PROBLEM

Build a sophisticated Machine Learning model that predicts the probability percentage of marketing leads purchasing their product, based on information provided in the given dataset.

IMPORTING ALL THE NECESSARY LIBRARIES

```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
import plotly.express as pe
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
```

GETTING TO KNOW THE DATASET

```
df=pd.read_csv("marketing_leads.csv")
df.head()
```

	Deal_title	Lead_name	Industry	Deal_value	Weighted_amount	Date_of_creati
0	TitleM5DZY	Davis, Perkins and Bishop Inc	Restaurants	320506$	2067263.7$	2020-03-
1	TitleKIW18	Bender PLC LLC	Construction Services	39488$	240876.8$	2019-07-
2	TitleFXSDN	Carter-Henry and Sons	Hospitals/Clinics	359392$	2407926.4$	2019-07-
3	TitlePSK4Y	Garcia Ltd Ltd	Real Estate	76774$	468321.4$	2021-01-
4	Title904GV	Lee and Sons PLC	Financial Services	483896$	NaN	2019-05-

```
df2=pd.read_csv("marketing_leads1.csv")
df2.head()
```

	Deal_title	Lead_name	Industry	Deal_value	Weighted_amount	Date_of_creati
0	TitleAD16O	Bonilla Ltd Inc	Investment Bank/Brokerage	200988$	NaN	2020-04·
1	TitleOW6CR	Williams, Rogers and Roach PLC	Electronics	409961$	2541758.2$	2021-01·
2	TitleVVJQ5	Wood, Vaughn and Morales Ltd	Banks	434433$	3041031.0$	2020-07·
3	TitleUS8NA	Durham-Crawford Inc	Music	218952$	1521716.4$	2020-02·
4	Title5VGWW	Simpson, Duncan and Long LLC	Real Estate	392835$	2455218.75$	2020-10·

```
df.shape
```

```
(7007, 23)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7007 entries, 0 to 7006
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   Deal_title            7007 non-null   object
 1   Lead_name             7007 non-null   object
 2   Industry              7006 non-null   object
 3   Deal_value            6956 non-null   object
 4   Weighted_amount       6482 non-null   object
 5   Date_of_creation      7007 non-null   object
 6   Pitch                 7007 non-null   object
 7   Contact_no            7007 non-null   object
 8   Lead_revenue          7007 non-null   object
 9   Fund_category         7007 non-null   object
 10  Geography             6035 non-null   object
 11  Location              6996 non-null   object
 12  POC_name              6999 non-null   object
 13  Designation           7007 non-null   object
 14  Lead_POC_email        7007 non-null   object
 15  Hiring_candidate_role 7007 non-null   object
 16  Lead_source           7007 non-null   object
 17  Level_of_meeting      7007 non-null   object
 18  Last_lead_update      6374 non-null   object
 19  Internal_POC          7007 non-null   object
 20  Resource              6858 non-null   object
 21  Internal_rating       7007 non-null   int64 
```

```
 22  Success_probability    7007 non-null   float64
dtypes: float64(1), int64(1), object(21)
memory usage: 1.2+ MB
```

```
df.columns
```

```
Index(['Deal_title', 'Lead_name', 'Industry', 'Deal_value', 'Weighted_amount',
       'Date_of_creation', 'Pitch', 'Contact_no', 'Lead_revenue',
       'Fund_category', 'Geography', 'Location', 'POC_name', 'Designation',
       'Lead_POC_email', 'Hiring_candidate_role', 'Lead_source',
       'Level_of_meeting', 'Last_lead_update', 'Internal_POC', 'Resource',
       'Internal_rating', 'Success_probability'],
      dtype='object')
```

```
df.isnull().sum()
```

```
Deal_title                0
Lead_name                 0
Industry                  1
Deal_value               51
Weighted_amount         525
Date_of_creation          0
Pitch                     0
Contact_no                0
Lead_revenue              0
Fund_category             0
Geography               972
Location                 11
POC_name                  8
Designation               0
Lead_POC_email            0
Hiring_candidate_role     0
Lead_source               0
Level_of_meeting          0
Last_lead_update        633
Internal_POC              0
Resource                149
Internal_rating           0
Success_probability       0
dtype: int64
```

```
df.nunique()
```

```
Deal_title              7007
Lead_name               7007
Industry                 171
Deal_value              6907
Weighted_amount         6480
Date_of_creation         777
Pitch                      2
Contact_no              7007
Lead_revenue               3
Fund_category              4
Geography                  2
Location                 597
POC_name                5261
Designation               10
Lead_POC_email          7007
```

```
Hiring_candidate_role     639
Lead_source                 4
Level_of_meeting            3
Last_lead_update           11
Internal_POC               60
Resource                    6
Internal_rating             5
Success_probability       248
dtype: int64
```

DATA CLEANING/DROPPING UNWANTED COLUMNS

```
df = df.drop(columns=['Lead_name', 'Contact_no','Lead_POC_email','POC_name','Date_of_creat
```

```
df2 = df2.drop(columns=['Lead_name', 'Contact_no','Lead_POC_email','POC_name','Date_of_cre
```

```
df.drop(['Deal_title','Lead_name','Date_of_creation','Contact_no','Geography','POC_name','
```

HANDLING ERRORS IN DATATYPES OF COLUMNS

```
df['Deal_value'] = df['Deal_value'].str.replace('$', '')
```

```
df2['Deal_value'] = df2['Deal_value'].str.replace('$', '')
```

```
df['Weighted_amount'] = df['Weighted_amount'].str.replace('$', '')
```

```
df2['Weighted_amount'] = df2['Weighted_amount'].str.replace('$', '')
```

```
df['Industry'].fillna('Hotels/Motels', inplace = True)
df2['Industry'].fillna('Hotels/Motels', inplace = True)
```

```
df['Deal_value']
```

```
0       320506
1        39488
2       359392
3        76774
4       483896
         ...
7002    192800
7003    220208
7004    253608
7005    118615
```

```
7006    258627
Name: Deal_value, Length: 7007, dtype: object
```

```
df["Deal_value"]= df["Deal_value"].astype(float)
df["Weighted_amount"]= df["Weighted_amount"].astype(float)
```

```
df2["Deal_value"]= df2["Deal_value"].fillna(0).astype(int)
```

```
df2["Weighted_amount"]= df2["Weighted_amount"].fillna(0).astype(float)
```

HANDLING MISSING VALUES

```
df.Location.fillna("NA",inplace=True)
```

```
df["Deal_value"]= df["Deal_value"].fillna(0).astype(int)
```

```
df["Weighted_amount"]= df["Weighted_amount"].fillna(0).astype(float)
```

```
df.Location.fillna(0,inplace=True)
```

ONE HOT ENCODING

```
cat=list(df.select_dtypes(exclude=(np.number)).columns)
```

```
cat
```

```
['Industry',
 'Pitch',
 'Lead_revenue',
 'Fund_category',
 'Location',
 'Lead_source',
 'Level_of_meeting',
 'Last_lead_update',
 'Resource']
```

```
df.isnull().sum()
```

```
Industry               1
Deal_value             0
Weighted_amount        0
Pitch                  0
Lead_revenue           0
Fund_category          0
Location               0
Lead_source            0
```

```
Level_of_meeting         0
Last_lead_update       633
Resource               149
Internal_rating          0
Success_probability      0
dtype: int64
```

```
df.dropna(inplace=True)
```

```
df
```

	Industry	Deal_value	Weighted_amount	Pitch	Lead_revenue	Fund_categ
1	Construction Services	39488.0	2.408768e+05	Product_2	500 Million - 1 Billion	Categ
2	Hospitals/Clinics	359392.0	2.407926e+06	Product_1	500 Million - 1 Billion	Categ
3	Real Estate	76774.0	4.683214e+05	Product_2	500 Million - 1 Billion	Categ
4	Financial Services	483896.0	1.569884e+06	Product_2	50 - 100 Million	Categ
5	Banks	418674.0	2.637646e+06	Product_1	50 - 100 Million	Categ
...	...	...	...	...	...	
6999	Financial Services	31429.0	2.200030e+05	Product_2	500 Million - 1 Billion	Categ
7000	Beverages (Alcoholic)	152908.0	9.709658e+05	Product_1	100 - 500 Million	Categ
7002	Banks	192800.0	1.195360e+06	Product_1	100 - 500 Million	Categ
7003	Hospitals/Clinics	220208.0	1.453373e+06	Product_2	100 - 500 Million	Categ
7006	Financial Services	258627.0	1.642281e+06	Product_2	500 Million - 1 Billion	Categ

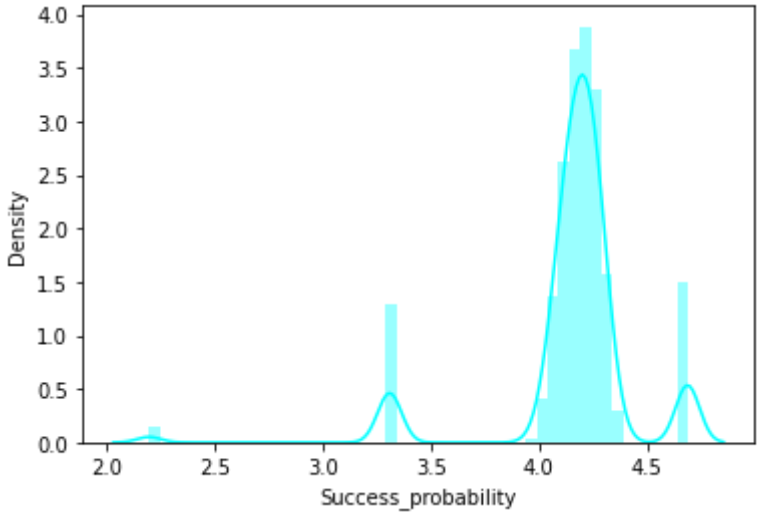
6237 rows × 13 columns

```
df=pd.get_dummies(df)
```

EDA - EXPLORATORY DATA ANALYSIS

```
import seaborn as sb
from matplotlib import pyplot as plt
g=sb.distplot(np.log1p(df['Success_probability']),color="Cyan")
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/series.py:726: RuntimeWarning:

invalid value encountered in log1p

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please a
```




```
f = np.mean(df['Deal_value'])
f
```

```
247838.9200799201
```

```
f1 = np.mean(df2['Deal_value'])
f1
```

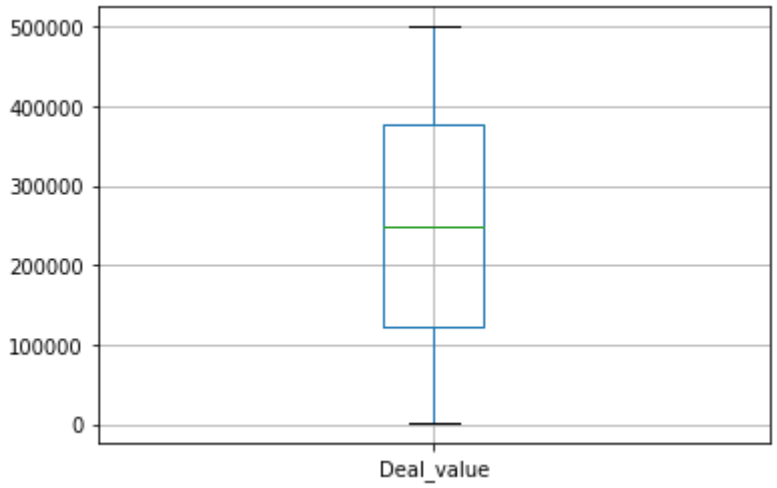
```
247459.56330625896
```

```
for i in range (0,len(df)):
  if(df['Deal_value'][i] == 0):
    df['Deal_value'][i] = 247839
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarnir


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```

```
for i in range (0,len(df2)):
```

```
  if(df2['Deal_value'][i] == 0):
    df2['Deal_value'][i] = 247459

boxplot = df.boxplot(column=['Deal_value'])
```

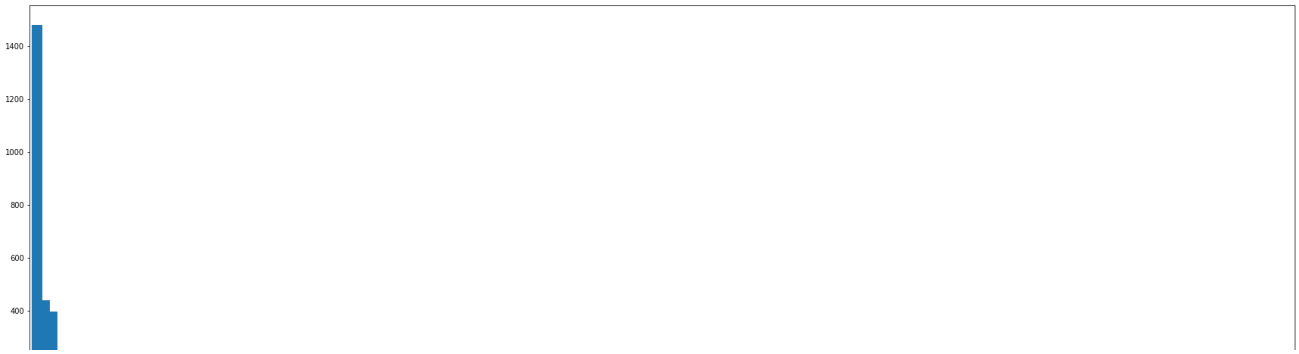



```
df[df['Industry'].isnull()]
```

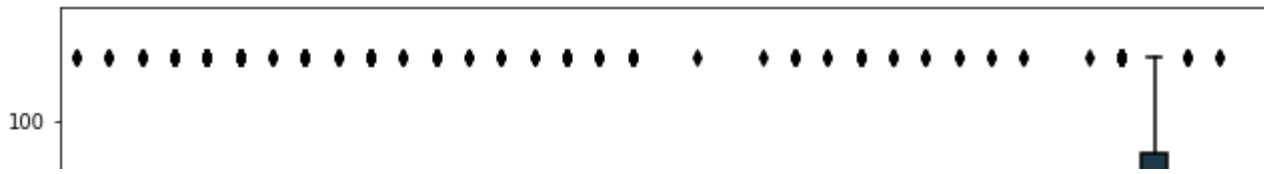
	Industry	Deal_value	Weighted_amount	Pitch	Lead_revenue	Fund_category	I
4653	NaN	209418$	1266978.9$	Product_2	50 - 100 Million	Category 4	

```
df["Industry"].value_counts().plot(kind='bar',width = 1.5,figsize = (30,10))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c58078650>
```

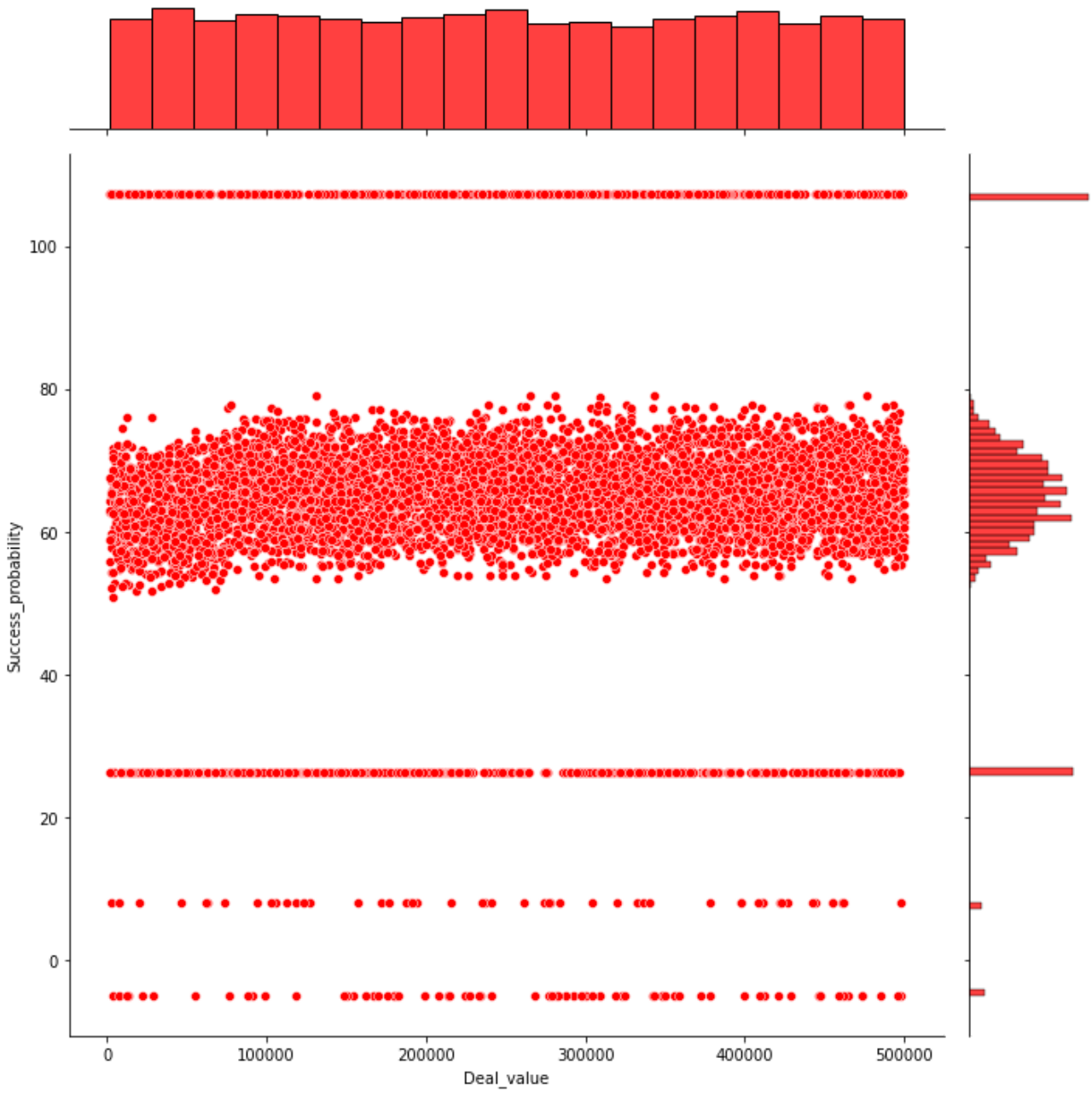



```
import seaborn as sb
plt.figure(figsize=(50, 10))
sb.boxplot(x="Industry", y="Success_probability", data=df,palette="cubehelix")
plt.xticks(rotation=90)
plt.show()
```




```
plt.figure(figsize=(50, 50))
sb.jointplot(x='Deal_value',y='Success_probability',data=df, color = 'red' , height =10, r
plt.show()
```

```
<Figure size 3600x3600 with 0 Axes>
```




```
df['Weighted_amount']
```

```
0        2067263.7$
1         240876.8$
2        2407926.4$
3         468321.4$
```

```
4               NaN
           ...
7002     1195360.0$
7003     1453372.8$
7004            NaN
7005      794720.5$
7006    1642281.45$
Name: Weighted_amount, Length: 7007, dtype: object
```

```
g = np.mean(df['Weighted_amount'])
g
```

```
1452260.0168117583
```

```
g2 = np.mean(df2['Weighted_amount'])
g2
```

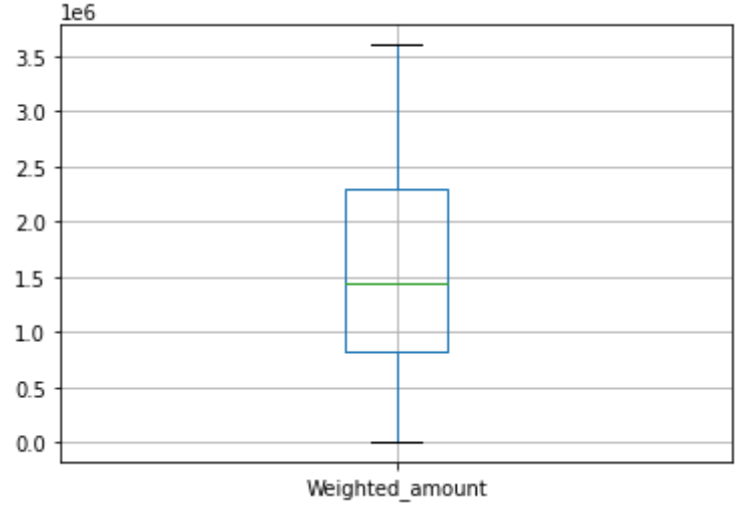
```
1512284.5661968442
```

```
for i in range (0,len(df)):
  if(df['Weighted_amount'][i] == 0):
    df['Weighted_amount'][i] = 1452260.0
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarnir


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```

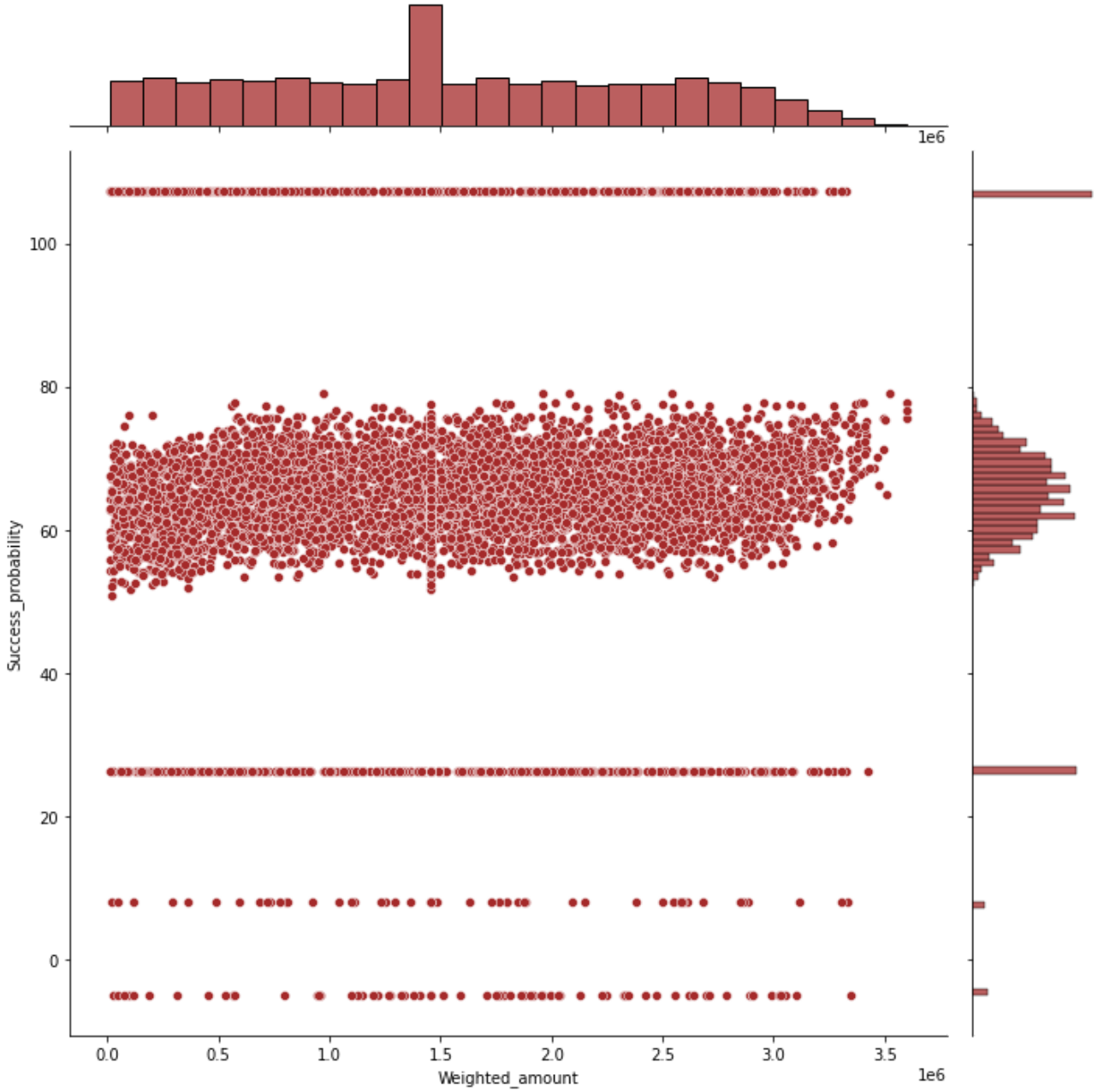
```
for i in range (0,len(df2)):
  if(df2['Weighted_amount'][i] == 0):
    df2['Weighted_amount'][i] = 1512284.5
```

```
boxplot = df.boxplot(column=['Weighted_amount'])
```

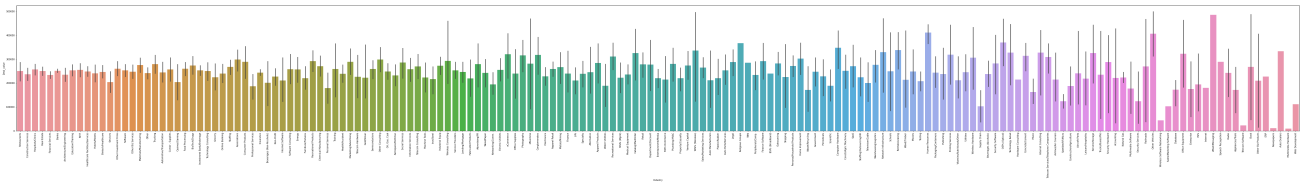


```
plt.figure(figsize=(50, 50))
sb.jointplot(x='Weighted_amount',y='Success_probability',data=df, color = 'brown' , height
plt.show()
```

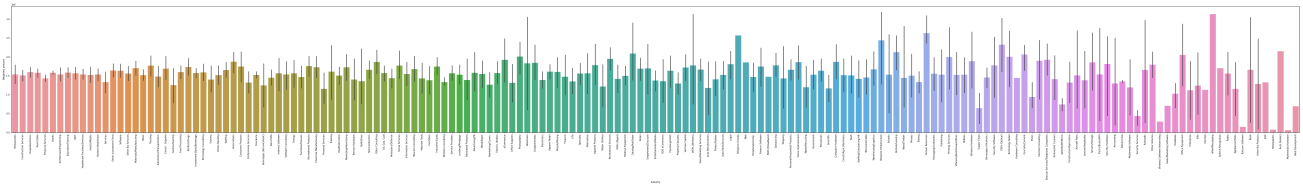
```
<Figure size 3600x3600 with 0 Axes>
```




```
plt.figure(figsize=(100, 10))
sb.barplot(x="Industry", y="Deal_value", data=df)
plt.xticks(rotation=90)
plt.show()
```



```
plt.figure(figsize=(100, 10))
sb.barplot(x="Industry", y="Weighted_amount", data=df)
plt.xticks(rotation=90)
plt.show()
```



```
df['Pitch'].unique()
```

```
array(['Product_2', 'Product_1'], dtype=object)
```

```
btype = df["Location"].str.split(",",n=1,expand = True)
df["City"]=btype[0]
df["State"]=btype[1]
```

```
btype2 = df2["Location"].str.split(",",n=1,expand = True)
df2["City"]=btype[0]
df2["State"]=btype[1]
```

```
df.head()
```

	Deal_title	Industry	Deal_value	Weighted_amount	Pitch	Lead_revenue	F
0	TitleM5DZY	Restaurants	200500	9067262.7	Product 2	50 - 100	

```
for i in range(0,len(df)):
  if(df['State'][i] == None):
    df['Geography'][i] = "India"
  else:
    df['Geography'][i] = "USA"
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarnir


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarnir


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```

```
for i in range(0,len(df2)):
  if(df2['State'][i] == None):
    df2['Geography'][i] = "India"
  else:
    df2['Geography'][i] = "USA"
```

```
df['Geography'].isnull().sum()
```

```
0
```

```
df = df.drop(columns=['Location', 'City','State','Last_lead_update'])
```

```
df2 = df2.drop(columns=['Location', 'City','State','Last_lead_update'])
```

```
df['Resource'].unique()
```

```
array([nan, 'No', 'We have all the requirements', 'Deliverable',
       'Cannot deliver', 'Not enough', 'Yes'], dtype=object)
```

```
df["Resource"].mode()
```

```
0    No
dtype: object
```

```
df['Resource'].fillna("No", inplace = True)
```

```
df2['Resource'].fillna("No", inplace = True)


df.reset_index(inplace=True)
df.drop('index',axis=1,inplace=True)


l=list(df.columns)
l.remove('Success_probability')
l
```

```
['Deal_title',
 'Industry',
 'Deal_value',
 'Weighted_amount',
 'Pitch',
 'Lead_revenue',
 'Fund_category',
 'Geography',
 'Designation',
 'Hiring_candidate_role',
 'Lead_source',
 'Level_of_meeting',
 'Internal_POC',
 'Resource',
 'Internal_rating']
```

```
X = df[['Deal_title', 'Industry', 'Pitch','Lead_revenue', 'Fund_category', 'Geography', 'D


X2 = df[['Deal_title', 'Industry', 'Pitch','Lead_revenue', 'Fund_category', 'Geography', '


X = df[l]
X.head()
```

itle	Industry	Deal_value	Weighted_amount	Pitch	Lead_revenue	Fund_catego
5DZY	Restaurants	320506	2067263.7	Product_2	50 - 100 Million	Category
IW18	Construction Services	39488	240876.8	Product_2	500 Million - 1 Billion	Category
SDN	Hospitals/Clinics	359392	2407926.4	Product_1	500 Million - 1 Billion	Category
3K4Y	Real Estate	76774	468321.4	Product_2	500 Million - 1 Billion	Category
)4GV	Financial Services	483896	1452260.0	Product_2	50 - 100 Million	Category

```
X2 = pd.get_dummies(data=X, drop_first=True)
X2.head()
```

	Deal_title_Title00IIZ	Deal_title_Title00VOR	Deal_title_Title013QQ	Deal_title_T
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	

5 rows × 7899 columns

```
Y = df['Success_probability']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

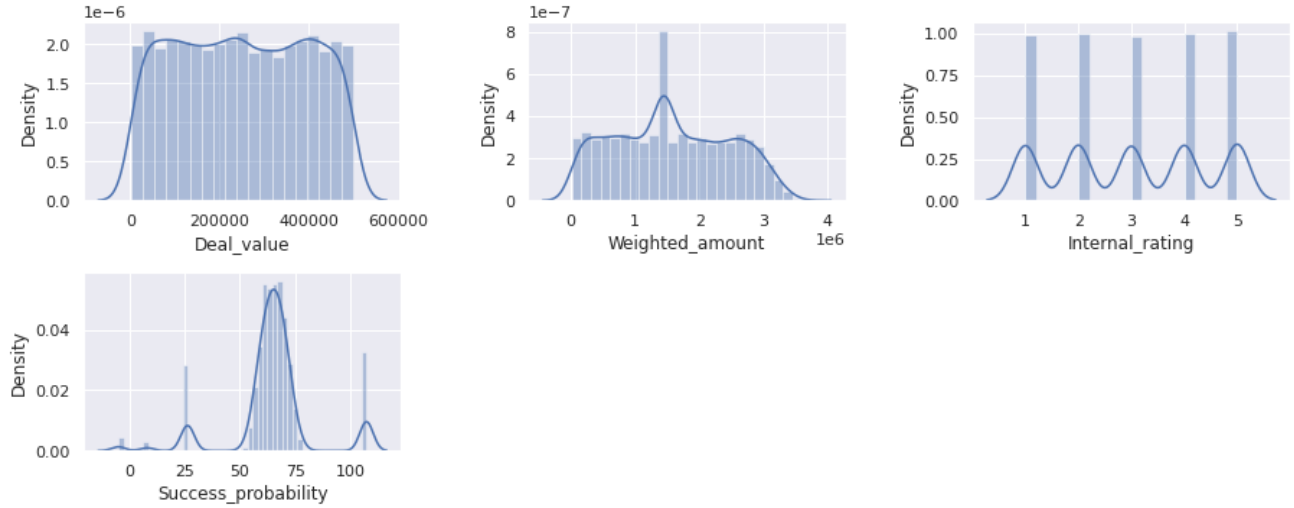
```
(5605, 15)
(1402, 15)
(5605,)
(1402,)
```

```
import warnings
warnings.filterwarnings('ignore')
```

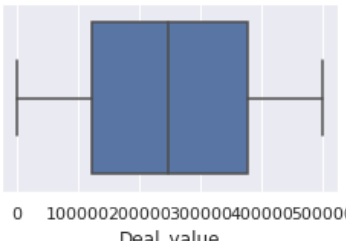
```
sb.heatmap(df.corr(),annot=True)
sb.set(rc={'figure.figsize':(15,15)})
```

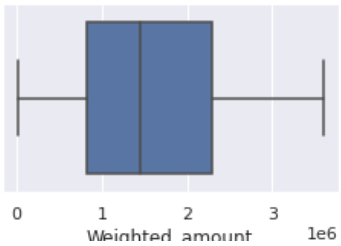
```
num=list(df.select_dtypes(include=(np.number)).columns)
```

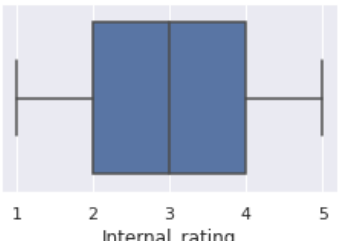

```
count=1
plt.subplots(figsize=(15,15))
for i in num:
  plt.subplot(5,3,count)
  sb.distplot(df[i])
  count+=1
plt.subplots_adjust(wspace=0.4, hspace=0.4)
plt.show()
```



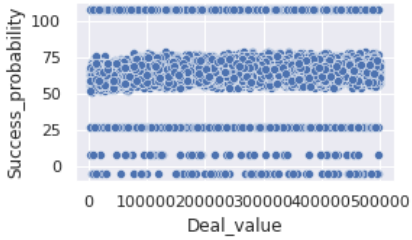

```
count=1
plt.subplots(figsize=(15,15))
for i in num:
  plt.subplot(5,3,count)
  sb.boxplot(df[i])
  count+=1
plt.subplots_adjust(wspace=0.4, hspace=0.4)
plt.show()
```

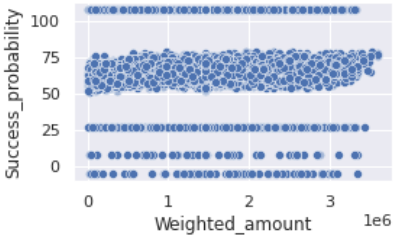


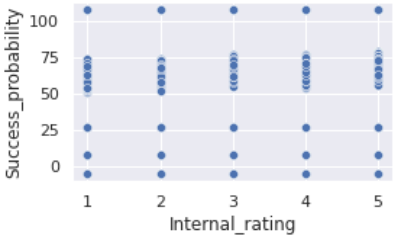


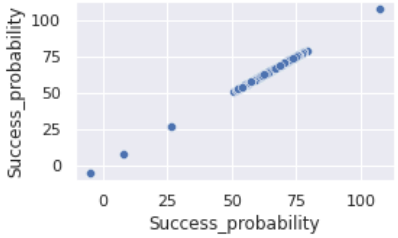



```
count=1
plt.subplots(figsize=(15,15))
for i in num:
  plt.subplot(5,3,count)
  sb.scatterplot(x=df[i],y=df['Success_probability'])
  count+=1
plt.subplots_adjust(wspace=0.4, hspace=0.4)
plt.show()
```










```
df.reset_index(inplace=True)
df.drop('index',axis=1,inplace=True)
```

```
df
```

	Deal_title	Industry	Deal_value	Weighted_amount	Pitch	Lead_revenue
0	TitleM5DZY	Restaurants	320506	2067263.70	Product_2	50 - 100 Millior
1	TitleKIW18	Construction Services	39488	240876.80	Product_2	500 Million - 1 Billior
2	TitleFXSDN	Hospitals/Clinics	359392	2407926.40	Product_1	500 Million - 1 Billior
3	TitlePSK4Y	Real Estate	76774	468321.40	Product_2	500 Million - 1 Billior
4	Title904GV	Financial Services	483896	1452260.00	Product_2	50 - 100 Millior
...	...	...	...	...	...	..
7002	TitleJ7TDY	Banks	192800	1195360.00	Product_1	100 - 500 Millior
7003	TitleO1IIN	Hospitals/Clinics	220208	1453372.80	Product_2	100 - 500 Millior

```
l=list(df.columns)
l.remove('Success_probability')
```

```
x=df[l]
y=df['Success_probability']
```

```
df.drop(columns=['Deal_title'])
```

	Industry	Deal_value	Weighted_amount	Pitch	Lead_revenue	Fund_categ
0	Restaurants	320506	2067263.70	Product_2	50 - 100 Million	Categ
1	Construction Services	39488	240876.80	Product_2	500 Million - 1 Billion	Categ
2	Hospitals/Clinics	359392	2407926.40	Product_1	500 Million - 1 Billion	Categ
3	Real Estate	76774	468321.40	Product_2	500 Million - 1 Billion	Categ
4	Financial Services	483896	1452260.00	Product_2	50 - 100 Million	Categ

```
X = df[['Deal_title', 'Industry', 'Pitch','Lead_revenue', 'Fund_category', 'Geography', 'D

X = pd.get_dummies(data=X, drop_first=True)
X.head()
```

	Deal_title_Title00IIZ	Deal_title_Title00VOR	Deal_title_Title013QQ	Deal_title_T
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	

5 rows × 7899 columns

```
Y = df['Success_probability']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.4, random_state=101)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(4204, 7899)
(2803, 7899)
(4204,)
(2803,)
```

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
print(model.intercept_)
```

```
56.15975114272731
```

```
coeff_parameter = pd.DataFrame(model.coef_,X.columns,columns=['Coefficient'])
coeff_parameter
```

	Coefficient
Deal_title_Title00IlZ	2.357888
Deal_title_Title00VOR	7.933857
Deal_title_Title013QQ	0.440326
Deal_title_Title017NC	-3.619940
Deal_title_Title01ANN	-32.834445
...	...
Resource_Deliverable	-0.395741
Resource_No	-0.759543
Resource_Not enough	-0.610481
Resource_We have all the requirements	0.912920
Resource_Yes	-0.361731

7899 rows × 1 columns

```
predictions = model.predict(X_test)
predictions
```

```
array([60.00355909, 65.76964256, 67.26488284, ..., 69.25676092,
       62.27983245, 59.95131948])
```

```
import statsmodels.api as sm
X_train_Sm= sm.add_constant(X_train)
X_train_Sm= sm.add_constant(X_train)
ls=sm.OLS(y_train,X_train_Sm).fit()
print(ls.summary())
```

```
Internal_POC_Gaskins Jr,Franklin D
Internal_POC_Georgakopoulos,Vasilios T
Internal_POC_Gilley,Janine
Internal_POC_Gould,Lisa D
Internal_POC_Green,Ann E
Internal_POC_Green,Candy
Internal_POC_Hameier,Kurt E
Internal_POC_Hanyok,John J
Internal_POC_Hebron,Artenia D
Internal_POC_Heidelberg,Andre D
Internal_POC_Himes,Maurice C
```

```
Internal_POC_Houston,Arnold E
Internal_POC_Irizarry,Yolanda
Internal_POC_Jones,Eyvette W
Internal_POC_Jones,Michael L
Internal_POC_Kiepea,Prince A
Internal_POC_Knox,Antonio D
Internal_POC_Leu,Darren L
Internal_POC_Logan,Kevin N
Internal_POC_Mabrey,Kevin C
Internal_POC_Maine,John P
Internal_POC_Massiah,Gerard F
Internal_POC_McKenstry,Loretta A
Internal_POC_Meli,Teresa V
Internal_POC_Moran,Natalie A
Internal_POC_Morsy,Omar A
Internal_POC_Murray,Younetta
Internal_POC_Pappas,Mark S
Internal_POC_Robinson,John C
Internal_POC_Rocks,Michael J
Internal_POC_Ross,Eric L
Internal_POC_Ryker,David
Internal_POC_Salyers,Daniel L
Internal_POC_Shelton,Sidney P
Internal_POC_Smith,Keenan H
Internal_POC_Sutton,Michelle R
Internal_POC_Thomas,Lori E
Internal_POC_Tondeur,Keith D
Internal_POC_Turner,Marlon D
Internal_POC_Ullrich,Rose Anne
Internal_POC_Van Arter,Derrick
Internal_POC_Vickers Jr.,Henry J
Internal_POC_Young,Valerie K
Resource_Deliverable
Resource_No
Resource_Not enough
Resource_We have all the requirements
Resource_Yes
==============================================================================
Omnibus:                     3254.379   Durbin-Watson:                   1.823
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           680796.322
Skew:                          -2.765   Prob(JB):                         0.00
Kurtosis:                      65.097   Cond. No.                         258.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
[2] The input rank is higher than the number of observations.
```

```
Predict = model.predict(X2)


solution = pd.DataFrame(Predict)


id = df2['Deal_title']
sample_solution = pd.concat([id,solution],axis=1)
sample_solution.columns = ['Deal_title','Success_Probability']
```

```
sample_solution.to_csv('sample_solution.csv')
```

```
ss=pd.read_csv("sample_solution.csv")
```

```
ss.head()
```

	Unnamed: 0	Deal_title	Success_Probability
0	0	TitleAD16O	76.467936
1	1	TitleOW6CR	58.900000
2	2	TitleVVJQ5	68.800000
3	3	TitleUS8NA	64.500000
4	4	Title5VGWW	62.400000

✓ 0s completed at 11:06 PM