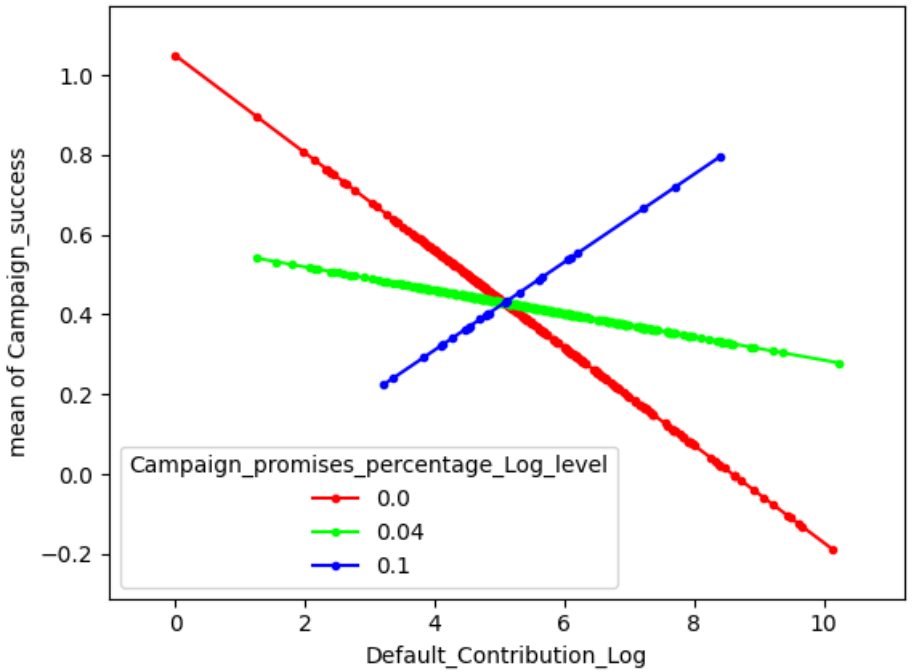
In [121…
```
def Linear_Regression(X, Y):
    # with statsmodels
    # X = sm.add_constant(X) # adding a constant
    model1 = sm.OLS(Y, X).fit()
    results1 = model1.summary()
    #predicts = model1._results
    print(results1)
    return model1
```

In [121…
```
from statsmodels.graphics.factorplots import interaction_plot
from statsmodels.formula.api import ols
def Interaction_plot(DV, interaction_variable1, interaction_variable2_catergial, Dataset):
    Y = Dataset[[DV]]
    X = Dataset[[interaction_variable1, interaction_variable2_catergial]]
    Reg7 = ols(formula = "%s ~ %s  * %s " % (DV, interaction_variable1, interaction_variable2_catergial), data = extract_data).fit()
    fig = interaction_plot(Dataset[interaction_variable1], Dataset[interaction_variable2_catergial], Reg7.fittedvalues,
                           ylabel= DV, xlabel = interaction_variable1)
    plt.show()
```

In [122…
```
from scipy.stats import gaussian_kde
def  residule_plot(x, y, data_set, remove_zero = 0):
    X = np.array(data_set[x])
    Y = np.array(y)
    # remove 0 line
    if remove_zero == 1:
        Y = np.delete(Y, X == 0)
        X = np.delete(X, X == 0)
    XY = np.vstack([X, Y])
    Z = gaussian_kde(XY)(XY)
    idx = Z.argsort()
    X = X[idx]
    Y = Y[idx]
    Z = Z[idx]

    plt.figure(figsize=(10,10))
    plt.scatter(X, Y, marker=',', c=Z, s=15, cmap='Blues' )
    #plt.hist2d(y, x, bins=100, norm=LogNorm(), cmap='Blues' )
    plt.axhline(y = 0, color='b', linestyle='--', lw=0.5)
    plt.xlabel(x)
    #plt.xlim((-0.5, 11))
    plt.ylabel("Residual")
    #plt.savefig('%s.png' % x, dpi=1600) # save as png or not
    plt.show()
```

In [122…
```
DV = 'Campaign_success'
interaction_variable1 = 'Default_Contribution_Log'
interaction_variable2_catergial = 'Campaign_promises_percentage_Log_level'
```

```
Interaction_plot(DV, interaction_variable1, interaction_variable2_catergial, extract_data)
```



In [122…

```
Model1_variables = ['Campaign_promises_percentage', 'Default_Contribution_Log']
import statsmodels.formula.api as smf
X = extract_data[Model1_variables]
Y = extract_data['Amount_raised_Log']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
print(results1)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Amount_raised_Log   R-squared:                       0.057
Model:                            OLS   Adj. R-squared:                  0.057
Method:                 Least Squares   F-statistic:                     220.8
Date:                Sat, 24 Jun 2023   Prob (F-statistic):           7.83e-94
Time:                        15:40:37   Log-Likelihood:                -15125.
No. Observations:                7340   AIC:                         3.026e+04
Df Residuals:                    7337   BIC:                         3.028e+04
Df Model:                           2
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                            7.1918      0.030    236.274      0.000       7.132       7.251
Campaign_promises_percentage    -0.0470      1.218     -0.039      0.969      -2.435       2.341
Default_Contribution_Log         0.1839      0.010     18.311      0.000       0.164       0.204
==============================================================================
Omnibus:                      145.505   Durbin-Watson:                   1.402
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              154.215
Skew:                          -0.344   Prob(JB):                     3.26e-34
Kurtosis:                       3.176   Cond. No.                         195.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

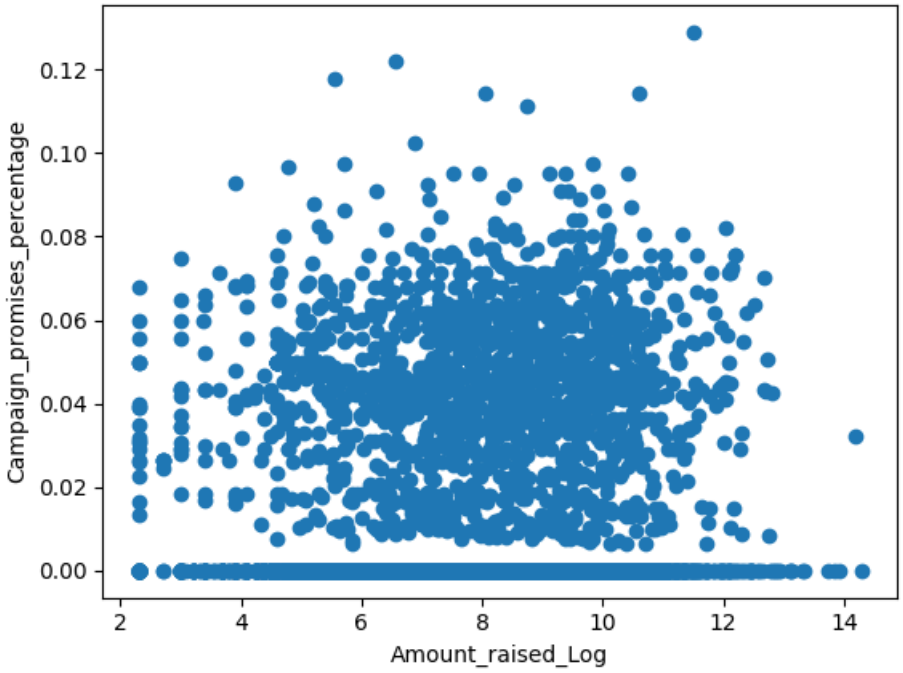
7. Linear regression model

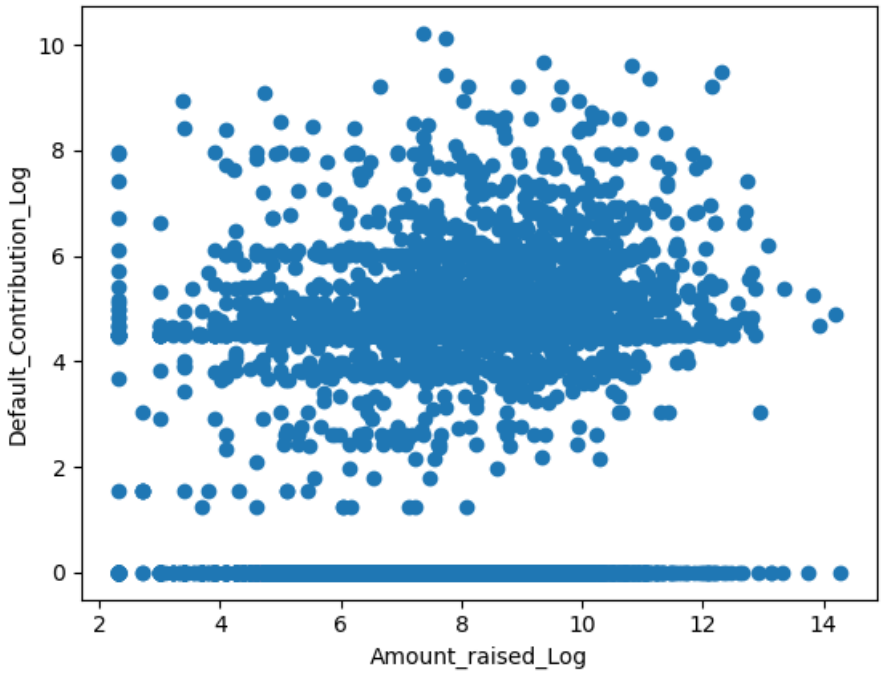
The Linear regression of selected variables Model 1

In [122…
```
extract_data['Org_causes'].value_counts()
```

Out[1224]:
```
4    5714
3     906
1     439
2     281
Name: Org_causes, dtype: int64
```

In [122…
```
plt.scatter(extract_data["Amount_raised_Log"], extract_data["Campaign_promises_percentage"])
#plt.xlim(0,30000)
#plt.ylim(0,30000)
plt.xlabel('Amount_raised_Log'); plt.ylabel('Campaign_promises_percentage')
plt.show()

plt.scatter(extract_data["Amount_raised_Log"], extract_data["Default_Contribution_Log"])
plt.xlabel('Amount_raised_Log'); plt.ylabel('Default_Contribution_Log')
plt.show()
```





In [122…

```
extract_data["Campaign_Start_Year"] = pd.to_numeric(extract_data["Campaign_Start_Year"])
extract_data["Creator_Type"] = pd.to_numeric(extract_data["Creator_Type"])
```

In [122…
```
# extract_data.to_csv('4-Jun-dataset.csv')
```

Base model

In [122…
```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    "Campaign_frequency"]

import statsmodels.formula.api as smf
X = extract_data[Model1_variables]
Y = extract_data['Amount_raised_Log']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
print(results1)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Amount_raised_Log   R-squared:                       0.342
Model:                            OLS   Adj. R-squared:                  0.341
Method:                 Least Squares   F-statistic:                     346.3
Date:                Sat, 24 Jun 2023   Prob (F-statistic):               0.00
Time:                        15:40:38   Log-Likelihood:                -13803.
No. Observations:                7340   AIC:                         2.763e+04
Df Residuals:                    7328   BIC:                         2.771e+04
Df Model:                          11
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                            1.2349      0.164      7.540      0.000       0.914       1.556
Campaign_Goal_Log                0.5187      0.012     44.873      0.000       0.496       0.541
Tax_duction_status               0.2141      0.100      2.149      0.032       0.019       0.409
Campaign_Duration                0.0013      0.000      7.055      0.000       0.001       0.002
Campaign_Video                   0.0604      0.039      1.531      0.126      -0.017       0.138
Number_of_images                 0.0213      0.012      1.733      0.083      -0.003       0.045
Words_of_campaigns               0.0029      0.000      9.983      0.000       0.002       0.003
Creator_Type                     0.1728      0.021      8.228      0.000       0.132       0.214
Financial_Size                   0.1007      0.018      5.494      0.000       0.065       0.137
Sector_type                     -0.0763      0.015     -5.054      0.000      -0.106      -0.047
Campaign_Start_Year_category     0.1183      0.014      8.270      0.000       0.090       0.146
Campaign_frequency              -0.0059      0.001     -7.390      0.000      -0.007      -0.004
==============================================================================
Omnibus:                     1003.655   Durbin-Watson:                   1.552
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1597.298
Skew:                          -0.951   Prob(JB):                         0.00
Kurtosis:                       4.268   Cond. No.                     1.81e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.81e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Model 2 = Base model + Future tense percentage + Default_Contribution_Log

In [122…

```python
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                    'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                   "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                   'Campaign_frequency' , "Campaign_promises_percentage", "Default_Contribution_Log"]

import statsmodels.formula.api as smf
X = extract_data[Model1_variables]
Y = extract_data['Amount_raised_Log']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
print(results1)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Amount_raised_Log   R-squared:                       0.344
Model:                            OLS   Adj. R-squared:                  0.343
Method:                 Least Squares   F-statistic:                     296.0
Date:                Sat, 24 Jun 2023   Prob (F-statistic):               0.00
Time:                        15:40:38   Log-Likelihood:                -13790.
No. Observations:                7340   AIC:                         2.761e+04
Df Residuals:                    7326   BIC:                         2.770e+04
Df Model:                          13
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                            1.2260      0.164      7.493      0.000       0.905       1.547
Campaign_Goal_Log                0.5419      0.013     43.042      0.000       0.517       0.567
Tax_duction_status               0.1906      0.100      1.913      0.056      -0.005       0.386
Campaign_Duration                0.0013      0.000      7.163      0.000       0.001       0.002
Campaign_Video                   0.0409      0.040      1.034      0.301      -0.037       0.119
Number_of_images                 0.0170      0.012      1.384      0.166      -0.007       0.041
Words_of_campaigns               0.0030      0.000     10.434      0.000       0.002       0.004
Creator_Type                     0.1045      0.026      3.951      0.000       0.053       0.156
Financial_Size                   0.0994      0.018      5.431      0.000       0.064       0.135
Sector_type                     -0.0692      0.015     -4.570      0.000      -0.099      -0.039
Campaign_Start_Year_category     0.1153      0.014      8.070      0.000       0.087       0.143
Campaign_frequency              -0.0063      0.001     -7.914      0.000      -0.008      -0.005
Campaign_promises_percentage     2.7688      1.025      2.702      0.007       0.760       4.777
Default_Contribution_Log        -0.0617      0.012     -5.060      0.000      -0.086      -0.038
==============================================================================
Omnibus:                     1023.723   Durbin-Watson:                   1.559
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1649.432
Skew:                          -0.960   Prob(JB):                         0.00
Kurtosis:                       4.306   Cond. No.                     1.11e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.11e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Model 3 = Model 2 + Campaign_promises_percentage_X_Default_Contribution_Log

In [123…

```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                    'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                   "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                   'Campaign_frequency' , "Campaign_promises_percentage", "Default_Contribution_Log",
                   'Campaign_promises_percentage_X_Default_Contribution_Log']

import statsmodels.formula.api as smf
X = extract_data[Model1_variables]
Y = extract_data['Amount_raised_Log']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()
```

```
results1 = model1.summary()
#predicts = model1._results
print(results1)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Amount_raised_Log   R-squared:                       0.347
Model:                            OLS   Adj. R-squared:                  0.346
Method:                 Least Squares   F-statistic:                     277.8
Date:                Sat, 24 Jun 2023   Prob (F-statistic):               0.00
Time:                        15:40:38   Log-Likelihood:                -13776.
No. Observations:                7340   AIC:                         2.758e+04
Df Residuals:                    7325   BIC:                         2.769e+04
Df Model:                          14
Covariance Type:            nonrobust
=============================================================================================================================
                                                               coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------------------------------------
const                                                        1.2958      0.164      7.907      0.000       0.975       1.617
Campaign_Goal_Log                                            0.5411      0.013     43.055      0.000       0.516       0.566
Tax_duction_status                                           0.2094      0.100      2.103      0.035       0.014       0.404
Campaign_Duration                                            0.0013      0.000      7.129      0.000       0.001       0.002
Campaign_Video                                               0.0339      0.040      0.857      0.391      -0.044       0.111
Number_of_images                                             0.0157      0.012      1.277      0.202      -0.008       0.040
Words_of_campaigns                                           0.0031      0.000     10.652      0.000       0.003       0.004
Creator_Type                                                 0.0832      0.027      3.114      0.002       0.031       0.136
Financial_Size                                               0.0966      0.018      5.283      0.000       0.061       0.132
Sector_type                                                 -0.0715      0.015     -4.731      0.000      -0.101      -0.042
Campaign_Start_Year_category                                 0.1106      0.014      7.737      0.000       0.083       0.139
Campaign_frequency                                          -0.0066      0.001     -8.217      0.000      -0.008      -0.005
Campaign_promises_percentage                               -18.8577      4.263     -4.423      0.000     -27.215     -10.501
Default_Contribution_Log                                    -0.0771      0.013     -6.156      0.000      -0.102      -0.053
Campaign_promises_percentage_X_Default_Contribution_Log      4.5530      0.871      5.225      0.000       2.845       6.261
==============================================================================
Omnibus:                     1019.778   Durbin-Watson:                   1.565
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1642.688
Skew:                          -0.957   Prob(JB):                         0.00
Kurtosis:                       4.307   Cond. No.                     4.70e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.7e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Robustness test 1

In [123…

```
'''
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    'Campaign_frequency' , "Campaign_promises_percentage", "Default_Contribution_Log",
                    'Campaign_promises_percentage_X_Default_Contribution_Log']

import statsmodels.formula.api as smf
```

```
X = extract_data[Model1_variables]
Y = extract_data['Distinct_Donors_Log']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
print(results1)
'''
```

Out[1231]:
```
'\nModel1_variables = [\'Campaign_Goal_Log\', \'Tax_duction_status\', \'Campaign_Duration\',\n                    \'Campaign_Video\',  \'Number_of_images\',\'Words_of_campaigns\',\n                    "Creator_Type", \'Financial_Size\', "Sector_type", "Campaign_Start_Year_category",\n                    \'Campaign_frequency\' , "Campaign_promises_percentage", "Default_Contribution_Log",\n                    \'Campaign_promises_percentage_X_Default_Contribution_Log\']\n\nimport statsmodels.formula.api as smf\nX = extract_data[Model1_variables] \nY = extract_data[\'Distinct_Donors_Log\']\n# with statsmodels\nX = sm.add_constant(X) # adding a constant\nmodel1 = sm.OLS(Y, X).fit()\n\nresults1 = model1.summary()\n#predicts = model1._results\nprint(results1)\n'
```

Calculate Residual Standard Error

In [123…
```
# Residual Standard Error of the model
np.sqrt(model1.scale)
```

Out[1232]:
```
1.5823701616420498
```

Calculate predicted_r2

In [123…
```
IV1 = extract_data['Campaign_promises_percentage']
len(model1.resid)
```

Out[1233]:
```
7340
```

Based on last model, get the ourliers

In [123…
```
#abnormal detection
outliers = model1.get_influence()
#High leverage points
leverage = outliers.hat_matrix_diag
#DFFITS value
dffits = outliers.dffits[0]
#standard residue
resid_stu = outliers.resid_studentized_external
#Cook distance
cook = outliers.cooks_distance[0]
#Merge all the abnormal values
contat1 = pd.concat([pd.Series(leverage, name='leverage'), pd.Series(dffits, name='dffits'), pd.Series(resid_stu, name='resid_stu'), pd.Series(cook, name='cook')],
#reset train data index
extract_data.index = range(extract_data.shape[0])
#Merge abnormal values with original data
profit_outliers = pd.concat([extract_data, contat1], axis=1)
#set display numbers
```

```
#pd.set_option('display.width', 200)
#set display columns
pd.set_option('display.max_columns',None)
#set display rows
pd.set_option('display.max_rows', None)
#print(profit_outliers)

#print(profit_outliers[np.abs(profit_outliers.resid_stu)>4])
#calculate the percentage of abnomal
#outliers_ratio = sum(np.where( (np.abs(profit_outliers.resid_stu)>2), 1, 0)) / profit_outliers.shape[0]
#print(outliers_ratio)
```

Remove ourliers

In [123…

```
print("Number of outliers:" + str(profit_outliers[np.abs(profit_outliers.resid_stu)>2].shape[0]))
print("Rows numbers before removing outliers:" + str(extract_data.shape[0]))
out_liers_Campaign_Id = profit_outliers[np.abs(profit_outliers.resid_stu)>2]['Campaign_Id'].to_list()
out_liers_Campaign_Id = profit_outliers[np.abs(profit_outliers.resid_stu)>2]['Campaign_Id'].to_list()
num1 = 0
for index, row in extract_data.iterrows():
    if extract_data.loc[index, 'Campaign_Id'] in out_liers_Campaign_Id:
        extract_data.drop(index, inplace=True)
        num1 += 1
print("Total delete out liers numbers :", num1)
print("Rows numbers after removing outliers:" + str(extract_data.shape[0]))
```

```
Number of outliers:368
Rows numbers before removing outliers:7340
Total delete out liers numbers : 368
Rows numbers after removing outliers:6972
```

Already removed outliers, left 6972

Delete unused columns

In [ ]:

```
unused_columns = ['Donation_per_donor', 'Campaign_Start_Day','Campaign_Start_Month',
       'Campaign_End_Day', 'Campaign_End_Month', 'Campaign_End_Year',
       'Campaign_Start', 'Campaign_End',
       'Num_desc_NPO',
       'Campaign_Duration_Log',
       'Campaign_promises_X_Campaign_frequency',
       'Campaign_frequency_X_Default_Contribution_Log',
       'Log_Org_causes',
       'Campaign_promises_percentage_X_Campaign_frequency',
       'Campaign_promises_percentage_X_Default_Contribution',
       'Amount_raised_Log_round']
extract_data = extract_data.drop(unused_columns, axis = 1)
extract_data.to_csv("24-Jun-dataset.csv")
```

VIF test 1

In [ ]:
```
numeric_features2 = ['Amount_raised_Log', 'Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    "Campaign_frequency", "Campaign_promises_percentage_Log", "Default_Contribution_Log",
                    'Campaign_promises_percentage_Log_X_Default_Contribution_Log']
```

In [123…
```
test_data = extract_data[numeric_features2]
for i in numeric_features2:
        print(i, "\t", vif(df=test_data, col_i=i))
```

```
Amount_raised_Log        1.9593106446166808
Campaign_Goal_Log        2.624003971179146
Tax_duction_status       1.2983320784865426
Campaign_Duration        1.2314179548281596
Campaign_Video   1.0400803122780522
Number_of_images         1.0792448800410415
Words_of_campaigns       1.1383598534155797
Creator_Type     1.851967070342675
Financial_Size   1.3432048759464552
Sector_type      1.0437708143972735
Campaign_Start_Year_category     1.131199489488718
Campaign_frequency       1.3942757589848307
Campaign_promises_percentage_Log         25.609168145916783
Default_Contribution_Log         2.9977379861637385
Campaign_promises_percentage_Log_X_Default_Contribution_Log      26.500138223471204
```

VIF test 2

In [123…
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
import numpy as np
test_data = extract_data[numeric_features2[1:]]
x = add_constant(test_data)
# 当VIF<10,说明不存在多重共线性；当10<=VIF<100,存在较强的多重共线性，当VIF>=100,存在严重多重共线性
vif_res = [variance_inflation_factor(x.values, x.columns.get_loc(i)) for i in x.columns]
j = 0
for i in numeric_features2:
        print(i, "\t", vif_res[j])
        j += 1
```

```
Amount_raised_Log        80.23518076228166
Campaign_Goal_Log        1.76015777952216
Tax_duction_status       1.2977996622549353
Campaign_Duration        1.221759481398104
Campaign_Video   1.0400478032426637
Number_of_images         1.0789683682930369
Words_of_campaigns       1.1195554309368771
Creator_Type     1.8489910740399722
Financial_Size   1.3365594899149154
Sector_type      1.040111030772338
Campaign_Start_Year_category     1.1211425272389306
Campaign_frequency       1.3737971462680727
Campaign_promises_percentage_Log         25.523379186538488
Default_Contribution_Log         2.9635937877468845
Campaign_promises_percentage_Log_X_Default_Contribution_Log      26.373043435709295
```

VIF test 3

In [123…

```
# When VIF<10, it means that there is no multicollinearity;
# when 10<=VIF<100, there is strong multicollinearity; when VIF>=100, there is severe multicollinearity
tol = [1./variance_inflation_factor(x.values, x.columns.get_loc(i)) for i in x.columns]
j = 0
for i in numeric_features2:
        print(i, "\t", tol[j])
        j += 1
```

```
Amount_raised_Log        0.012463360716576055
Campaign_Goal_Log        0.5681308866932803
Tax_duction_status       0.7705349516446117
Campaign_Duration        0.8184917041574037
Campaign_Video   0.9614942667848512
Number_of_images         0.9268112294914007
Words_of_campaigns       0.8932116913256991
Creator_Type     0.5408354934970235
Financial_Size   0.7481896672355822
Sector_type      0.9614358183062884
Campaign_Start_Year_category     0.8919472553259827
Campaign_frequency       0.7279095044828892
Campaign_promises_percentage_Log         0.03917976505741916
Default_Contribution_Log         0.337428160409347
Campaign_promises_percentage_Log_X_Default_Contribution_Log      0.03791750475965139
```

Add Log Avg custom amount level

In [124…

```
print(extract_data['Default_Contribution_Log'].min())
print(extract_data['Default_Contribution_Log'].max())
print(np.median(extract_data['Default_Contribution_Log']))
print(extract_data['Default_Contribution_Log'].mean())
```

```
0.0
10.130164852798476
0.0
2.3553805042430627
```

In [124…
```
extract_data['Default_Contribution_Log_level'] = 0
for index, row in extract_data.iterrows():
    if extract_data.loc[index,'Default_Contribution_Log'] < 5.35 and extract_data.loc[index,'Default_Contribution_Log'] > 2.35:
        extract_data.loc[index,'Default_Contribution_Log_level'] = 2.35
    if extract_data.loc[index,'Default_Contribution_Log'] >= 5.35:
        extract_data.loc[index,'Default_Contribution_Log_level'] = 5.35

    if extract_data.loc[index,'Default_Contribution_Log'] <= 2.35:
        extract_data.loc[index,'Default_Contribution_Log_level'] = 0.35
```

In [124…
```
extract_data['Default_Contribution_Log_level'].unique()
```

Out[1242]:
```
array([2.35, 0.35, 5.35])
```

Another Base model

In [124…
```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    "Campaign_frequency"]

import statsmodels.formula.api as smf
X = extract_data[Model1_variables]
Y = extract_data['Amount_raised_Log']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
print(results1)
y_true= Y
y_pred = model1.predict(X)
xs = X
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Amount_raised_Log   R-squared:                       0.483
Model:                            OLS   Adj. R-squared:                  0.482
Method:                 Least Squares   F-statistic:                     590.2
Date:                Sat, 24 Jun 2023   Prob (F-statistic):               0.00
Time:                        15:41:17   Log-Likelihood:                -11592.
No. Observations:                6972   AIC:                         2.321e+04
Df Residuals:                    6960   BIC:                         2.329e+04
Df Model:                          11
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                            1.0893      0.136      7.985      0.000       0.822       1.357
Campaign_Goal_Log                0.5840      0.010     60.295      0.000       0.565       0.603
Tax_duction_status               0.1502      0.083      1.803      0.071      -0.013       0.313
Campaign_Duration                0.0011      0.000      7.320      0.000       0.001       0.001
Campaign_Video                   0.0473      0.033      1.450      0.147      -0.017       0.111
Number_of_images                 0.0198      0.010      1.951      0.051   -9.65e-05       0.040
Words_of_campaigns               0.0023      0.000      9.757      0.000       0.002       0.003
Creator_Type                     0.1838      0.017     10.618      0.000       0.150       0.218
Financial_Size                   0.0938      0.015      6.142      0.000       0.064       0.124
Sector_type                     -0.0683      0.013     -5.442      0.000      -0.093      -0.044
Campaign_Start_Year_category     0.1011      0.012      8.531      0.000       0.078       0.124
Campaign_frequency              -0.0058      0.001     -8.932      0.000      -0.007      -0.005
==============================================================================
Omnibus:                      300.073   Durbin-Watson:                   1.612
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              326.894
Skew:                          -0.514   Prob(JB):                     1.04e-71
Kurtosis:                       2.739   Cond. No.                     1.83e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.83e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [124…
```
# Residual Standard Error of the model
print("Residual Standard Error: "+ str(model1.scale) )
print("Predicted R2 : "+ str(predicted_r2(y_true, y_pred, xs)) )
```

```
Residual Standard Error: 1.63113465960979
Predicted R2 : 0.48068882472442764
```

In [124…
```
'''

test_data = extract_data[numeric_features2]
for i in numeric_features2:
        print(i, "\t", vif(df=test_data, col_i=i))

'''
```

Out[1245]:
```
'\n\ntest_data = extract_data[numeric_features2]\nfor i in numeric_features2:\n\tprint(i, "\t", vif(df=test_data, col_i=i))\n    \n'
```

Change future tense and custom amount into category variables

In [124…
```
extract_data.columns
```

Out[1246]:
```
Index(['Campaign_Id', 'Campaign_Title_x', 'Receiving_NPO_Name',
       'Receiving_NPO_Id', 'NPO_Status', 'Campaign_frequency',
       'Public_Campaign_Access', 'Creator_Type', 'Creator_Id',
       'Campaign_Status', 'Amount_raised', 'Distinct_Donors', 'Campaign_Goal',
       'Campaign_Completion_Rate', 'Days_Left_for_Campaign',
       'Campaign_Start_Date', 'Campaign_End_Date', 'Tax_duction_status',
       'Campaign_Image1_Id', 'Campaign_Image2_Id', 'Campaign_Image3_Id',
       'Campaign_Image4_Id', 'Campaign_Image5_Id', 'Campaign_Video',
       'Impact_Message_1', 'Impact_Message_2', 'Impact_Message_3',
       'Impact_Message_4', 'Impact_Message_5', 'Custom_Amount_1',
       'Custom_Amount_2', 'Custom_Amount_3', 'Custom_Amount_4',
       'Description_of_Campaign', 'Description_of_NPO', 'Campaign_Title_y',
       'Org_Cause_Animal_Welfare', 'Org_Cause_Arts_&_Heritage',
       'Org_Cause_Children_&_Youth', 'Org_Cause_Community',
       'Org_Cause_Disability', 'Org_Cause_Education', 'Org_Cause_Elderly',
       'Org_Cause_Environment', 'Org_Cause_Families', 'Org_Cause_Health',
       'Org_Cause_Humanitarian', 'Org_Cause_Social_Service',
       'Org_Cause_Sports', 'Org_Cause_Women_&_Girls',
       'Cam_Cause_Animal_Welfare', 'Cam_Cause_Arts_&_Heritage',
       'Cam_Cause_Children_&_Youth', 'Cam_Cause_Community',
       'Cam_Cause_Disability', 'Cam_Cause_Education', 'Cam_Cause_Elderly',
       'Cam_Cause_Environment', 'Cam_Cause_Families', 'Cam_Cause_Health',
       'Cam_Cause_Humanitarian', 'Cam_Cause_Social_Service',
       'Cam_Cause_Sports', 'Cam_Cause_Women_&_Girls', 'Pub_Enquiry_Person',
       'Pub_Enquiry_Contact', 'Pub_Enquiry_Email', 'Web_URL', 'Facebook_Link',
       'Org_causes', 'Cam_causes', 'S/N', 'Type', 'UEN', 'IPC_Period',
       'Sector', 'Classification', 'Activities', 'Financial_Size',
       'Campaign_Start_Year', 'Campaign_Duration', 'Sector_type',
       'Campaign_Start_Year_category', 'Campaign_Funds_Raised',
       'Number_of_images', 'Words_of_campaigns', 'Default_Contribution',
       'Campaign_promises', 'Campaign_promises_percentage',
       'Amount_raised_Log', 'Distinct_Donors_Log', 'Campaign_Goal_Log',
       'Default_Contribution_Log', 'Campaign_success', 'Campaign_success_Log',
       'Campaign_promises_percentage_X_Default_Contribution_Log',
       'Campaign_promises_percentage_Log_level',
       'Campaign_promises_percentage_Log',
       'Campaign_promises_percentage_Log_X_Default_Contribution_Log',
       'Default_Contribution_Log_level'],
      dtype='object')
```

In [124…
```
extract_data['Campaign_promises_categoty'] = 0
extract_data['Custom_amount_categoty'] = 0
for index, row in extract_data.iterrows():
    if extract_data.loc[index,'Campaign_promises'] > 0:
        extract_data.loc[index,'Campaign_promises_categoty'] = 1
    if extract_data.loc[index,'Default_Contribution'] > 0:
        extract_data.loc[index,'Custom_amount_categoty'] = 1
```

Another model 2

In [124…
```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                    'Campaign_Video',  'Number_of_images','Words_of_campaigns',
```

```
                  "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                  "Campaign_frequency", "Campaign_promises_categoty", "Custom_amount_categoty"]

import statsmodels.formula.api as smf
X = extract_data[Model1_variables]
Y = extract_data['Amount_raised_Log']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
print(results1)
y_true= Y
y_pred = model1.predict(X)
xs = X
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Amount_raised_Log   R-squared:                       0.488
Model:                            OLS   Adj. R-squared:                  0.487
Method:                 Least Squares   F-statistic:                     510.7
Date:                Sat, 24 Jun 2023   Prob (F-statistic):               0.00
Time:                        15:41:18   Log-Likelihood:                -11554.
No. Observations:                6972   AIC:                         2.314e+04
Df Residuals:                    6958   BIC:                         2.323e+04
Df Model:                          13
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                            1.1399      0.136      8.385      0.000       0.873       1.406
Campaign_Goal_Log                0.6167      0.010     59.477      0.000       0.596       0.637
Tax_duction_status               0.1258      0.083      1.516      0.130      -0.037       0.289
Campaign_Duration                0.0012      0.000      7.523      0.000       0.001       0.001
Campaign_Video                   0.0167      0.033      0.511      0.609      -0.047       0.081
Number_of_images                 0.0129      0.010      1.271      0.204      -0.007       0.033
Words_of_campaigns               0.0026      0.000     10.714      0.000       0.002       0.003
Creator_Type                     0.0635      0.022      2.830      0.005       0.020       0.107
Financial_Size                   0.0921      0.015      6.058      0.000       0.062       0.122
Sector_type                     -0.0599      0.013     -4.777      0.000      -0.084      -0.035
Campaign_Start_Year_category     0.0917      0.012      7.741      0.000       0.068       0.115
Campaign_frequency              -0.0067      0.001    -10.256      0.000      -0.008      -0.005
Campaign_promises_categoty       0.1239      0.044      2.792      0.005       0.037       0.211
Custom_amount_categoty          -0.4731      0.054     -8.715      0.000      -0.580      -0.367
==============================================================================
Omnibus:                      298.341   Durbin-Watson:                   1.626
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              326.695
Skew:                          -0.516   Prob(JB):                     1.15e-71
Kurtosis:                       2.754   Cond. No.                     1.83e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.83e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [124…
```
'''
test_data = extract_data[numeric_features2]
for i in numeric_features2:
        print(i, "\t", vif(df=test_data, col_i=i))
'''
```

Out[1249]:
```
'\ntest_data = extract_data[numeric_features2]\nfor i in numeric_features2:\n\tprint(i, "\t", vif(df=test_data, col_i=i))\n'
```

In [125…
```
# Residual Standard Error of the model
print("Residual Standard Error: "+ str(model1.scale) )
print("Predicted R2 : "+ str(predicted_r2(y_true, y_pred, xs)) )
```

```
Residual Standard Error: 1.6137221760814024
Predicted R2 : 0.48606785166163413
```

Lastest Model 1

In [125…
```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    "Campaign_frequency"]

import statsmodels.formula.api as smf
X = extract_data[Model1_variables]
Y = extract_data['Amount_raised_Log']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
print(results1)
y_true= Y
y_pred = model1.predict(X)
xs = X
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Amount_raised_Log   R-squared:                       0.483
Model:                            OLS   Adj. R-squared:                  0.482
Method:                 Least Squares   F-statistic:                     590.2
Date:                Sat, 24 Jun 2023   Prob (F-statistic):               0.00
Time:                        15:41:19   Log-Likelihood:                -11592.
No. Observations:                6972   AIC:                         2.321e+04
Df Residuals:                    6960   BIC:                         2.329e+04
Df Model:                          11
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                            1.0893      0.136      7.985      0.000       0.822       1.357
Campaign_Goal_Log                0.5840      0.010     60.295      0.000       0.565       0.603
Tax_duction_status               0.1502      0.083      1.803      0.071      -0.013       0.313
Campaign_Duration                0.0011      0.000      7.320      0.000       0.001       0.001
Campaign_Video                   0.0473      0.033      1.450      0.147      -0.017       0.111
Number_of_images                 0.0198      0.010      1.951      0.051   -9.65e-05       0.040
Words_of_campaigns               0.0023      0.000      9.757      0.000       0.002       0.003
Creator_Type                     0.1838      0.017     10.618      0.000       0.150       0.218
Financial_Size                   0.0938      0.015      6.142      0.000       0.064       0.124
Sector_type                     -0.0683      0.013     -5.442      0.000      -0.093      -0.044
Campaign_Start_Year_category     0.1011      0.012      8.531      0.000       0.078       0.124
Campaign_frequency              -0.0058      0.001     -8.932      0.000      -0.007      -0.005
==============================================================================
Omnibus:                      300.073   Durbin-Watson:                   1.612
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              326.894
Skew:                          -0.514   Prob(JB):                     1.04e-71
Kurtosis:                       2.739   Cond. No.                     1.83e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.83e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Model 1 robust test 1

In [125…

```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                    'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                   "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                   "Campaign_frequency"]

import statsmodels.formula.api as smf
X = extract_data[Model1_variables]
Y = extract_data['Distinct_Donors_Log']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
```

```
print(results1)
y_true= Y
y_pred = model1.predict(X)
xs = X
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Distinct_Donors_Log   R-squared:                       0.336
Model:                             OLS   Adj. R-squared:                  0.335
Method:                  Least Squares   F-statistic:                     320.2
Date:                 Sat, 24 Jun 2023   Prob (F-statistic):               0.00
Time:                         15:41:19   Log-Likelihood:                -10882.
No. Observations:                6972   AIC:                         2.179e+04
Df Residuals:                    6960   BIC:                         2.187e+04
Df Model:                          11
Covariance Type:             nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                           -1.3980      0.123    -11.348      0.000      -1.639      -1.156
Campaign_Goal_Log                0.3468      0.009     39.638      0.000       0.330       0.364
Tax_duction_status               0.0066      0.075      0.088      0.930      -0.141       0.154
Campaign_Duration                0.0014      0.000     10.074      0.000       0.001       0.002
Campaign_Video                   0.0352      0.029      1.196      0.232      -0.023       0.093
Number_of_images                 0.0294      0.009      3.211      0.001       0.011       0.047
Words_of_campaigns               0.0026      0.000     11.797      0.000       0.002       0.003
Creator_Type                     0.1621      0.016     10.370      0.000       0.131       0.193
Financial_Size                   0.0938      0.014      6.800      0.000       0.067       0.121
Sector_type                     -0.0360      0.011     -3.177      0.001      -0.058      -0.014
Campaign_Start_Year_category     0.0494      0.011      4.615      0.000       0.028       0.070
Campaign_frequency              -0.0050      0.001     -8.506      0.000      -0.006      -0.004
==============================================================================
Omnibus:                       32.659   Durbin-Watson:                   1.547
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               33.007
Skew:                          -0.165   Prob(JB):                     6.80e-08
Kurtosis:                       2.930   Cond. No.                     1.83e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.83e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [125…
```
# Residual Standard Error of the model
print("Residual Standard Error: "+ str(model1.scale) )
print("Predicted R2 : "+ str(predicted_r2(y_true, y_pred, xs)) )
```

```
Residual Standard Error: 1.3303573442351524
Predicted R2 : 0.3335809510365161
```

Model 1 robust test 2

In [125…
```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                    'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                   "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
```

```
                  "Campaign_frequency"]

import statsmodels.formula.api as smf
X = extract_data[Model1_variables]
Y = extract_data['Campaign_success']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
print(results1)
y_true= Y
y_pred = model1.predict(X)
xs = X
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       Campaign_success   R-squared:                       0.116
Model:                            OLS   Adj. R-squared:                  0.114
Method:                 Least Squares   F-statistic:                     82.64
Date:                Sat, 24 Jun 2023   Prob (F-statistic):          2.95e-176
Time:                        15:41:19   Log-Likelihood:                -10645.
No. Observations:                6972   AIC:                         2.131e+04
Df Residuals:                    6960   BIC:                         2.140e+04
Df Model:                          11
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                            2.2703      0.119     19.064      0.000       2.037       2.504
Campaign_Goal_Log               -0.2193      0.008    -25.937      0.000      -0.236      -0.203
Tax_duction_status               0.0236      0.073      0.324      0.746      -0.119       0.166
Campaign_Duration                0.0008      0.000      6.032      0.000       0.001       0.001
Campaign_Video                   0.0207      0.028      0.727      0.467      -0.035       0.077
Number_of_images                -0.0074      0.009     -0.834      0.404      -0.025       0.010
Words_of_campaigns               0.0005      0.000      2.271      0.023     6.5e-05       0.001
Creator_Type                     0.0938      0.015      6.206      0.000       0.064       0.123
Financial_Size                   0.0376      0.013      2.821      0.005       0.011       0.064
Sector_type                     -0.0175      0.011     -1.597      0.110      -0.039       0.004
Campaign_Start_Year_category     0.0150      0.010      1.448      0.148      -0.005       0.035
Campaign_frequency              -0.0041      0.001     -7.261      0.000      -0.005      -0.003
==============================================================================
Omnibus:                    11967.764   Durbin-Watson:                   1.889
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         16861378.119
Skew:                          11.736   Prob(JB):                         0.00
Kurtosis:                     242.774   Cond. No.                     1.83e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.83e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [125…

```
'''
test_data = extract_data[numeric_features2]
for i in numeric_features2:
```

```
        print(i, "\t", vif(df=test_data, col_i=i))
'''
```

Out[1255]: '\ntest_data = extract_data[numeric_features2]\nfor i in numeric_features2:\n\tprint(i, "\t", vif(df=test_data, col_i=i))\n'

In [125…
```
# Residual Standard Error of the model
print("Residual Standard Error: "+ str(model1.scale) )
print("Predicted R2 : "+ str(predicted_r2(y_true, y_pred, xs)) )
```

```
Residual Standard Error: 1.2431022080488225
Predicted R2 : 0.11200001557904404
```

In [125…
```
extract_data['Sqr_Campaign_promises_percentage'] = np.sqrt(extract_data['Campaign_promises_percentage'])
```

In [125…
```
extract_data['Campaign_promises_percentage_Log'] = np.log(extract_data['Campaign_promises_percentage'] + 1)
extract_data['Default_Contribution_Log'] = np.log(extract_data['Default_Contribution'] + 1)
extract_data['Campaign_promises_percentage_Log_X_Default_Contribution_Log'] = extract_data['Campaign_promises_percentage_Log'] * extract_data['Default_Contribution
```

In [125…
```
min(extract_data['Default_Contribution_Log'])
```

Out[1259]: 0.0

Lastest Model2

In [126…
```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    "Campaign_frequency", "Campaign_promises_percentage_Log", "Default_Contribution_Log"]
X = extract_data[Model1_variables]
#X = sm.add_constant(X) # adding a constant
Y = extract_data['Amount_raised_Log']
Model2 = Linear_Regression(X, Y)
y_true= Y
y_pred = Model2.predict(X)
xs = X
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Amount_raised_Log   R-squared (uncentered):                   0.975
Model:                            OLS   Adj. R-squared (uncentered):              0.974
Method:                 Least Squares   F-statistic:                          2.049e+04
Date:                Sat, 24 Jun 2023   Prob (F-statistic):                        0.00
Time:                        15:41:19   Log-Likelihood:                         -11593.
No. Observations:                6972   AIC:                                  2.321e+04
Df Residuals:                    6959   BIC:                                  2.330e+04
Df Model:                          13
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Campaign_Goal_Log                0.6583      0.009     73.063      0.000       0.641       0.676
Tax_duction_status               0.3336      0.079      4.221      0.000       0.179       0.489
Campaign_Duration                0.0011      0.000      7.205      0.000       0.001       0.001
Campaign_Video                   0.0374      0.033      1.144      0.253      -0.027       0.102
Number_of_images                 0.0364      0.010      3.717      0.000       0.017       0.056
Words_of_campaigns               0.0027      0.000     11.326      0.000       0.002       0.003
Creator_Type                     0.1123      0.022      5.159      0.000       0.070       0.155
Financial_Size                   0.1345      0.014      9.406      0.000       0.106       0.163
Sector_type                     -0.0506      0.013     -4.033      0.000      -0.075      -0.026
Campaign_Start_Year_category     0.1292      0.011     11.590      0.000       0.107       0.151
Campaign_frequency              -0.0054      0.001     -8.463      0.000      -0.007      -0.004
Campaign_promises_percentage_Log 3.2911      0.875      3.759      0.000       1.575       5.007
Default_Contribution_Log        -0.0800      0.010     -7.879      0.000      -0.100      -0.060
==============================================================================
Omnibus:                      306.117   Durbin-Watson:                   1.618
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              341.091
Skew:                          -0.532   Prob(JB):                     8.57e-75
Kurtosis:                       2.798   Cond. No.                     1.14e+04
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 1.14e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [126…

```
# Residual Standard Error of the model
print("Residual Standard Error: "+ str(Model2.scale) )
print("Predicted R2 : "+ str(predicted_r2(y_true, y_pred, xs)) )
```

```
Residual Standard Error: 1.631513006859936
Predicted R2 : 0.48048309285062185
```

Model 2 Robust test 1

In [126…

```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                    'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                   "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                   "Campaign_frequency", "Campaign_promises_percentage_Log", "Default_Contribution_Log"]
```

```
import statsmodels.formula.api as smf
X = extract_data[Model1_variables]
Y = extract_data['Distinct_Donors_Log']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
print(results1)
y_true= Y
y_pred = model1.predict(X)
xs = X
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Distinct_Donors_Log   R-squared:                       0.351
Model:                             OLS   Adj. R-squared:                  0.350
Method:                  Least Squares   F-statistic:                     290.0
Date:                 Sat, 24 Jun 2023   Prob (F-statistic):               0.00
Time:                         15:41:19   Log-Likelihood:                -10800.
No. Observations:                6972   AIC:                         2.163e+04
Df Residuals:                    6958   BIC:                         2.172e+04
Df Model:                          13
Covariance Type:             nonrobust
=================================================================================================
                                    coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------------
const                            -1.4095      0.122    -11.565      0.000      -1.648      -1.171
Campaign_Goal_Log                 0.3940      0.009     41.769      0.000       0.376       0.412
Tax_duction_status               -0.0404      0.075     -0.542      0.588      -0.186       0.106
Campaign_Duration                 0.0014      0.000     10.253      0.000       0.001       0.002
Campaign_Video                   -0.0003      0.029     -0.011      0.991      -0.058       0.057
Number_of_images                  0.0222      0.009      2.442      0.015       0.004       0.040
Words_of_campaigns                0.0028      0.000     13.224      0.000       0.002       0.003
Creator_Type                      0.0176      0.020      0.897      0.370      -0.021       0.056
Financial_Size                    0.0915      0.014      6.708      0.000       0.065       0.118
Sector_type                      -0.0246      0.011     -2.183      0.029      -0.047      -0.003
Campaign_Start_Year_category      0.0436      0.011      4.115      0.000       0.023       0.064
Campaign_frequency               -0.0060      0.001    -10.195      0.000      -0.007      -0.005
Campaign_promises_percentage_Log  2.9061      0.782      3.716      0.000       1.373       4.439
Default_Contribution_Log         -0.1160      0.009    -12.799      0.000      -0.134      -0.098
==============================================================================
Omnibus:                       39.180   Durbin-Watson:                   1.582
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               39.579
Skew:                          -0.179   Prob(JB):                     2.54e-09
Kurtosis:                       2.909   Cond. No.                     1.14e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.14e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [126…

```
# Residual Standard Error of the model
print("Residual Standard Error: "+ str(model1.scale) )
print("Predicted R2 : "+ str(predicted_r2(y_true, y_pred, xs)) )
```

```
Residual Standard Error: 1.2999851842415353
Predicted R2 : 0.3485599688377843
```

Model 2 Robust test 2

In [126…

```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    "Campaign_frequency", "Campaign_promises_percentage_Log", "Default_Contribution_Log"]
X = extract_data[Model1_variables]
X = sm.add_constant(X) # adding a constant
Y = extract_data['Campaign_success']
model_robust_test = Linear_Regression(X, Y)
y_pre_robust_test = model_robust_test.predict(X)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       Campaign_success   R-squared:                       0.118
Model:                            OLS   Adj. R-squared:                  0.117
Method:                 Least Squares   F-statistic:                     71.74
Date:                Sat, 24 Jun 2023   Prob (F-statistic):          7.44e-179
Time:                        15:41:20   Log-Likelihood:                -10635.
No. Observations:                6972   AIC:                         2.130e+04
Df Residuals:                    6958   BIC:                         2.139e+04
Df Model:                          13
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                            2.2707      0.119     19.078      0.000       2.037       2.504
Campaign_Goal_Log               -0.2024      0.009    -21.976      0.000      -0.220      -0.184
Tax_duction_status               0.0049      0.073      0.067      0.946      -0.138       0.148
Campaign_Duration                0.0008      0.000      6.017      0.000       0.001       0.001
Campaign_Video                   0.0086      0.029      0.301      0.763      -0.047       0.065
Number_of_images                -0.0095      0.009     -1.071      0.284      -0.027       0.008
Words_of_campaigns               0.0006      0.000      2.751      0.006       0.000       0.001
Creator_Type                     0.0406      0.019      2.127      0.033       0.003       0.078
Financial_Size                   0.0368      0.013      2.763      0.006       0.011       0.063
Sector_type                     -0.0141      0.011     -1.284      0.199      -0.036       0.007
Campaign_Start_Year_category     0.0129      0.010      1.251      0.211      -0.007       0.033
Campaign_frequency              -0.0045      0.001     -7.832      0.000      -0.006      -0.003
Campaign_promises_percentage_Log 0.3258      0.764      0.427      0.670      -1.171       1.823
Default_Contribution_Log        -0.0392      0.009     -4.425      0.000      -0.057      -0.022
==============================================================================
Omnibus:                    11982.077   Durbin-Watson:                   1.898
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         16984223.012
Skew:                          11.764   Prob(JB):                         0.00
Kurtosis:                     243.649   Cond. No.                     1.14e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.14e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [126…
```
'''
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    "Campaign_frequency", "Campaign_promises_percentage_Log", "Default_Contribution_Log"]

import statsmodels.formula.api as smf
X = extract_data[Model1_variables]
Y = extract_data['Campaign_success']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
print(results1)
y_true= Y
y_pred = model1.predict(X)
xs = X
'''
```

Out[1265]:
```
'\nModel1_variables = [\'Campaign_Goal_Log\', \'Tax_duction_status\', \'Campaign_Duration\',\n                     \'Campaign_Video\',  \'Number_of_images\',\'Words_of_campaigns\',\n                    "Creator_Type", \'Financial_Size\', "Sector_type", "Campaign_Start_Year_category",\n                    "Campaign_frequency", "Campaign_promises_percentage_Log", "Default_Contribution_Log"]\n\nimport statsmodels.formula.api as smf\nX = extract_data[Model1_variables] \nY = extract_data[\'Campaign_success\']\n# with statsmodels\nX = sm.add_constant(X) # adding a constant\nmodel1 = sm.OLS(Y, X).fit()\n\nresults1 = model1.summary()\n#predicts = model1._results\nprint(results1)\ny_true= Y\ny_pred = model1.predict(X)\nxs = X\n'
```

In [126…
```
# Residual Standard Error of the model
print("Residual Standard Error: "+ str(model1.scale) )
print("Predicted R2 : "+ str(predicted_r2(y_true, y_pred, xs)) )
```

```
Residual Standard Error: 1.2999851842415353
Predicted R2 : 0.3485599688377843
```

Lastest Model 3

Campaign_promises_percentage_X_Default_Contribution_Log

In [128…
```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    "Campaign_frequency", "Campaign_promises_percentage_Log", "Default_Contribution_Log",
                    'Campaign_promises_percentage_Log_X_Default_Contribution_Log']
X = extract_data[Model1_variables]
X = sm.add_constant(X) # adding a constant
Y = extract_data['Amount_raised_Log']
Model3 = Linear_Regression(X, Y)
y_true= Y
y_pred = Model3.predict(X)
xs = X
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Amount_raised_Log   R-squared:                       0.490
Model:                            OLS   Adj. R-squared:                  0.489
Method:                 Least Squares   F-statistic:                     476.7
Date:                Sat, 24 Jun 2023   Prob (F-statistic):               0.00
Time:                        15:51:29   Log-Likelihood:                -11545.
No. Observations:                6972   AIC:                         2.312e+04
Df Residuals:                    6957   BIC:                         2.322e+04
Df Model:                          14
Covariance Type:            nonrobust
================================================================================================================================
                                                                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------------------------
const                                                            1.1410      0.136      8.384      0.000       0.874       1.408
Campaign_Goal_Log                                                0.6135      0.010     58.432      0.000       0.593       0.634
Tax_duction_status                                               0.1402      0.083      1.689      0.091      -0.022       0.303
Campaign_Duration                                                0.0011      0.000      7.416      0.000       0.001       0.001
Campaign_Video                                                   0.0152      0.033      0.466      0.641      -0.049       0.079
Number_of_images                                                 0.0135      0.010      1.335      0.182      -0.006       0.033
Words_of_campaigns                                               0.0026      0.000     10.810      0.000       0.002       0.003
Creator_Type                                                     0.0736      0.022      3.346      0.001       0.030       0.117
Financial_Size                                                   0.0893      0.015      5.881      0.000       0.060       0.119
Sector_type                                                     -0.0620      0.013     -4.948      0.000      -0.087      -0.037
Campaign_Start_Year_category                                     0.0933      0.012      7.900      0.000       0.070       0.116
Campaign_frequency                                              -0.0066      0.001    -10.184      0.000      -0.008      -0.005
Campaign_promises_percentage_Log                               -18.2669      3.778     -4.836      0.000     -25.672     -10.862
Default_Contribution_Log                                        -0.0926      0.010     -8.953      0.000      -0.113      -0.072
Campaign_promises_percentage_Log_X_Default_Contribution_Log      4.4715      0.772      5.790      0.000       2.958       5.985
==============================================================================
Omnibus:                      297.218   Durbin-Watson:                   1.628
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              325.431
Skew:                          -0.515   Prob(JB):                     2.16e-71
Kurtosis:                       2.755   Cond. No.                     5.05e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.05e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [130…

```
test_data = extract_data[Model1_variables]
for i in Model1_variables:
        print(i, "\t", vif(df=test_data, col_i=i))
```

```
Campaign_Goal_Log        1.7601577795221595
Tax_duction_status       1.297799662254935
Campaign_Duration        1.2217594813981039
Campaign_Video   1.0400478032426634
Number_of_images         1.0789683682930369
Words_of_campaigns       1.1195554309368771
Creator_Type     1.8489910740399722
Financial_Size   1.3365594899149154
Sector_type      1.040111030772338
Campaign_Start_Year_category     1.1211425272389306
Campaign_frequency       1.3737971462680727
Campaign_promises_percentage_Log         25.523379186538488
Default_Contribution_Log         2.9635937877468845
Campaign_promises_percentage_Log_X_Default_Contribution_Log      26.373043435709295
```

Breusch–Pagan test

In [128…

```
 # Conduct the Breusch-Pagan test
import statsmodels.stats.api as sms
from statsmodels.compat import lzip
fit = smf.ols('Amount_raised_Log ~ Campaign_Goal_Log+Tax_duction_status+Campaign_Duration \
                     +Campaign_Video+Number_of_images+Words_of_campaigns+ \
                    Creator_Type+Financial_Size+Sector_type+Campaign_Start_Year_category+ \
                    +Campaign_frequency+Campaign_promises_percentage_Log+Default_Contribution_Log+ \
                    Campaign_promises_percentage_Log_X_Default_Contribution_Log', data=extract_data).fit()

bp_lm, bp_lm_pvalue, bp_fvalue, bp_f_pvalue = sm.stats.diagnostic.het_breuschpagan(model1.resid, model1.model.exog)

print("Lagrange multiplier statistic---> " + str(bp_lm))
print("Lagrange multiplier p-value-----> " + str(bp_lm_pvalue))
print("F-statistic---------------------> " + str(bp_fvalue))
print("P-value of F-statistic----------> " + str(bp_f_pvalue))
```

```
Lagrange multiplier statistic---> 522.9596078821811
Lagrange multiplier p-value-----> 1.9348650332939003e-103
F-statistic---------------------> 43.40243760071765
P-value of F-statistic----------> 1.085823479250943e-107
```

In [129…

```
# Test for heteroscedasticity using the Breusch-Pagan test
# NOTE: statsmodels refers to X variables as `exog` for exogenous
bp_lm, bp_lm_pvalue, bp_fvalue, bp_f_pvalue = sm.stats.diagnostic.het_breuschpagan(model1.resid, model1.model.exog)

print("Lagrange multiplier statistic---> " + str(bp_lm))
print("Lagrange multiplier p-value-----> " + str(bp_lm_pvalue))
print("F-statistic---------------------> " + str(bp_fvalue))
print("P-value of F-statistic----------> " + str(bp_f_pvalue))
```

```
Lagrange multiplier statistic---> 522.9596078821811
Lagrange multiplier p-value-----> 1.9348650332939003e-103
F-statistic---------------------> 43.40243760071765
P-value of F-statistic----------> 1.085823479250943e-107
```

Breusch–Godfrey test

In [129…
```
import statsmodels.stats.diagnostic as dg

#perform Breusch-Godfrey test at order p = 3

print(dg.acorr_breusch_godfrey(model1, nlags=3))
```

```
(591.159252640908, 8.316190458140966e-128, 214.78426667725014, 2.989610990169272e-133)
```

In [129…
```
# Residual Standard Error of the model
print("Residual Standard Error: "+ str(model1.scale) )
print("Predicted R2 : "+ str(predicted_r2(y_true, y_pred, xs)) )
```

```
Residual Standard Error: 1.2999851842415353
Predicted R2 : 0.4872441428242723
```

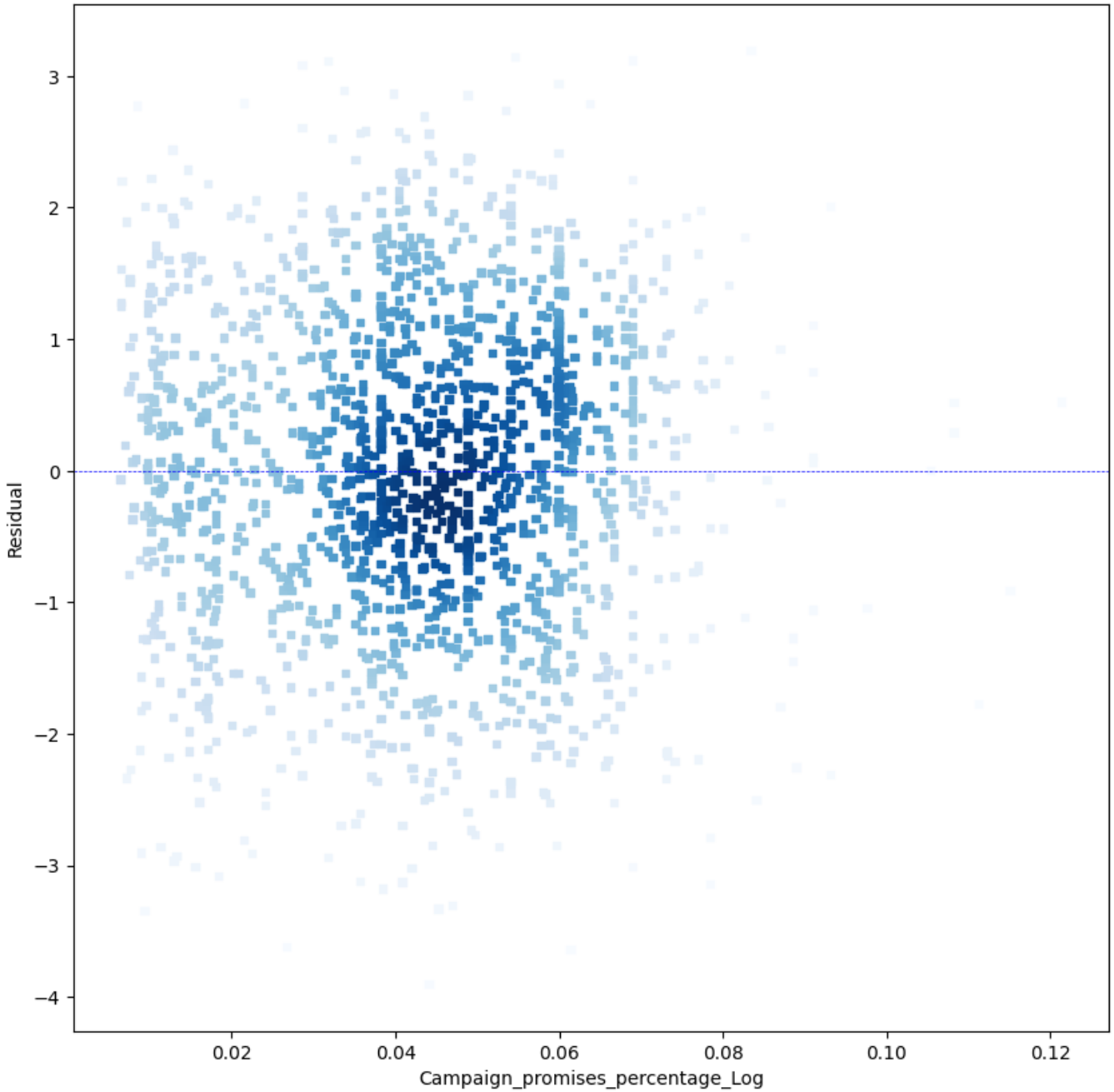
New robostness test 1 --- 6 Jun

In [129…
```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    "Campaign_frequency", "Campaign_promises_percentage_Log", "Default_Contribution_Log",
                    'Campaign_promises_percentage_Log_X_Default_Contribution_Log']
X = extract_data[Model1_variables]
X = sm.add_constant(X) # adding a constant
Y = extract_data['Distinct_Donors_Log']
model_robust_test = Linear_Regression(X, Y)
y_pre_robust_test = model_robust_test.predict(X)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Distinct_Donors_Log   R-squared:                       0.352
Model:                             OLS   Adj. R-squared:                  0.350
Method:                  Least Squares   F-statistic:                     269.6
Date:                 Sat, 24 Jun 2023   Prob (F-statistic):               0.00
Time:                         15:51:50   Log-Likelihood:                -10799.
No. Observations:                6972   AIC:                         2.163e+04
Df Residuals:                    6957   BIC:                         2.173e+04
Df Model:                          14
Covariance Type:            nonrobust
================================================================================================================================
                                                                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------------------------
const                                                           -1.4282      0.122    -11.678      0.000      -1.668      -1.188
Campaign_Goal_Log                                                0.3944      0.009     41.805      0.000       0.376       0.413
Tax_duction_status                                              -0.0453      0.075     -0.608      0.543      -0.191       0.101
Campaign_Duration                                                0.0014      0.000     10.270      0.000       0.001       0.002
Campaign_Video                                                   0.0019      0.029      0.066      0.947      -0.055       0.059
Number_of_images                                                 0.0224      0.009      2.468      0.014       0.005       0.040
Words_of_campaigns                                               0.0028      0.000     13.135      0.000       0.002       0.003
Creator_Type                                                     0.0227      0.020      1.151      0.250      -0.016       0.061
Financial_Size                                                   0.0923      0.014      6.767      0.000       0.066       0.119
Sector_type                                                     -0.0239      0.011     -2.126      0.034      -0.046      -0.002
Campaign_Start_Year_category                                     0.0447      0.011      4.215      0.000       0.024       0.066
Campaign_frequency                                              -0.0059      0.001    -10.071      0.000      -0.007      -0.005
Campaign_promises_percentage_Log                                 8.9111      3.394      2.625      0.009       2.257      15.565
Default_Contribution_Log                                        -0.1122      0.009    -12.065      0.000      -0.130      -0.094
Campaign_promises_percentage_Log_X_Default_Contribution_Log     -1.2615      0.694     -1.818      0.069      -2.622       0.099
==============================================================================
Omnibus:                       40.021   Durbin-Watson:                   1.584
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               40.435
Skew:                          -0.181   Prob(JB):                     1.66e-09
Kurtosis:                       2.908   Cond. No.                     5.05e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.05e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [129…

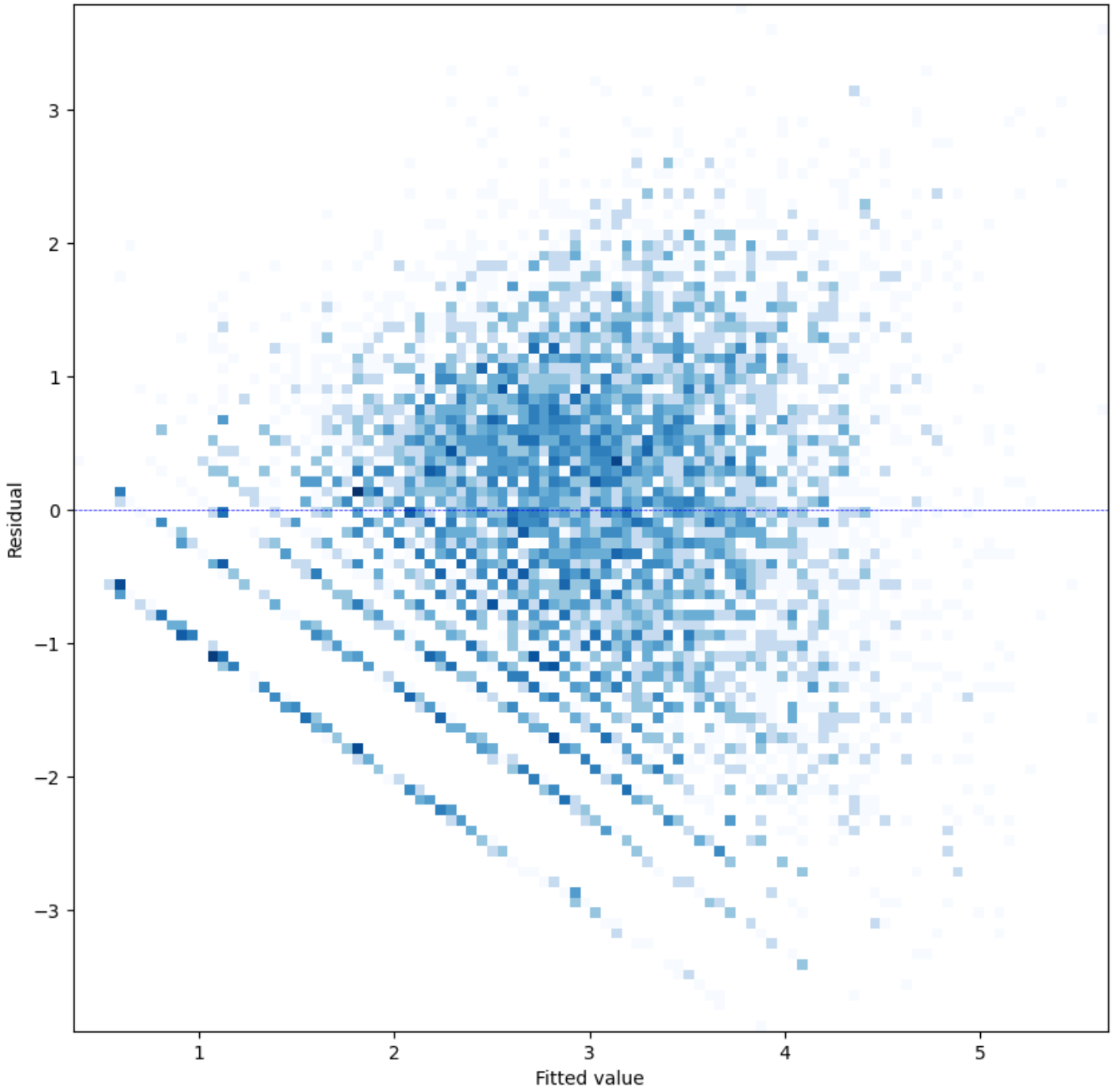
```
x = 'Campaign_promises_percentage_Log'
y = np.array(model_robust_test.resid)
residule_plot(x, y, extract_data, remove_zero = 1)
```



In [129…

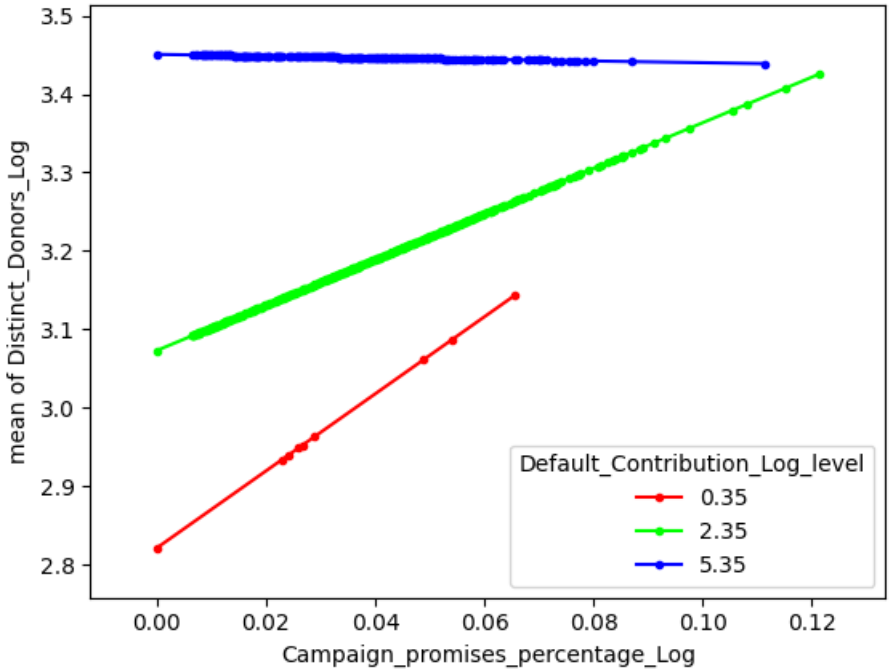
```
from matplotlib.colors import LogNorm
y_pred = model_robust_test.predict(X)
x = np.array(model_robust_test.resid)
y =np.array(y_pred)
xy = np.vstack([x,y])
z = gaussian_kde(xy)(xy)
```

```
idx = z.argsort()
x = x[idx]
y = y[idx]
z = z[idx]
# plt.scatter(y, x, marker=',', c=z, s=1, cmap='Blues' )
plt.figure(figsize=(10,10))

plt.hist2d(y, x, bins=100, norm=LogNorm(), cmap='Blues' )

plt.axhline(y=0, color='b', linestyle='--', lw=0.5)
plt.xlabel("Fitted value")
plt.ylabel("Residual")
#plt.savefig('Robust_test1_residule.png', dpi=2000)
plt.show()
```



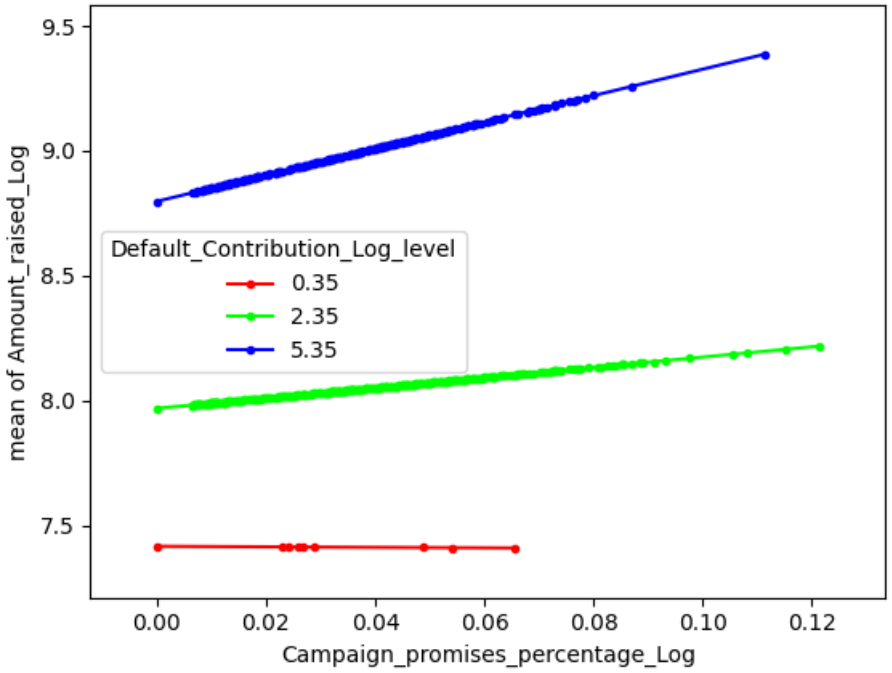
In [129…

```
DV = 'Distinct_Donors_Log'
interaction_variable1 = 'Campaign_promises_percentage_Log'
interaction_variable2_catergial = 'Default_Contribution_Log_level'
Interaction_plot(DV, interaction_variable1, interaction_variable2_catergial, extract_data)
```



In [129…

```
DV = 'Amount_raised_Log'
interaction_variable1 = 'Campaign_promises_percentage_Log'
interaction_variable2_catergial = 'Default_Contribution_Log_level'
Interaction_plot(DV, interaction_variable1, interaction_variable2_catergial, extract_data)
```



In [130…

```
'''
Consumption = [51, 52, 53, 54, 56, 57, 55, 56, 58, 59, 62, 63]
Gender  = ["Male", "Male", "Male", "Male", "Male", "Male", "Female", "Female", "Female", "Female", "Female", "Female"]
Income = [80, 80, 90, 90, 100, 100, 80, 80, 90, 90, 100, 100]

import pandas as pd
df6 = pd.DataFrame(
 {
   "Consumption": Consumption
 , "Gender": Gender
 , "Income": Income
 }
 )

from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm
Reg6 = ols(formula = "Consumption ~ Gender + Income", data = df6)
Fit6 = Reg6.fit()

Reg7 = ols(formula = "Consumption ~ Gender*Income", data = df6)
Fit7 = Reg7.fit()

from statsmodels.graphics.factorplots import interaction_plot
fig = interaction_plot(Income, Gender, Consumption,
             colors=['black','gray'], markers=['D','^'], ylabel='Consumption', xlabel='Income')
fig = interaction_plot(Income, Gender, Fit6.fittedvalues,
             colors=['red','blue'], markers=['D','^'], ylabel='Consumption', xlabel='Income')
fig = interaction_plot(Income, Gender, Fit7.fittedvalues,
             colors=['green','orange'], markers=['D','^'], ylabel='Consumption', xlabel='Income')
```

```
import matplotlib.pyplot as plt
plt.show()
'''
```

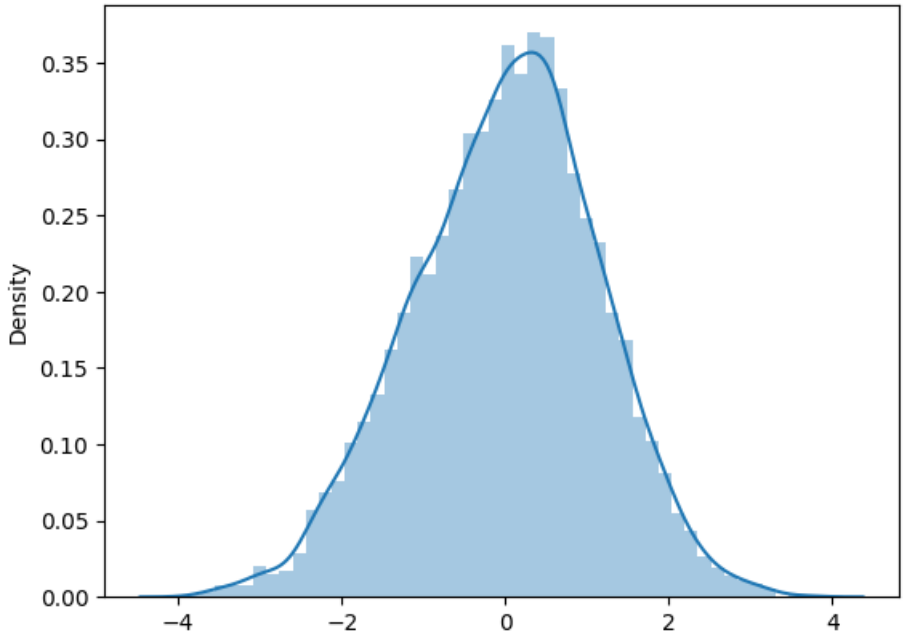
Out[1300]:
```
'\nConsumption = [51, 52, 53, 54, 56, 57, 55, 56, 58, 59, 62, 63]\nGender  = ["Male", "Male", "Male", "Male", "Male", "Male", "Female", "Female", "Female", "Female", "Female", "Female"]\nIncome = [80, 80, 90, 90, 100, 100, 80, 80, 90, 90, 100, 100]\n\nimport pandas as pd\ndf6 = pd.DataFrame(\n {\n   "Consumption": Consumption\n , "Gender": Gender\n , "Income": Income\n }\n )\n\nfrom statsmodels.formula.api import ols\nfrom statsmodels.stats.anova import anova_lm\nReg6 = ols(formula = "Consumption ~ Gender + Income", data = df6)\nFit6 = Reg6.fit()\n\nReg7 = ols(formula = "Consumption ~ Gender*Income", data = df6)\nFit7 = Reg7.fit()\n\nfrom statsmodels.graphics.factorplots import interaction_plot\nfig = interaction_plot(Income, Gender, Consumption,\n             colors=[\'black\',\'gray\'], markers=[\'D\',\'^\'], ylabel=\'Consumption\', xlabel=\'Income\')\nfig = interaction_plot(Income, Gender, Fit6.fittedvalues,\n             colors=[\'red\',\'blue\'], markers=[\'D\',\'^\'], ylabel=\'Consumption\', xlabel=\'Income\')\nfig = interaction_plot(Income, Gender, Fit7.fittedvalues,\n             colors=[\'green\',\'orange\'], markers=[\'D\',\'^\'], ylabel=\'Consumption\', xlabel=\'Income\')\n\nimport matplotlib.pyplot as plt\nplt.show()\n'
```

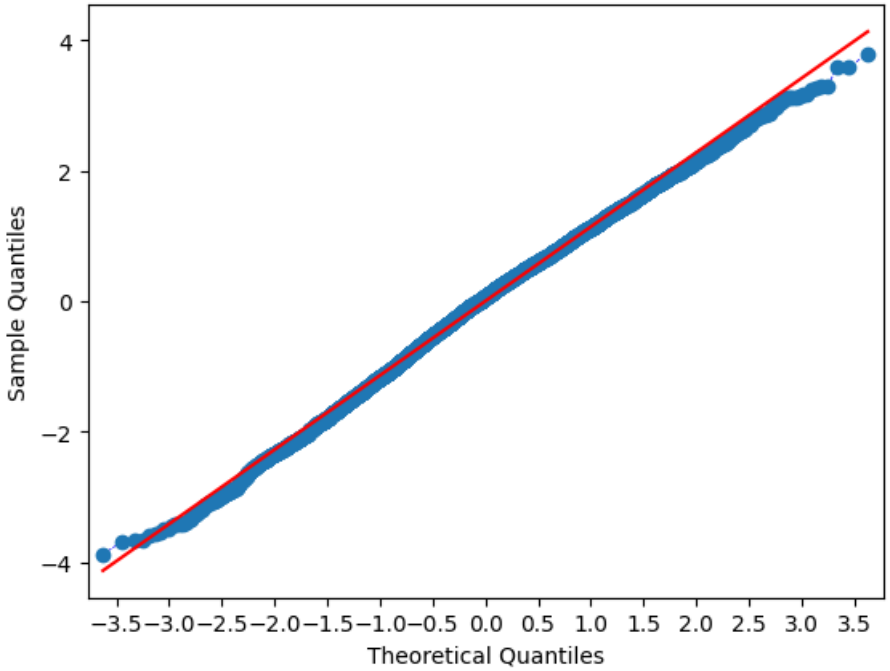
In [130…
```
# Residual Standard Error of the model
print("Residual Standard Error: "+ str(model1.scale) )
print("Predicted R2 : "+ str(predicted_r2(y_true, y_pred, xs)) )
```

```
Residual Standard Error: 1.2999851842415353
Predicted R2 : -6.910873667918411
```

QQ plot

In [130…
```
ax = plt.axes()
res = model1.resid # 获取了构造的模型的残差，获取了数据
# 主要调用方法
sns.distplot(res)
plt.show()
probplot = sm.ProbPlot(res) # 实例probplt
probplot.qqplot(line='s', linestyle='--', lw=0.5) # 调用函数
x_major_locator=MultipleLocator(0.5)
ax = plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.show()
```





New robostness test 2 --- 6 Jun

In [130…

```
'''
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
```

```
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    "Campaign_frequency", "Campaign_promises_percentage_Log", "Default_Contribution_Log",
                    'Campaign_promises_percentage_Log_X_Default_Contribution_Log']

import statsmodels.formula.api as smf
X = extract_data[Model1_variables]
Y = extract_data['Campaign_success']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
print(results1)
y_true= Y
y_pred = model1.predict(X)
xs = X
'''
```

Out[1303]:
```
'\nModel1_variables = [\'Campaign_Goal_Log\', \'Tax_duction_status\', \'Campaign_Duration\',\n                    \'Campaign_Video\',  \'Number_of_images\',\'Words_of_campaigns\',\n                    "Creator_Type", \'Financial_Size\', "Sector_type", "Campaign_Start_Year_category",\n                    "Campaign_frequency", "Campaign_promises_percentage_Log", "Default_Contribution_Log",\n                    \'Campaign_promises_percentage_Log_X_Default_Contribution_Log\']\n\nimport statsmodels.formula.api as smf\nX = extract_data[Model1_variables] \nY = extract_data[\'Campaign_success\']\n# with statsmodels\nX = sm.add_constant(X) # adding a constant\nmodel1 = sm.OLS(Y, X).fit()\n\nresults1 = model1.summary()\n#predicts = model1._results\nprint(results1)\ny_true= Y\ny_pred = model1.predict(X)\nxs = X\n'
```

In [130…
```
# Residual Standard Error of the model
print("Residual Standard Error: "+ str(model1.scale) )
print("Predicted R2 : "+ str(predicted_r2(y_true, y_pred, xs)) )
```

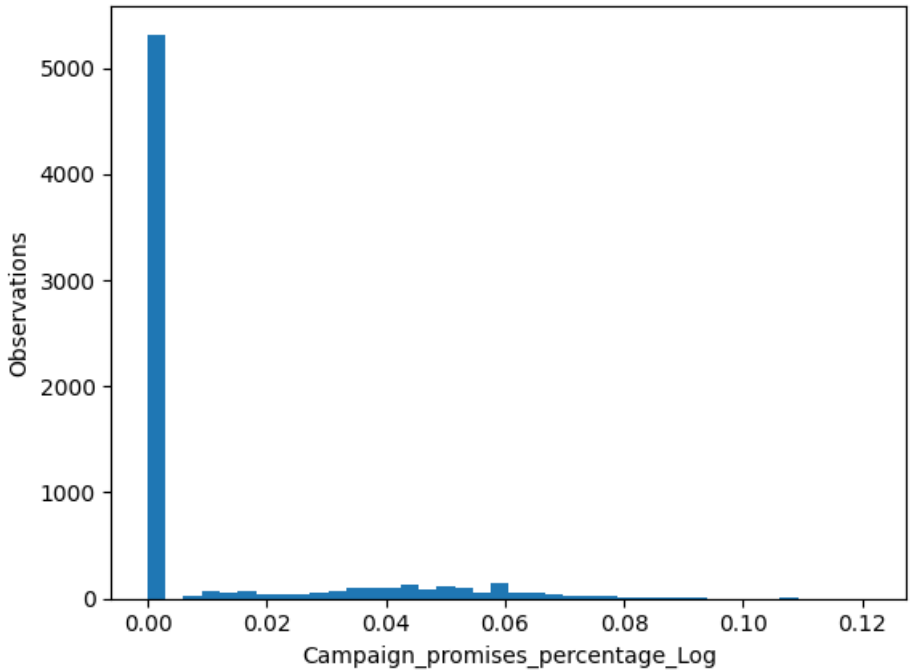
```
Residual Standard Error: 1.2999851842415353
Predicted R2 : -6.910873667918411
```

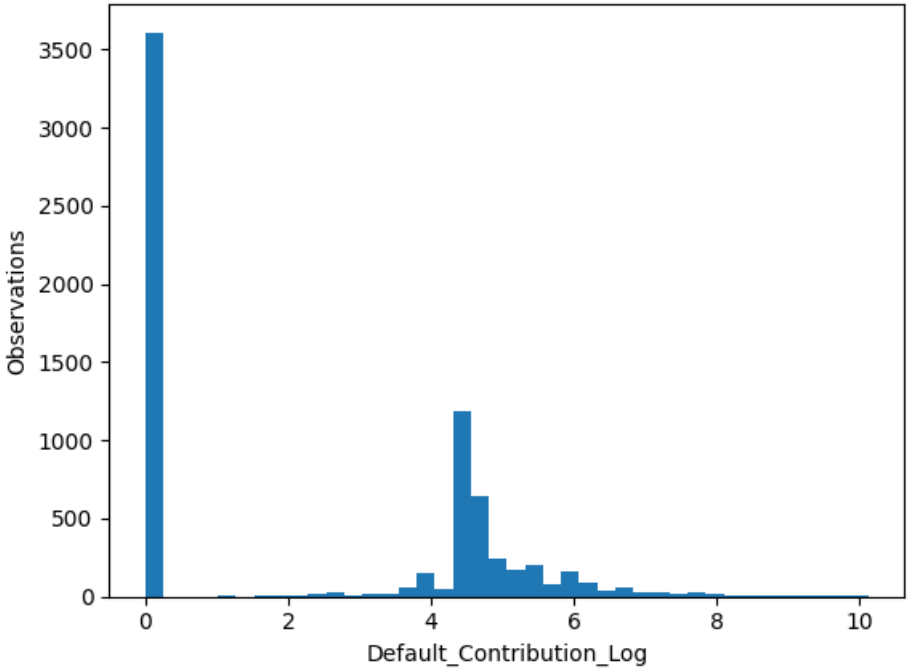
In [130…
```
extract_data.columns
```

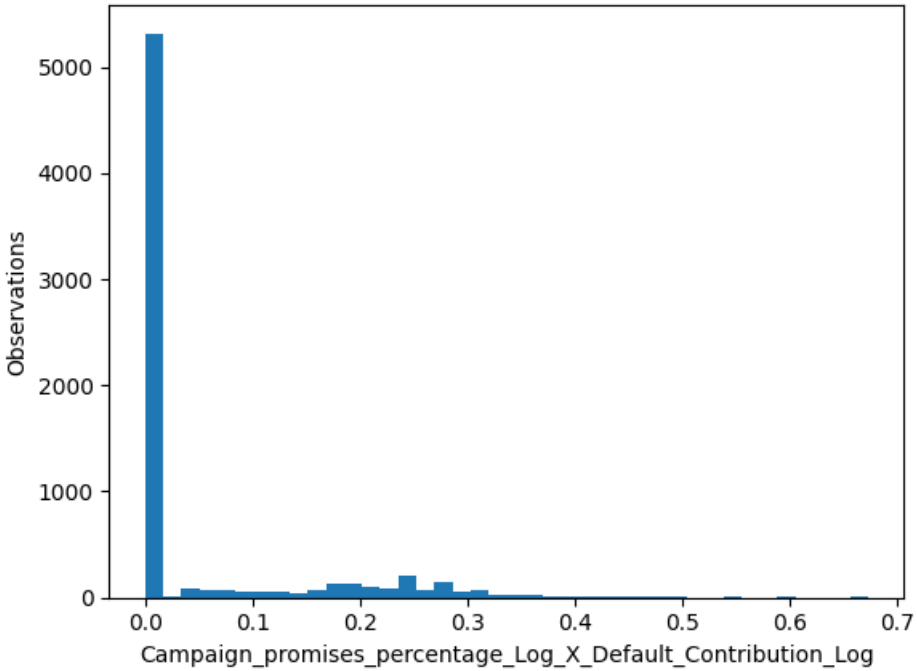
Out[1305]:
```
Index(['Campaign_Id', 'Campaign_Title_x', 'Receiving_NPO_Name',
       'Receiving_NPO_Id', 'NPO_Status', 'Campaign_frequency',
       'Public_Campaign_Access', 'Creator_Type', 'Creator_Id',
       'Campaign_Status',
       ...
       'Campaign_success', 'Campaign_success_Log',
       'Campaign_promises_percentage_X_Default_Contribution_Log',
       'Campaign_promises_percentage_Log_level',
       'Campaign_promises_percentage_Log',
       'Campaign_promises_percentage_Log_X_Default_Contribution_Log',
       'Default_Contribution_Log_level', 'Campaign_promises_categoty',
       'Custom_amount_categoty', 'Sqr_Campaign_promises_percentage'],
      dtype='object', length=103)
```

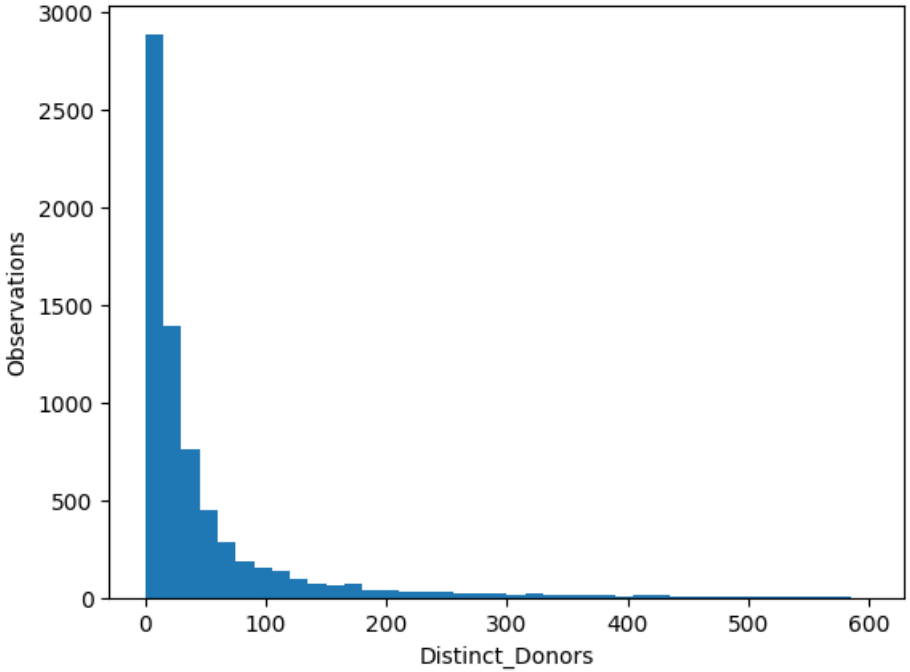
In [130…
```
#plt.figure(figsize=(20,20))

#plt.subplot(441)
plt.hist(extract_data['Campaign_promises_percentage_Log'], bins=40)
plt.xlabel('Campaign_promises_percentage_Log'); plt.ylabel('Observations')
plt.show()

#plt.subplot(442)
```

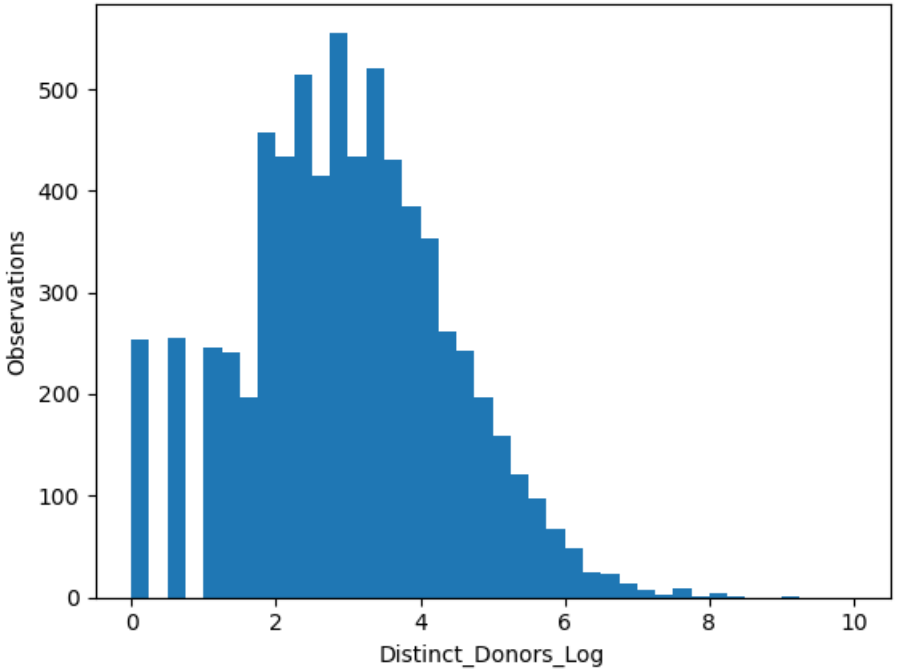
```
plt.hist(extract_data['Default_Contribution_Log'], bins=40 )
plt.xlabel('Default_Contribution_Log'); plt.ylabel('Observations')
plt.show()

#plt.subplot(449)
plt.hist(extract_data['Campaign_promises_percentage_Log_X_Default_Contribution_Log'], bins=40)
plt.xlabel('Campaign_promises_percentage_Log_X_Default_Contribution_Log'); plt.ylabel('Observations')
plt.show()


plt.hist(extract_data['Distinct_Donors'], bins=40, range=[0, 600])
plt.xlabel('Distinct_Donors'); plt.ylabel('Observations')
plt.show()

plt.hist(extract_data['Distinct_Donors_Log'], bins=40,  range=[0, 10])
plt.xlabel('Distinct_Donors_Log'); plt.ylabel('Observations')
plt.show()


plt.hist(extract_data['Campaign_success'], bins=40, range=[0, 30])
plt.xlabel('Campaign_success'); plt.ylabel('Observations')
plt.show()
```

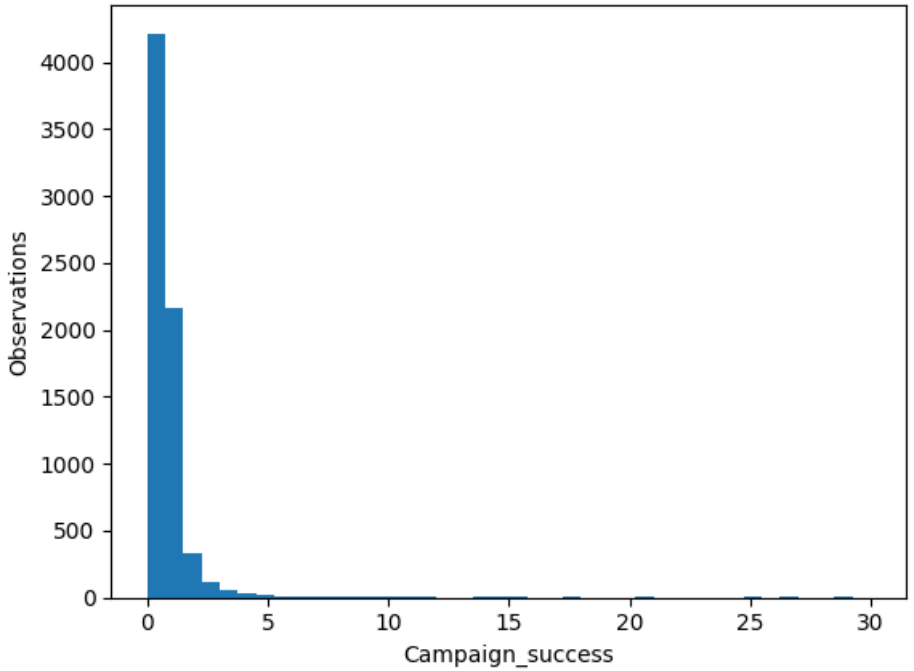



Observations
3500
3000
2500
2000
1500
1000
500
0
0
2
4
6
8
10
Default_Contribution_Log

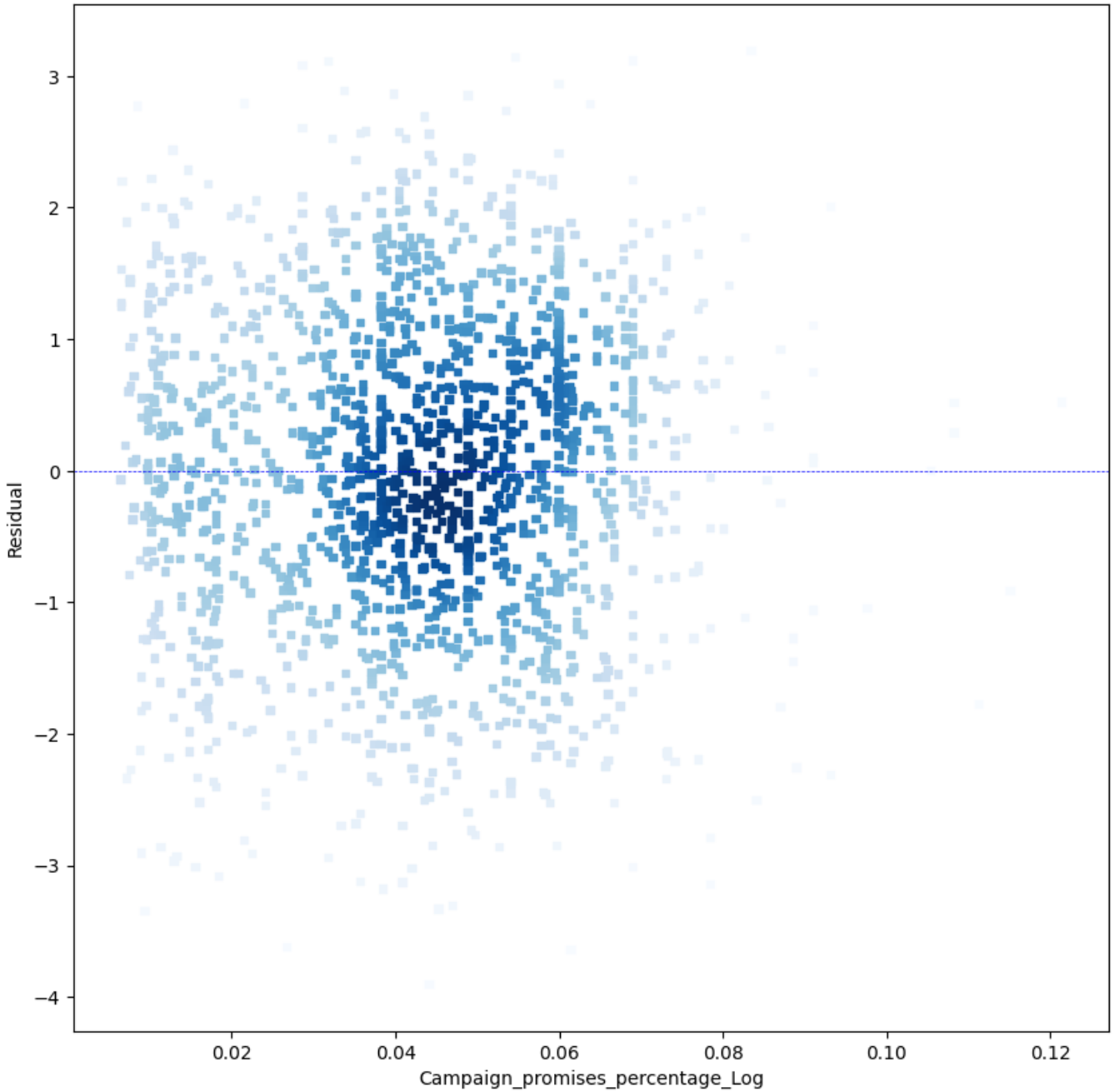

Observations
5000
4000
3000
2000
1000
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Campaign_promises_percentage_Log_X_Default_Contribution_Log


3000
2500
2000
1500
1000
500
0
Observations
0
100
200
300
400
500
600
Distinct_Donors

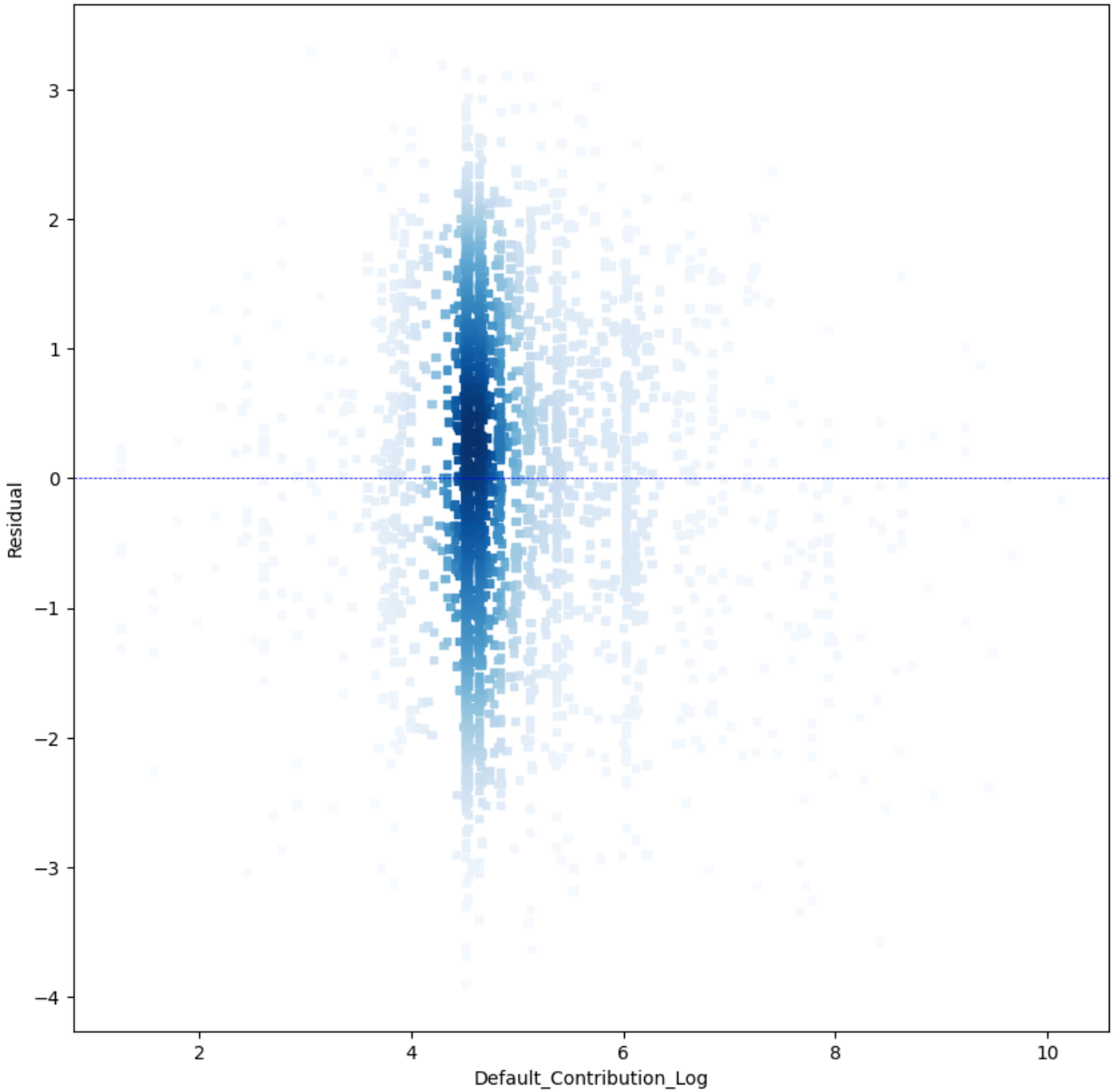

500
400
300
200
100
0
Observations
0
2
4
6
8
10
Distinct_Donors_Log



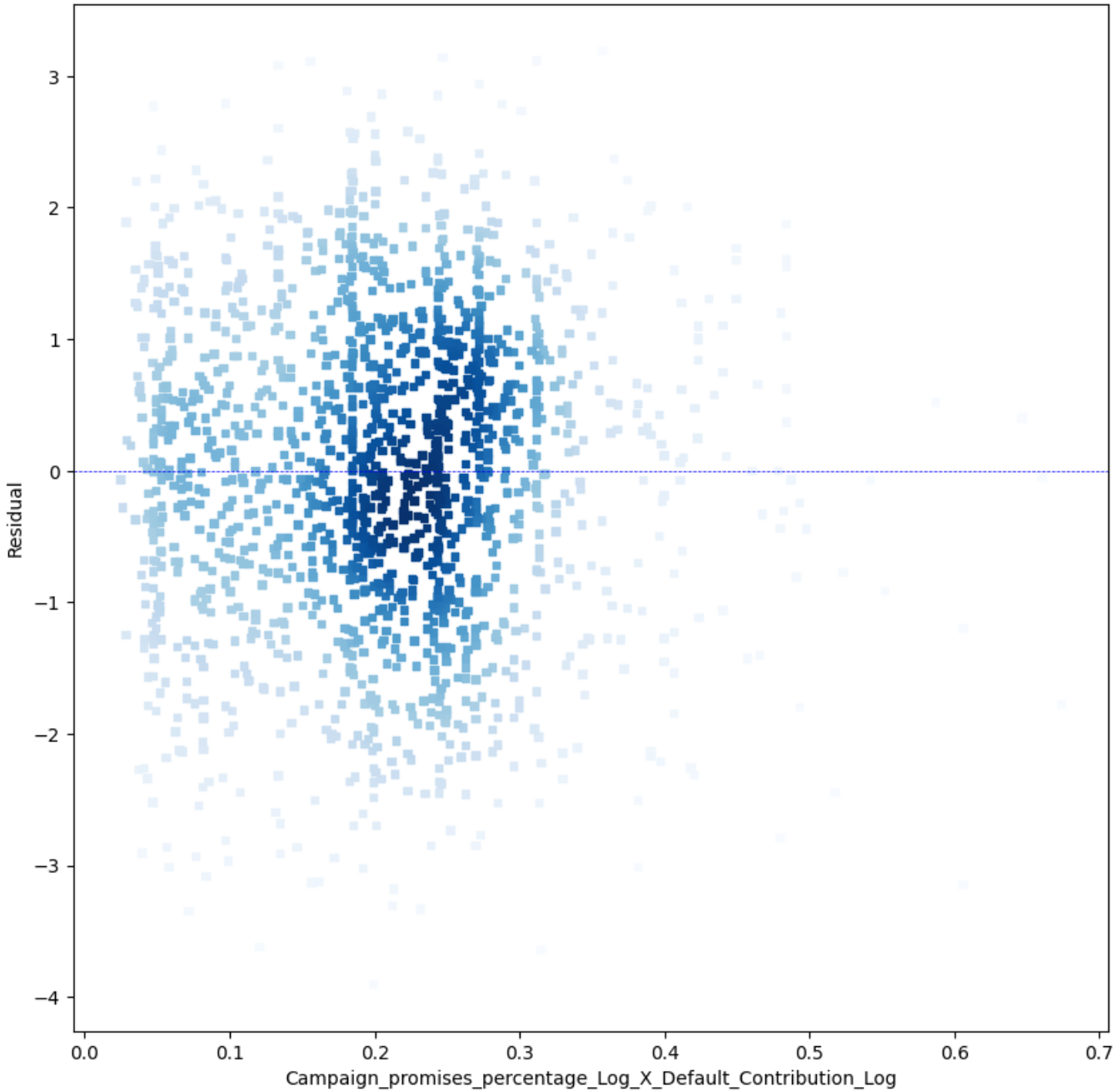
In [ ]:
```
x = 'Campaign_promises_percentage_Log'
y = model1.resid
residule_plot(x, y, extract_data, remove_zero = 1)
```



In [ ]:
```
x = 'Default_Contribution_Log'
y = model1.resid
residule_plot(x, y, extract_data, remove_zero = 1)
```



In [ ]:
```
x = 'Campaign_promises_percentage_Log_X_Default_Contribution_Log'
y = model1.resid
residule_plot(x, y, extract_data, remove_zero = 1)
```



Stop here

In [ ]:
```
stop here
```

```
  Cell In [473], line 1
    stop here
         ^
SyntaxError: invalid syntax
```

Other codes used in past

In [ ]:
```
extract_data.shape
```

Out[ ]:
```
(6972, 114)
```

In [ ]:
```
short_list_campaigns = extract_data[extract_data['Campaign_promises_categoty'] == 1 ]
```

In [ ]:
```
short_list_campaigns.shape
```

Out[ ]:
```
(1651, 116)
```

In [ ]:
```
short_list_campaigns = short_list_campaigns[short_list_campaigns['Custom_amount_categoty'] == 1 ]
```

In [ ]:
```
short_list_campaigns.shape
```

Out[ ]:
```
(1651, 116)
```

In [ ]:
```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    "Campaign_frequency", "Campaign_promises", "Default_Contribution_Log"]

import statsmodels.formula.api as smf
X = short_list_campaigns[Model1_variables]
Y = short_list_campaigns['Amount_raised_Log']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
print(results1)
y_true= Y
y_pred = model1.predict(X)
xs = X
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Log_Actual_Donation_Amount   R-squared:                       0.428
Model:                                    OLS   Adj. R-squared:                  0.424
Method:                         Least Squares   F-statistic:                     102.1
Date:                        Mon, 05 Jun 2023   Prob (F-statistic):          1.06e-188
Time:                                16:58:45   Log-Likelihood:                -2852.8
No. Observations:                        1651   AIC:                             5732.
Df Residuals:                            1638   BIC:                             5802.
Df Model:                                  12
Covariance Type:                    nonrobust
=========================================================================================================
                                            coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------------------------
const                                     0.0556      0.335      0.166      0.868      -0.601       0.713
Log_Campaign_Goal                         0.5654      0.024     23.325      0.000       0.518       0.613
NPO_Ipc_Status_For_Tax_Deductibility      0.1455      0.149      0.978      0.328      -0.146       0.437
Campaign_Duration                         0.0011      0.000      3.279      0.001       0.000       0.002
Campaign_Video                            0.1496      0.072      2.064      0.039       0.007       0.292
Campaign_Image_Number                     0.0349      0.023      1.520      0.129      -0.010       0.080
Number_of_words_describing_campaign       0.0022      0.001      3.843      0.000       0.001       0.003
Creator_Type                           1.094e-17    1.8e-17      0.609      0.543   -2.43e-17    4.62e-17
Scale_type                                0.1202      0.032      3.813      0.000       0.058       0.182
Sector_type                              -0.0982      0.027     -3.651      0.000      -0.151      -0.045
Campaign_Start_Year_category              0.1713      0.026      6.490      0.000       0.120       0.223
Campaign_frequency                       -0.0046      0.005     -0.962      0.336      -0.014       0.005
Future_tense                              0.0475      0.020      2.359      0.018       0.008       0.087
Log_avg_custom_amount                     0.1290      0.036      3.550      0.000       0.058       0.200
==============================================================================
Omnibus:                       85.188   Durbin-Watson:                   1.557
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               76.384
Skew:                          -0.463   Prob(JB):                     2.59e-17
Kurtosis:                       2.498   Cond. No.                     2.94e+18
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 9.34e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [ ]:
```
# Residual Standard Error of the model
print("Residual Standard Error: "+ str(model1.scale) )
print("Predicted R2 : "+ str(predicted_r2(y_true, y_pred, xs)) )
```

```
Residual Standard Error: 1.8894558537607167
Predicted R2 : 0.41329028876287266
```

进行Box-Cox变换

In [ ]:
```
#进行Box-Cox变换
#box-cox
#stats.boxcox(data1)[0]
'''
extract_data['BC_Campaign_promises'] = extract_data['Campaign_promises']
extract_data['BC_Custom_amount'] = extract_data['Default_Contribution']
```

```
for index, row in extract_data.iterrows():
    if extract_data.loc[index,'BC_Campaign_promises'] == 0:
        extract_data.loc[index, "BC_Campaign_promises"] = 0.0001
    if extract_data.loc[index,'BC_Custom_amount'] == 0:
        extract_data.loc[index, "BC_Custom_amount"] = 0.0001
'''
```

In [ ]:
```
extract_data["BC_Campaign_promises"], best_lambda = stats.boxcox(extract_data["Campaign_promises"] + 0.00001)
extract_data["BC_Custom_amount"], best_lambda = stats.boxcox(extract_data["Default_Contribution"] + 0.00001)
```

In [ ]:
```
Model1_variables = ['Campaign_Goal_Log', 'Tax_duction_status', 'Campaign_Duration',
                     'Campaign_Video',  'Number_of_images','Words_of_campaigns',
                    "Creator_Type", 'Financial_Size', "Sector_type", "Campaign_Start_Year_category",
                    "Campaign_frequency", "BC_Campaign_promises", "BC_Custom_amount"]

import statsmodels.formula.api as smf
X = extract_data[Model1_variables]
Y = extract_data['Amount_raised_Log']
# with statsmodels
X = sm.add_constant(X) # adding a constant
model1 = sm.OLS(Y, X).fit()

results1 = model1.summary()
#predicts = model1._results
print(results1)
y_true= Y
y_pred = model1.predict(X)
xs = X
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Log_Actual_Donation_Amount   R-squared:                       0.488
Model:                                    OLS   Adj. R-squared:                  0.487
Method:                         Least Squares   F-statistic:                     510.1
Date:                        Mon, 05 Jun 2023   Prob (F-statistic):               0.00
Time:                                15:55:05   Log-Likelihood:                -11556.
No. Observations:                        6972   AIC:                         2.314e+04
Df Residuals:                            6958   BIC:                         2.324e+04
Df Model:                                  13
Covariance Type:                    nonrobust
==============================================================================================================
                                         coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------
const                                  0.9048      0.141      6.425      0.000       0.629       1.181
Log_Campaign_Goal                      0.6174      0.010     59.153      0.000       0.597       0.638
NPO_Ipc_Status_For_Tax_Deductibility   0.1237      0.083      1.490      0.136      -0.039       0.287
Campaign_Duration                      0.0012      0.000      7.489      0.000       0.001       0.001
Campaign_Video                         0.0178      0.033      0.545      0.586      -0.046       0.082
Campaign_Image_Number                  0.0132      0.010      1.306      0.192      -0.007       0.033
Number_of_words_describing_campaign    0.0026      0.000     10.672      0.000       0.002       0.003
Creator_Type                           0.0672      0.022      2.997      0.003       0.023       0.111
Scale_type                             0.0923      0.015      6.073      0.000       0.063       0.122
Sector_type                           -0.0599      0.013     -4.775      0.000      -0.084      -0.035
Campaign_Start_Year_category           0.0930      0.012      7.861      0.000       0.070       0.116
Campaign_frequency                    -0.0066      0.001    -10.182      0.000      -0.008      -0.005
BC_Future_tense                        0.0011      0.000      2.695      0.007       0.000       0.002
BC_Custom_amount                      -0.0266      0.003     -8.474      0.000      -0.033      -0.020
==============================================================================
Omnibus:                      298.901   Durbin-Watson:                   1.626
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              327.567
Skew:                          -0.517   Prob(JB):                     7.41e-72
Kurtosis:                       2.755   Cond. No.                     2.01e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.01e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [ ]:
```
# Residual Standard Error of the model
print("Residual Standard Error: "+ str(model1.scale) )
print("Predicted R2 : "+ str(predicted_r2(y_true, y_pred, xs)) )
```

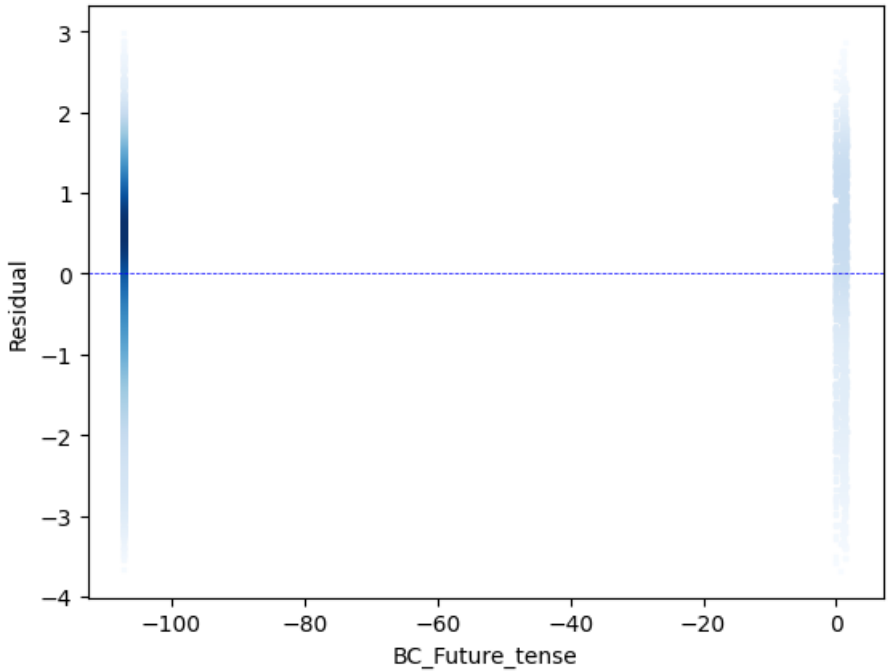
```
Residual Standard Error: 1.6146826687097922
Predicted R2 : 0.4857616243507188
```

In [ ]:
```
extract_data['BC_Campaign_promises'].unique()
```

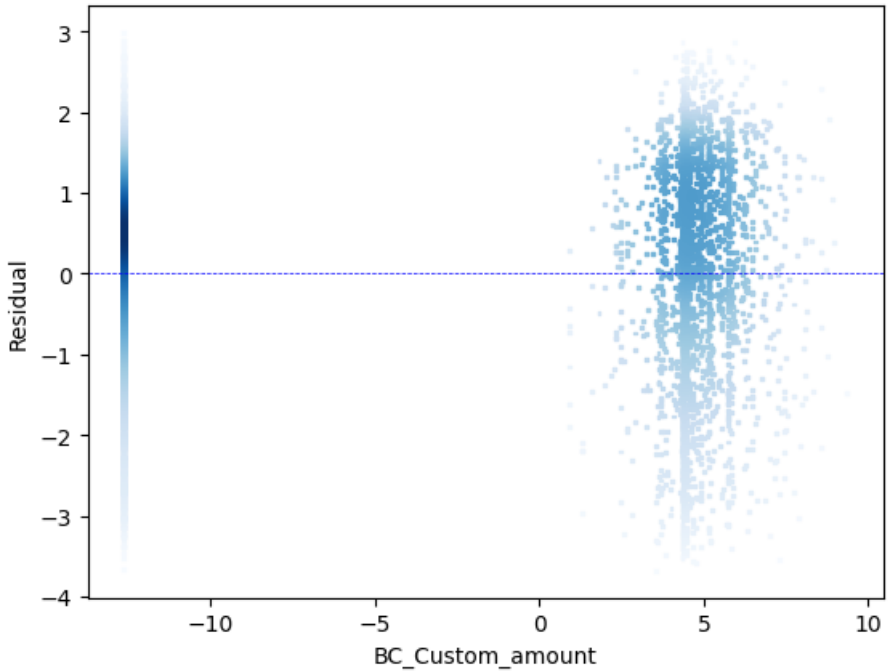
Out[ ]:
```
array([ 9.99993472e-06, -1.07155770e+02,  1.37964635e+00,  1.60017509e+00,
        6.24644918e-01,  9.33178581e-01,  1.13002046e+00,  1.27121831e+00,
        1.46672268e+00,  1.53890544e+00,  1.65314449e+00])
```

In [ ]:
```
IV1 = extract_data['BC_Campaign_promises']
#Interaction_v = extract_data.Campaign_promises_percentage_X_Default_Contribution_Log
x = np.array(model1.resid)
y =np.array(IV1)
xy = np.vstack([x,y])
```

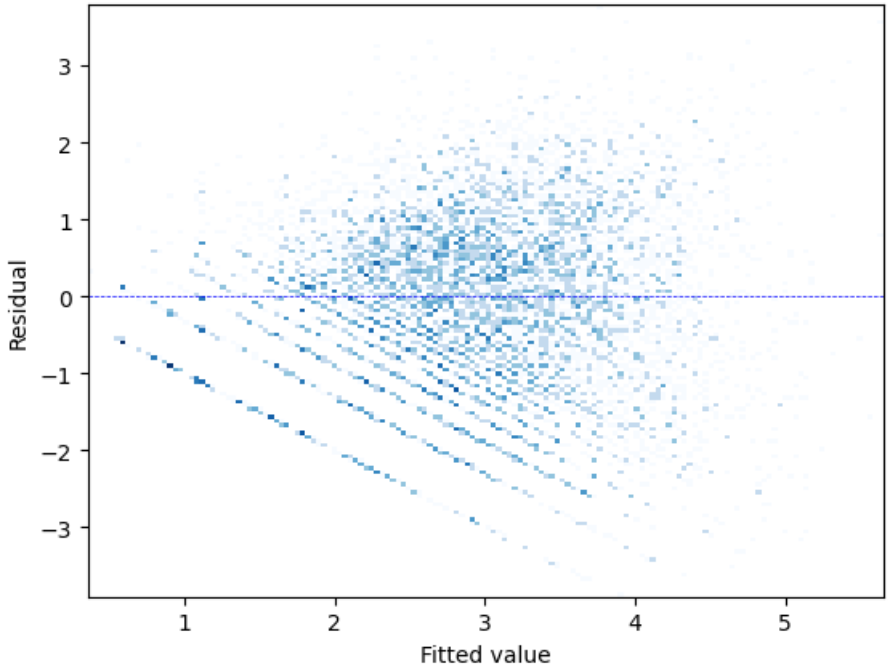
```
z = gaussian_kde(xy)(xy)
idx = z.argsort()
x = x[idx]
y = y[idx]
z = z[idx]
plt.scatter(y, x, marker=',', c=z, s=2, cmap='Blues' )
#plt.hist2d(y, x, bins=100, norm=LogNorm(), cmap='Blues' )
plt.axhline(y=0, color='b', linestyle='--', lw=0.5)
plt.xlabel("BC_Campaign_promises")
#plt.xlim((-0.5, 3))
plt.ylabel("Residual")
plt.show()
```



In [ ]:
```
IV2 = extract_data['BC_Custom_amount']
#Interaction_v = extract_data.Campaign_promises_percentage_X_Default_Contribution_Log
x = np.array(model1.resid)
y =np.array(IV2)
xy = np.vstack([x,y])
z = gaussian_kde(xy)(xy)
idx = z.argsort()
x = x[idx]
y = y[idx]
z = z[idx]
plt.scatter(y, x, marker=',', c=z, s=2, cmap='Blues' )
#plt.hist2d(y, x, bins=100, norm=LogNorm(), cmap='Blues' )
plt.axhline(y=0, color='b', linestyle='--', lw=0.5)
plt.xlabel("BC_Custom_amount")
#plt.xlim((-0.5, 11))
plt.ylabel("Residual")
plt.show()
```



In [ ]:

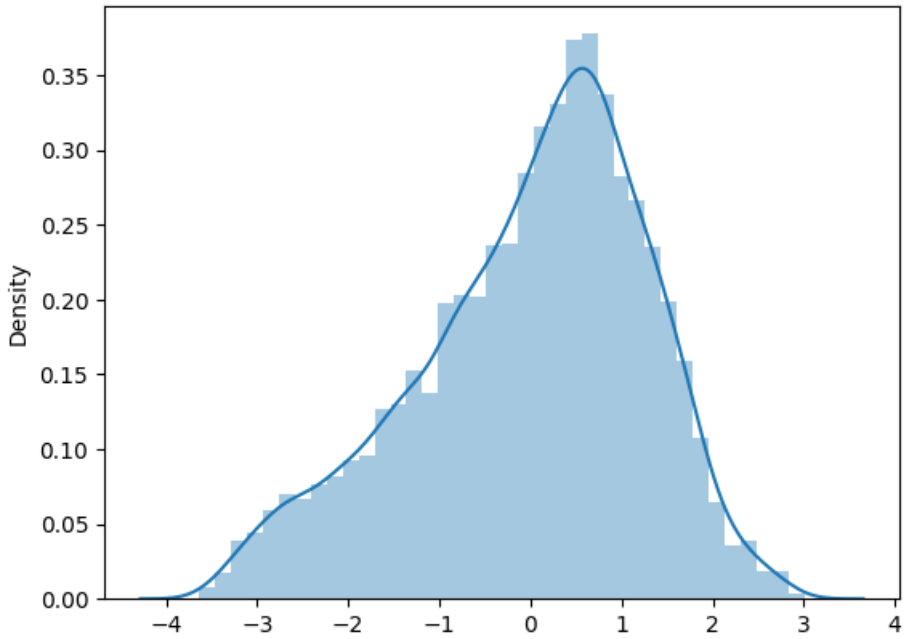


In [ ]:
```
min(extract_data["Campaign_promises"])
```

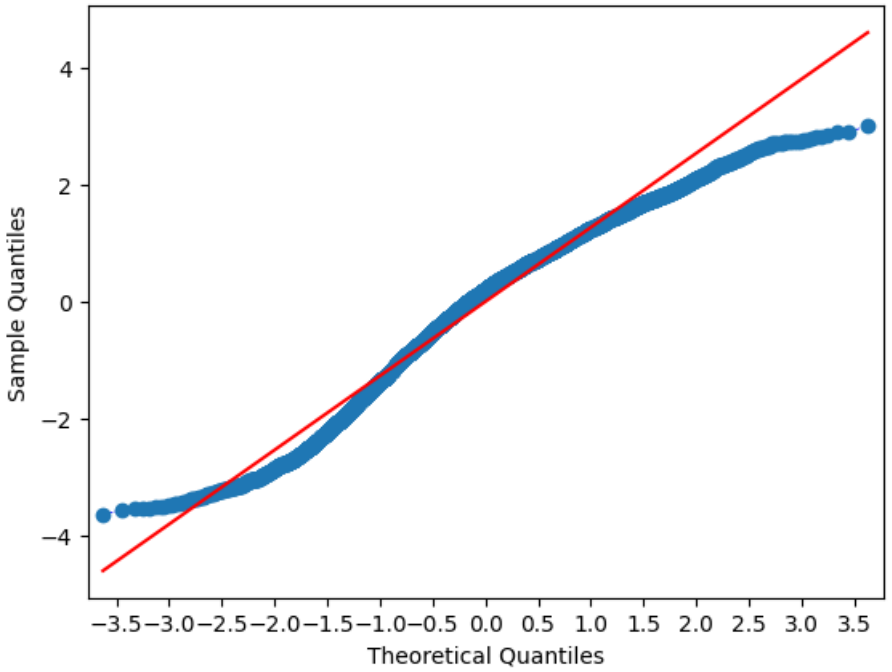
Out[ ]: 0

QQ plot

In [ ]:
```
import statsmodels.api as sm
from matplotlib import pyplot as plt
import matplotlib.ticker as ticker
# 样例1
ax = plt.axes()
res = model1.resid # 获取了构造的模型的残差，获取了数据
# 主要调用方法


sns.distplot(res)
plt.show()
```



In [ ]:
```
probplot = sm.ProbPlot(res) # 实例probplot
probplot.qqplot(line='s', linestyle='--', lw=0.5) # 调用函数
x_major_locator=MultipleLocator(0.5)
ax = plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.show()
```

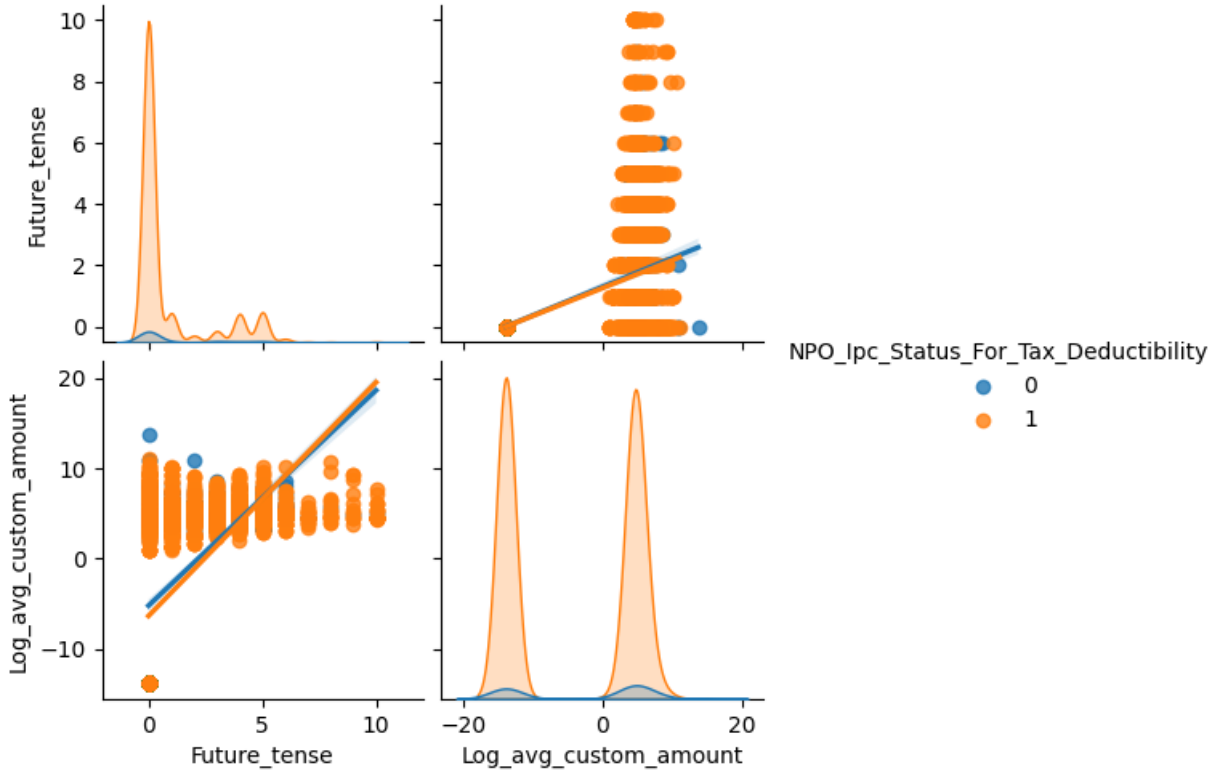


Add Campaign_promises(not Campaign_promises_percentage) result

In [ ]:
```
import pandas as pd

df_X = extract_data[['Campaign_promises_percentage','Amount_raised_Log','Campaign_frequency']]
from pyprocessmacro import Process
p = Process(data=df_X, model=1, x="Campaign_promises_percentage", y="Amount_raised_Log",m=["Campaign_frequency"],controls_in="all_to_y")
p.summary()

import matplotlib.pyplot as plt
g = p.plot_conditional_direct_effects(x = 'Campaign_frequency')
plt.show()
```

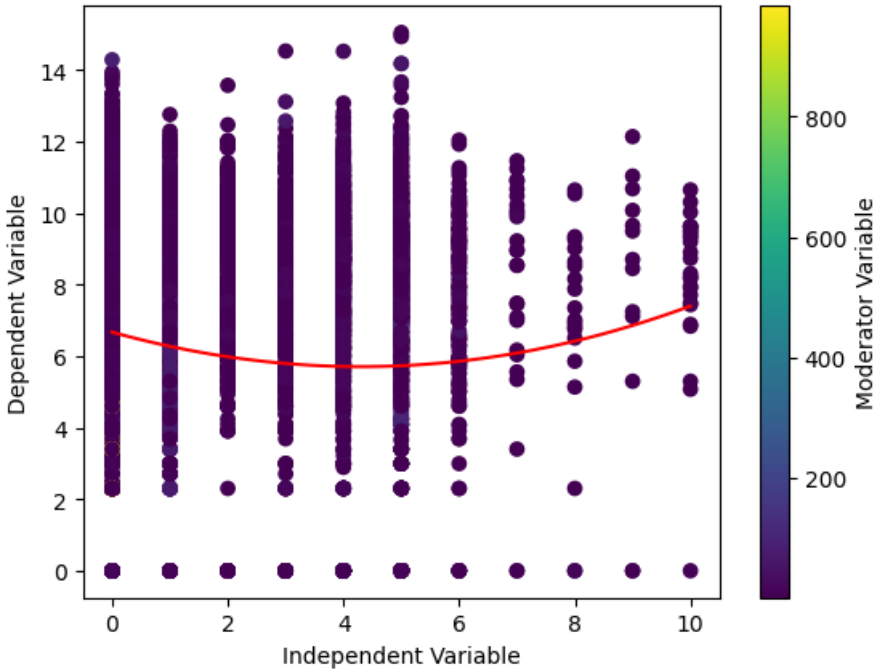
In [ ]:
```
import seaborn
seaborn.pairplot(extract_data, vars=['Campaign_promises','Default_Contribution_Log'],   kind='reg', hue='Tax_duction_status')
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x291b5b7c0>
```



In [ ]:

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
df = extract_data
extract_data['interaction_term'] = df['Campaign_promises'] * df['Campaign_frequency']\

X = sm.add_constant(df[['Campaign_promises', 'Campaign_frequency', 'interaction_term']])
model = sm.OLS(df['Amount_raised_Log'], X).fit()

independent_vals = np.linspace(df['Campaign_promises'].min(), df['Campaign_promises'].max(), 100)
moderator_vals = np.linspace(df['Campaign_frequency'].min(), df['Campaign_frequency'].max(), 100)
interaction_vals = independent_vals * moderator_vals
new_data = pd.DataFrame({'Campaign_promises': independent_vals, 'Campaign_frequency': moderator_vals, 'interaction_term': interaction_vals})
new_data = sm.add_constant(new_data)
new_data['predicted_dependent_variable'] = model.predict(new_data)

fig, ax = plt.subplots()
scatter = ax.scatter(df['Campaign_promises'], df['Amount_raised_Log'], c=df['Campaign_frequency'], cmap='viridis')
line = ax.plot(new_data['Campaign_promises'], new_data['predicted_dependent_variable'], color='red')
ax.set_xlabel('Independent Variable')
ax.set_ylabel('Dependent Variable')
fig.colorbar(scatter, label='Moderator Variable')
plt.show()
```


Dependent Variable
Independent Variable
Moderator Variable
0
2
4
6
8
10
12
14
200
400
600
800