Project Machine Learning Spring 2022

CREDIT APPROVAL DATA ANALYSIS USING CLASSIFICATION AND REGRESSION MODELS

By- Boddu upender

Under the guidence of Muhammad Aminul Islam

I have selected this project because it is one of the main aspects of life. It is important to know if your credit card gets approved or not well before in time to plan life and other situations. It is very challenging to predict this information by humans as there are several dependent variables. A machine can predict the aproval if well trained. In this project several methods, models, and parameters are tested to get highest prediction accuracy

Abstract

Algorithms that are used to decide the outcome of credit applications vary from one provider to another and across sectors and geographies. There are however, high degrees of similarity in the attributes used to generate those algorithms. The differences may be in the weights applied to individual attributes. This paoject analyses the credit application data taken from the kaggle(A Credit Card Dataset for Machine Learning) Various preprocessing techniques like data exploratory analysis and data transformations like handling missing values, continuous values, and categorical values are computed on the data. Several data visualization techniques are adopted to understand the data. The analytical models like regression and classification techniques are generated and implemented on the data. The results are evaluated using various performance metrics.

Data sets I am using

Link -https://www.kaggle.com/rikdifos/credit-card-approval-prediction

In this project I am using data sets from kaggle open source platform

Mainly two data sets

1.application_record.csv (All the applicants who applied for credit card)

2.credit_record.csv (Their credit history record)

Describing Datasets

Importing usefull libraries for Describeing data sets

In [263…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Describing data set one(1)= data1

In [264…
```
data1 = pd.read_csv('application_record.csv')
```

In [265…
```
data1.head()
```

Out[265…

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL
0	5008804	M	Y	Y	0	427500.0
1	5008805	M	Y	Y	0	427500.0
2	5008806	M	Y	Y	0	112500.0
3	5008808	F	N	Y	0	270000.0
4	5008809	F	N	Y	0	270000.0

In [266…
```
data1.shape
```

Out[266…
```
(438557, 18)
```

In [267…
```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 438557 entries, 0 to 438556
Data columns (total 18 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   ID                   438557 non-null  int64
 1   CODE_GENDER          438557 non-null  object
```

```
 2   FLAG_OWN_CAR          438557 non-null  object
 3   FLAG_OWN_REALTY       438557 non-null  object
 4   CNT_CHILDREN          438557 non-null  int64
 5   AMT_INCOME_TOTAL      438557 non-null  float64
 6   NAME_INCOME_TYPE      438557 non-null  object
 7   NAME_EDUCATION_TYPE   438557 non-null  object
 8   NAME_FAMILY_STATUS    438557 non-null  object
 9   NAME_HOUSING_TYPE     438557 non-null  object
 10  DAYS_BIRTH            438557 non-null  int64
 11  DAYS_EMPLOYED         438557 non-null  int64
 12  FLAG_MOBIL            438557 non-null  int64
 13  FLAG_WORK_PHONE       438557 non-null  int64
 14  FLAG_PHONE            438557 non-null  int64
 15  FLAG_EMAIL            438557 non-null  int64
 16  OCCUPATION_TYPE       304354 non-null  object
 17  CNT_FAM_MEMBERS       438557 non-null  float64
dtypes: float64(2), int64(8), object(8)
memory usage: 60.2+ MB
```

In [268…

```
data1.describe()
```

Out[268…

	ID	CNT_CHILDREN	AMT_INCOME_TOTAL	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOI
count	4.385570e+05	438557.000000	4.385570e+05	438557.000000	438557.000000	43855
mean	6.022176e+06	0.427390	1.875243e+05	-15997.904649	60563.675328	
std	5.716370e+05	0.724882	1.100869e+05	4185.030007	138767.799647	
min	5.008804e+06	0.000000	2.610000e+04	-25201.000000	-17531.000000	
25%	5.609375e+06	0.000000	1.215000e+05	-19483.000000	-3103.000000	
50%	6.047745e+06	0.000000	1.607805e+05	-15630.000000	-1467.000000	
75%	6.456971e+06	1.000000	2.250000e+05	-12514.000000	-371.000000	
max	7.999952e+06	19.000000	6.750000e+06	-7489.000000	365243.000000	

Describeing data set two(2)= data2

In [269…

```
data2 = pd.read_csv('credit_record.csv')
```

In [270…

```
data2.head()
```

Out[270…

	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	C

In [271…
```
data2.shape
```

Out[271…
```
(1048575, 3)
```

In [272…
```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 3 columns):
 #   Column          Non-Null Count    Dtype 
---  ------          --------------    ----- 
 0   ID              1048575 non-null  int64 
 1   MONTHS_BALANCE  1048575 non-null  int64 
 2   STATUS          1048575 non-null  object
dtypes: int64(2), object(1)
memory usage: 24.0+ MB
```

In [273…
```
data2.describe()
```

Out[273…

	ID	MONTHS_BALANCE
count	1.048575e+06	1.048575e+06
mean	5.068286e+06	-1.913700e+01
std	4.615058e+04	1.402350e+01
min	5.001711e+06	-6.000000e+01
25%	5.023644e+06	-2.900000e+01
50%	5.062104e+06	-1.700000e+01
75%	5.113856e+06	-7.000000e+00
max	5.150487e+06	0.000000e+00

References

Link to research article that I used for implement

I read this article from (found at) International Peer Reviewed & Refereed Journal, Open Access Journal

I felt intresting when I was reading this article and decided to take this journal as refernce for this project many important points in this article cought my eye and i followed this steps

2018 IJRAR September 2018, Volume 5, Issue 3 www.ijrar.org (E-ISSN 2348-1269, P- ISSN 2349-5138)

link - https://www.ijrar.org/papers/IJRAR190B030.pdf

Exploratory Data Analysis (EDA) on data1(application_record.csv)

In [274…
```
data1.describe()
```

Out[274…

	ID	CNT_CHILDREN	AMT_INCOME_TOTAL	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOI
count	4.385570e+05	438557.000000	4.385570e+05	438557.000000	438557.000000	43855
mean	6.022176e+06	0.427390	1.875243e+05	-15997.904649	60563.675328	
std	5.716370e+05	0.724882	1.100869e+05	4185.030007	138767.799647	
min	5.008804e+06	0.000000	2.610000e+04	-25201.000000	-17531.000000	
25%	5.609375e+06	0.000000	1.215000e+05	-19483.000000	-3103.000000	
50%	6.047745e+06	0.000000	1.607805e+05	-15630.000000	-1467.000000	
75%	6.456971e+06	1.000000	2.250000e+05	-12514.000000	-371.000000	
max	7.999952e+06	19.000000	6.750000e+06	-7489.000000	365243.000000	

chechking null values in the data set

In [275…
```
data1.isnull().sum()
```

Out[275…
```
ID                          0
CODE_GENDER                 0
FLAG_OWN_CAR                0
FLAG_OWN_REALTY             0
CNT_CHILDREN                0
AMT_INCOME_TOTAL            0
NAME_INCOME_TYPE            0
NAME_EDUCATION_TYPE         0
NAME_FAMILY_STATUS          0
NAME_HOUSING_TYPE           0
DAYS_BIRTH                  0
DAYS_EMPLOYED               0
FLAG_MOBIL                  0
FLAG_WORK_PHONE             0
FLAG_PHONE                  0
FLAG_EMAIL                  0
OCCUPATION_TYPE        134203
```

```
CNT_FAM_MEMBERS              0
dtype: int64
```

so now we found null valeus in occupation_type around 134203 we should remove them because it will be noise and that will effect in data analysis further

In [276…
```
data1.drop('OCCUPATION_TYPE', axis=1, inplace=True)
```

checking if there are any duplicates in ID column

In [282…
```
len(data1['ID']) - len(data1['ID'].unique())
```

Out[282…
```
47
```

dropping duplicates from the ID column

In [285…
```
data1 = data1.drop_duplicates('ID', keep='last')
```

checking the columns which has non numerical values

In [286…
```
check_col = data1.columns[(data1.dtypes =='object').values].tolist()
check_col
```

Out[286…
```
['CODE_GENDER',
 'FLAG_OWN_CAR',
 'FLAG_OWN_REALTY',
 'NAME_INCOME_TYPE',
 'NAME_EDUCATION_TYPE',
 'NAME_FAMILY_STATUS',
 'NAME_HOUSING_TYPE']
```

Checking the columns which has numerical values

In [287…
```
data1.columns[(data1.dtypes !='object').values].tolist()
```

Out[287…
```
['ID',
 'CNT_CHILDREN',
 'AMT_INCOME_TOTAL',
 'DAYS_BIRTH',
 'DAYS_EMPLOYED',
 'FLAG_MOBIL',
 'FLAG_WORK_PHONE',
 'FLAG_PHONE',
 'FLAG_EMAIL',
 'CNT_FAM_MEMBERS']
```

Checking unique values from Numerical Columns

In [288…
```
data1['CNT_CHILDREN'].value_counts()
```

Out[288…
```
0     304038
1      88518
2      39879
3       5430
4        486
5        133
7          9
9          5
12         4
6          4
14         3
19         1
Name: CNT_CHILDREN, dtype: int64
```

Checking for the unique values from different types of Columns

In [289…
```
for i in data1.columns[(data1.dtypes =='object').values].tolist():
    print(i,'\n')
    print(data1[i].value_counts())
    print('----------------------------------------------')
```

```
CODE_GENDER

F    294412
M    144098
Name: CODE_GENDER, dtype: int64
----------------------------------------------
FLAG_OWN_CAR

N    275428
Y    163082
Name: FLAG_OWN_CAR, dtype: int64
----------------------------------------------
FLAG_OWN_REALTY

Y    304043
N    134467
Name: FLAG_OWN_REALTY, dtype: int64
----------------------------------------------
NAME_INCOME_TYPE

Working                 226087
Commercial associate    100739
Pensioner                75483
State servant            36184
Student                     17
Name: NAME_INCOME_TYPE, dtype: int64
----------------------------------------------
NAME_EDUCATION_TYPE
```

```
Secondary / secondary special    301789
Higher education                 117509
Incomplete higher                 14849
Lower secondary                    4051
Academic degree                     312
Name: NAME_EDUCATION_TYPE, dtype: int64
------------------------------------------------
NAME_FAMILY_STATUS

Married                 299798
Single / not married     55268
Civil marriage           36524
Separated                27249
Widow                    19671
Name: NAME_FAMILY_STATUS, dtype: int64
------------------------------------------------
NAME_HOUSING_TYPE

House / apartment      393788
With parents            19074
Municipal apartment     14213
Rented apartment         5974
Office apartment         3922
Co-op apartment          1539
Name: NAME_HOUSING_TYPE, dtype: int64
------------------------------------------------
```

Checking Minimum , Maximum values from 'DAYS_BIRTH' column

In [290…

```
print('Min DAYS_BIRTH :', data1['DAYS_BIRTH'].min(),'\nMax DAYS_BIRTH :', data1['DAYS_B
```

```
Min DAYS_BIRTH : -25201
Max DAYS_BIRTH : -7489
```

Changing the 'DAYS_BIRTH' column values from Day to Year

In [291…

```
data1['DAYS_BIRTH'] = round(data1['DAYS_BIRTH']/-365,0)
data1.rename(columns={'DAYS_BIRTH':'AGE_YEARS'}, inplace=True)
```

Finding out the unique values which are greater than zero

In [292…

```
data1[data1['DAYS_EMPLOYED']>0]['DAYS_EMPLOYED'].unique()
```

Out[292…

```
array([365243], dtype=int64)
```

In [293…

```
## The column 'DAYS_EMPLOYED' has the positive value that clearly states that the preso
## replacing the value with zero
data1['DAYS_EMPLOYED'].replace(365243, 0, inplace=True)
```

Changing the 'DAYS_EMPLOYED' column values from Day to Year

In [294…
```
data1['DAYS_EMPLOYED'] = abs(round(data1['DAYS_EMPLOYED']/-365,0))
data1.rename(columns={'DAYS_EMPLOYED':'YEARS_EMPLOYED'}, inplace=True)
```

'FLAG_MOBIL'

In [295…
```
data1['FLAG_MOBIL'].value_counts()
```

Out[295…
```
1    438510
Name: FLAG_MOBIL, dtype: int64
```

All the values of the column is 1 so dropping the column

In [296…
```
data1.drop('FLAG_MOBIL', axis=1, inplace=True)
```

'FLAG_WORK_PHONE'

In [297…
```
data1['FLAG_WORK_PHONE'].value_counts()
```

Out[297…
```
0    348118
1     90392
Name: FLAG_WORK_PHONE, dtype: int64
```

This column has 0 & 1 values for the those who have submitted the phone numnber.

In [298…
```
## so dropping this phone number
data1.drop('FLAG_WORK_PHONE', axis=1, inplace=True)
```

'FLAG_PHONE'

In [299…
```
data1['FLAG_PHONE'].value_counts()
```

Out[299…
```
0    312323
1    126187
Name: FLAG_PHONE, dtype: int64
```

In [300…
```
## This column has 0 & 1 values for the those who have submitted the phone numnber.
## so dropping this phone number
data1.drop('FLAG_PHONE', axis=1, inplace=True)
```

'FLAG_EMAIL'

In [301…
```
data1['FLAG_EMAIL'].value_counts()
```

Out[301…
```
0    391062
1     47448
Name: FLAG_EMAIL, dtype: int64
```

In [302…
```
## This column has 0 & 1 values for the those who have submitted the Email ID.
## so dropping this Email ID column
data1.drop('FLAG_EMAIL', axis=1, inplace=True)
```

'CNT_FAM_MEMBERS'

In [303…
```
data1['CNT_FAM_MEMBERS'].value_counts()
```

Out[303…
```
2.0     233867
1.0      84483
3.0      77119
4.0      37351
5.0       5081
6.0        459
7.0        124
9.0          9
11.0         5
14.0         4
8.0          4
15.0         3
20.0         1
Name: CNT_FAM_MEMBERS, dtype: int64
```

In [304…
```
data1.head()
```

Out[304…

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL
0	5008804	M	Y	Y	0	427500.0
1	5008805	M	Y	Y	0	427500.0
2	5008806	M	Y	Y	0	112500.0
3	5008808	F	N	Y	0	270000.0
4	5008809	F	N	Y	0	270000.0

In [305…
```
data1.tail()
```

Out[305…

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_T
438552	6840104	M	N	Y	0	13

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_T
438553	6840222	F	N	N	0	10
438554	6841878	F	N	N	0	5
438555	6842765	F	N	Y	0	7
438556	6842885	F	N	Y	0	12

Visualization

In [306…
```
import seaborn as sns
import os
```

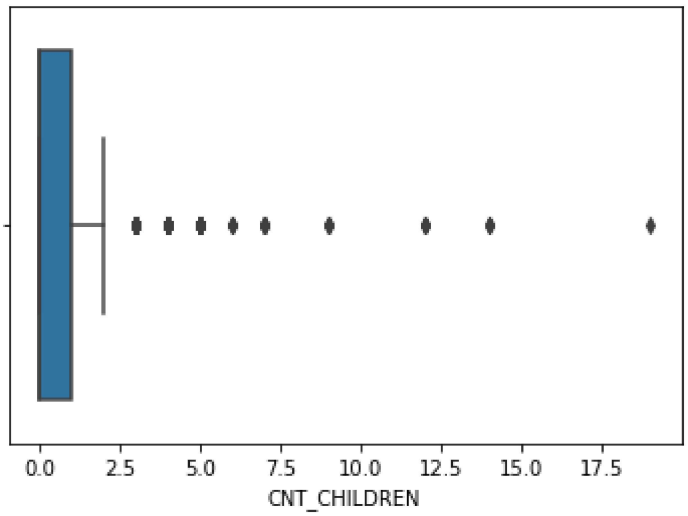
Here we are checking for outliars

In [307…
```
# Here we are checking for outliars

sns.boxplot(data1['CNT_CHILDREN'])
```

```
C:\Users\boddu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

Out[307…
```
<AxesSubplot:xlabel='CNT_CHILDREN'>
```

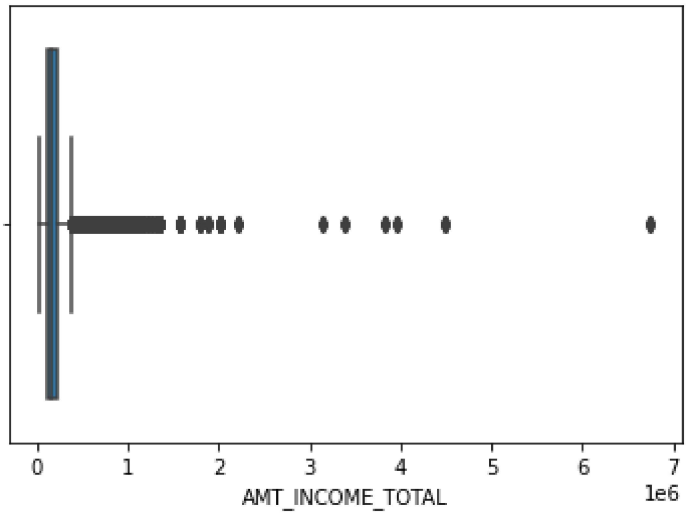


In [308…
```
sns.boxplot(data1['AMT_INCOME_TOTAL'])
```

```
C:\Users\boddu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
```

```
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

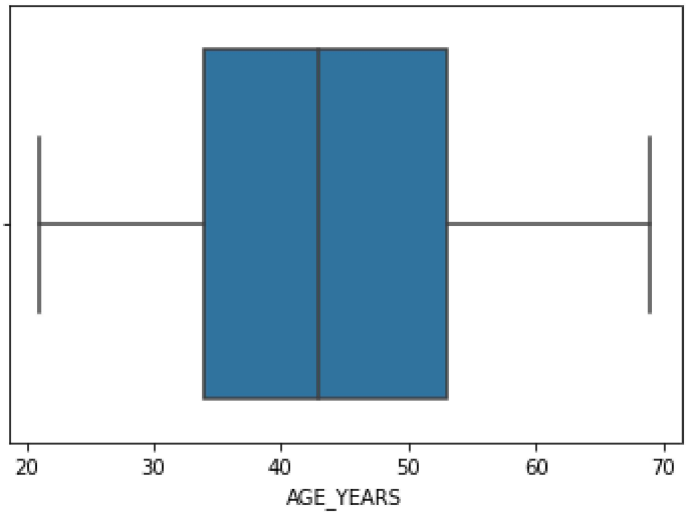
Out[308…
```
<AxesSubplot:xlabel='AMT_INCOME_TOTAL'>
```



In [309…
```
sns.boxplot(data1['AGE_YEARS'])
```

```
C:\Users\boddu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

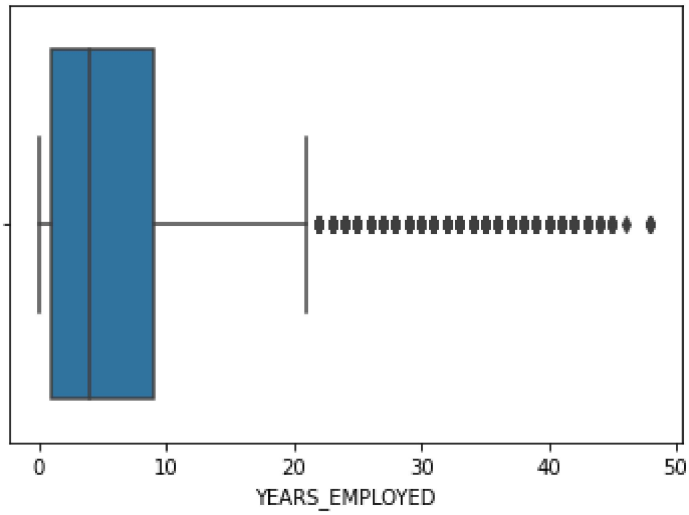
Out[309…
```
<AxesSubplot:xlabel='AGE_YEARS'>
```



In [310…
```
sns.boxplot(data1['YEARS_EMPLOYED'])
```

```
C:\Users\boddu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

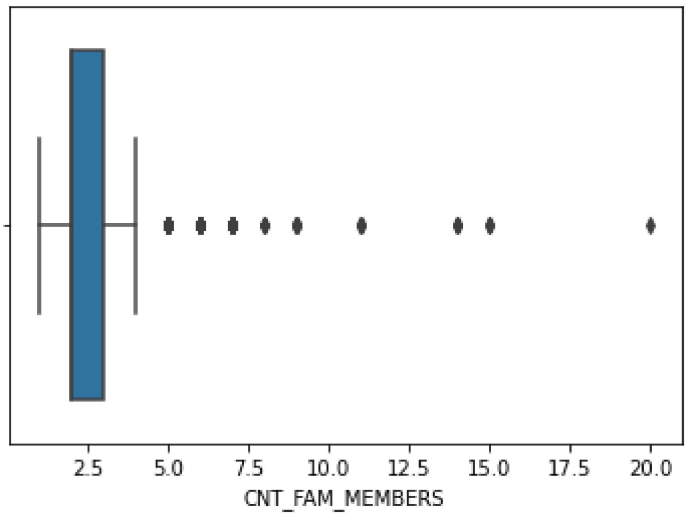
Out[310…
```
<AxesSubplot:xlabel='YEARS_EMPLOYED'>
```



In [311…
```
sns.boxplot(data1

            ['CNT_FAM_MEMBERS'])
```

```
C:\Users\boddu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```
Out[311…
```
<AxesSubplot:xlabel='CNT_FAM_MEMBERS'>
```



outliers removal

In [312…
```
higher_boundries = data1['CNT_CHILDREN'].quantile(0.999)
print('higher_boundries :', higher_boundries)
lower_boundries = data1['CNT_CHILDREN'].quantile(0.001)
print('lower_boundries :', lower_boundries)
data1 = data1[(data1['CNT_CHILDREN']>=lower_boundries) & (data1['CNT_CHILDREN']<=higher
```

```
higher_boundries : 4.0
lower_boundries : 0.0
```

In [313…

```
higher_boundries = data1['AMT_INCOME_TOTAL'].quantile(0.999)
print('higher_boundries :', higher_boundries)
lower_boundries = data1['AMT_INCOME_TOTAL'].quantile(0.001)
print('lower_boundries :', lower_boundries)
data1 = data1[(data1['AMT_INCOME_TOTAL']>=lower_boundries) & (data1['AMT_INCOME_TOTAL']
```

```
higher_boundries : 990000.0
lower_boundries : 36000.0
```

In [314…

```
higher_boundries = data1['YEARS_EMPLOYED'].quantile(0.999)
print('higher_boundries :', higher_boundries)
lower_boundries = data1['YEARS_EMPLOYED'].quantile(0.001)
print('lower_boundries :', lower_boundries)
data1 = data1[(data1['YEARS_EMPLOYED']>=lower_boundries) & (data1['YEARS_EMPLOYED']<=hi
```

```
higher_boundries : 40.0
lower_boundries : 0.0
```

In [315…

```
higher_boundries = data1['CNT_FAM_MEMBERS'].quantile(0.999)
print('higher_boundries :', higher_boundries)
lower_boundries = data1['CNT_FAM_MEMBERS'].quantile(0.001)
print('lower_boundries :', lower_boundries)
data1 = data1[(data1['CNT_FAM_MEMBERS']>=lower_boundries) & (data1


                                                    ['CNT_FAM_MEMBERS']<=highe
```

```
higher_boundries : 6.0
lower_boundries : 1.0
```

In [316…

```
data1.head()
```

Out[316…

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL
0	5008804	M	Y	Y	0	427500.0
1	5008805	M	Y	Y	0	427500.0
2	5008806	M	Y	Y	0	112500.0
3	5008808	F	N	Y	0	270000.0
4	5008809	F	N	Y	0	270000.0

Exploratory Data Analysis (EDA) on data2(Credit Record.csv)

In [317…

```
data2.head()
```

Out[317…

	ID	MONTHS_BALANCE	STATUS

	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	C

Checking for null values.

In [318…
```
data2.isnull().sum()
```

Out[318…
```
ID                0
MONTHS_BALANCE    0
STATUS            0
dtype: int64
```

There are no null values

In [319…
```
# Different status and their counts are listed below
data2['STATUS'].value_counts()
```

Out[319…
```
C    442031
0    383120
X    209230
1     11090
5      1693
2       868
3       320
4       223
Name: STATUS, dtype: int64
```

In [320…
```
# categorizing 'STATUS' column to binary classification   0 : Good Customer and 1 : Bad
data2['STATUS'].replace(['C', 'X'],0, inplace=True)
data2['STATUS'].replace(['2','3','4','5'],1, inplace=True)
data2['STATUS'] = data2['STATUS'].astype('int')
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 3 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   ID              1048575 non-null  int64
 1   MONTHS_BALANCE  1048575 non-null  int64
 2   STATUS          1048575 non-null  int32
dtypes: int32(1), int64(2)
memory usage: 20.0 MB
```

In [321…
```
data2['STATUS'].value_counts(normalize=True)*100
```

Out[321…
```
0    98.646353
1     1.353647
Name: STATUS, dtype: float64
```

In [322…
```
data2_trans = data2.groupby('ID').agg(max).reset_index()
data2_trans.drop('MONTHS_BALANCE', axis=1, inplace=True)
data2_trans.head()
```

Out[322…

	ID	STATUS
0	5001711	0
1	5001712	0
2	5001713	0
3	5001714	0
4	5001715	0

In [323…
```
data2_trans['STATUS'].value_counts(normalize=True)*100
```

Out[323…
```
0    88.365771
1    11.634229
Name: STATUS, dtype: float64
```

Merging the Dataframes

In [324…
```
## Now merging the datasets based on the 'ID'
Merge_data1 = pd.merge(data1, data2_trans, on='ID', how='inner')
Merge_data1.head()
```

Out[324…

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL
0	5008804	M	Y	Y	0	427500.0
1	5008805	M	Y	Y	0	427500.0
2	5008806	M	Y	Y	0	112500.0
3	5008808	F	N	Y	0	270000.0
4	5008809	F	N	Y	0	270000.0

In [325…
```
Merge_data1.shape
```

Out[325…
```
(36326, 14)
```

drop the 'ID' column as has unique values which is not

required for ML Model

In [326…
```
Merge_data1.drop('ID', axis=1, inplace=True)
```

In [327…
```
## Finding out that there are any duplicates rows in Final Dataframe
len(Merge_data1) - len(Merge_data1.drop_duplicates())
```

Out[327…
```
25268
```

In [328…
```
# Dropping the duplicates from the records
Merge_data1 = Merge_data1.drop_duplicates()
Merge_data1.reset_index(drop=True ,inplace=True)
Merge_data1.shape
```

Out[328…
```
(11058, 13)
```

In [329…
```
Merge_data1.isnull().sum()
```

Out[329…
```
CODE_GENDER            0
FLAG_OWN_CAR           0
FLAG_OWN_REALTY        0
CNT_CHILDREN           0
AMT_INCOME_TOTAL       0
NAME_INCOME_TYPE       0
NAME_EDUCATION_TYPE    0
NAME_FAMILY_STATUS     0
NAME_HOUSING_TYPE      0
AGE_YEARS              0
YEARS_EMPLOYED         0
CNT_FAM_MEMBERS        0
STATUS                 0
dtype: int64
```

In [330…
```
Merge_data1['STATUS'].value_counts(normalize=True)*100
```

Out[330…
```
0    78.513294
1    21.486706
Name: STATUS, dtype: float64
```

Merge Vizualization

In [331…
```
Merge_data1.head()
```

Out[331…

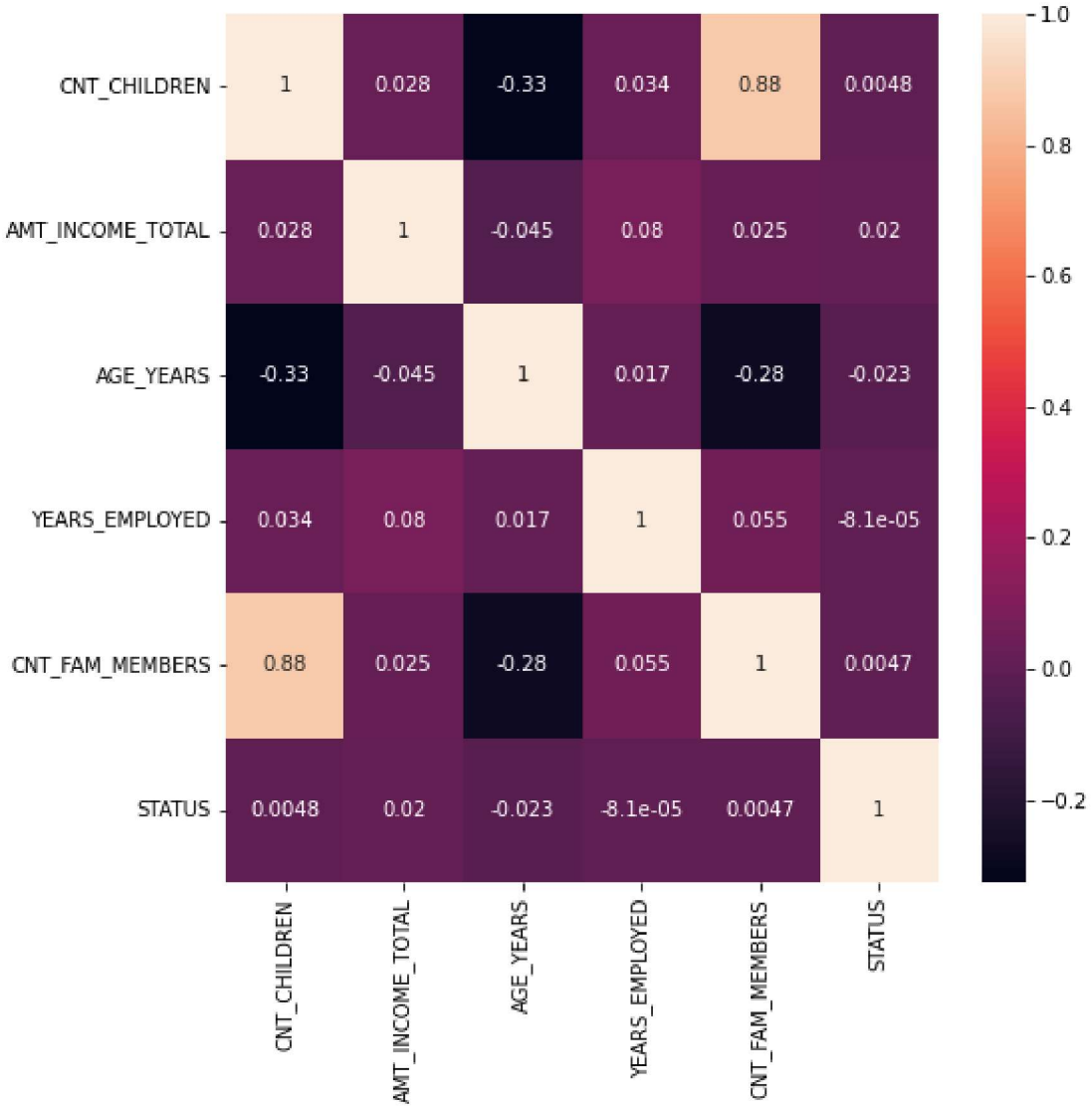
	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_I
0	M	Y	Y	0	427500.0	
1	M	Y	Y	0	112500.0	
2	F	N	Y	0	270000.0	Comme

	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_I
3	F	N	Y	0	283500.0	
4	M	Y	Y	0	270000.0	

Heatmap

In [332…

```
# The below heatmap shows that, there are no such column / Feature which is highly co-r
# wherever there is 1 or light colors, there is high coorelation between the variables
plt.figure(figsize = (8,8))
sns.heatmap(Merge_data1.corr(), annot=True)
plt.show()
```

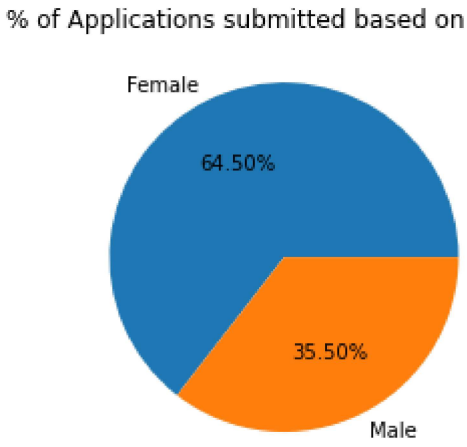


pie charts

In [333…

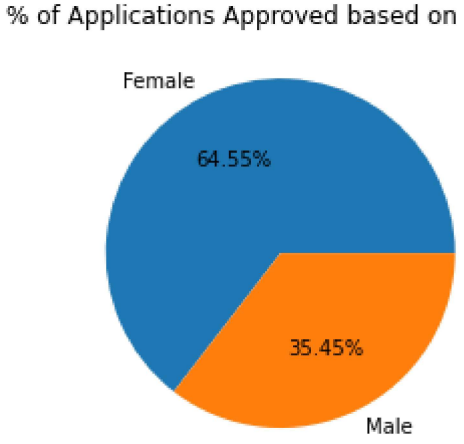
```
## The below pie chart shows that the applications which are submitted for the credit c
plt.pie(Merge_data1['CODE_GENDER'].value_counts(), labels=['Female', 'Male'], autopct='
```

```
plt.title('% of Applications submitted based on Gender')
plt.show()
```



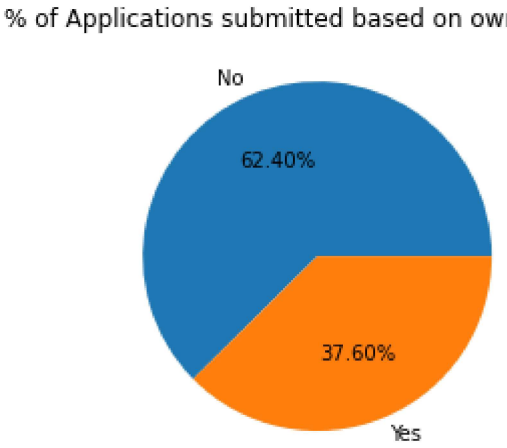
In [334…

```
## The below pie charts shows that the applications which are approved for the credit c
plt.pie(Merge_data1[Merge_data1['STATUS']==0]['CODE_GENDER'].value_counts(), labels=['F
plt.title('% of Applications Approved based on Gender')
plt.show()
```



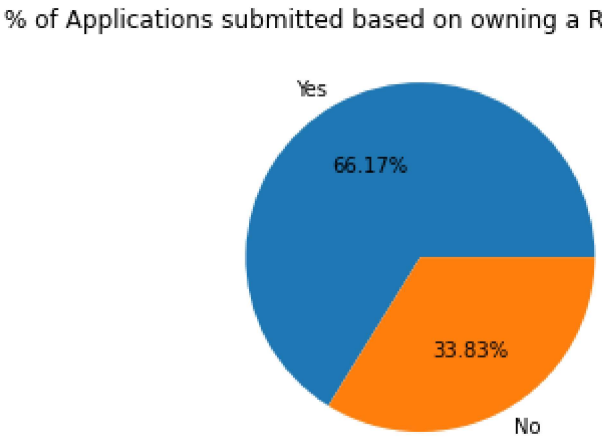
In [335…

```
## The below pie charts shows that the majority of customers's who applied for credit c
plt.pie(Merge_data1['FLAG_OWN_CAR'].value_counts(), labels=['No', 'Yes'], autopct='%1.2
plt.title('% of Applications submitted based on owning a Car')
plt.show()
```



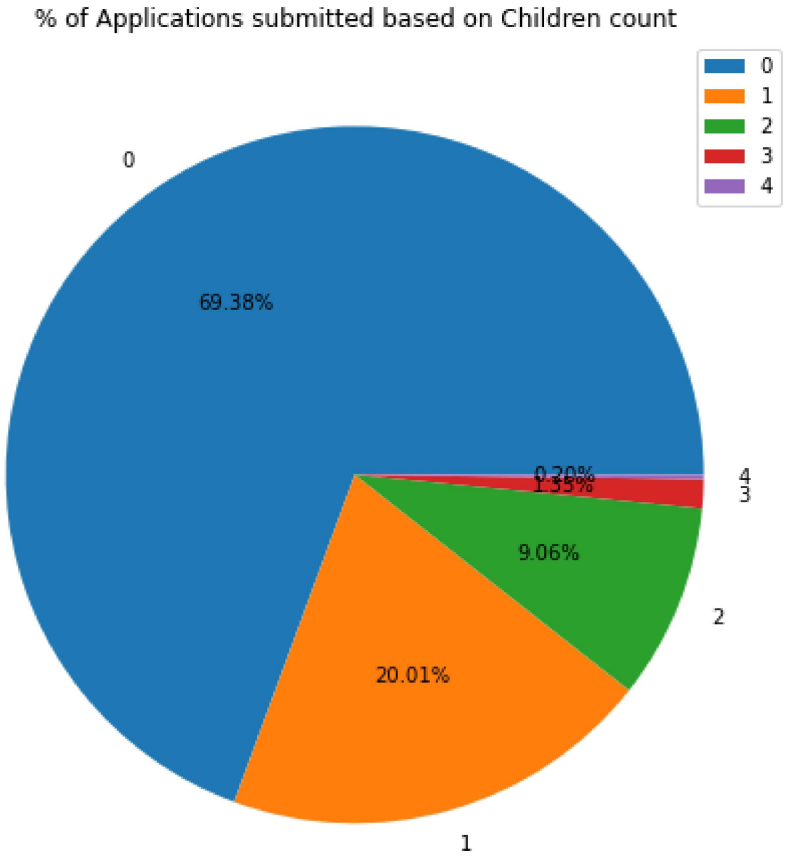
In [336…

```
# The below pie charts shows that the majority of customer's who applied for credit car
plt.pie(Merge_data1['FLAG_OWN_REALTY'].value_counts(), labels=['Yes','No'], autopct='%1
plt.title('% of Applications submitted based on owning a Real estate property')
plt.show()
```



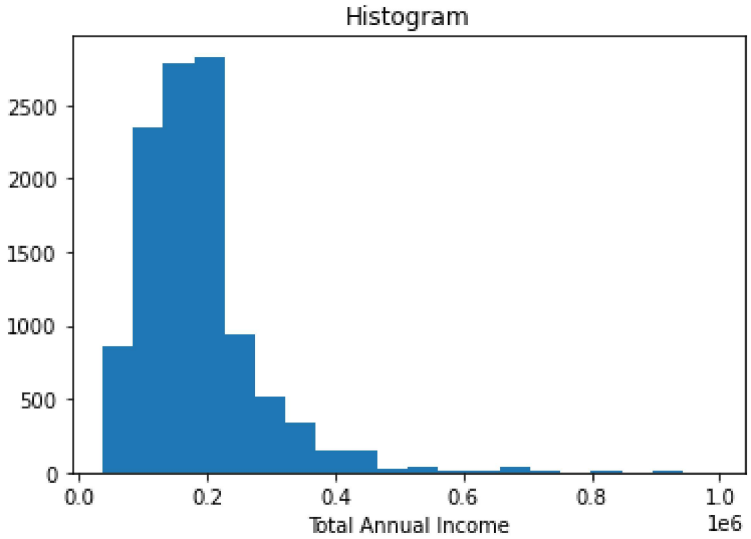
In [337…

```
##  The below pie charts shows that the majority of customer's who applied for credit c
plt.figure(figsize = (8,8))
plt.pie(Merge_data1['CNT_CHILDREN'].value_counts(), labels=Merge_data1['CNT_CHILDREN'].
plt.title('% of Applications submitted based on Children count')
plt.legend()
plt.show()
```



In [338…

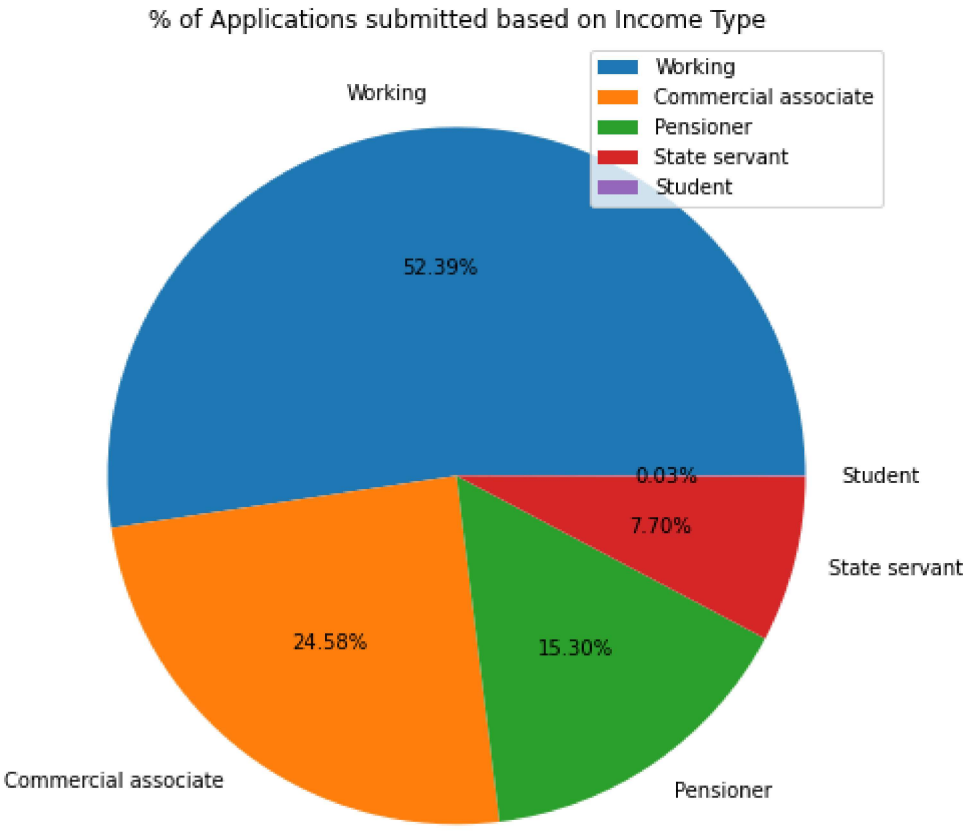
```
## The below graph shows that the majority of customer's who applied for credit card th
plt.hist(Merge_data1['AMT_INCOME_TOTAL'], bins=20)
plt.xlabel('Total Annual Income')
plt.title('Histogram')
plt.show()
```



In [339…

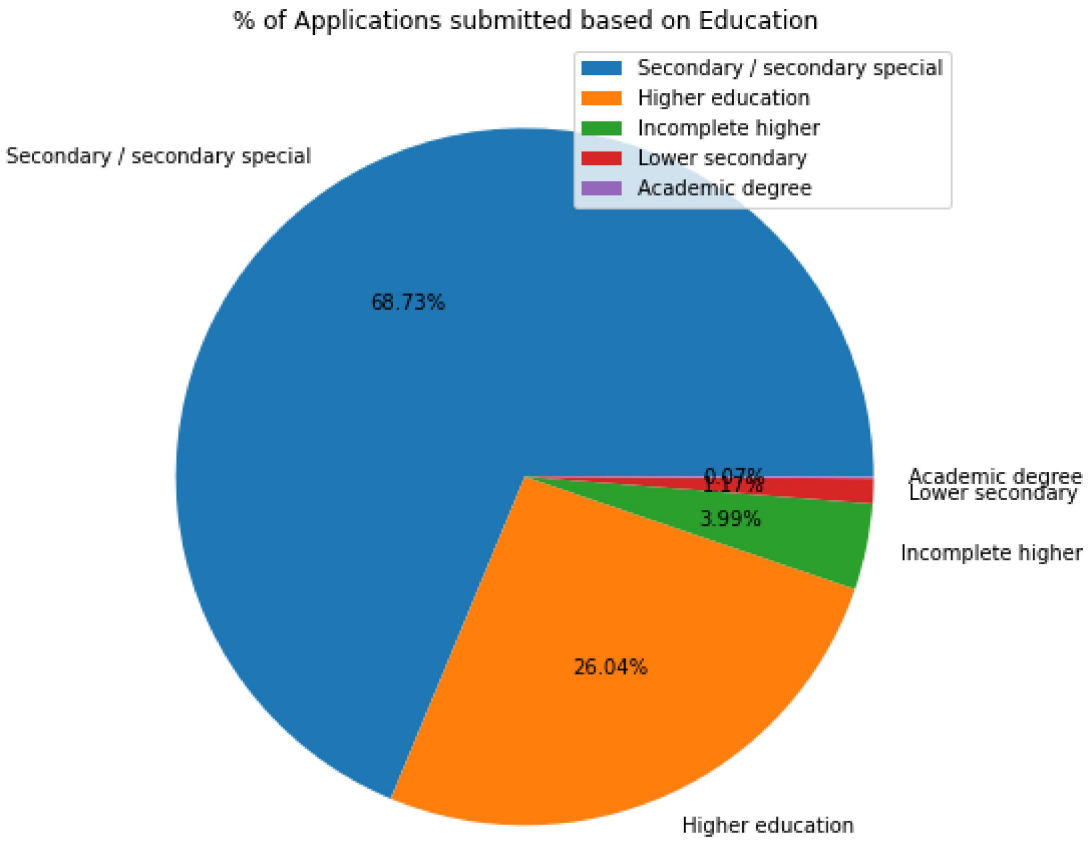
```
## The below pie charts shows that the majority of customer's who applied for credit ca
plt.figure(figsize = (8,8))
plt.pie(Merge_data1['NAME_INCOME_TYPE'].value_counts(), labels=Merge_data1['NAME_INCOME
plt.title('% of Applications submitted based on Income Type')
```

```
plt.legend()
plt.show()
```



In [340…

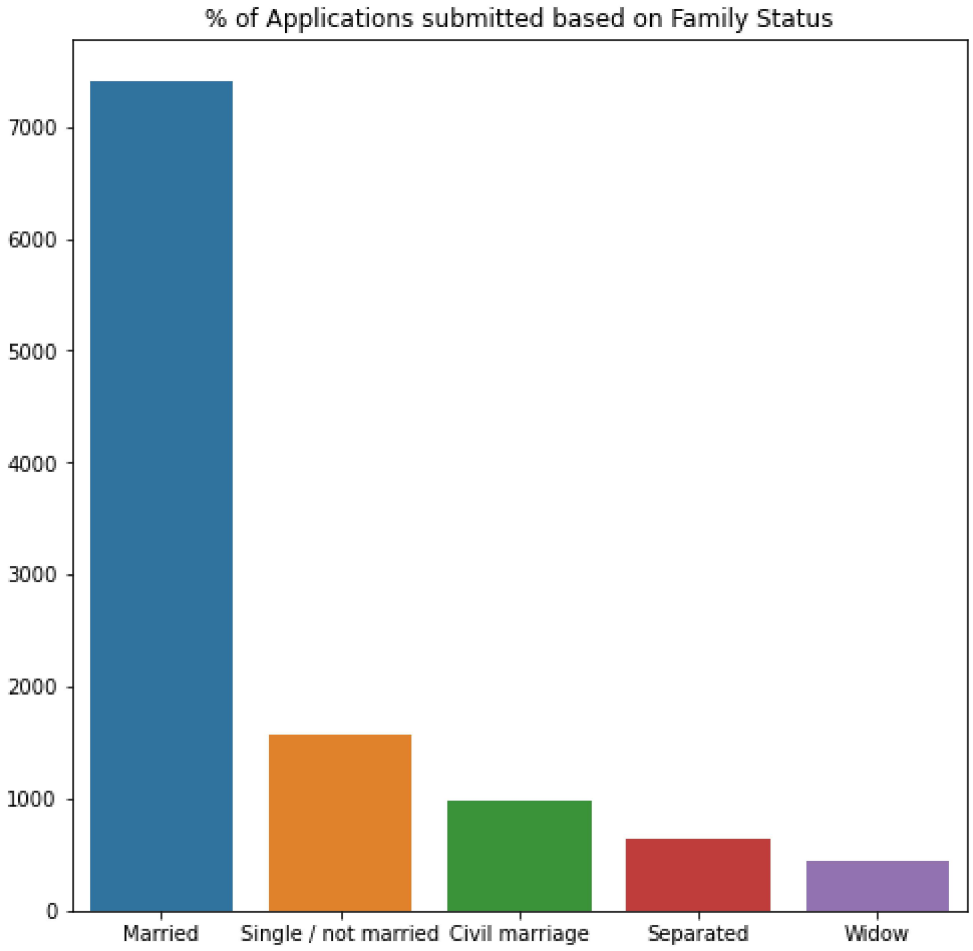
```
## The below pie charts shows that the majority of customer's who applied for credit ca
plt.figure(figsize=(8,8))
plt.pie(Merge_data1['NAME_EDUCATION_TYPE'].value_counts(), labels=Merge_data1['NAME_EDU
plt.title('% of Applications submitted based on Education')
plt.legend()
plt.show()
```



In [341…

```
## The below graph shows that the majority of customer's who applied for credit card ar
plt.figure(figsize=(8,8))
sns.barplot(Merge_data1['NAME_FAMILY_STATUS'].value_counts().index, Merge_data1['NAME_F
plt.title('% of Applications submitted based on Family Status')
plt.show()
```

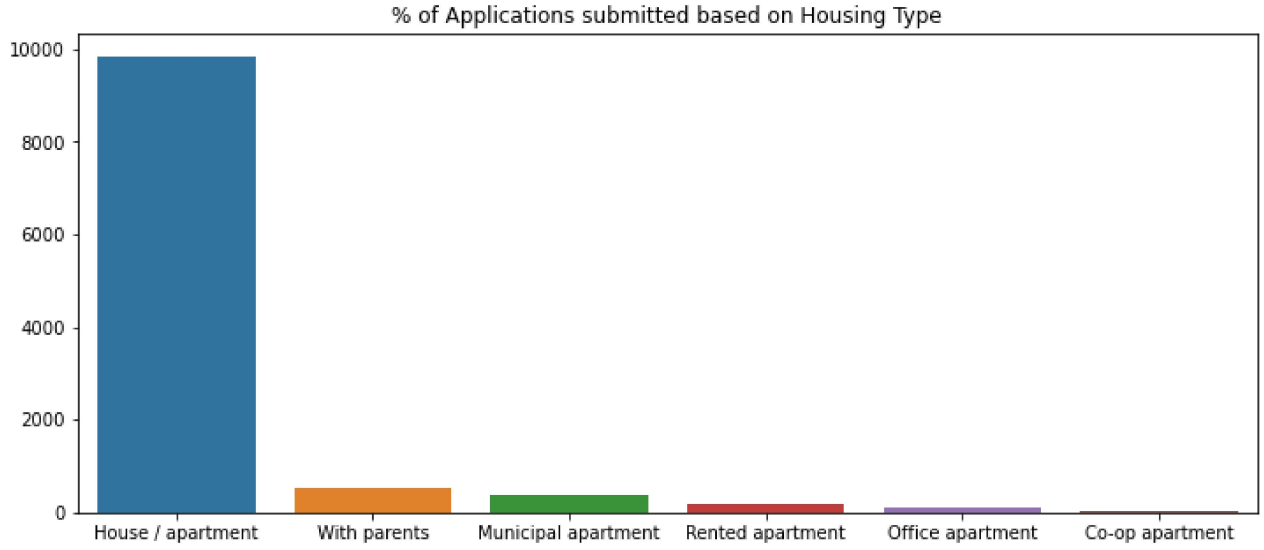
```
C:\Users\boddu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variables as keyword args: x, y. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```



In [342…

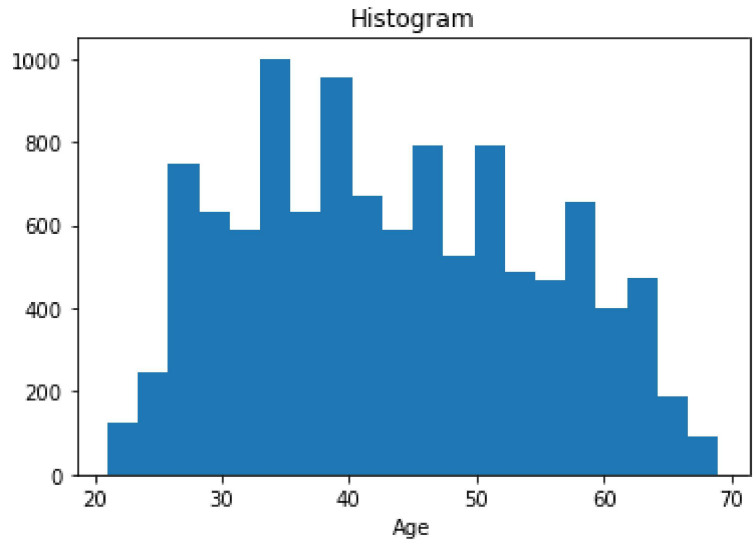
```
## The below graph shows that the majority of customer's who applied for credit card li
plt.figure(figsize=(12,5))
sns.barplot(Merge_data1['NAME_HOUSING_TYPE'].value_counts().index, Merge_data1['NAME_HO
plt.title('% of Applications submitted based on Housing Type')
plt.show()
```

```
C:\Users\boddu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variables as keyword args: x, y. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```



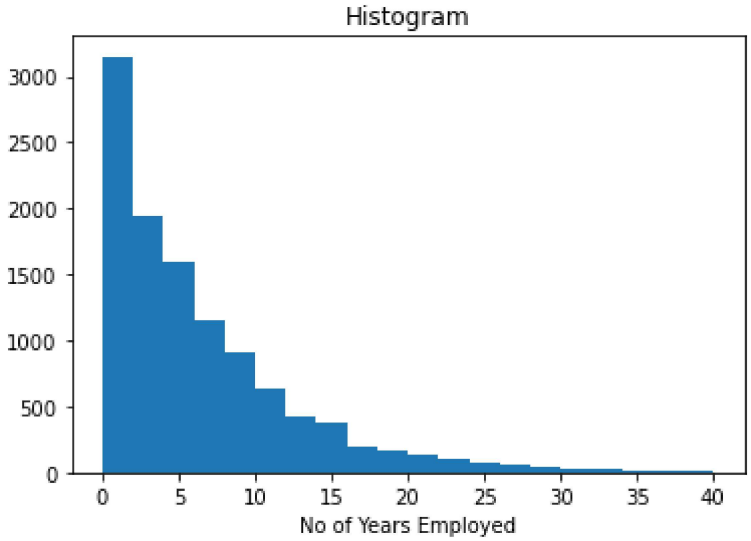
In [343…

```
## The below graph shows that the majority of customer's who applied for credit card ar
plt.hist(Merge_data1['AGE_YEARS'], bins=20)
plt.xlabel('Age')
plt.title('Histogram')
plt.show()
```



In [344…

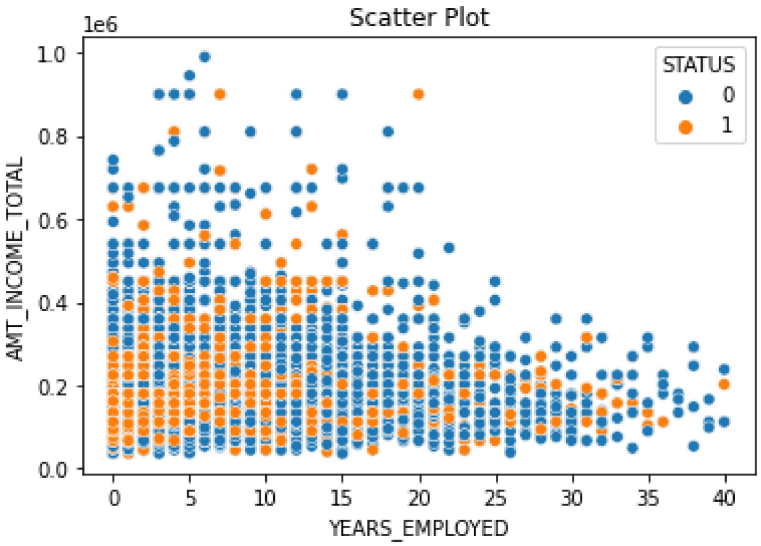
```
## The below graph shows that the majority of customer's who applied for credit card ar
plt.hist(Merge_data1['YEARS_EMPLOYED'], bins=20)
plt.xlabel('No of Years Employed')
plt.title('Histogram')
plt.show()
```



In [345…

```
## The below graph shows that the majority of customer's who applied for credit card
## are rejected if Total income & years of Employment is less
sns.scatterplot(Merge_data1['YEARS_EMPLOYED'], Merge_data1['AMT_INCOME_TOTAL'], hue= Me
plt.title('Scatter Plot')
plt.show()
```

```
C:\Users\boddu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variables as keyword args: x, y. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```



Feature Selection

In [346…

```
Merge_data1.head()
```

Out[346…

	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_I
0	M	Y	Y	0	427500.0	

	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_I
1	M	Y	Y	0	112500.0	
2	F	N	Y	0	270000.0	Comme
3	F	N	Y	0	283500.0	
4	M	Y	Y	0	270000.0	

In [347…

```
check_col = Merge_data1.columns[(Merge_data1.dtypes =='object').values].tolist()
check_col
```

Out[347…

```
['CODE_GENDER',
 'FLAG_OWN_CAR',
 'FLAG_OWN_REALTY',
 'NAME_INCOME_TYPE',
 'NAME_EDUCATION_TYPE',
 'NAME_FAMILY_STATUS',
 'NAME_HOUSING_TYPE']
```

In [348…

```
## Converting all the Non-Numerical Columns into Numerical columns
from sklearn.preprocessing import LabelEncoder

for column in check_col:
        globals()['LE_{}'.format(column)] = LabelEncoder()
        Merge_data1[column] = globals()['LE_{}'.format(column)].fit_transform(Merge_dat
Merge_data1.head()
```

Out[348…

	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_I
0	1	1	1	0	427500.0	
1	1	1	1	0	112500.0	
2	0	0	1	0	270000.0	
3	0	0	1	0	283500.0	
4	1	1	1	0	270000.0	

In [349…

```
for column in check_col:
    print(column , "  : ", globals()['LE_{}'.format(column)].classes_)
```

```
CODE_GENDER   :  ['F' 'M']
FLAG_OWN_CAR   :  ['N' 'Y']
FLAG_OWN_REALTY   :  ['N' 'Y']
NAME_INCOME_TYPE   :  ['Commercial associate' 'Pensioner' 'State servant' 'Student' 'Wor
king']
NAME_EDUCATION_TYPE   :  ['Academic degree' 'Higher education' 'Incomplete higher'
 'Lower secondary' 'Secondary / secondary special']
NAME_FAMILY_STATUS   :  ['Civil marriage' 'Married' 'Separated' 'Single / not married'
'Widow']
```

```
NAME_HOUSING_TYPE   :  ['Co-op apartment' 'House / apartment' 'Municipal apartment'
 'Office apartment' 'Rented apartment' 'With parents']
```

In [350…

```
# Checking for correlation between variables
Merge_data1.corr()
```

Out[350…

	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_IN
CODE_GENDER	1.000000	0.348307	-0.052647	0.052351	
FLAG_OWN_CAR	0.348307	1.000000	-0.000126	0.087407	
FLAG_OWN_REALTY	-0.052647	-0.000126	1.000000	0.001740	
CNT_CHILDREN	0.052351	0.087407	0.001740	1.000000	
AMT_INCOME_TOTAL	0.199358	0.218026	0.036549	0.027806	
NAME_INCOME_TYPE	0.072725	0.030008	-0.034673	0.086772	
NAME_EDUCATION_TYPE	0.013080	-0.085706	0.003771	-0.025059	
NAME_FAMILY_STATUS	-0.075315	-0.125707	-0.011871	-0.166742	
NAME_HOUSING_TYPE	0.052185	-0.011203	-0.178070	0.010091	
AGE_YEARS	-0.157507	-0.106634	0.121351	-0.326642	
YEARS_EMPLOYED	-0.043657	0.006570	-0.011665	0.033979	
CNT_FAM_MEMBERS	0.078853	0.138978	0.009698	0.884676	
STATUS	0.002039	-0.003826	-0.022887	0.004820	

In [351…

```
Property = Merge_data1.drop(['STATUS'], axis=1)
Identification = Merge_data1['STATUS']
```

In [352…

```
Property.head()
```

Out[352…

	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_I
0	1	1	1	0	427500.0	
1	1	1	1	0	112500.0	
2	0	0	1	0	270000.0	
3	0	0	1	0	283500.0	
4	1	1	1	0	270000.0	

In [353…

```
Identification.head()
```

Out[353…

```
0    1
1    0
2    0
```

```
3    0
4    0
Name: STATUS, dtype: int32
```

Train and Test the model

In [354…

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(Property, Identification, test_size
```

Logistic Regression Model

In [355…

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

logistic_modl = LogisticRegression()
logistic_modl.fit(X_train, Y_train)

print('Accuracy of Logistic Model : ', logistic_modl.score(X_test, Y_test)*100, '%')

prdictn = logistic_modl.predict(X_test)
print('\n The Confusion matrix is as follows :')
print(confusion_matrix(Y_test, prdictn))

print('\n The Classification report is as follows:')
print(classification_report(Y_test, prdictn))
```

```
Accuracy of Logistic Model :  78.84267631103074 %

 The Confusion matrix is as follows :
[[1744    0]
 [ 468    0]]

 The Classification report is as follows:
              precision    recall  f1-score   support

           0       0.79      1.00      0.88      1744
           1       0.00      0.00      0.00       468

    accuracy                           0.79      2212
   macro avg       0.39      0.50      0.44      2212
weighted avg       0.62      0.79      0.70      2212

C:\Users\boddu\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Unde
finedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\boddu\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Unde
finedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\boddu\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Unde
finedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Decision Tree classification Model

In [356…

```
from sklearn.tree import DecisionTreeClassifier

DTCM_modl = DecisionTreeClassifier(max_depth=12,min_samples_split=8)

DTCM_modl.fit(X_train, Y_train)

print(' The accuracy of Decision Tree Model is as follows : ', DTCM_modl.score(X_test,

prdictn = DTCM_modl.predict(X_test)
print('\n The Confusion matrix is as follows :')
print(confusion_matrix(Y_test, prdictn))

print('\n The Classification report is as follows:')
print(classification_report(Y_test, prdictn))
```

```
 The accuracy of Decision Tree Model is as follows :  73.50813743218806 %

 The Confusion matrix is as follows :
[[1615  129]
 [ 457   11]]

 The Classification report is as follows:
              precision    recall  f1-score   support

           0       0.78      0.93      0.85      1744
           1       0.08      0.02      0.04       468

    accuracy                           0.74      2212
   macro avg       0.43      0.47      0.44      2212
weighted avg       0.63      0.74      0.68      2212
```

In [ ]:

From the above Machine Learning Models Logistic Regression Model HAS LITTEL MORE ACCURACY THEN Decision Tree classification Model

In [ ]:

In [ ]:

In [ ]: