In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import pyplot
%matplotlib inline
import sklearn
import scipy
import asgl
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression, Lasso
from optbinning.scorecard import plot_auc_roc, plot_cap, plot_ks
from scipy import stats
import statsmodels.api as sm
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score, roc_auc_score, precision_score, recall_score
import statsmodels.stats.proportion as proportion
from sklearn.preprocessing import OneHotEncoder
from optbinning import OptimalBinning
from optbinning import OptimalBinningSketch
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier
from scipy.stats import linregress
import pingouin as pg
from varclushi import VarClusHi
from mlxtend.feature_selection import SequentialFeatureSelector
from sklearn import linear_model
from yellowbrick.model_selection import RFECV
from sklearn.svm import SVC
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from stepwise_regression import step_reg
import xgboost as xgb
from sklearn.svm import LinearSVC
from xgboost import XGBClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import plot_importance
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
from sklearn.linear_model import ElasticNet, ElasticNetCV
from sklearn.linear_model import Ridge, RidgeCV, RidgeClassifier
from sklearn.linear_model import LassoLarsIC, LassoCV, LassoLarsCV
from sklearn.pipeline import make_pipeline
from sklearn.pipeline import Pipeline
from sklearn.linear_model import RANSACRegressor
from sklearn.linear_model import HuberRegressor
from sklearn import ensemble
from sklearn.linear_model import SGDRegressor
from sklearn.linear_model import SGDClassifier
from sklearn.preprocessing import RobustScaler
import lightgbm as lgb
from xgboost import cv
from sklearn.model_selection import RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier
import statsmodels.formula.api as smf
from sklearn.metrics import log_loss
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import mean_squared_error
from sklearn.compose import ColumnTransformer
import warnings
```

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

```
D:\Anaconda\lib\site-packages\outdated\utils.py:14: OutdatedPackageWarning: The package pingo
uin is out of date. Your version is 0.5.3, the latest is 0.5.4.
Set the environment variable OUTDATED_IGNORE=1 to disable these warnings.
  return warn(
```

In [2]:
```
df= pd.read_csv("D:\Documentos\CreditDataForCheco2.csv")
df
```

Out[2]:

	OBS#	CHK_ACCT	DURATION	HISTORY	NEW_CAR	USED_CAR	FURNITURE	RADIO/TV	EDUCATION	RETRA
0	1	0	6	4	0	0	0	1	0	
1	2	1	48	2	0	0	0	1	0	
2	3	3	12	4	0	0	0	0	1	
3	4	0	42	2	0	0	1	0	0	
4	5	0	24	3	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
995	996	3	12	2	0	0	1	0	0	
996	997	0	30	2	0	1	0	0	0	
997	998	3	12	2	0	0	0	1	0	
998	999	0	45	2	0	0	0	1	0	
999	1000	1	45	4	0	1	0	0	0	

1000 rows × 32 columns

In [3]:
```
df.describe()
```

Out[3]:

	OBS#	CHK_ACCT	DURATION	HISTORY	NEW_CAR	USED_CAR	FURNITURE	RADIO/TV	EDU
count	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000	1000.000000	1000.000000	100C
mean	500.500000	1.577000	20.903000	2.54500	0.234000	0.103000	0.181000	0.280000	C
std	288.819436	1.257638	12.058814	1.08312	0.423584	0.304111	0.385211	0.449224	C
min	1.000000	0.000000	4.000000	0.00000	0.000000	0.000000	0.000000	0.000000	C
25%	250.750000	0.000000	12.000000	2.00000	0.000000	0.000000	0.000000	0.000000	C
50%	500.500000	1.000000	18.000000	2.00000	0.000000	0.000000	0.000000	0.000000	C
75%	750.250000	3.000000	24.000000	4.00000	0.000000	0.000000	0.000000	1.000000	C
max	1000.000000	3.000000	72.000000	4.00000	1.000000	1.000000	1.000000	1.000000	1

8 rows × 32 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 32 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   OBS#              1000 non-null   int64
 1   CHK_ACCT          1000 non-null   int64
 2   DURATION          1000 non-null   int64
 3   HISTORY           1000 non-null   int64
 4   NEW_CAR           1000 non-null   int64
 5   USED_CAR          1000 non-null   int64
 6   FURNITURE         1000 non-null   int64
 7   RADIO/TV          1000 non-null   int64
 8   EDUCATION         1000 non-null   int64
 9   RETRAINING        1000 non-null   int64
 10  AMOUNT            1000 non-null   int64
 11  SAV_ACCT          1000 non-null   int64
 12  EMPLOYMENT        1000 non-null   int64
 13  INSTALL_RATE      1000 non-null   int64
 14  MALE_DIV          1000 non-null   int64
 15  MALE_SINGLE       1000 non-null   int64
 16  MALE_MAR_or_WID   1000 non-null   int64
 17  CO-APPLICANT      1000 non-null   int64
 18  GUARANTOR         1000 non-null   int64
 19  PRESENT_RESIDENT  1000 non-null   int64
 20  REAL_ESTATE       1000 non-null   int64
 21  PROP_UNKN_NONE    1000 non-null   int64
 22  AGE               1000 non-null   int64
 23  OTHER_INSTALL     1000 non-null   int64
 24  RENT              1000 non-null   int64
 25  OWN_RES           1000 non-null   int64
 26  NUM_CREDITS       1000 non-null   int64
 27  JOB               1000 non-null   int64
 28  NUM_DEPENDENTS    1000 non-null   int64
 29  TELEPHONE         1000 non-null   int64
 30  FOREIGN           1000 non-null   int64
 31  DEFAULT           1000 non-null   int64
dtypes: int64(32)
memory usage: 250.1 KB
```

In [5]:
```
df.isnull().sum()
#All variables are good, none has null data or blanck cells
```

Out[5]:
```
OBS#                0
CHK_ACCT            0
DURATION            0
HISTORY             0
NEW_CAR             0
USED_CAR            0
FURNITURE           0
RADIO/TV            0
EDUCATION           0
RETRAINING          0
AMOUNT              0
SAV_ACCT            0
EMPLOYMENT          0
INSTALL_RATE        0
MALE_DIV            0
MALE_SINGLE         0
MALE_MAR_or_WID     0
CO-APPLICANT        0
GUARANTOR           0
PRESENT_RESIDENT    0
REAL_ESTATE         0
PROP_UNKN_NONE      0
AGE                 0
OTHER_INSTALL       0
RENT                0
OWN_RES             0
NUM_CREDITS         0
JOB                 0
NUM_DEPENDENTS      0
TELEPHONE           0
FOREIGN             0
DEFAULT             0
dtype: int64
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
OBS#               0
CHK_ACCT           0
DURATION           0
HISTORY            0
NEW_CAR            0
USED_CAR           0
FURNITURE          0
RADIO/TV           0
EDUCATION          0
RETRAINING         0
AMOUNT             0
SAV_ACCT           0
EMPLOYMENT         0
INSTALL_RATE       0
MALE_DIV           0
MALE_SINGLE        0
MALE_MAR_or_WID    0
CO-APPLICANT       0
GUARANTOR          0
PRESENT_RESIDENT   0
REAL_ESTATE        0
PROP_UNKN_NONE     0
AGE                0
OTHER_INSTALL      0
RENT               0
OWN_RES            0
NUM_CREDITS        0
JOB                0
NUM_DEPENDENTS     0
TELEPHONE          0
FOREIGN            0
DEFAULT            0
dtype: int64
```

In [7]:
```
df= df.drop(columns=['OBS#'])
df.reset_index(inplace=True)
df= df.drop(columns=['index'])
df
```

Out[7]:

	CHK_ACCT	DURATION	HISTORY	NEW_CAR	USED_CAR	FURNITURE	RADIO/TV	EDUCATION	RETRAINING
0	0	6	4	0	0	0	1	0	0
1	1	48	2	0	0	0	1	0	0
2	3	12	4	0	0	0	0	1	0
3	0	42	2	0	0	1	0	0	0
4	0	24	3	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
995	3	12	2	0	0	1	0	0	0
996	0	30	2	0	1	0	0	0	0
997	3	12	2	0	0	0	1	0	0
998	0	45	2	0	0	0	1	0	0
999	1	45	4	0	1	0	0	0	0

1000 rows × 31 columns

In [8]:
```
df['PRESENT_RESIDENT']= df['PRESENT_RESIDENT'].replace({1:0,2:1,3:2,4:3})
df.rename(columns={'CO-APPLICANT': 'CO_APPLICANT','RADIO/TV': 'RADIO_TV'}, inplace=True)
df.rename(columns={'CO_APPLICANT': 'GUARANTOR','GUARANTOR': 'CO_APPLICANT'}, inplace=True)
```

In [9]:
```
def type(df,column,types):
    df[column]= df[column].astype(types)
    return
```

In [10]:
```
type(df,'CHK_ACCT',"category")
type(df,'HISTORY', "category")
type(df,'SAV_ACCT',"category")
type(df,'EMPLOYMENT',"category")
type(df,'PRESENT_RESIDENT',"category")
type(df,'JOB',"category")

type(df,'NEW_CAR',"uint8")
type(df,'USED_CAR',"uint8")
type(df,'FURNITURE',"uint8")
type(df,'RADIO_TV',"uint8")
type(df,'EDUCATION',"uint8")
type(df,'RETRAINING',"uint8")
type(df,'MALE_DIV',"uint8")
type(df,'MALE_SINGLE',"uint8")
type(df,'MALE_MAR_or_WID',"uint8")
type(df, 'CO_APPLICANT',"uint8")
type(df,'GUARANTOR',"uint8")
type(df,'REAL_ESTATE',"uint8")
type(df,'PROP_UNKN_NONE',"uint8")
type(df, 'OTHER_INSTALL',"uint8")
type(df,'RENT',"uint8")
type(df,'OWN_RES',"uint8")
type(df, 'TELEPHONE',"uint8")
type(df,'FOREIGN',"uint8")

type(df,'DEFAULT',"uint8")
```

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 31 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   CHK_ACCT         1000 non-null   category
 1   DURATION         1000 non-null   int64
 2   HISTORY          1000 non-null   category
 3   NEW_CAR          1000 non-null   uint8
 4   USED_CAR         1000 non-null   uint8
 5   FURNITURE        1000 non-null   uint8
 6   RADIO_TV         1000 non-null   uint8
 7   EDUCATION        1000 non-null   uint8
 8   RETRAINING       1000 non-null   uint8
 9   AMOUNT           1000 non-null   int64
 10  SAV_ACCT         1000 non-null   category
 11  EMPLOYMENT       1000 non-null   category
 12  INSTALL_RATE     1000 non-null   int64
 13  MALE_DIV         1000 non-null   uint8
 14  MALE_SINGLE      1000 non-null   uint8
 15  MALE_MAR_or_WID  1000 non-null   uint8
 16  GUARANTOR        1000 non-null   uint8
 17  CO_APPLICANT     1000 non-null   uint8
 18  PRESENT_RESIDENT 1000 non-null   category
 19  REAL_ESTATE      1000 non-null   uint8
 20  PROP_UNKN_NONE   1000 non-null   uint8
 21  AGE              1000 non-null   int64
 22  OTHER_INSTALL    1000 non-null   uint8
 23  RENT             1000 non-null   uint8
 24  OWN_RES          1000 non-null   uint8
 25  NUM_CREDITS      1000 non-null   int64
 26  JOB              1000 non-null   category
 27  NUM_DEPENDENTS   1000 non-null   int64
 28  TELEPHONE        1000 non-null   uint8
 29  FOREIGN          1000 non-null   uint8
 30  DEFAULT          1000 non-null   uint8
dtypes: category(6), int64(6), uint8(19)
memory usage: 72.6 KB
```

In [12]:
```
def outlier(column):
    q1= df[column].quantile(0.25)
    q3= df[column].quantile(0.72)
    IQR= q3-q1
    outliers= df[column][((df[column]<(q1-3.5*IQR))|(df[column]>(q3+3.5*IQR)))]
    return outliers
```

In [13]:
```
outlier('AGE')
```

Out[13]:
```
Series([], Name: AGE, dtype: int64)
```

In [14]:
```
outlier('AMOUNT')
```

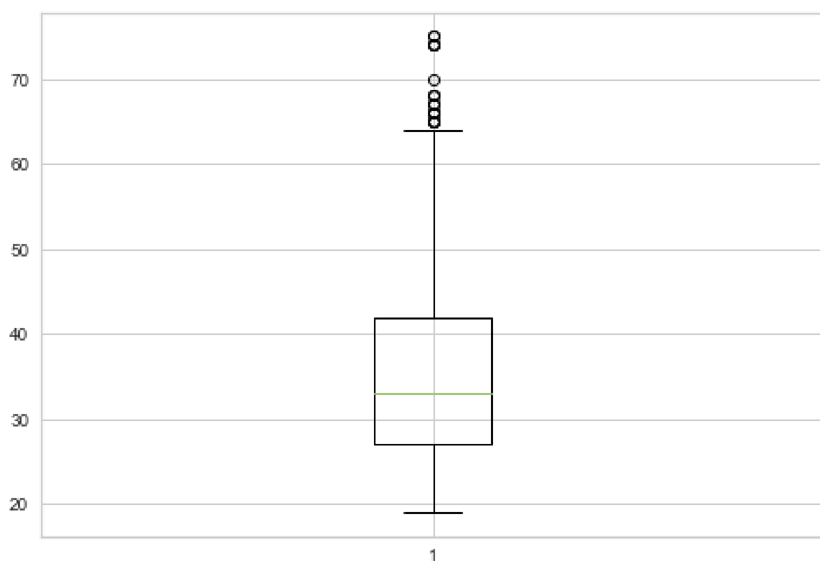
Out[14]:
```
18     12579
63     14421
87     12612
95     15945
236    14555
272    12169
274    11998
373    13756
374    14782
378    14318
381    12976
563    12389
615    12204
637    15653
714    14027
744    14179
763    12680
818    15857
887    15672
915    18424
917    14896
921    12749
Name: AMOUNT, dtype: int64
```

In [15]:
```
outlier('DURATION')
```

Out[15]:
```
677    72
Name: DURATION, dtype: int64
```

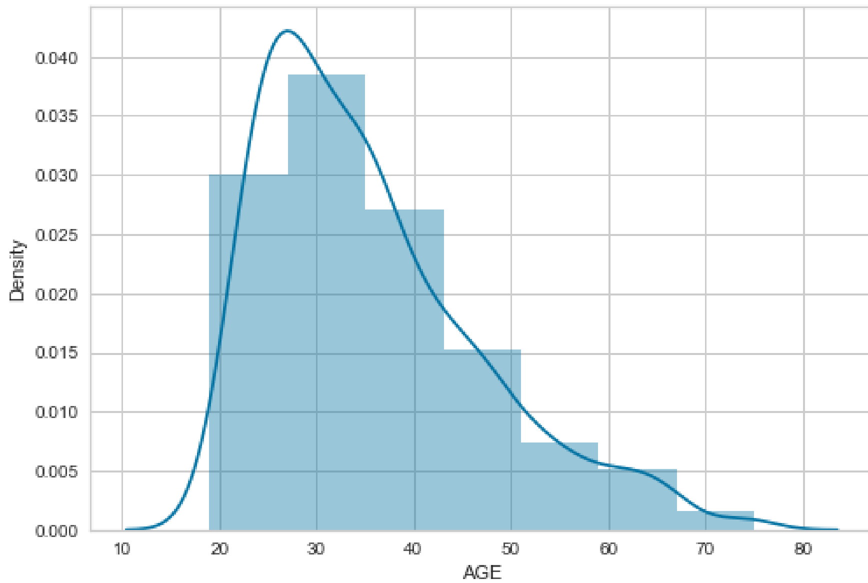
In [16]:
```
plt.boxplot(df['AGE'])
```

Out[16]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1ceb1736730>,
  <matplotlib.lines.Line2D at 0x1ceb1736a00>],
 'caps': [<matplotlib.lines.Line2D at 0x1ceb1736d90>,
  <matplotlib.lines.Line2D at 0x1ceb1751160>],
 'boxes': [<matplotlib.lines.Line2D at 0x1ceb17362b0>],
 'medians': [<matplotlib.lines.Line2D at 0x1ceb17514f0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1ceb1751880>],
 'means': []}
```



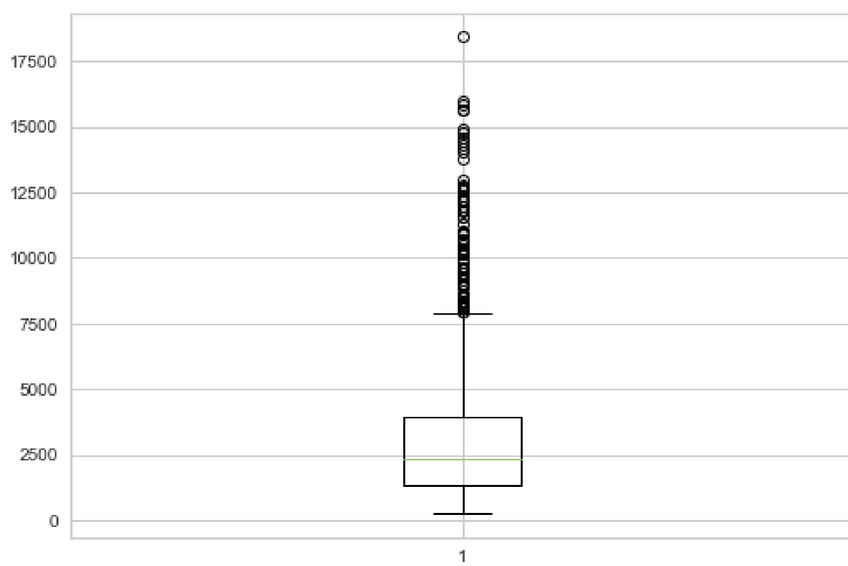
In [17]:
```
sns.distplot(df['AGE'],bins=7)
```

Out[17]:
```
<AxesSubplot:xlabel='AGE', ylabel='Density'>
```



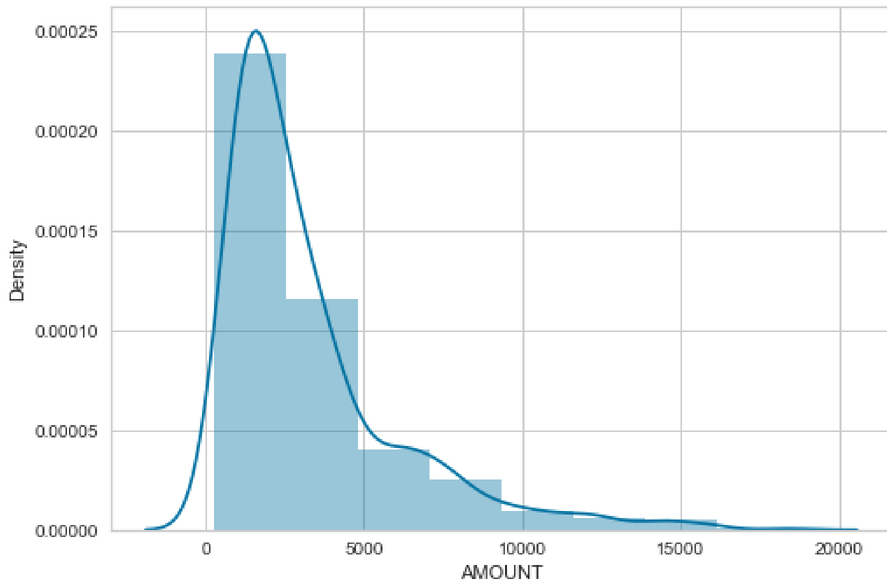
In [18]:
```
plt.boxplot(df['AMOUNT'])
```

Out[18]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1ceb18c9a00>,
  <matplotlib.lines.Line2D at 0x1ceb18c9d90>],
 'caps': [<matplotlib.lines.Line2D at 0x1ceb18d5160>,
  <matplotlib.lines.Line2D at 0x1ceb18d54f0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1ceb18c9640>],
 'medians': [<matplotlib.lines.Line2D at 0x1ceb18d5880>],
 'fliers': [<matplotlib.lines.Line2D at 0x1ceb18d5c10>],
 'means': []}
```



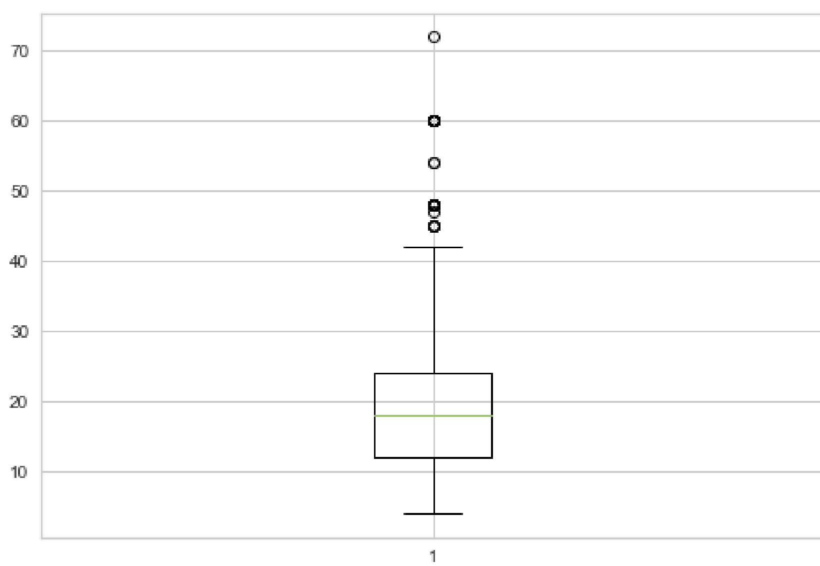
In [19]:
```
sns.distplot(df['AMOUNT'],bins=8)
```

Out[19]:
```
<AxesSubplot:xlabel='AMOUNT', ylabel='Density'>
```



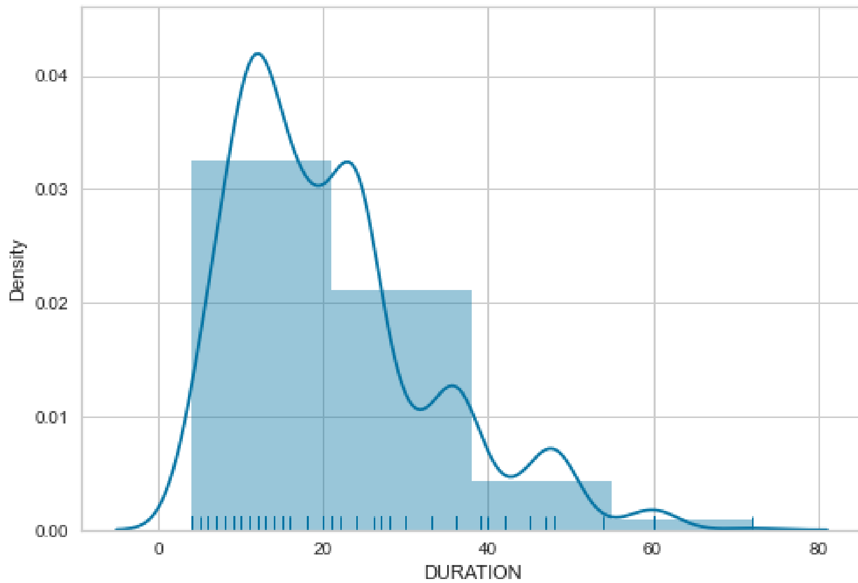
In [20]:
```
plt.boxplot(df['DURATION'])
```

Out[20]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1ceb19a6940>,
  <matplotlib.lines.Line2D at 0x1ceb19a6cd0>],
 'caps': [<matplotlib.lines.Line2D at 0x1ceb19b30a0>,
  <matplotlib.lines.Line2D at 0x1ceb19b3430>],
 'boxes': [<matplotlib.lines.Line2D at 0x1ceb19a65b0>],
 'medians': [<matplotlib.lines.Line2D at 0x1ceb19b37f0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1ceb19b3b80>],
 'means': []}
```



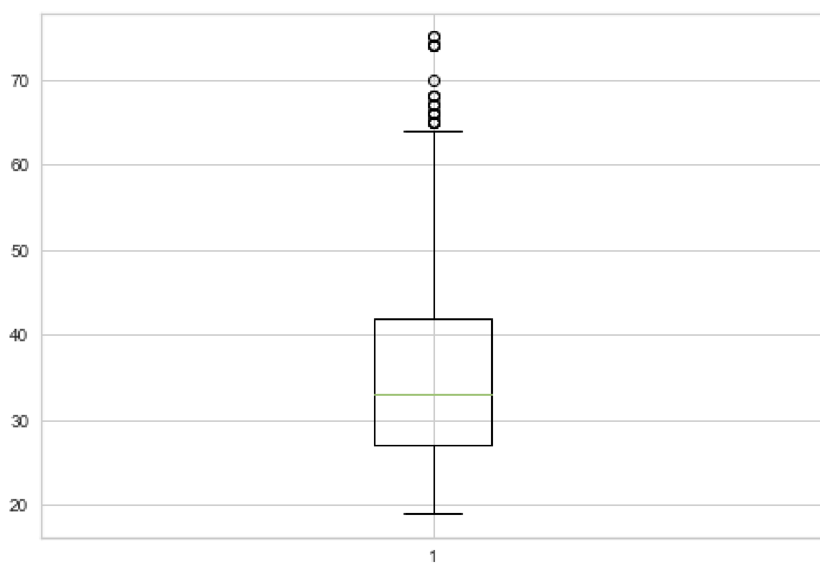
In [21]:
```
sns.distplot(df['DURATION'],bins=4,rug=True)
```

Out[21]:
```
<AxesSubplot:xlabel='DURATION', ylabel='Density'>
```



In [22]:
```
plt.boxplot(df['AGE'])
```

Out[22]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1ceb1b06ca0>,
  <matplotlib.lines.Line2D at 0x1ceb1b18070>],
 'caps': [<matplotlib.lines.Line2D at 0x1ceb1b18430>,
  <matplotlib.lines.Line2D at 0x1ceb1b187c0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1ceb1b06910>],
 'medians': [<matplotlib.lines.Line2D at 0x1ceb1b18b50>],
 'fliers': [<matplotlib.lines.Line2D at 0x1ceb1b18f10>],
 'means': []}
```



In [23]:
```
df= df.drop(df[df['AMOUNT']>=11998].index)
df.reset_index(inplace=True)
df= df.drop(columns=['index'])
df
```

Out[23]:

	CHK_ACCT	DURATION	HISTORY	NEW_CAR	USED_CAR	FURNITURE	RADIO_TV	EDUCATION	RETRAINING
0	0	6	4	0	0	0	1	0	0
1	1	48	2	0	0	0	1	0	0
2	3	12	4	0	0	0	0	1	0
3	0	42	2	0	0	1	0	0	0
4	0	24	3	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
973	3	12	2	0	0	1	0	0	0
974	0	30	2	0	1	0	0	0	0
975	3	12	2	0	0	0	1	0	0
976	0	45	2	0	0	0	1	0	0
977	1	45	4	0	1	0	0	0	0

978 rows × 31 columns

In [24]:
```
df= df.drop(df[df['DURATION']==72].index)
df.reset_index(inplace=True)
df= df.drop(columns=['index'])
df
```

Out[24]:

	CHK_ACCT	DURATION	HISTORY	NEW_CAR	USED_CAR	FURNITURE	RADIO_TV	EDUCATION	RETRAINING
0	0	6	4	0	0	0	1	0	0
1	1	48	2	0	0	0	1	0	0
2	3	12	4	0	0	0	0	1	0
3	0	42	2	0	0	1	0	0	0
4	0	24	3	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
972	3	12	2	0	0	1	0	0	0
973	0	30	2	0	1	0	0	0	0
974	3	12	2	0	0	0	1	0	0
975	0	45	2	0	0	0	1	0	0
976	1	45	4	0	1	0	0	0	0

977 rows × 31 columns

In [25]:
```
df= df.drop(df[df['AGE']>69].index)
df.reset_index(inplace=True)
df= df.drop(columns=['index'])
df
```

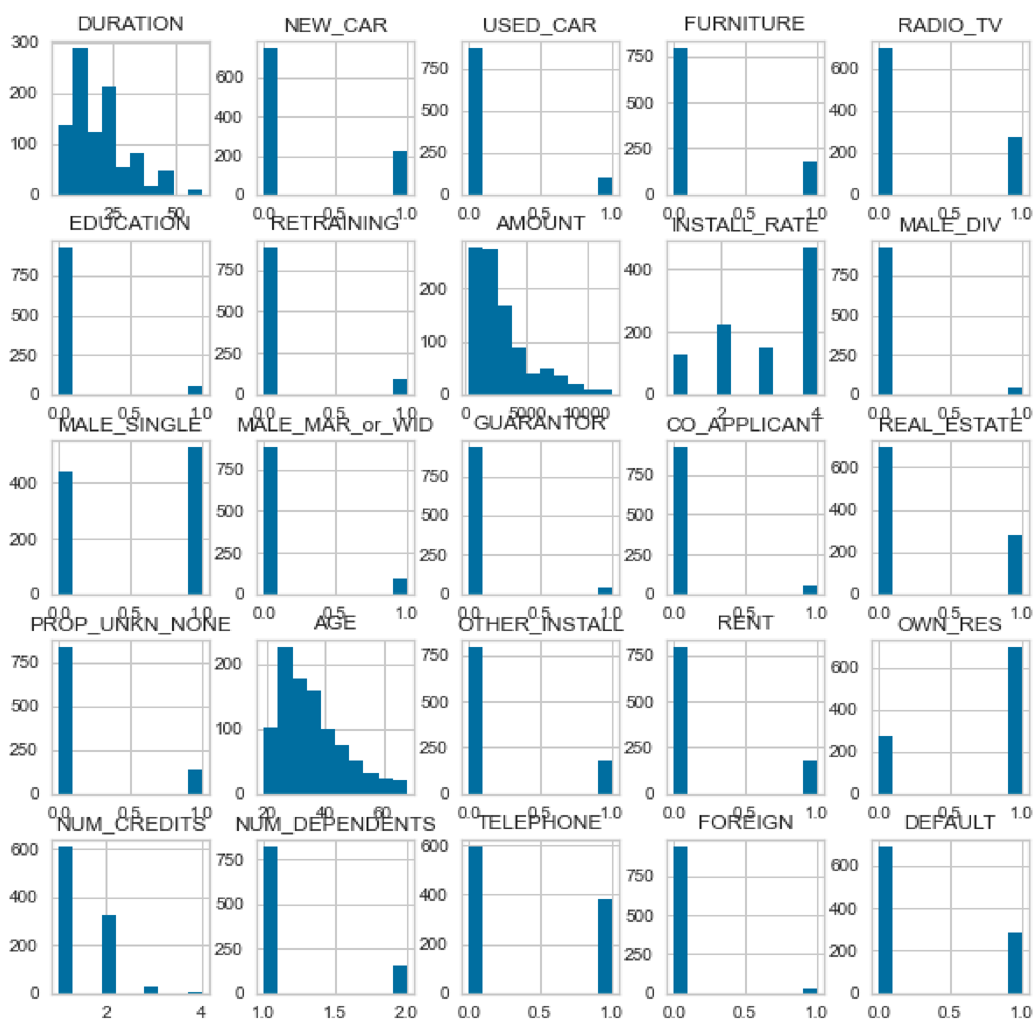
Out[25]:

	CHK_ACCT	DURATION	HISTORY	NEW_CAR	USED_CAR	FURNITURE	RADIO_TV	EDUCATION	RETRAINING
0	0	6	4	0	0	0	1	0	0
1	1	48	2	0	0	0	1	0	0
2	3	12	4	0	0	0	0	1	0
3	0	42	2	0	0	1	0	0	0
4	0	24	3	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
965	3	12	2	0	0	1	0	0	0
966	0	30	2	0	1	0	0	0	0
967	3	12	2	0	0	0	1	0	0
968	0	45	2	0	0	0	1	0	0
969	1	45	4	0	1	0	0	0	0

970 rows × 31 columns

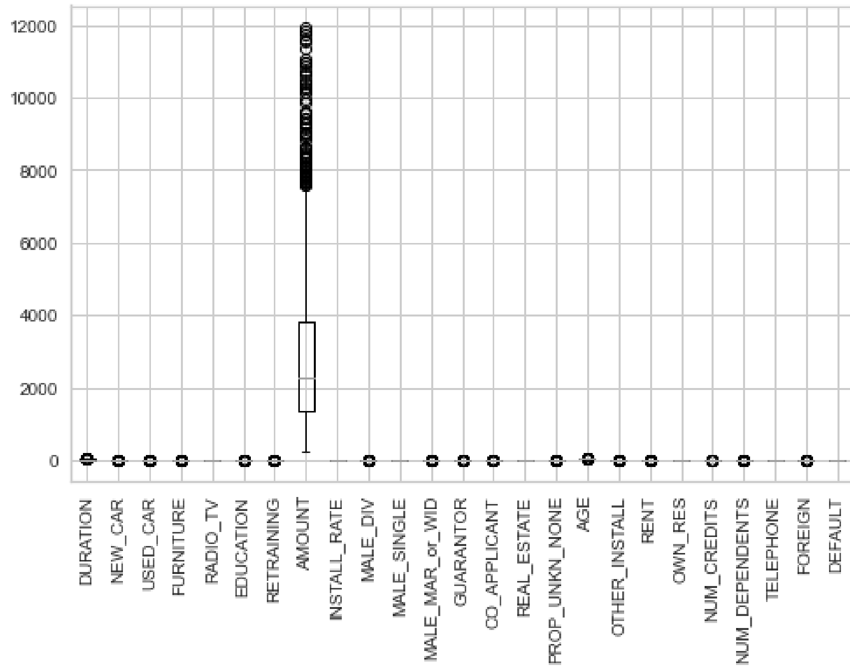
In [29]:
```
df.hist(figsize=(10,10))
```

Out[29]:
```
array([[<AxesSubplot:title={'center':'DURATION'}>,
        <AxesSubplot:title={'center':'NEW_CAR'}>,
        <AxesSubplot:title={'center':'USED_CAR'}>,
        <AxesSubplot:title={'center':'FURNITURE'}>,
        <AxesSubplot:title={'center':'RADIO_TV'}>],
       [<AxesSubplot:title={'center':'EDUCATION'}>,
        <AxesSubplot:title={'center':'RETRAINING'}>,
        <AxesSubplot:title={'center':'AMOUNT'}>,
        <AxesSubplot:title={'center':'INSTALL_RATE'}>,
        <AxesSubplot:title={'center':'MALE_DIV'}>],
       [<AxesSubplot:title={'center':'MALE_SINGLE'}>,
        <AxesSubplot:title={'center':'MALE_MAR_or_WID'}>,
        <AxesSubplot:title={'center':'GUARANTOR'}>,
        <AxesSubplot:title={'center':'CO_APPLICANT'}>,
        <AxesSubplot:title={'center':'REAL_ESTATE'}>],
       [<AxesSubplot:title={'center':'PROP_UNKN_NONE'}>,
        <AxesSubplot:title={'center':'AGE'}>,
        <AxesSubplot:title={'center':'OTHER_INSTALL'}>,
        <AxesSubplot:title={'center':'RENT'}>,
        <AxesSubplot:title={'center':'OWN_RES'}>],
       [<AxesSubplot:title={'center':'NUM_CREDITS'}>,
        <AxesSubplot:title={'center':'NUM_DEPENDENTS'}>,
        <AxesSubplot:title={'center':'TELEPHONE'}>,
        <AxesSubplot:title={'center':'FOREIGN'}>,
        <AxesSubplot:title={'center':'DEFAULT'}>]], dtype=object)
```



In [27]:
```
df.boxplot(figsize=(8,5), rot=90)
```

Out[27]: <AxesSubplot:>



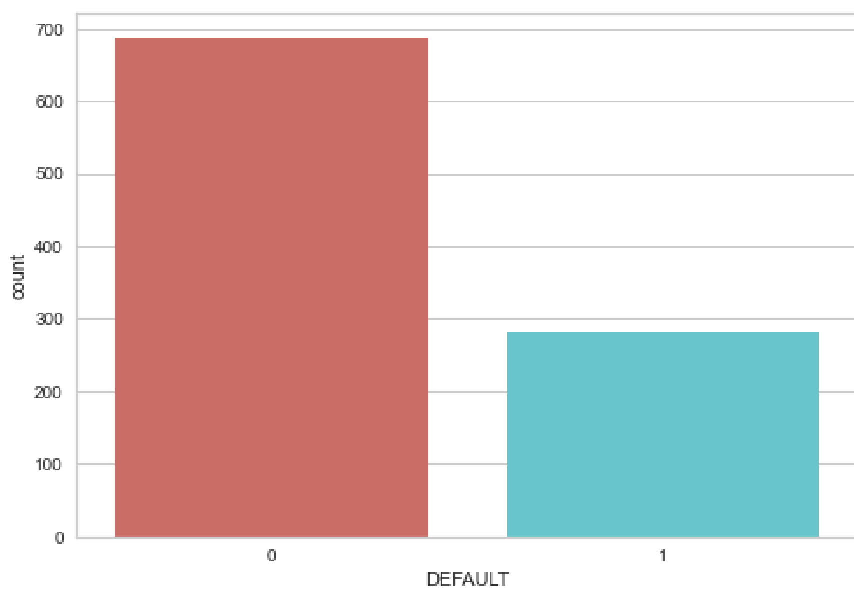
In [28]:
```
df['DEFAULT'].value_counts()
```

Out[28]:
```
0    687
1    283
Name: DEFAULT, dtype: int64
```

In [30]:
```
df['DEFAULT'].value_counts(normalize=True)
```

Out[30]:
```
0    0.708247
1    0.291753
Name: DEFAULT, dtype: float64
```

In [31]:
```
sns.countplot(x='DEFAULT', data=df, palette='hls')
plt.show()
```



In [32]:
```python
grouped_describe = df.groupby('DEFAULT', axis=0).describe()
grouped_describe['AMOUNT']
```

Out[32]:

DEFAULT	count	mean	std	min	25%	50%	75%	max
0	687.0	2865.622999	2142.632882	250.0	1369.0	2221.0	3604.0	11760.0
1	283.0	3386.713781	2690.947940	433.0	1332.0	2384.0	4598.0	11938.0

In [33]:
```python
df.corr()
```

Out[33]:

	DURATION	NEW_CAR	USED_CAR	FURNITURE	RADIO_TV	EDUCATION	RETRAINING	AMOUN
DURATION	1.000000	-0.113984	0.171956	-0.054344	-0.048302	0.006105	0.160366	0.6358
NEW_CAR	-0.113984	1.000000	-0.184205	-0.260804	-0.345434	-0.126026	-0.175800	-0.0789
USED_CAR	0.171956	-0.184205	1.000000	-0.160928	-0.213148	-0.077764	-0.108476	0.3178
FURNITURE	-0.054344	-0.260804	-0.160928	1.000000	-0.301784	-0.110101	-0.153585	-0.0025
RADIO_TV	-0.048302	-0.345434	-0.213148	-0.301784	1.000000	-0.145828	-0.203423	-0.1701
EDUCATION	0.006105	-0.126026	-0.077764	-0.110101	-0.145828	1.000000	-0.074215	-0.0029
RETRAINING	0.160366	-0.175800	-0.108476	-0.153585	-0.203423	-0.074215	1.000000	0.0950
AMOUNT	0.635864	-0.078976	0.317897	-0.002545	-0.170131	-0.002949	0.095038	1.0000
INSTALL_RATE	0.089199	-0.046090	-0.101248	-0.074669	0.134502	0.048460	-0.016383	-0.2867
MALE_DIV	0.005127	-0.011693	-0.029818	0.074499	-0.070586	-0.030924	0.089617	0.0235
MALE_SINGLE	0.119547	0.013990	0.104007	-0.073434	-0.029755	-0.005875	0.025268	0.1521
MALE_MAR_or_WID	-0.094681	-0.007736	-0.038398	-0.089919	0.117525	-0.058071	0.005613	-0.1398
GUARANTOR	0.020467	0.000424	-0.051667	0.064296	-0.001593	-0.047209	-0.029856	0.0736
CO_APPLICANT	-0.033681	-0.010384	-0.034883	-0.031193	0.112989	-0.054897	-0.045183	-0.0544
REAL_ESTATE	-0.232493	0.047950	-0.131470	-0.057256	0.122662	-0.104968	0.014261	-0.2389
PROP_UNKN_NONE	0.201616	0.001923	0.116160	-0.057757	-0.100674	0.162094	-0.029823	0.1861
AGE	-0.032398	0.061815	0.038645	-0.119132	-0.023342	0.076420	-0.020013	-0.0125
OTHER_INSTALL	0.068442	-0.032537	-0.011141	-0.001480	-0.030104	0.011622	0.102150	0.0397
RENT	-0.059681	-0.005834	0.042520	0.102552	-0.075657	0.000100	-0.015519	0.0067
OWN_RES	-0.071384	-0.000045	-0.128857	-0.048034	0.127987	-0.095764	0.044807	-0.1070
NUM_CREDITS	0.002604	0.041515	-0.005451	-0.079745	-0.037495	-0.010216	0.100024	0.0776
NUM_DEPENDENTS	-0.007340	0.108215	0.051398	-0.089044	-0.084173	0.030064	0.007200	0.0571
TELEPHONE	0.140919	-0.049743	0.135613	-0.044286	-0.069941	0.018389	0.083654	0.2240
FOREIGN	-0.150172	0.157043	-0.028708	-0.007036	-0.061129	-0.044627	-0.043293	-0.0735
DEFAULT	0.206579	0.096692	-0.111501	0.031996	-0.099494	0.069425	0.034617	0.1018

25 rows × 25 columns

In [34]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 970 entries, 0 to 969
Data columns (total 31 columns):
 #   Column            Non-Null Count  Dtype   
---  ------            --------------  -----   
 0   CHK_ACCT          970 non-null    category
 1   DURATION          970 non-null    int64   
 2   HISTORY           970 non-null    category
 3   NEW_CAR           970 non-null    uint8   
 4   USED_CAR          970 non-null    uint8   
 5   FURNITURE         970 non-null    uint8   
 6   RADIO_TV          970 non-null    uint8   
 7   EDUCATION         970 non-null    uint8   
 8   RETRAINING        970 non-null    uint8   
 9   AMOUNT            970 non-null    int64   
 10  SAV_ACCT          970 non-null    category
 11  EMPLOYMENT        970 non-null    category
 12  INSTALL_RATE      970 non-null    int64   
 13  MALE_DIV          970 non-null    uint8   
 14  MALE_SINGLE       970 non-null    uint8   
 15  MALE_MAR_or_WID   970 non-null    uint8   
 16  GUARANTOR         970 non-null    uint8   
 17  CO_APPLICANT      970 non-null    uint8   
 18  PRESENT_RESIDENT  970 non-null    category
 19  REAL_ESTATE       970 non-null    uint8   
 20  PROP_UNKN_NONE    970 non-null    uint8   
 21  AGE               970 non-null    int64   
 22  OTHER_INSTALL     970 non-null    uint8   
 23  RENT              970 non-null    uint8   
 24  OWN_RES           970 non-null    uint8   
 25  NUM_CREDITS       970 non-null    int64   
 26  JOB               970 non-null    category
 27  NUM_DEPENDENTS    970 non-null    int64   
 28  TELEPHONE         970 non-null    uint8   
 29  FOREIGN           970 non-null    uint8   
 30  DEFAULT           970 non-null    uint8   
dtypes: category(6), int64(6), uint8(19)
memory usage: 70.5 KB
```

DATA EXPLORATION AND TEST

In [35]:
```
def contingency(column,rot=0):
    y = df['DEFAULT']
    x = df[column]
    table = pd.crosstab(x, y)
    plot = table.div(table.sum(1), axis=0).plot(kind='bar', stacked=True, legend=False, rot=ro
    return plot
```

In [36]:
```
def graf_func(column):
    column= df[column].astype('int64')
    plot = sns.JointGrid(data=df, x=column)
    plot.plot_joint(sns.histplot)
    plot.plot_marginals(sns.boxplot)
    return plot
```

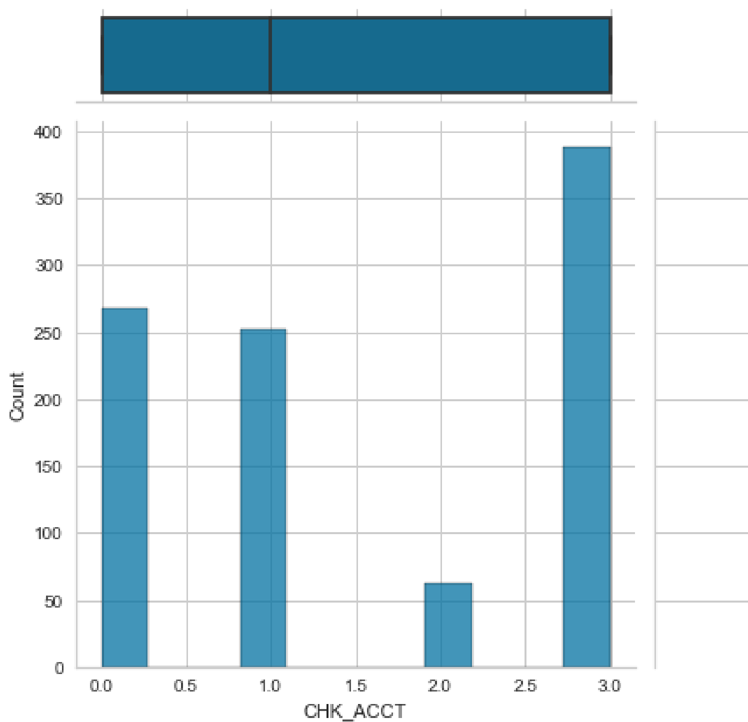
In [37]:
```
def data_tabla(column):
    column= df[column].astype('int64')
    tabla= column.describe(include='all')
    return pd.concat([tabla], axis=1)
```

In [38]:
```
def logit(column):
    x= df[column].astype('int')
    y= df['DEFAULT']
    logit= smf.logit('y~x',data=df).fit()
    return (logit.wald_test_terms())
```

In [39]:
```
def logplot1(column, df):
    y = df['DEFAULT']
    x = df.drop(columns='DEFAULT')
    column_data = df[[column]]

    log = LogisticRegression(penalty=None)
    log.fit(X=x, y=y)
    y_pred = log.predict_proba(x)[:, 1]

    new = pd.DataFrame(data={'Default': y_pred})
    df3 = pd.concat([column_data, new], axis=1)

    plot = sns.lmplot(x=column, y='Default', data=df3)
    return plot
```

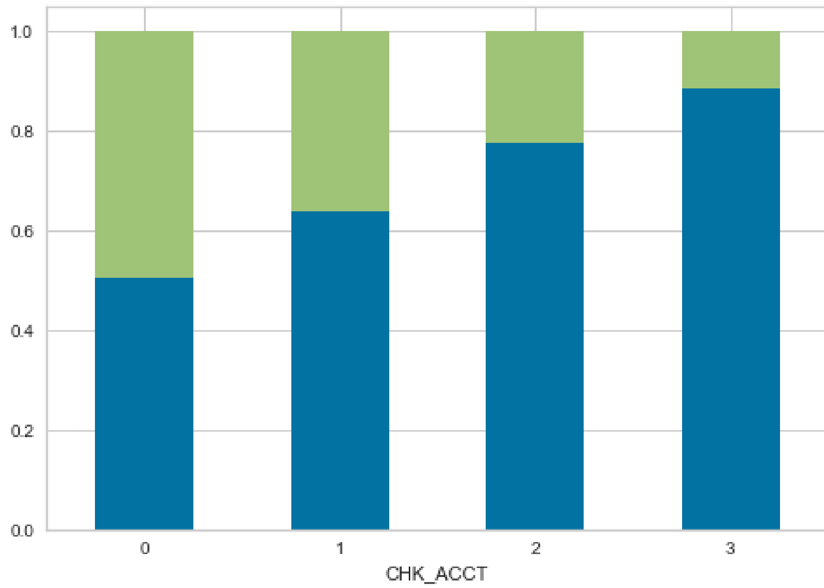
In [40]:
```
graf_func('CHK_ACCT')
```

Out[40]: <seaborn.axisgrid.JointGrid at 0x1ceb4293580>



In [41]:
```
contingency('CHK_ACCT')
```

Out[41]: <AxesSubplot:xlabel='CHK_ACCT'>



In [42]:
```
data_tabla('CHK_ACCT')
```

Out[42]:

	CHK_ACCT
count	970.000000
mean	1.587629
std	1.263544
min	0.000000
25%	0.000000
50%	1.000000
75%	3.000000
max	3.000000

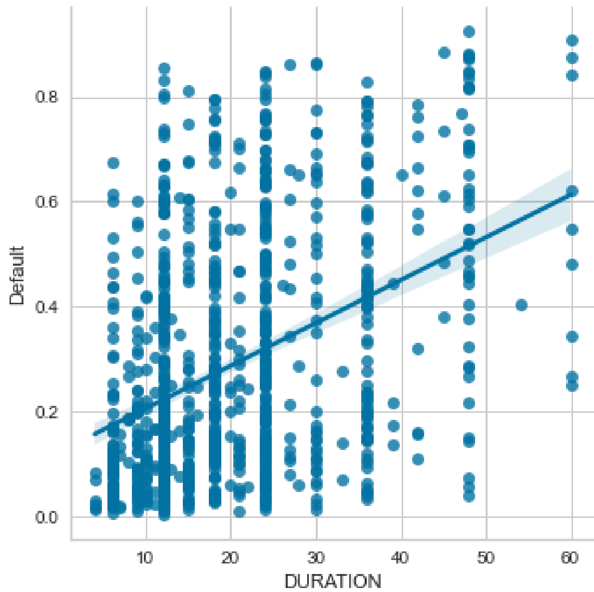
In [43]:
```
logit('CHK_ACCT')
```

```
Optimization terminated successfully.
         Current function value: 0.539398
         Iterations 6
```

Out[43]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                              chi2                 P>chi2  df constraint
Intercept  [[0.07932389489160871]]      0.7782157465220065              1
x           [[107.43733659832935]]  3.5701702785829496e-25              1
```

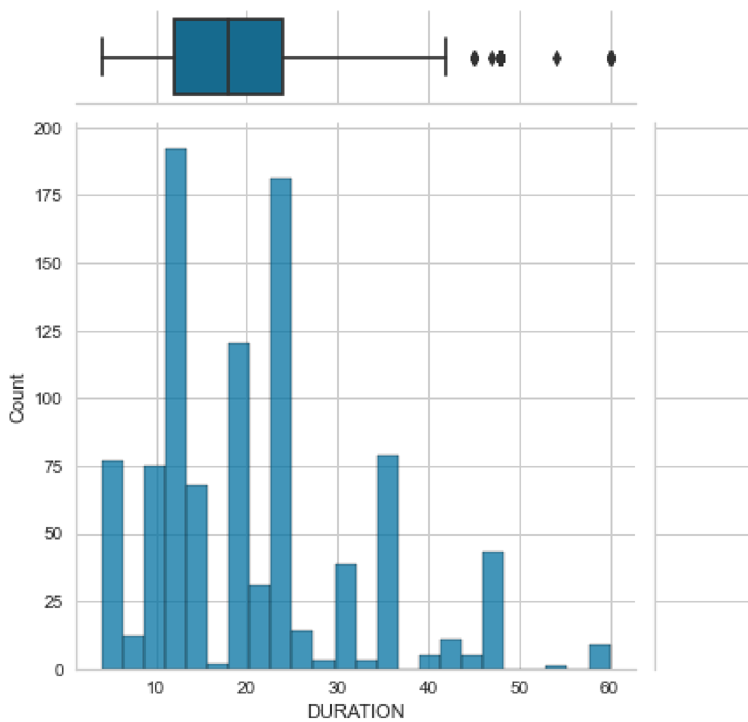
In [44]:
```
logplot1('DURATION',df)
```

Out[44]: <seaborn.axisgrid.FacetGrid at 0x1ceb4448760>



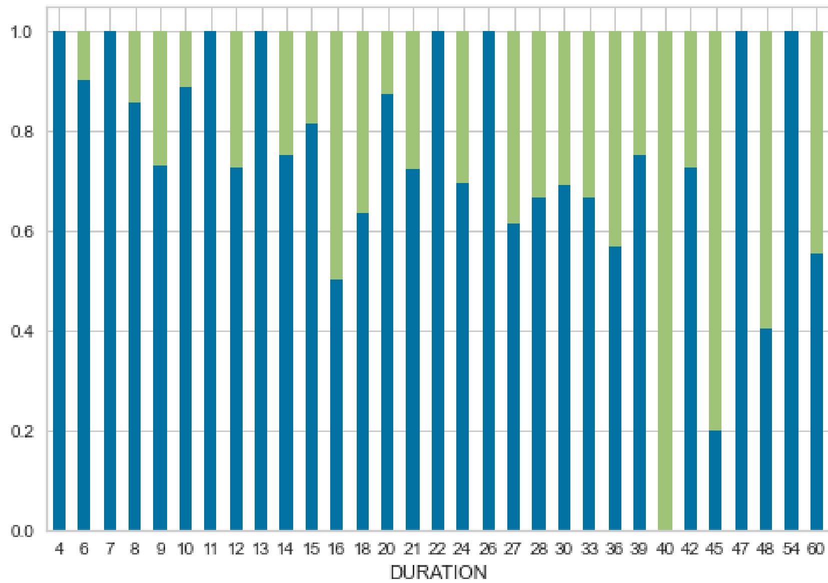
In [45]:
```
graf_func('DURATION')
```

Out[45]: <seaborn.axisgrid.JointGrid at 0x1ceb426d3d0>



In [46]:
```
contingency('DURATION' )
```

Out[46]: <AxesSubplot:xlabel='DURATION'>



In [47]:
```
data_tabla('DURATION')
```

Out[47]:

	DURATION
count	970.000000
mean	20.491753
std	11.498572
min	4.000000
25%	12.000000
50%	18.000000
75%	24.000000
max	60.000000

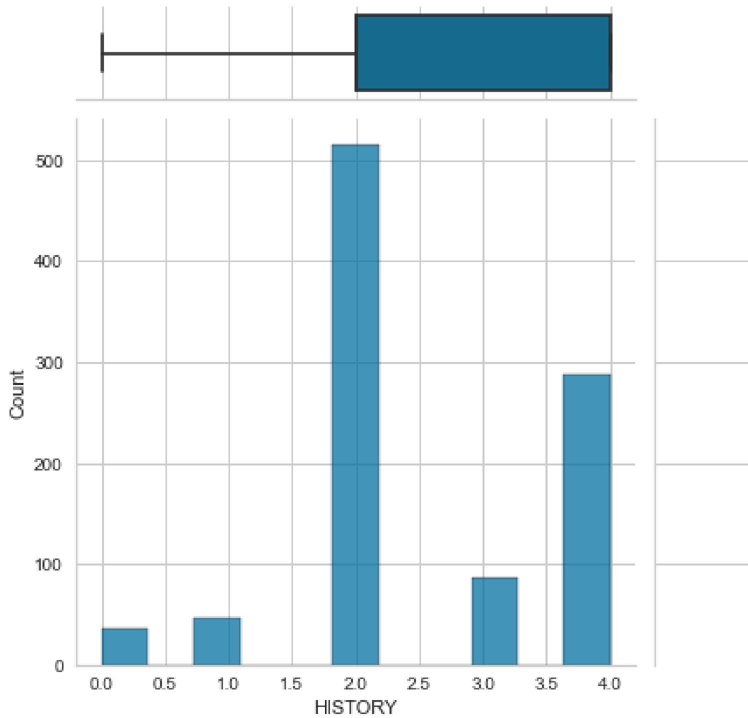
In [48]:
```
logit('DURATION')
```

```
Optimization terminated successfully.
         Current function value: 0.583128
         Iterations 5
```

Out[48]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                  P>chi2  df constraint
Intercept  [[123.95707180575724]]   8.608603012383594e-29              1
x           [[39.06655494523789]]  4.0960105790993724e-10              1
```

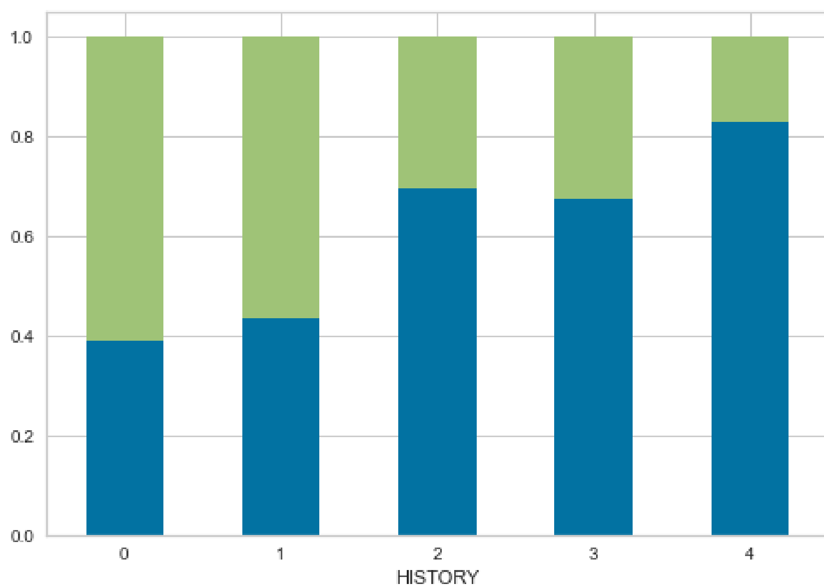
In [49]:
```
graf_func('HISTORY')
```

Out[49]: <seaborn.axisgrid.JointGrid at 0x1ceb4448280>



In [50]:
```
contingency('HISTORY')
```

Out[50]: <AxesSubplot:xlabel='HISTORY'>



In [51]:
```
data_tabla('HISTORY')
```

Out[51]:

	HISTORY
count	970.000000
mean	2.558763
std	1.075648
min	0.000000
25%	2.000000
50%	2.000000
75%	4.000000
max	4.000000

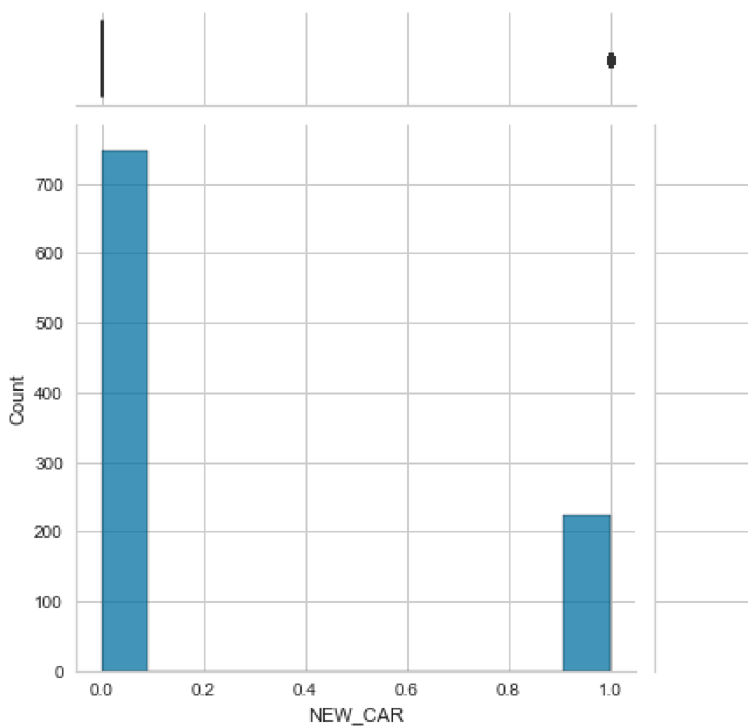
In [52]:
```
logit('HISTORY')
```

```
Optimization terminated successfully.
         Current function value: 0.579958
         Iterations 5
```

Out[52]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                 P>chi2  df constraint
Intercept  [[1.853860196653683]]    0.17333509730209645              1
x          [[42.67666675535496]]  6.457802595135919e-11              1
```

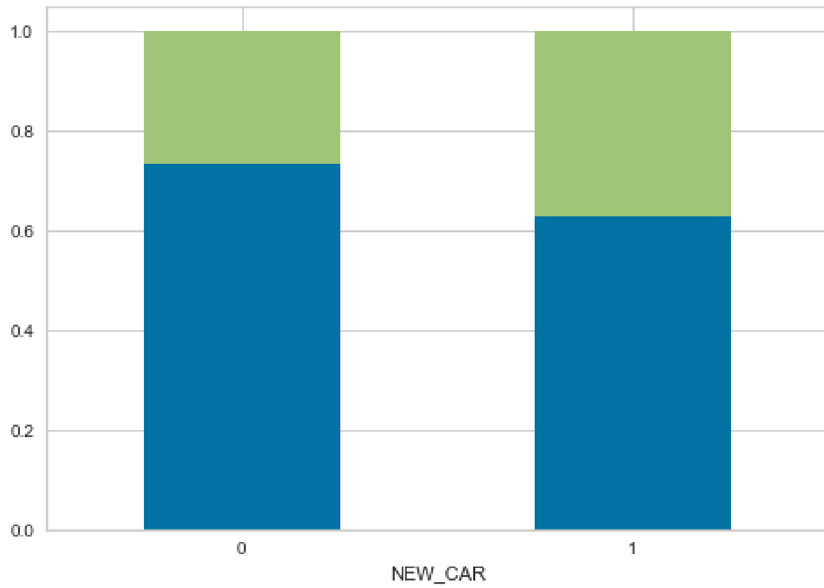
In [53]:
```
graf_func('NEW_CAR')
```

Out[53]:
```
<seaborn.axisgrid.JointGrid at 0x1ceb62d8ac0>
```



In [54]:
```
contingency('NEW_CAR')
```

Out[54]: <AxesSubplot:xlabel='NEW_CAR'>



In [55]:
```
data_tabla('NEW_CAR')
```

Out[55]:

	NEW_CAR
count	970.000000
mean	0.229897
std	0.420983
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

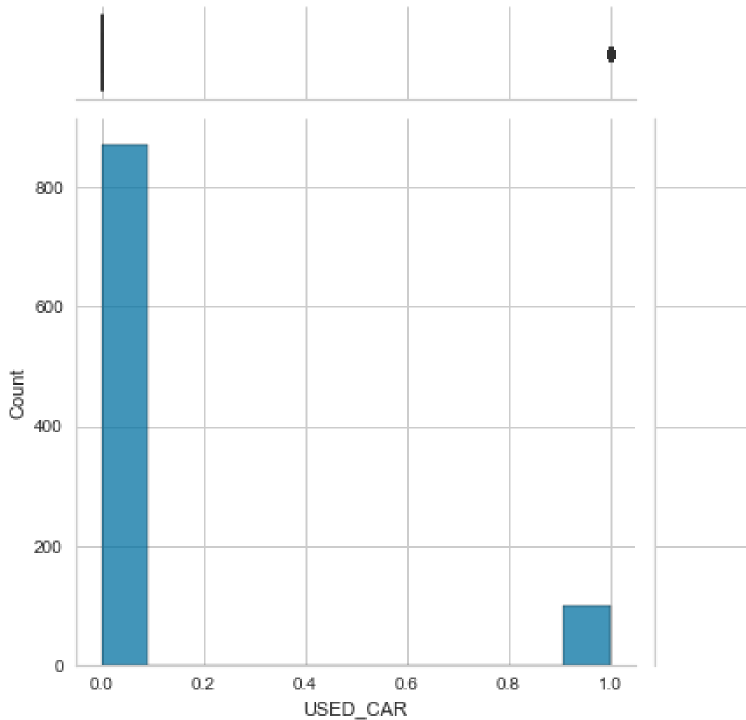
In [56]:
```
logit('NEW_CAR')
```

```
Optimization terminated successfully.
         Current function value: 0.599185
         Iterations 5
```

Out[56]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                P>chi2  df constraint
Intercept  [[148.25391031882685]]   4.1747245871188e-34              1
x            [[8.978273404302126]]  0.002732086893586821              1
```

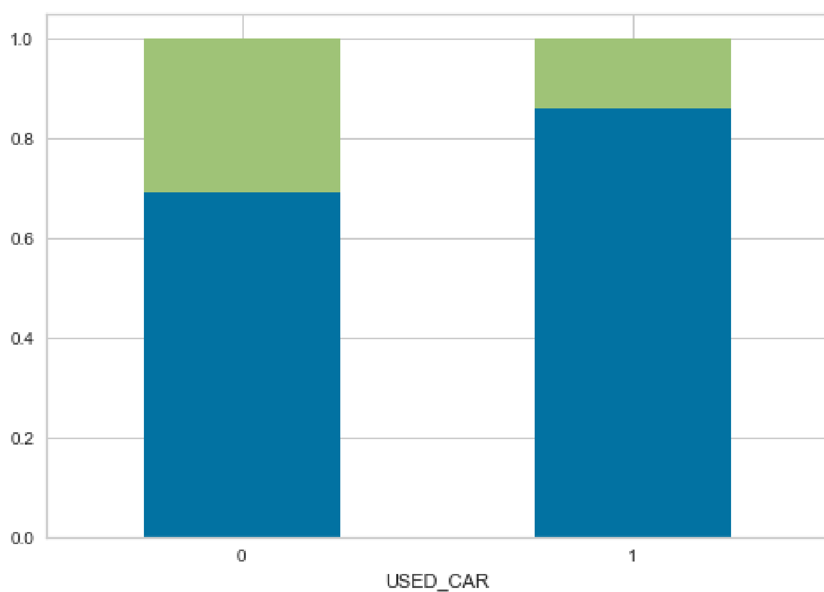
In [57]:
```
graf_func('USED_CAR')
```

Out[57]: <seaborn.axisgrid.JointGrid at 0x1ceb6406c40>



In [58]:
```
contingency('USED_CAR')
```

Out[58]: <AxesSubplot:xlabel='USED_CAR'>



In [59]:
```
data_tabla('USED_CAR')
```

Out[59]:

	USED_CAR
count	970.000000
mean	0.102062
std	0.302886
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

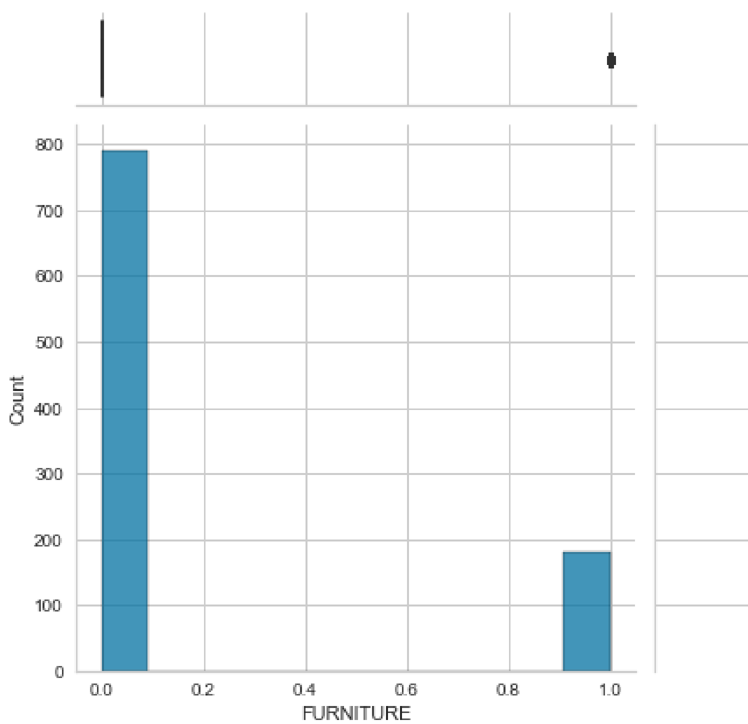
In [60]:
```
logit('USED_CAR')
```

```
Optimization terminated successfully.
         Current function value: 0.596671
         Iterations 6
```

Out[60]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                              chi2                 P>chi2  df constraint
Intercept  [[120.64556956067288]]  4.568817030369174e-28              1
x          [[11.246227859801635]]  0.0007978499569013929              1
```

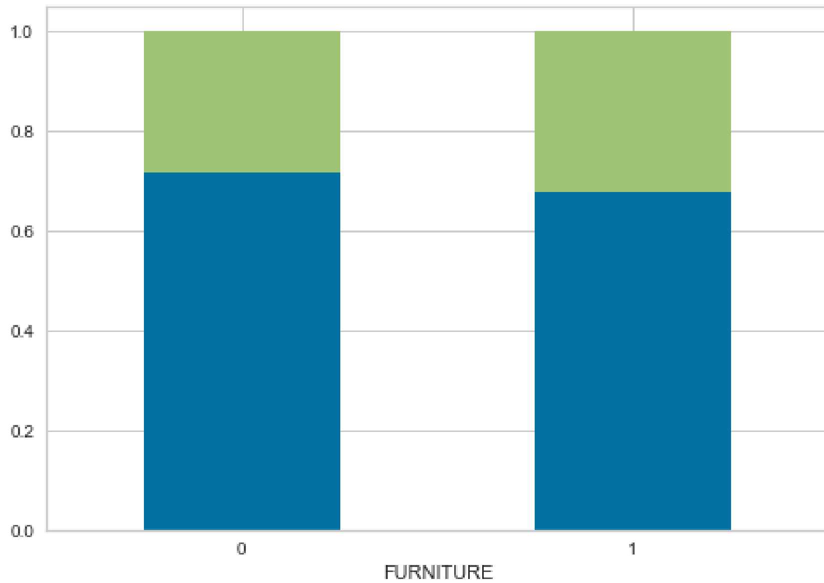
In [61]:
```
graf_func('FURNITURE')
```

Out[61]:
```
<seaborn.axisgrid.JointGrid at 0x1ceb64d8730>
```



In [62]:
```
contingency('FURNITURE')
```

Out[62]: <AxesSubplot:xlabel='FURNITURE'>



In [63]:
```
data_tabla('FURNITURE')
```

Out[63]:

	FURNITURE
count	970.000000
mean	0.185567
std	0.388957
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

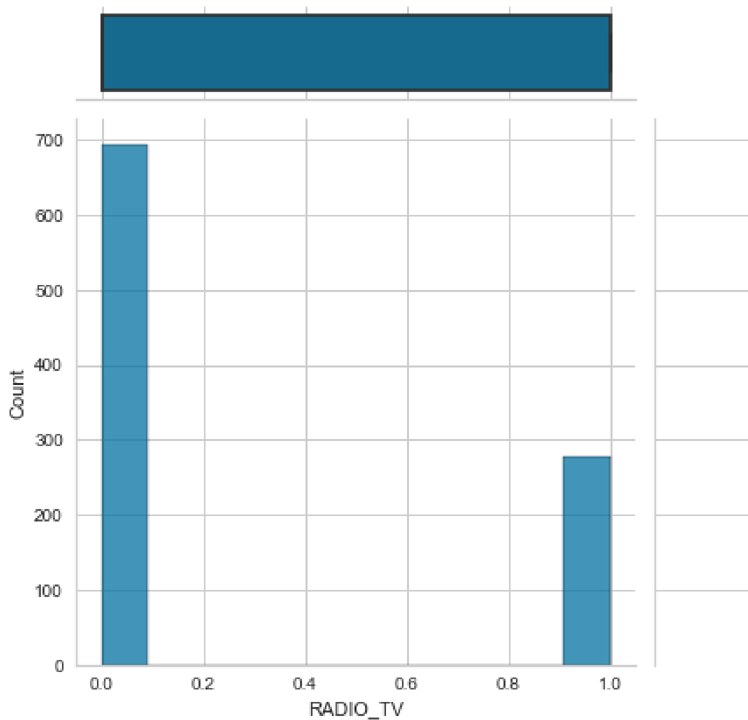
In [64]:
```
logit('FURNITURE')
```

```
Optimization terminated successfully.
         Current function value: 0.603209
         Iterations 5
```

Out[64]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                P>chi2  df constraint
Intercept  [[136.41560578931114]]  1.618483732005395e-31              1
x          [[0.9914290212829611]]     0.3193933598604689              1
```

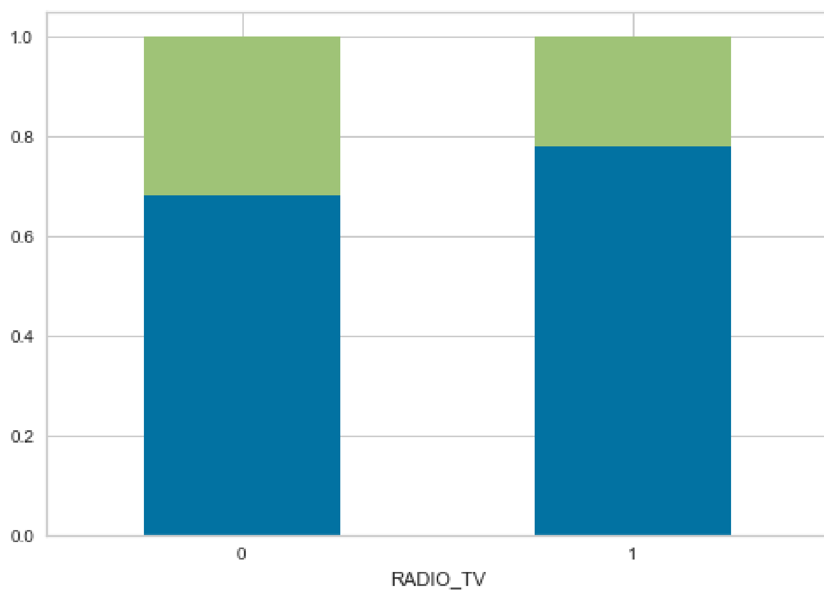
In [65]:
```
graf_func('RADIO_TV')
```

Out[65]: <seaborn.axisgrid.JointGrid at 0x1ceb670fee0>



In [66]:
```
contingency('RADIO_TV')
```

Out[66]: <AxesSubplot:xlabel='RADIO_TV'>



In [67]:
```
data_tabla('RADIO_TV')
```

Out[67]:

	RADIO_TV
count	970.000000
mean	0.285567
std	0.451917
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	1.000000

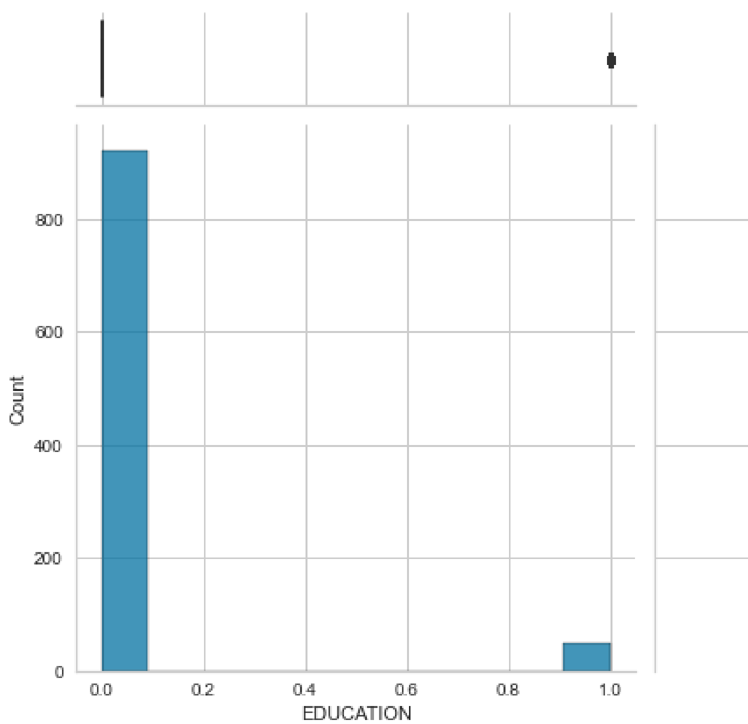
In [68]:
```
logit('RADIO_TV')
```

```
Optimization terminated successfully.
         Current function value: 0.598588
         Iterations 5
```

Out[68]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                            chi2                  P>chi2  df constraint
Intercept   [[85.3660197402369]]  2.4794507100266995e-20              1
x          [[9.488840176189827]]    0.002067254623200734              1
```

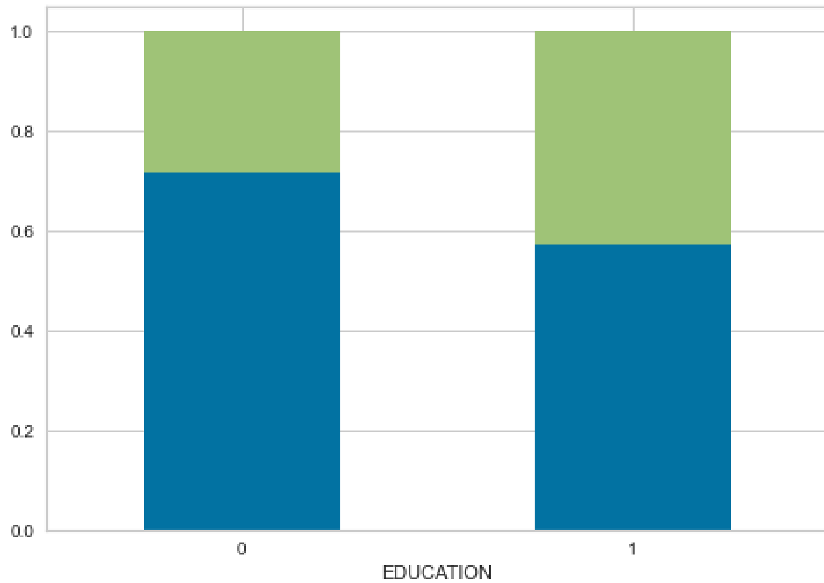
In [69]:
```
graf_func('EDUCATION')
```

Out[69]:
```
<seaborn.axisgrid.JointGrid at 0x1ceb66520a0>
```



In [70]:
```
contingency('EDUCATION')
```

Out[70]:
```
<AxesSubplot:xlabel='EDUCATION'>
```



In [71]:
```
data_tabla('EDUCATION')
```

Out[71]:

	EDUCATION
count	970.000000
mean	0.050515
std	0.219119
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

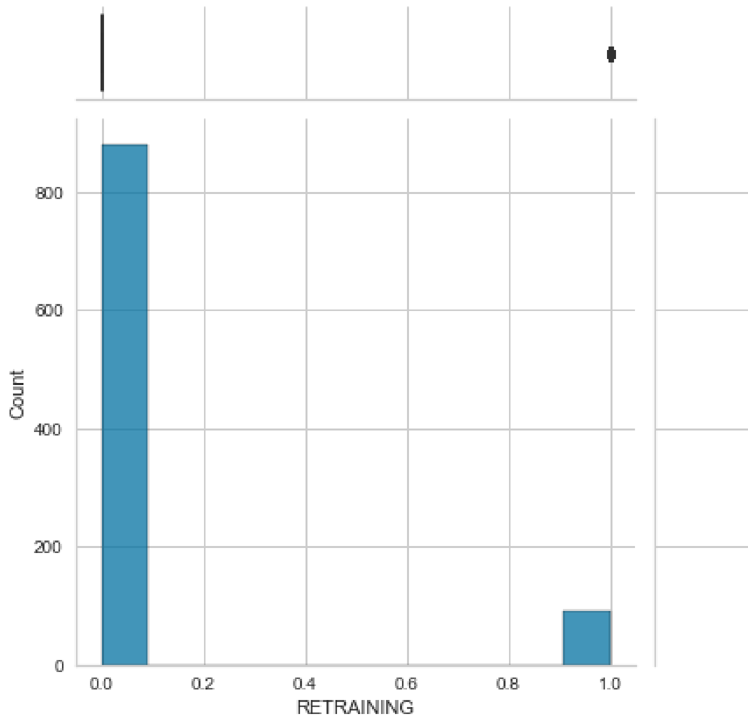
In [72]:
```
logit('EDUCATION')
```

```
Optimization terminated successfully.
         Current function value: 0.601462
         Iterations 5
```

Out[72]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                  P>chi2  df constraint
Intercept  [[159.49457589217067]]  1.4590778861657847e-36              1
x          [[4.5432641599177295]]     0.03304851471701444              1
```

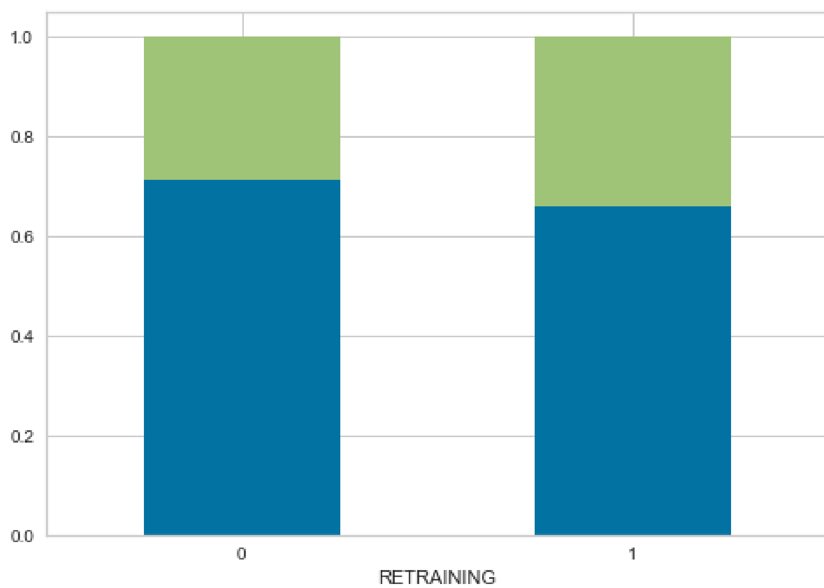
In [73]:
```
graf_func('RETRAINING')
```

Out[73]: <seaborn.axisgrid.JointGrid at 0x1ceb77d1430>



In [74]:
```
contingency('RETRAINING')
```

Out[74]: <AxesSubplot:xlabel='RETRAINING'>



In [75]:
```
data_tabla('RETRAINING')
```

Out[75]:

	RETRAINING
count	970.000000
mean	0.093814
std	0.291721
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

In [76]:
```
logit('RETRAINING')
```

```
Optimization terminated successfully.
         Current function value: 0.603130
         Iterations 5
```

Out[76]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                              chi2                  P>chi2  df constraint
Intercept  [[149.35136074720734]]  2.4029172756811e-34              1
x           [[1.157714188860484]]  0.28194010243856416              1
```

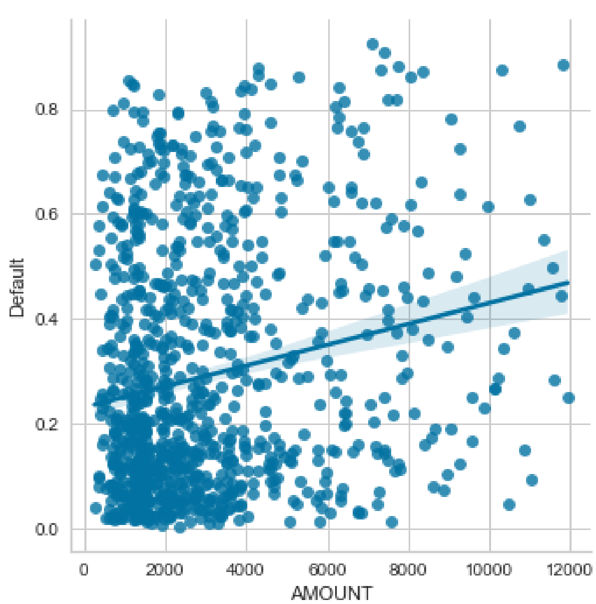
In [77]:
```
logit('AMOUNT')
```

```
Optimization terminated successfully.
         Current function value: 0.598712
         Iterations 5
```

Out[77]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                              chi2                     P>chi2  df constraint
Intercept  [[100.04764790745219]]  1.4877450330074576e-23              1
x            [[9.88233628466132]]   0.0016687292559115906              1
```

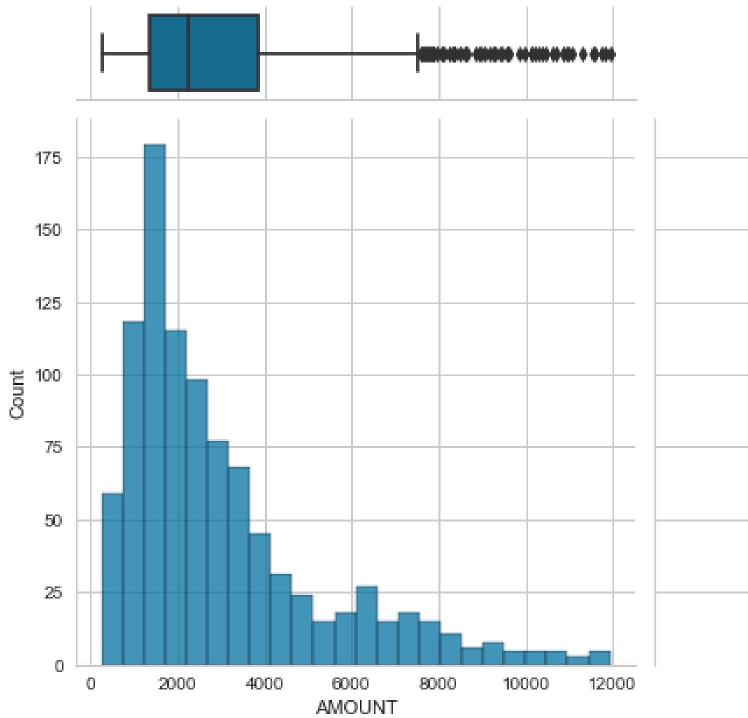
In [78]:
```
logplot1('AMOUNT',df)
```

Out[78]:
```
<seaborn.axisgrid.FacetGrid at 0x1ceb7ba66d0>
```



In [79]:
```python
graf_func('AMOUNT')
```

Out[79]: <seaborn.axisgrid.JointGrid at 0x1ceb7c0c9a0>



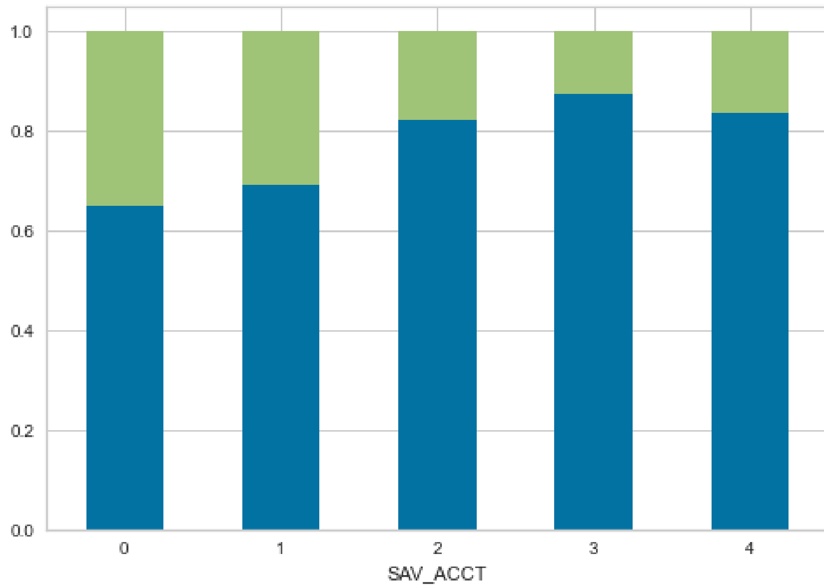
In [80]:
```python
data_tabla('AMOUNT')
```

Out[80]:

	AMOUNT
count	970.000000
mean	3017.652577
std	2326.715732
min	250.000000
25%	1352.750000
50%	2253.000000
75%	3834.250000
max	11938.000000

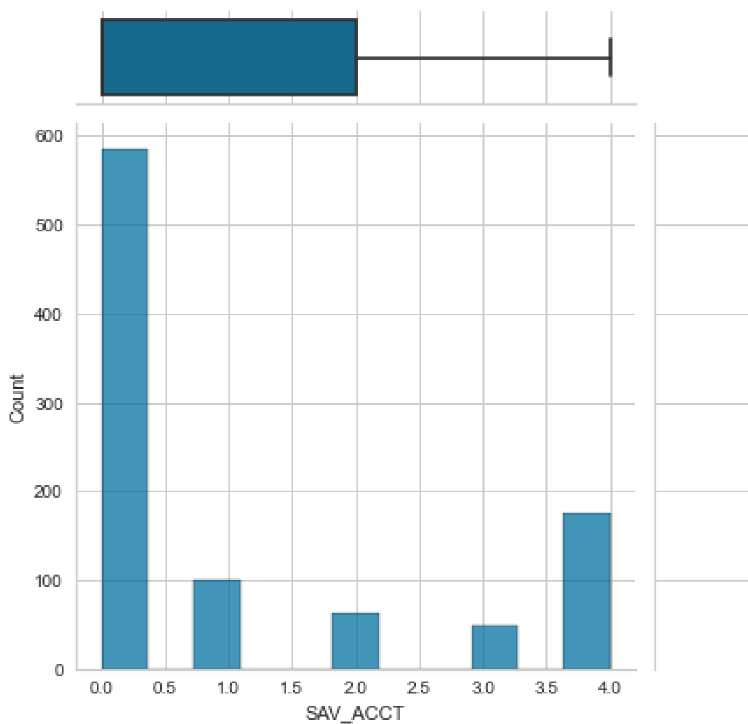
In [81]:
```
contingency('SAV_ACCT')
```

Out[81]: <AxesSubplot:xlabel='SAV_ACCT'>



In [82]:
```
graf_func('SAV_ACCT')
```

Out[82]: <seaborn.axisgrid.JointGrid at 0x1ceb7ce6e50>



In [83]:
```
data_tabla('SAV_ACCT')
```

Out[83]:

	SAV_ACCT
count	970.000000
mean	1.101031
std	1.575122
min	0.000000
25%	0.000000
50%	0.000000
75%	2.000000
max	4.000000

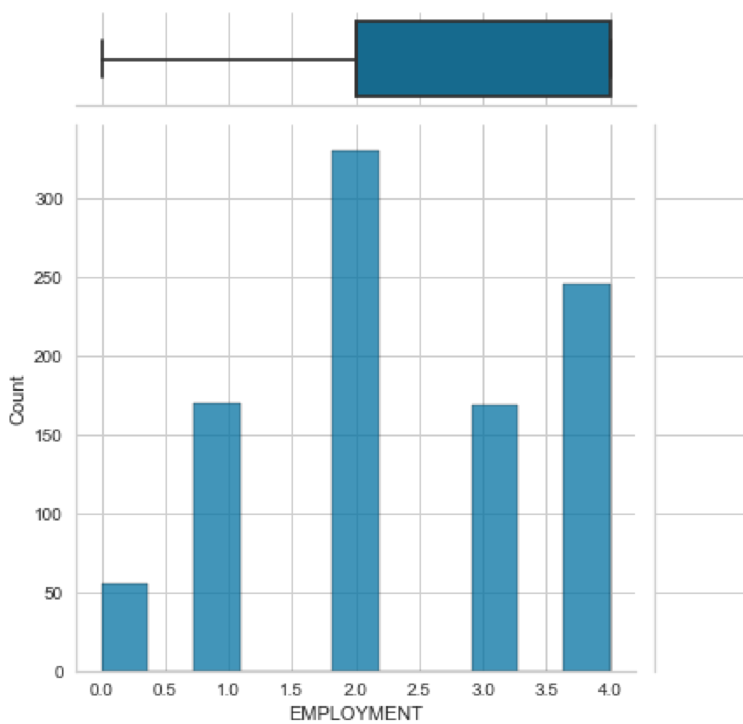
In [84]:
```
logit('SAV_ACCT')
```

```
Optimization terminated successfully.
         Current function value: 0.586262
         Iterations 5
```

Out[84]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                   P>chi2  df constraint
Intercept   [[55.34954180380802]]   1.0089338355679l8e-13               1
x          [[29.710263993791347]]  5.0168488591286185e-08               1
```

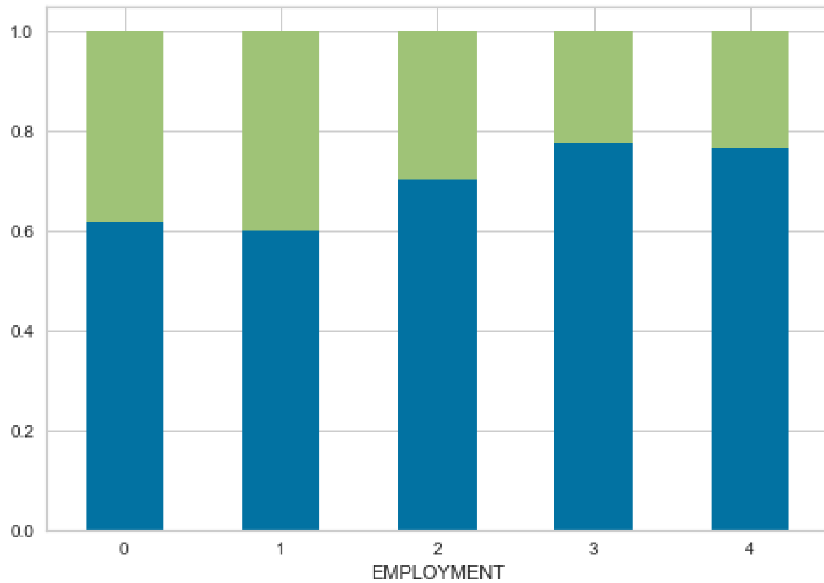
In [85]:
```
graf_func('EMPLOYMENT')
```

Out[85]:
```
<seaborn.axisgrid.JointGrid at 0x1ceb7c12580>
```



In [86]:
```
contingency('EMPLOYMENT')
```

Out[86]: <AxesSubplot:xlabel='EMPLOYMENT'>



In [87]:
```
data_tabla('EMPLOYMENT')
```

Out[87]:

	EMPLOYMENT
count	970.000000
mean	2.392784
std	1.199135
min	0.000000
25%	2.000000
50%	2.000000
75%	4.000000
max	4.000000

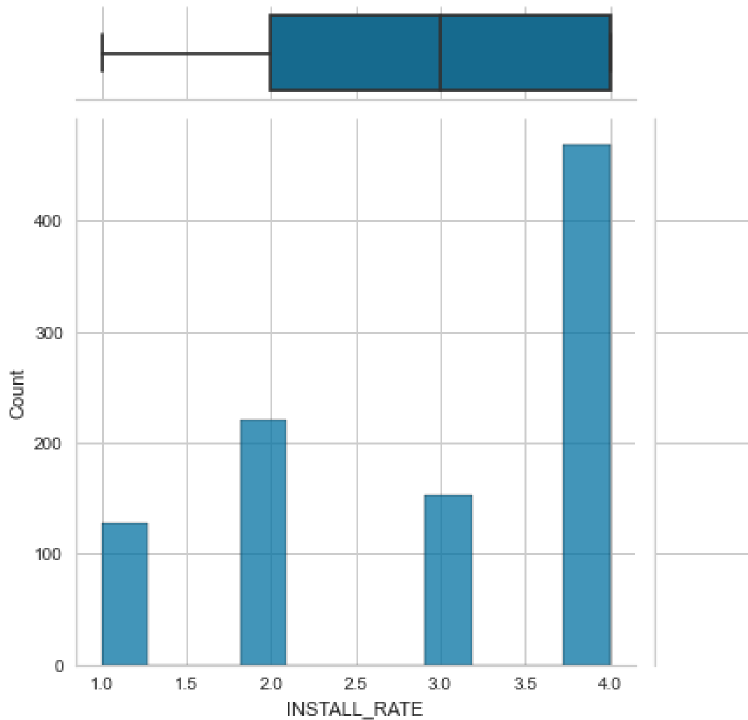
In [88]:
```
logit('EMPLOYMENT')
```

```
Optimization terminated successfully.
         Current function value: 0.595617
         Iterations 5
```

Out[88]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                 P>chi2  df constraint
Intercept   [[4.994577798931114]]    0.02542685603547374               1
x          [[15.448403074630523]]  8.478855583355783e-05               1
```

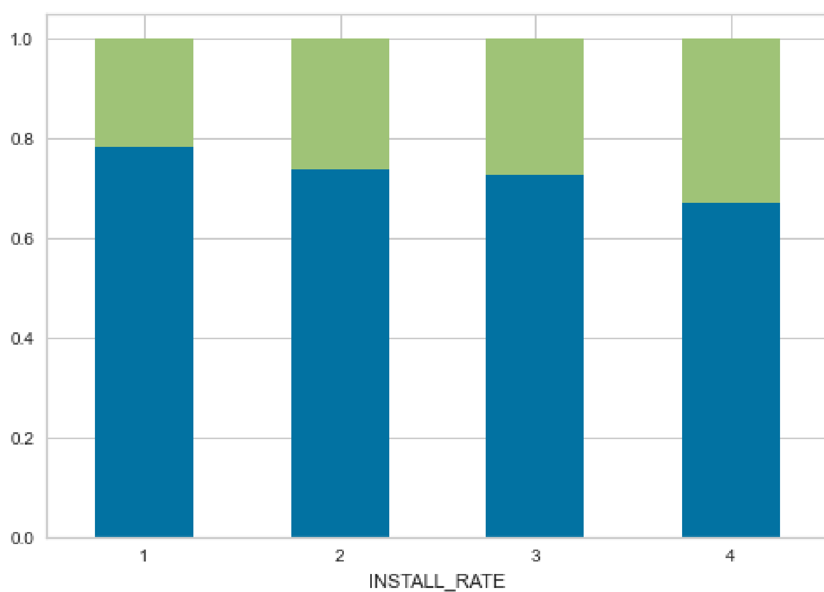
In [89]:
```
graf_func('INSTALL_RATE')
```

Out[89]: <seaborn.axisgrid.JointGrid at 0x1ceb9007e20>



In [90]:
```
contingency('INSTALL_RATE')
```

Out[90]: <AxesSubplot:xlabel='INSTALL_RATE'>



In [91]:
```
data_tabla('INSTALL_RATE')
```

Out[91]:

	INSTALL_RATE
count	970.000000
mean	2.990722
std	1.113255
min	1.000000
25%	2.000000
50%	3.000000
75%	4.000000
max	4.000000

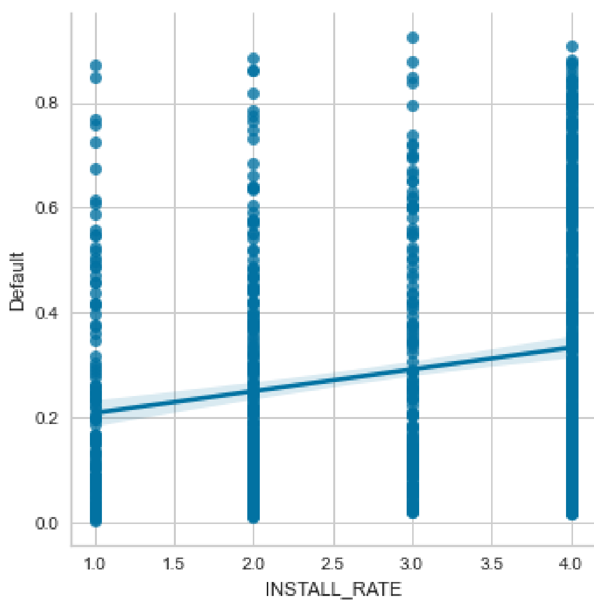
In [92]:
```
logit('INSTALL_RATE')
```

```
Optimization terminated successfully.
         Current function value: 0.599685
         Iterations 5
```

Out[92]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                 P>chi2  df constraint
Intercept  [[44.82874766929636]]  2.150422320460727e-11              1
x          [[7.615645716936369]]   0.005786403556472997              1
```

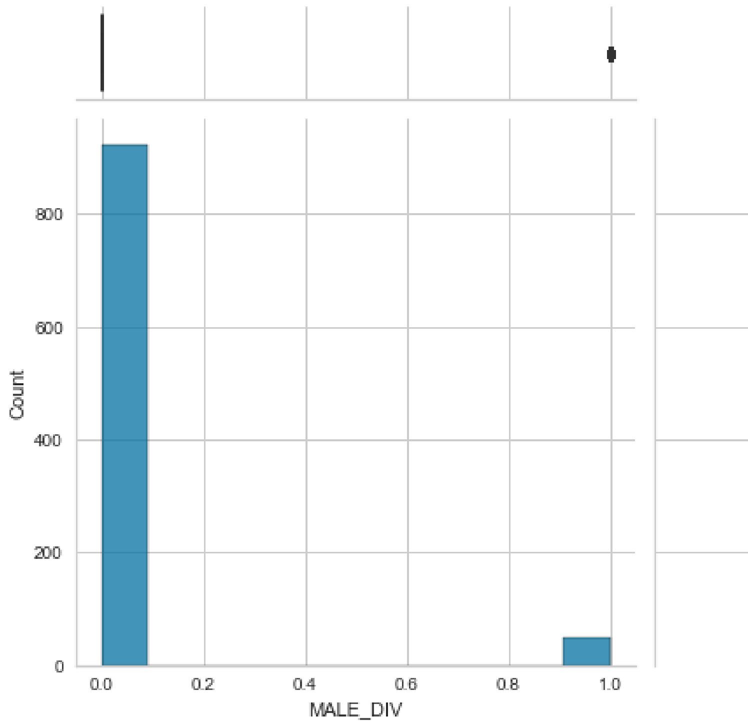
In [93]:
```
logplot1('INSTALL_RATE',df)
```

Out[93]:
```
<seaborn.axisgrid.FacetGrid at 0x1ceb9238cd0>
```



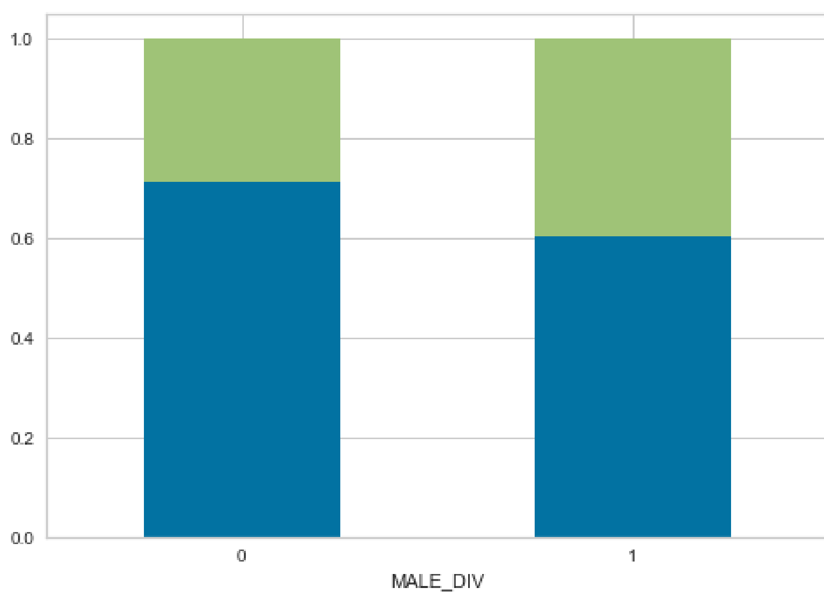
In [94]:
```
graf_func('MALE_DIV')
```

Out[94]: <seaborn.axisgrid.JointGrid at 0x1ceb9286d60>



In [95]:
```
contingency('MALE_DIV')
```

Out[95]: <AxesSubplot:xlabel='MALE_DIV'>



In [96]:
```
data_tabla('MALE_DIV')
```

Out[96]:

	MALE_DIV
count	970.000000
mean	0.049485
std	0.216989
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

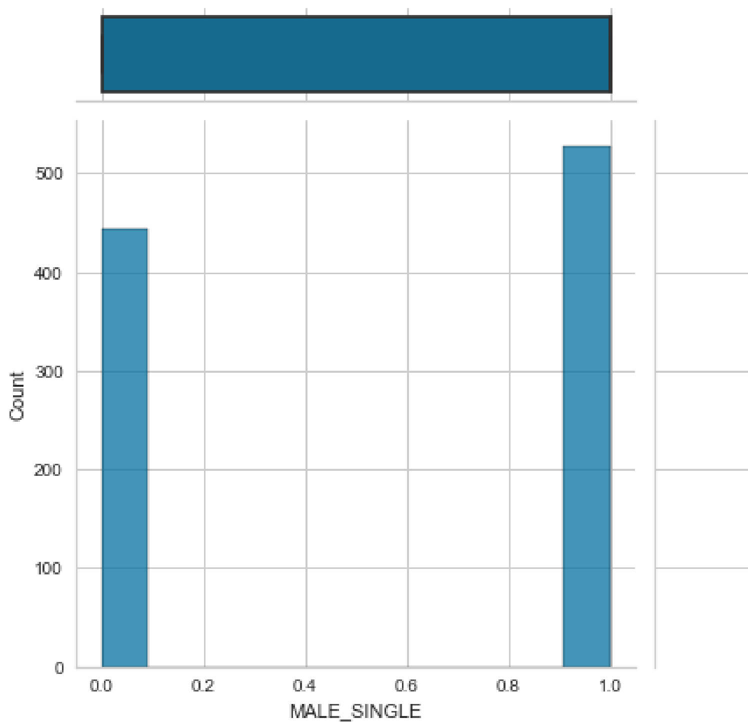
In [97]:
```
logit('MALE_DIV')
```

```
Optimization terminated successfully.
         Current function value: 0.602422
         Iterations 5
```

Out[97]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                               chi2                  P>chi2  df constraint
Intercept  [[157.13893461224387]]  4.7729946771856004e-36               1
x           [[2.602099356694339]]     0.10672226228752835               1
```

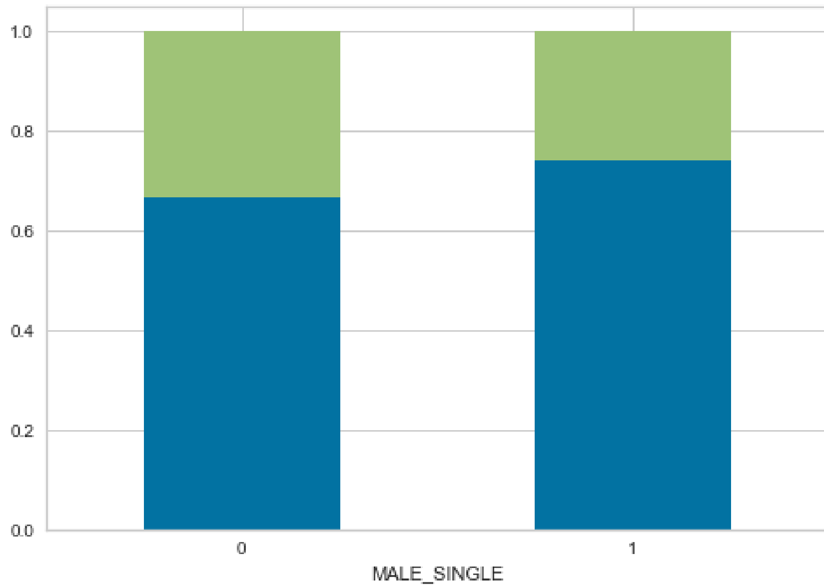
In [98]:
```
graf_func('MALE_SINGLE')
```

Out[98]:
```
<seaborn.axisgrid.JointGrid at 0x1ceb93f42e0>
```



In [99]:
```
contingency('MALE_SINGLE')
```

Out[99]: <AxesSubplot:xlabel='MALE_SINGLE'>



In [100]:
```
data_tabla('MALE_SINGLE')
```

Out[100]:

	MALE_SINGLE
count	970.000000
mean	0.543299
std	0.498379
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

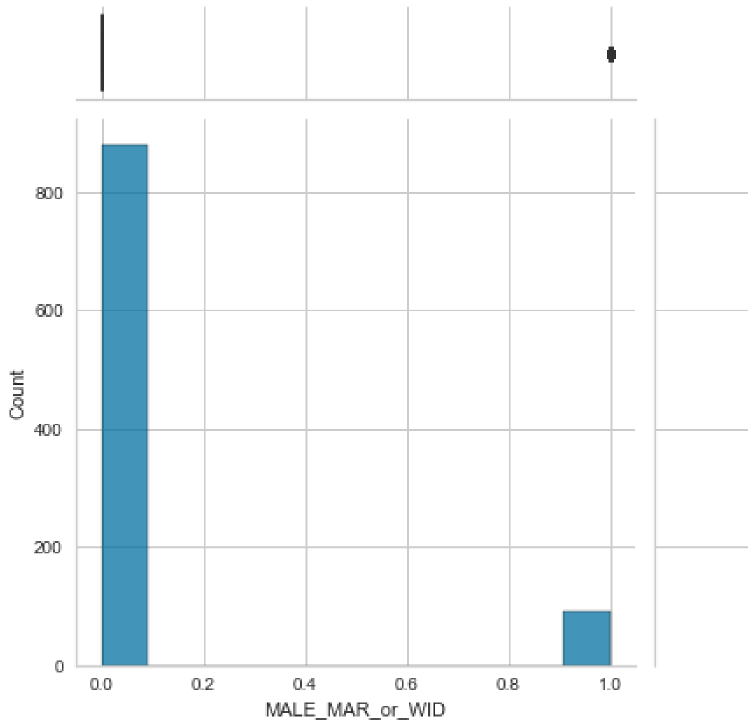
In [101]:
```
logit('MALE_SINGLE')
```

```
Optimization terminated successfully.
         Current function value: 0.600454
         Iterations 5
```

Out[101]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                P>chi2  df constraint
Intercept  [[48.118341394138035]]  4.012554937316995e-12              1
x           [[6.312373447570872]]   0.011989822769184323              1
```

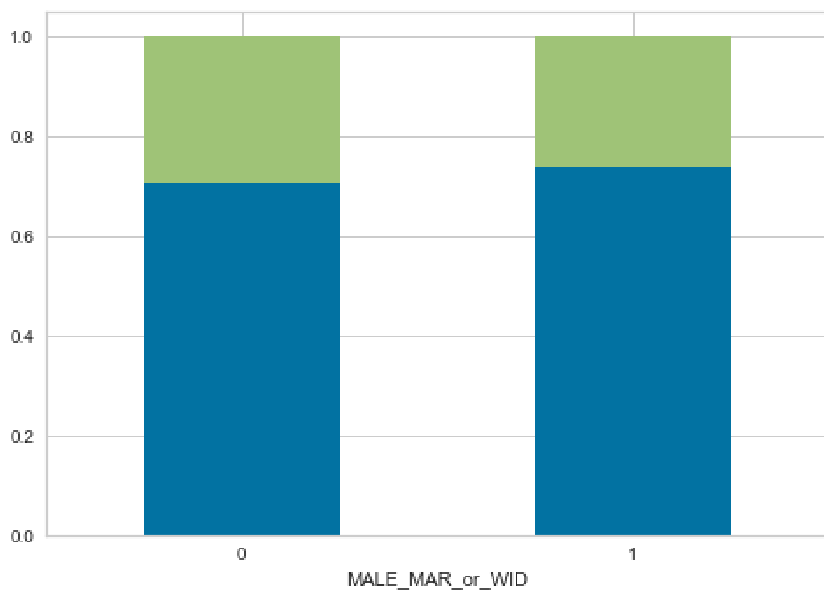
In [102]:
```python
graf_func('MALE_MAR_or_WID')
```

Out[102]: <seaborn.axisgrid.JointGrid at 0x1ceb94b6e50>



In [103]:
```python
contingency('MALE_MAR_or_WID')
```

Out[103]: <AxesSubplot:xlabel='MALE_MAR_or_WID'>



In [104]:
```
data_tabla('MALE_MAR_or_WID')
```

Out[104]:

	MALE_MAR_or_WID
count	970.000000
mean	0.093814
std	0.291721
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

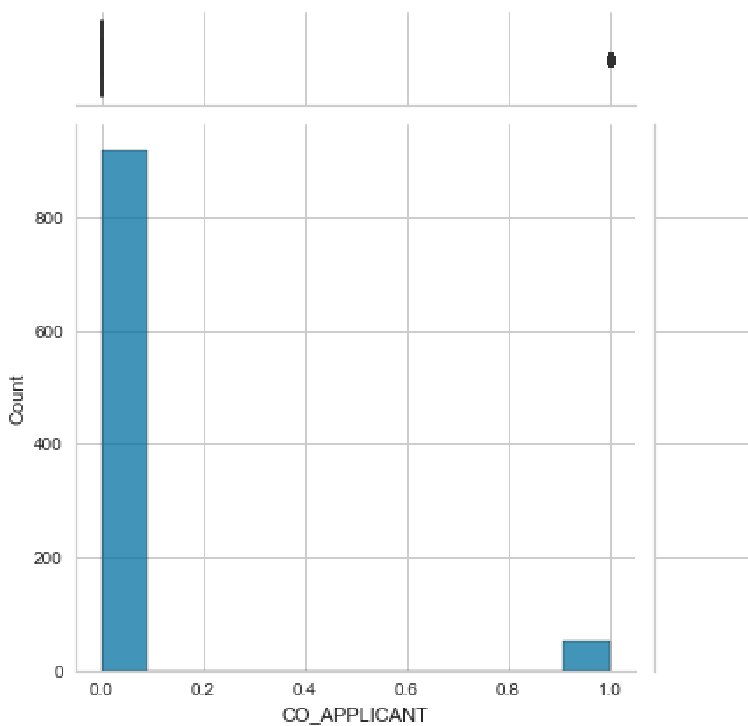
In [105]:
```
logit('MALE_MAR_or_WID')
```

```
Optimization terminated successfully.
         Current function value: 0.603513
         Iterations 5
```

Out[105]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                              chi2                 P>chi2  df constraint
Intercept  [[139.19480411994894]]  3.992960178584304e-32              1
x          [[0.3808570689485347]]     0.5371449041508316              1
```

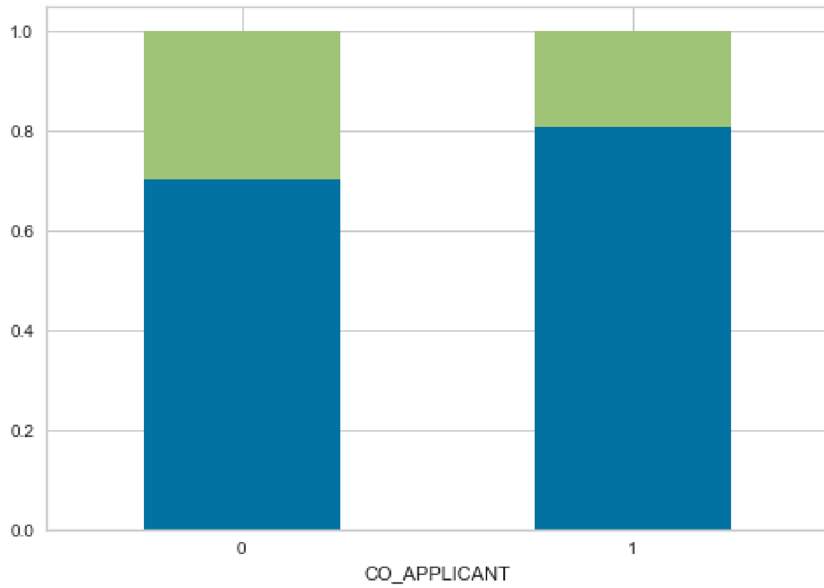
In [106]:
```
graf_func('CO_APPLICANT')
```

Out[106]:
```
<seaborn.axisgrid.JointGrid at 0x1ceb937ef10>
```



In [107]:
```
contingency('CO_APPLICANT')
```

Out[107]: <AxesSubplot:xlabel='CO_APPLICANT'>



In [108]:
```
data_tabla('CO_APPLICANT')
```

Out[108]:

	CO_APPLICANT
count	970.000000
mean	0.053608
std	0.225359
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

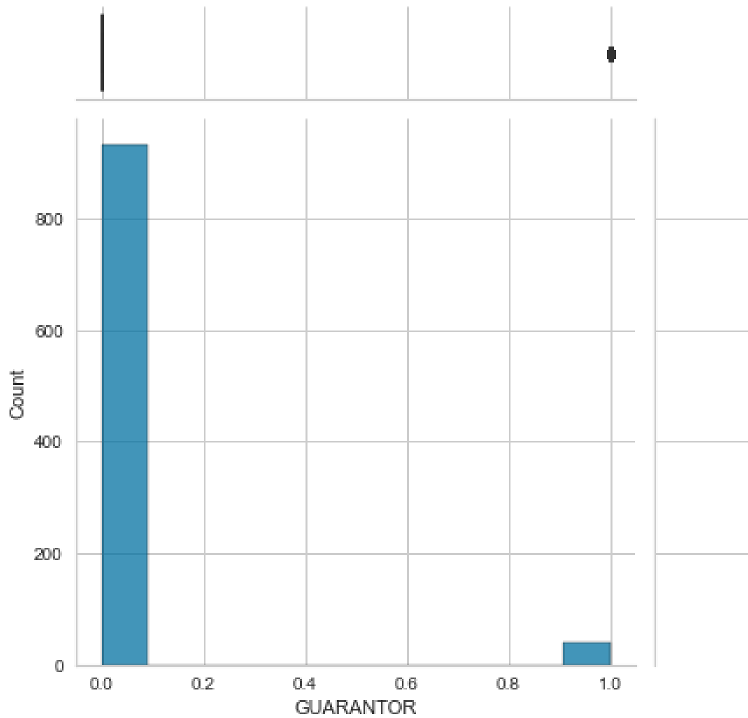
In [109]:
```
logit('CO_APPLICANT')
```

```
Optimization terminated successfully.
         Current function value: 0.602249
         Iterations 5
```

Out[109]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                            chi2                  P>chi2  df constraint
Intercept   [[141.7923707182207]]  1.0796447146874074e-32              1
x          [[2.5652574791884177]]     0.10923477116334147              1
```

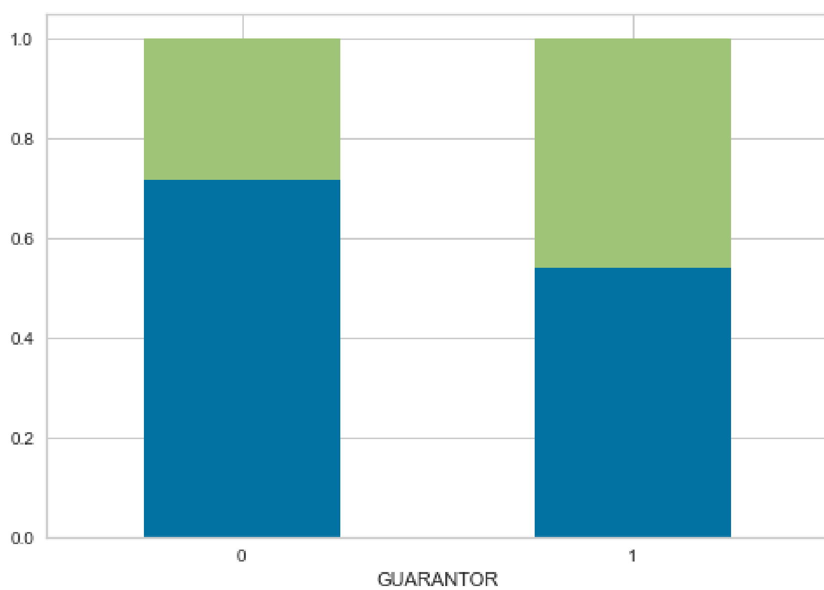
In [110]:
```
graf_func('GUARANTOR')
```

Out[110]: <seaborn.axisgrid.JointGrid at 0x1ceb94b6bb0>



In [111]:
```
contingency('GUARANTOR')
```

Out[111]: <AxesSubplot:xlabel='GUARANTOR'>



In [112]:
```
data_tabla('GUARANTOR')
```

Out[112]:

	GUARANTOR
count	970.000000
mean	0.040206
std	0.196544
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

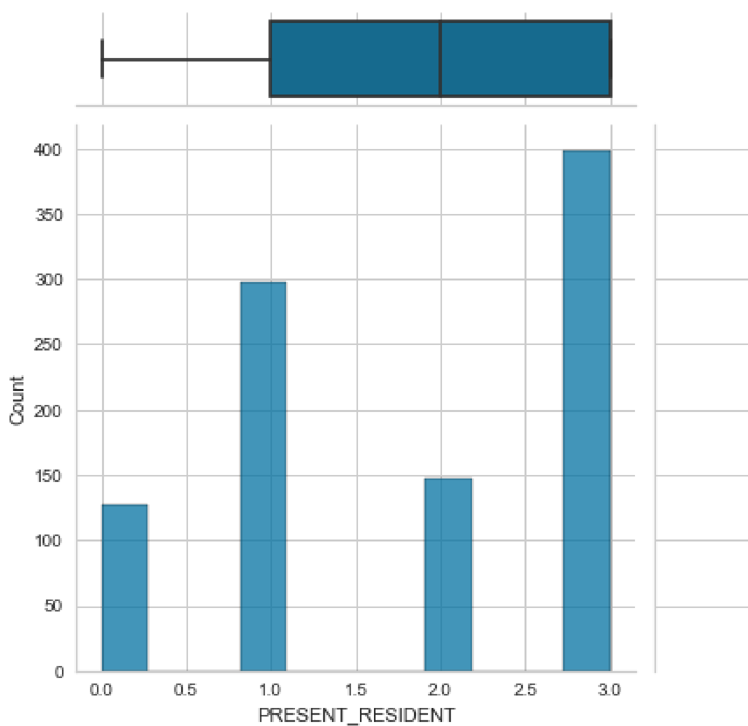
In [113]:
```
logit('GUARANTOR')
```

```
Optimization terminated successfully.
         Current function value: 0.601018
         Iterations 5
```

Out[113]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                P>chi2  df constraint
Intercept  [[160.99689778382512]]  6.852393228360414e-37              1
x            [[5.43032283797218]]   0.019790020314397704              1
```

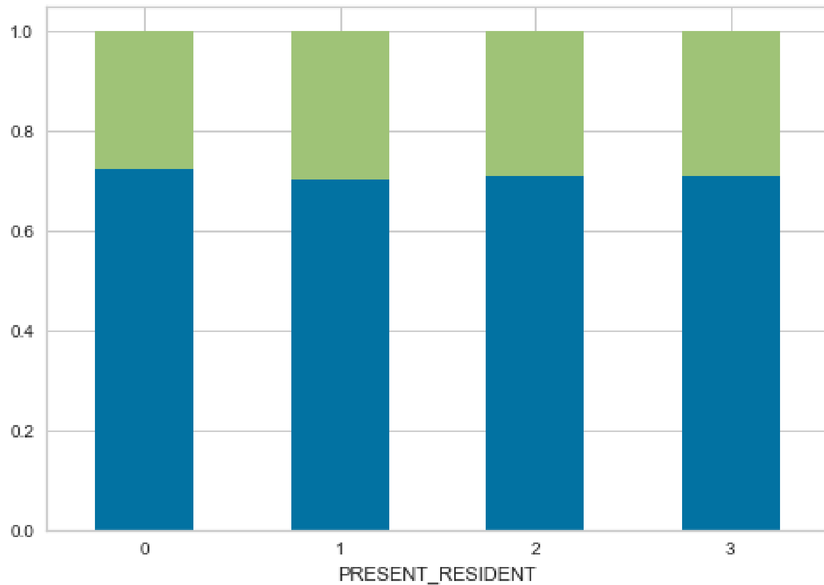
In [114]:
```
graf_func('PRESENT_RESIDENT')
```

Out[114]: <seaborn.axisgrid.JointGrid at 0x1ceba997be0>



In [115]:
```python
contingency('PRESENT_RESIDENT')
```

Out[115]: <AxesSubplot:xlabel='PRESENT_RESIDENT'>



In [116]:
```python
data_tabla('PRESENT_RESIDENT')
```

Out[116]:

	PRESENT_RESIDENT
count	970.000000
mean	1.841237
std	1.103307
min	0.000000
25%	1.000000
50%	2.000000
75%	3.000000
max	3.000000

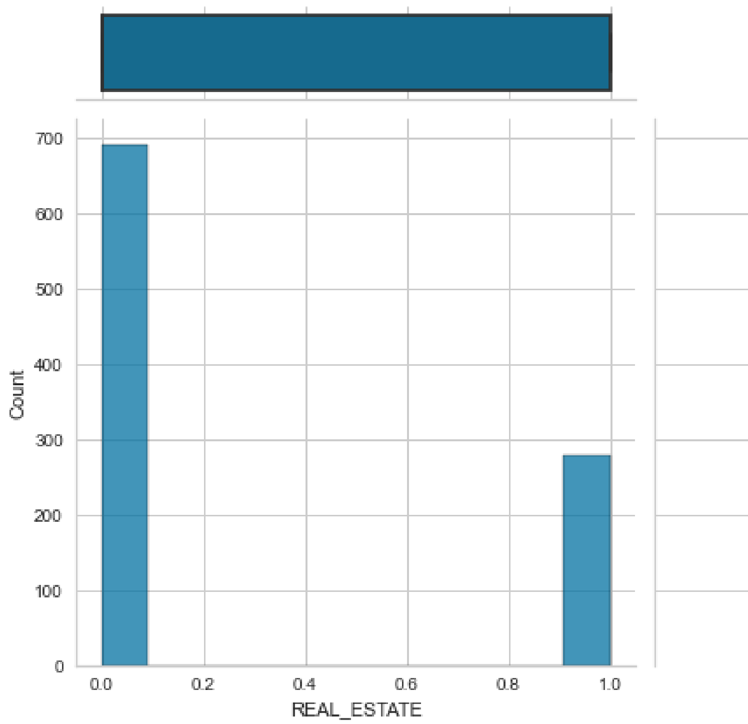
In [117]:
```python
logit('PRESENT_RESIDENT')
```

```
Optimization terminated successfully.
         Current function value: 0.603706
         Iterations 5
```

Out[117]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                              chi2                 P>chi2  df constraint
Intercept   [[42.832784325472154]]  5.962449716629546e-11              1
x          [[0.015280826553333353]]     0.9016196082186669              1
```

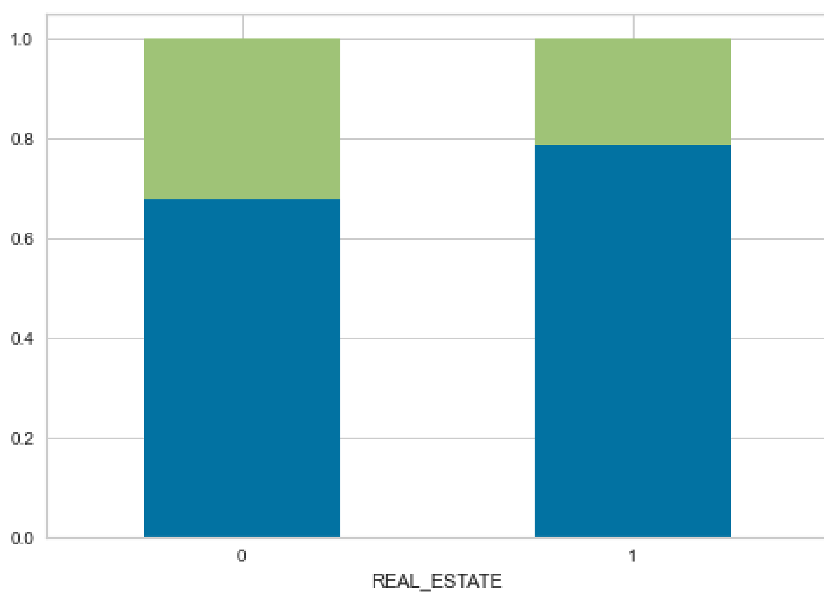
In [118]:
```
graf_func('REAL_ESTATE')
```

Out[118]: <seaborn.axisgrid.JointGrid at 0x1cebab9b340>



In [119]:
```
contingency('REAL_ESTATE')
```

Out[119]: <AxesSubplot:xlabel='REAL_ESTATE'>



In [120]:
```
data_tabla('REAL_ESTATE')
```

Out[120]:

	REAL_ESTATE
count	970.000000
mean	0.287629
std	0.452891
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	1.000000

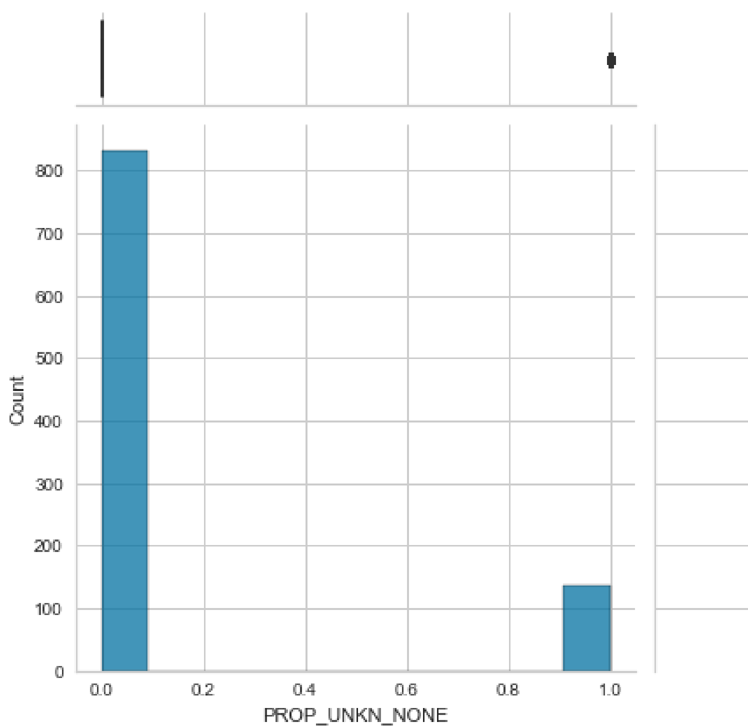
In [121]:
```
logit('REAL_ESTATE')
```

```
Optimization terminated successfully.
         Current function value: 0.597745
         Iterations 5
```

Out[121]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                 P>chi2  df constraint
Intercept   [[82.99589195586319]]  8.222408914419341e-20              1
x          [[10.996129995930378]]  0.0009130233025229733              1
```

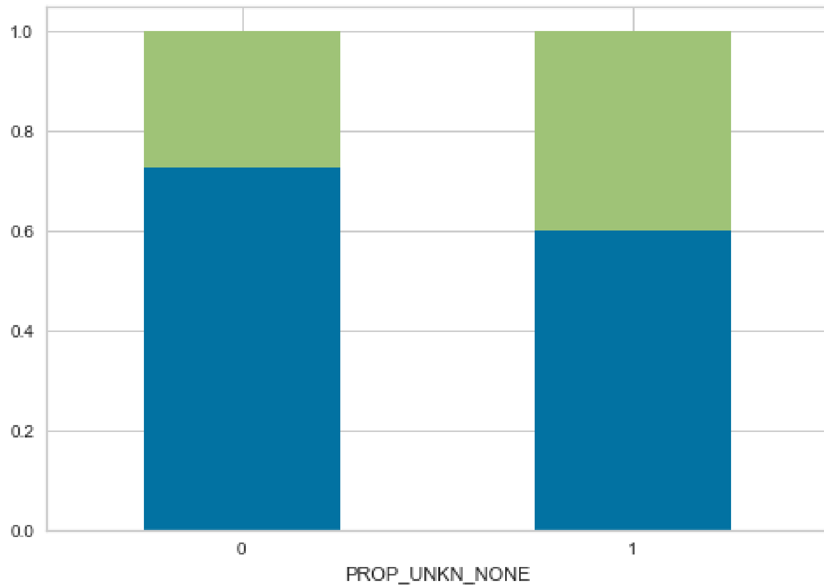
In [122]:
```
graf_func('PROP_UNKN_NONE')
```

Out[122]:
```
<seaborn.axisgrid.JointGrid at 0x1cebacc6490>
```



In [123]:
```
contingency('PROP_UNKN_NONE')
```

Out[123]: <AxesSubplot:xlabel='PROP_UNKN_NONE'>



In [124]:
```
data_tabla('PROP_UNKN_NONE')
```

Out[124]:

	PROP_UNKN_NONE
count	970.000000
mean	0.142268
std	0.349505
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

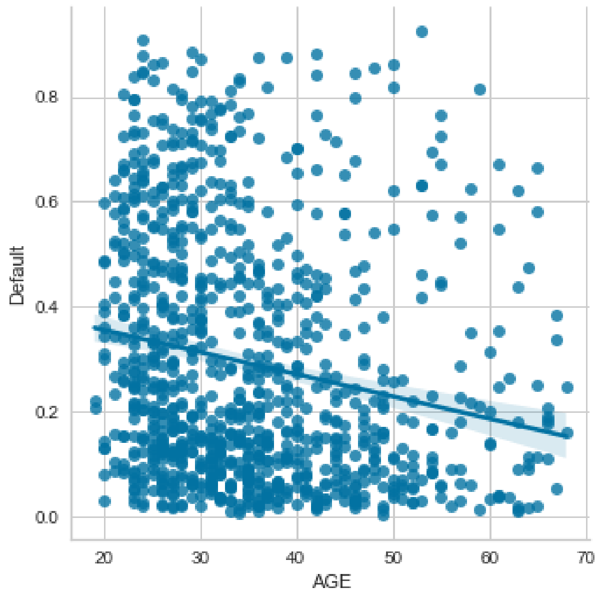
In [125]:
```
logit('PROP_UNKN_NONE')
```

```
Optimization terminated successfully.
         Current function value: 0.599353
         Iterations 5
```

Out[125]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                            chi2                 P>chi2  df constraint
Intercept  [[157.0978282469823]]  4.872739498504066e-36              1
x          [[8.730109643255638]]   0.003129976746550412              1
```

In [126]:
```
logplot1('AGE',df)
```

Out[126]:
```
<seaborn.axisgrid.FacetGrid at 0x1cebbeb8f40>
```



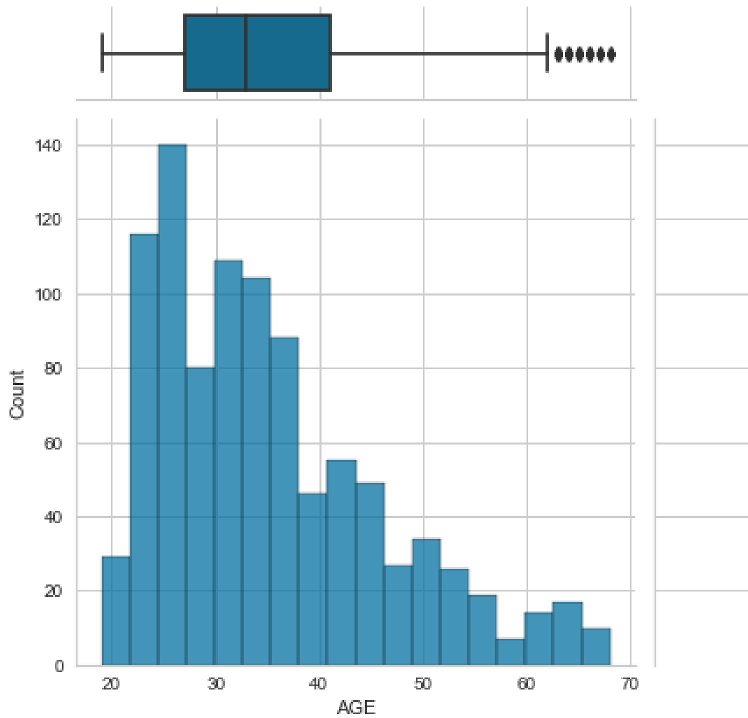
In [127]:
```
logit('AGE')
```

```
Optimization terminated successfully.
         Current function value: 0.598761
         Iterations 5
```

Out[127]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                              chi2                 P>chi2  df constraint
Intercept  [[0.42595248539651426]]     0.5139817148327255              1
x            [[9.172009468367161]]  0.0024574459391196123              1
```

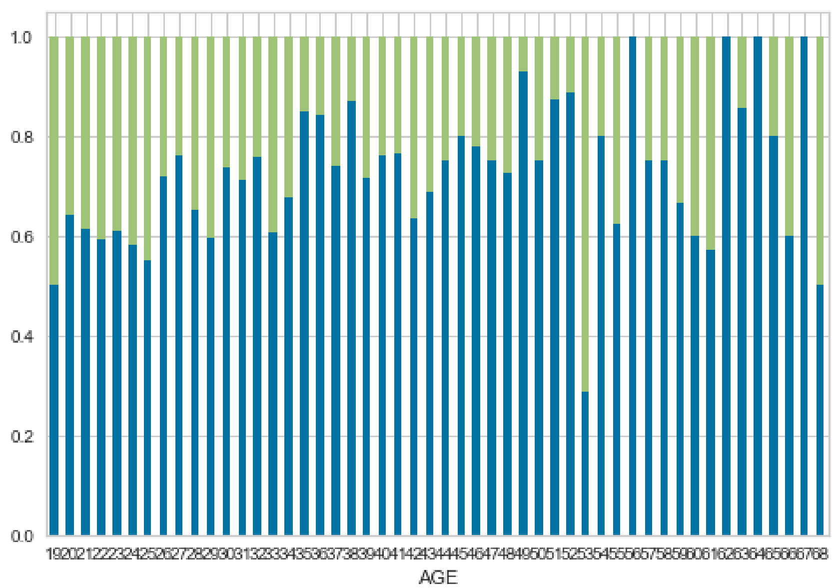
In [128]:
```
graf_func('AGE')
```

Out[128]: <seaborn.axisgrid.JointGrid at 0x1cebbe37b50>



In [129]:
```
contingency('AGE')
```

Out[129]: <AxesSubplot:xlabel='AGE'>



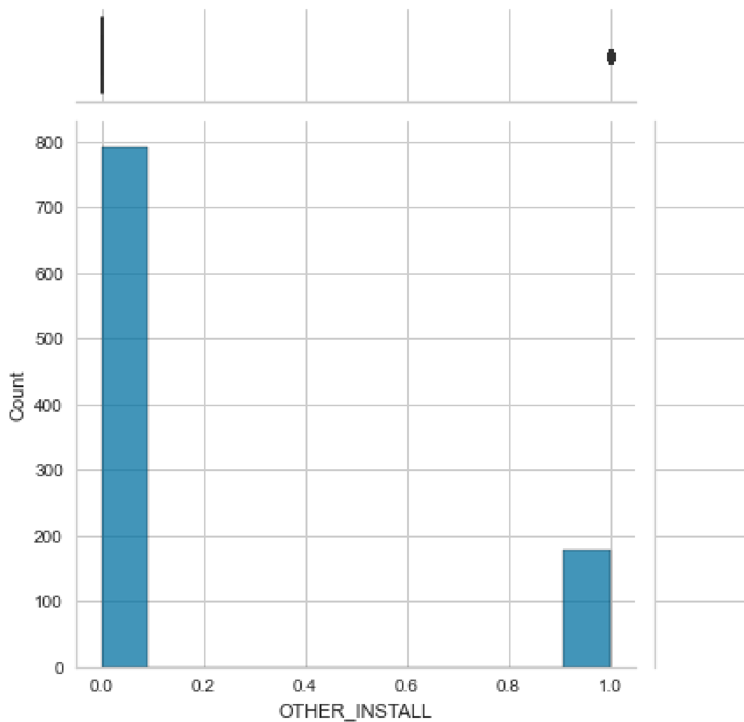
In [130]:
```
data_tabla('AGE')
```

Out[130]:

	AGE
count	970.000000
mean	35.180412
std	10.856671
min	19.000000
25%	27.000000
50%	33.000000
75%	41.000000
max	68.000000

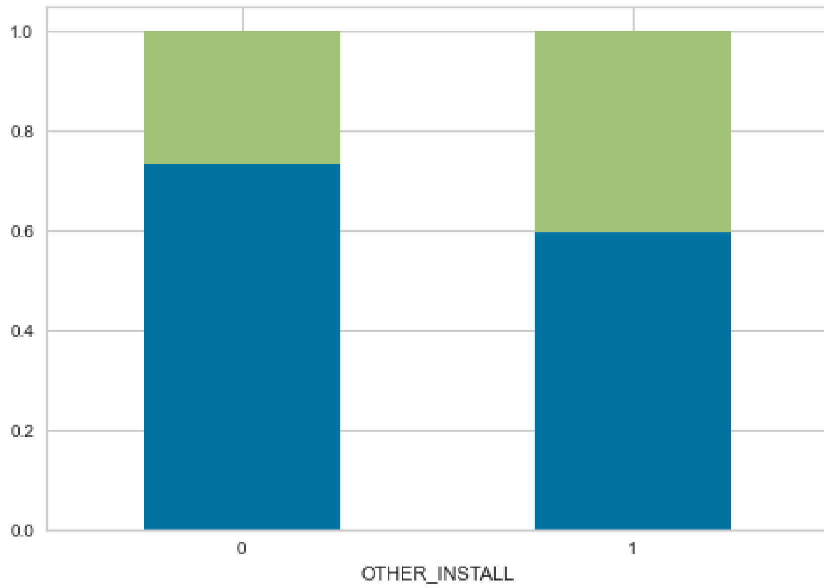
In [131]:
```
graf_func('OTHER_INSTALL')
```

Out[131]: <seaborn.axisgrid.JointGrid at 0x1cebc1d97f0>



In [132]:
```
contingency('OTHER_INSTALL')
```

Out[132]: <AxesSubplot:xlabel='OTHER_INSTALL'>



In [133]:
```
data_tabla('OTHER_INSTALL')
```

Out[133]:

	OTHER_INSTALL
count	970.000000
mean	0.184536
std	0.388121
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

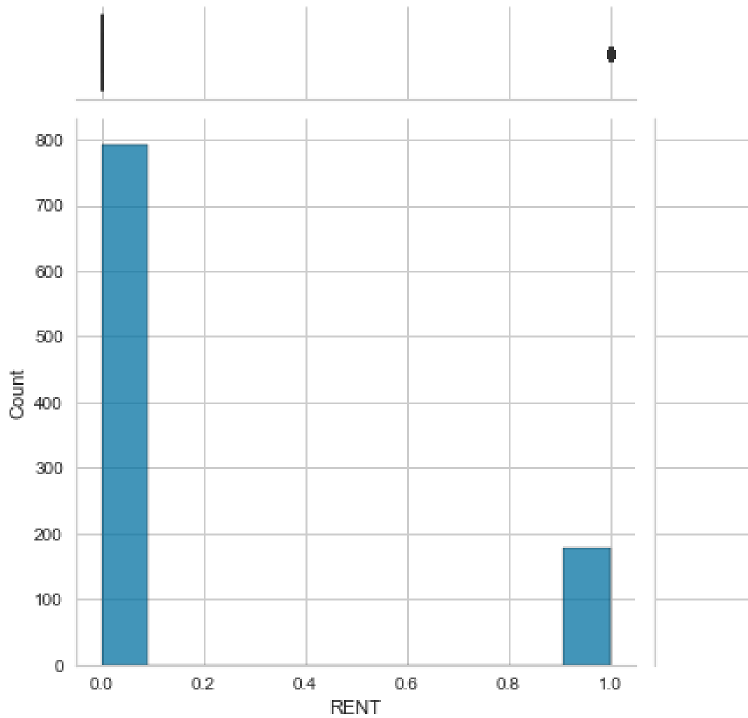
In [134]:
```
logit('OTHER_INSTALL')
```

```
Optimization terminated successfully.
         Current function value: 0.597330
         Iterations 5
```

Out[134]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                  P>chi2  df constraint
Intercept  [[158.19118964488155]]  2.8110285456033937e-36              1
x          [[12.735938942281992]]   0.0003586957020612892              1
```

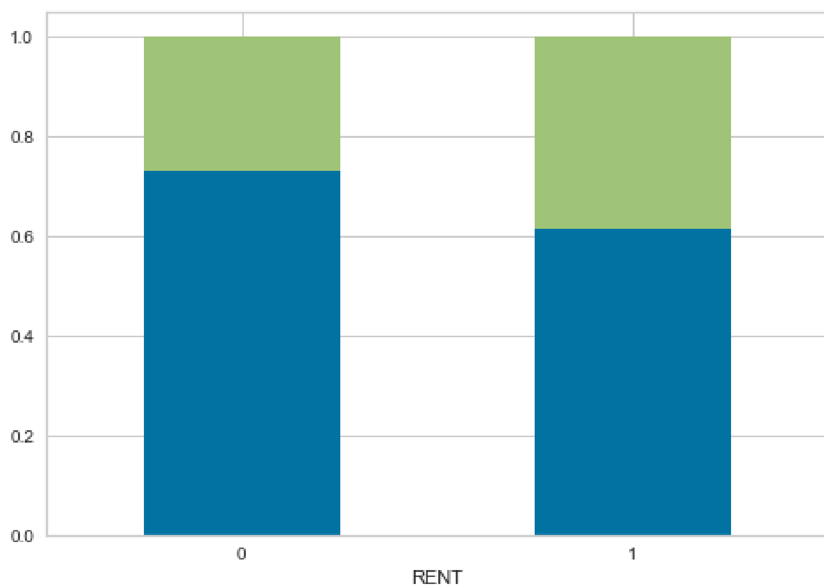
In [135]:
```
graf_func('RENT')
```

Out[135]: <seaborn.axisgrid.JointGrid at 0x1cebc3b5280>



In [136]:
```
contingency('RENT')
```

Out[136]: <AxesSubplot:xlabel='RENT'>



In [137]:
```
data_tabla('RENT')
```

Out[137]:

	RENT
count	970.000000
mean	0.183505
std	0.387280
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

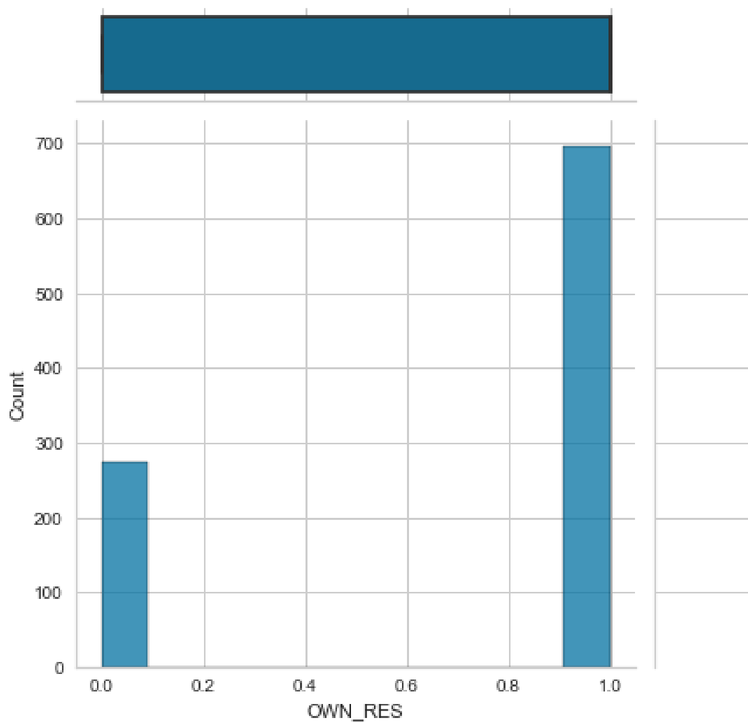
In [138]:
```
logit('RENT')
```

```
Optimization terminated successfully.
         Current function value: 0.598915
         Iterations 5
```

Out[138]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                P>chi2  df constraint
Intercept  [[154.18343592247024]]  2.1117428206297677e-35              1
x           [[9.566931581842052]]   0.0019811363739606403              1
```

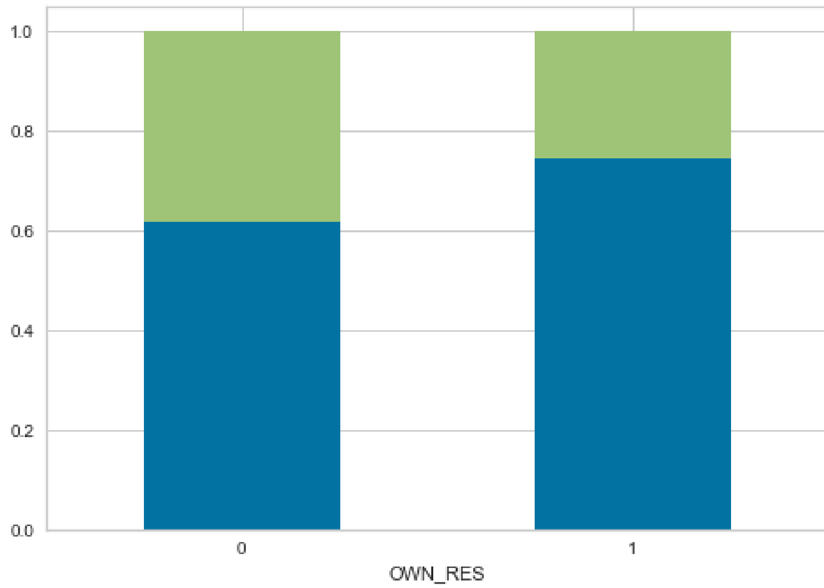
In [139]:
```
graf_func('OWN_RES')
```

Out[139]:
```
<seaborn.axisgrid.JointGrid at 0x1cebc4ed910>
```



In [140]:
```
contingency('OWN_RES')
```

Out[140]:
```
<AxesSubplot:xlabel='OWN_RES'>
```



In [141]:
```
data_tabla('OWN_RES')
```

Out[141]:

	OWN_RES
count	970.000000
mean	0.717526
std	0.450435
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

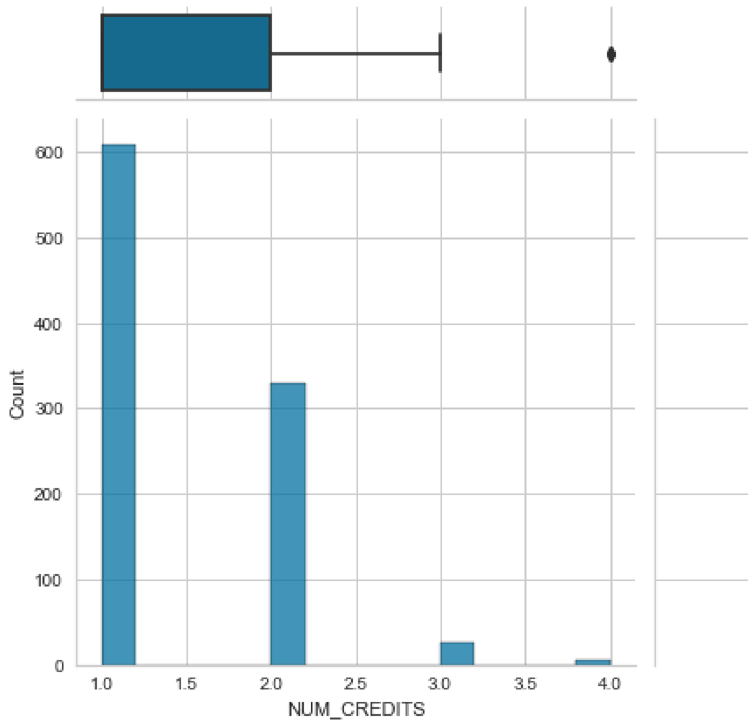
In [142]:
```
logit('OWN_RES')
```

```
Optimization terminated successfully.
         Current function value: 0.595976
         Iterations 5
```

Out[142]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                               chi2                P>chi2  df constraint
Intercept  [[14.66989039636329]]  0.0001280759466964206              1
x          [[15.25760828201866]]  9.379815084928388e-05              1
```

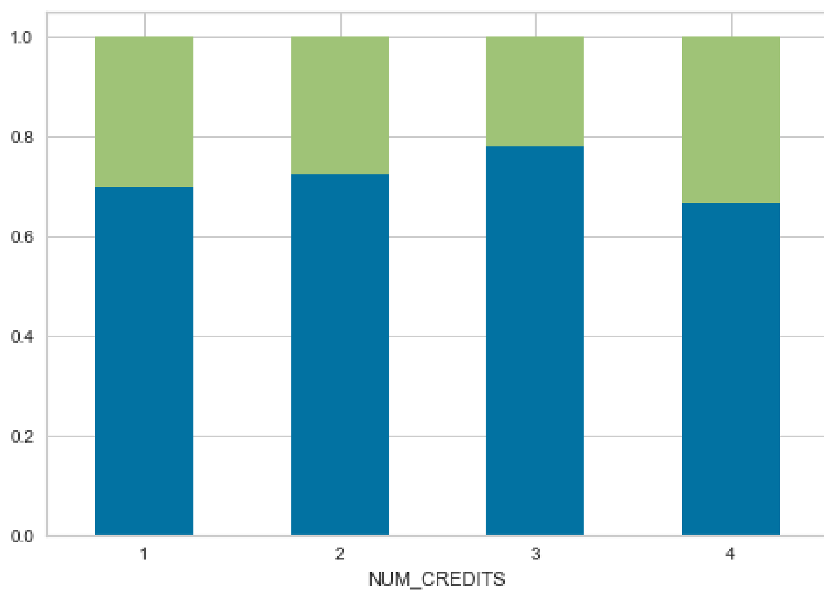
In [143]:
```
graf_func('NUM_CREDITS')
```

Out[143]: <seaborn.axisgrid.JointGrid at 0x1cebd5c1f40>



In [144]:
```
contingency('NUM_CREDITS')
```

Out[144]: <AxesSubplot:xlabel='NUM_CREDITS'>



In [145]:
```
data_tabla('NUM_CREDITS')
```

Out[145]:

	NUM_CREDITS
count	970.000000
mean	1.413402
std	0.579336
min	1.000000
25%	1.000000
50%	1.000000
75%	2.000000
max	4.000000

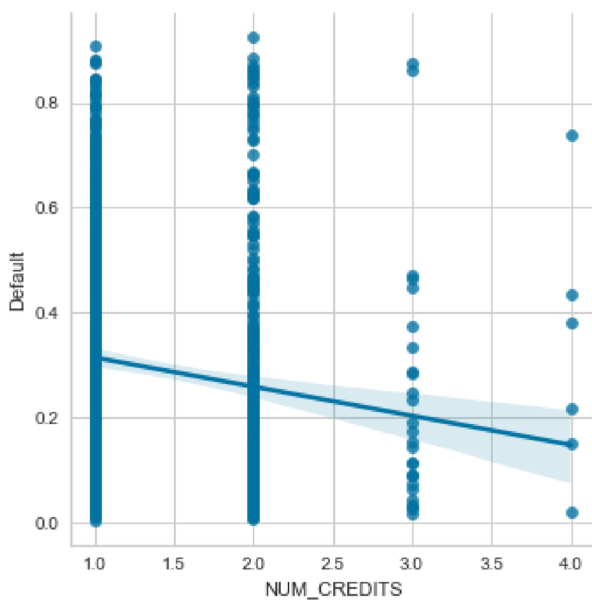
In [146]:
```
logit('NUM_CREDITS')
```

```
Optimization terminated successfully.
         Current function value: 0.603217
         Iterations 5
```

Out[146]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                             chi2                 P>chi2  df constraint
Intercept  [[14.58129960449505]]  0.0001342401241657876              1
x          [[0.949184981777088]]     0.3299268399728925              1
```

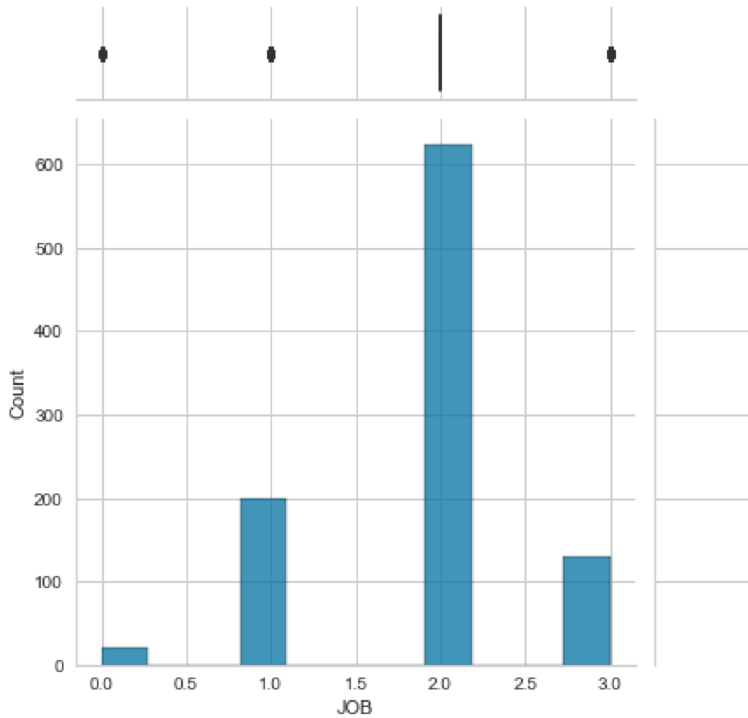
In [147]:
```
logplot1('NUM_CREDITS',df)
```

Out[147]:
```
<seaborn.axisgrid.FacetGrid at 0x1cebd825f70>
```



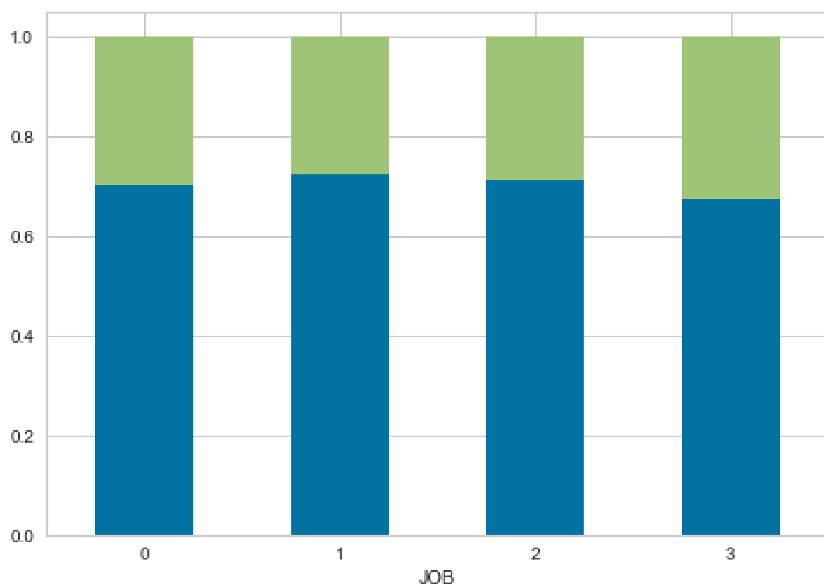
In [148]:
```
graf_func('JOB')
```

Out[148]: <seaborn.axisgrid.JointGrid at 0x1cebd8065e0>



In [149]:
```
contingency('JOB')
```

Out[149]: <AxesSubplot:xlabel='JOB'>



In [150]:
```
data_tabla('JOB')
```

Out[150]:

	JOB
count	970.000000
mean	1.887629
std	0.638264
min	0.000000
25%	2.000000
50%	2.000000
75%	2.000000
max	3.000000

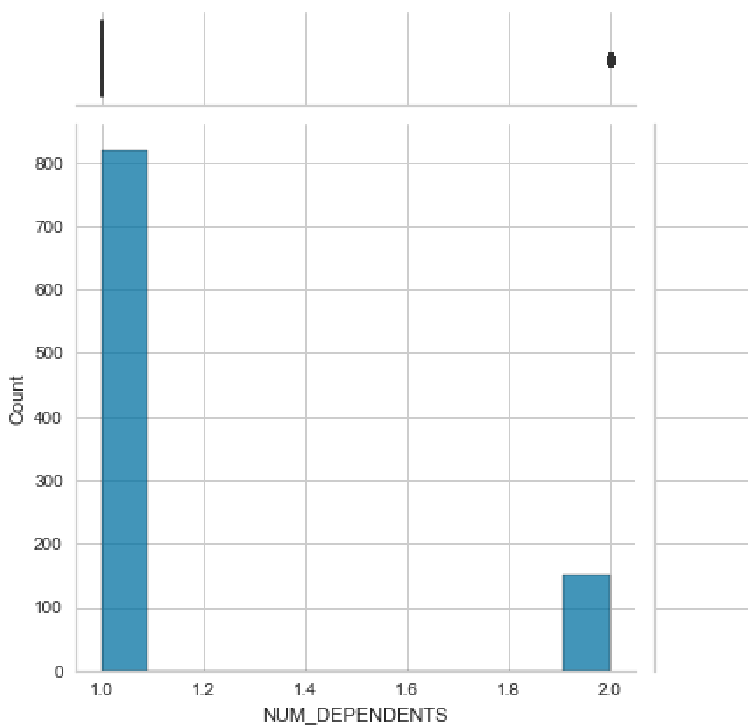
In [151]:
```
logit('JOB')
```

```
Optimization terminated successfully.
         Current function value: 0.603420
         Iterations 5
```

Out[151]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                               chi2                 P>chi2  df constraint
Intercept  [[21.955806466039892]]  2.790014978110281e-06              1
x          [[0.5666815862690652]]     0.4515805870011226              1
```

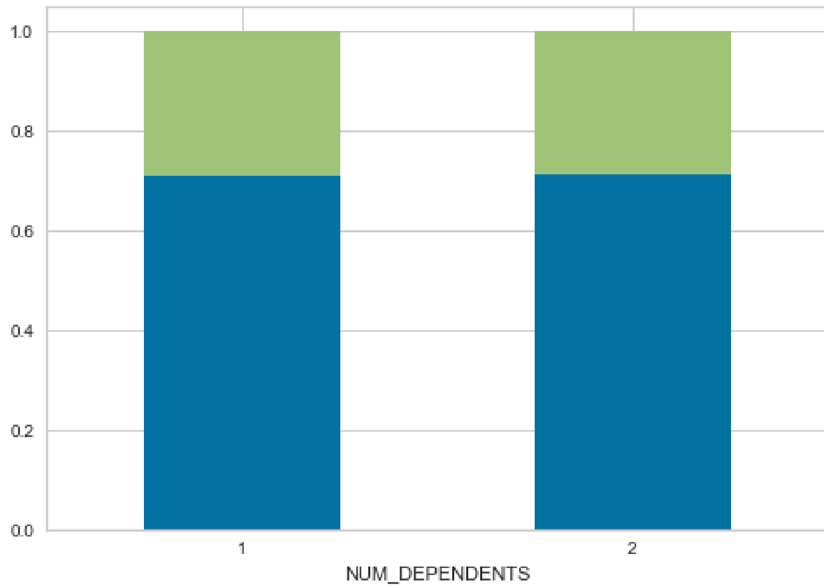
In [152]:
```
graf_func('NUM_DEPENDENTS')
```

Out[152]: <seaborn.axisgrid.JointGrid at 0x1cebd9aa700>



In [153]:
```
contingency('NUM_DEPENDENTS')
```

Out[153]: <AxesSubplot:xlabel='NUM_DEPENDENTS'>



In [154]:
```
data_tabla('NUM_DEPENDENTS')
```

Out[154]:

	NUM_DEPENDENTS
count	970.000000
mean	1.156701
std	0.363706
min	1.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	2.000000

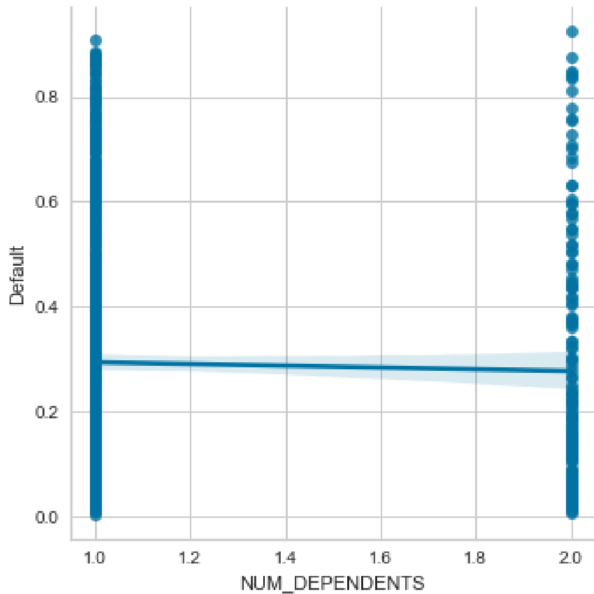
In [155]:
```
logit('NUM_DEPENDENTS')
```

```
Optimization terminated successfully.
         Current function value: 0.603711
         Iterations 5
```

Out[155]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                                chi2                 P>chi2  df constraint
Intercept    [[13.659986106490326]]  0.0002190731210630657               1
x          [[0.004530081754561938]]     0.9463381943343964               1
```

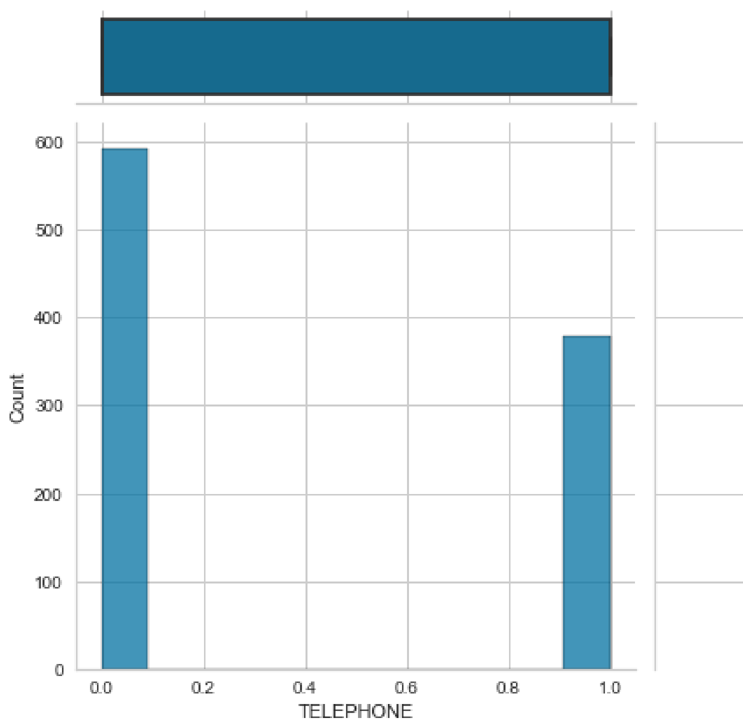
In [156]:
```
logplot1('NUM_DEPENDENTS',df)
```

Out[156]: <seaborn.axisgrid.FacetGrid at 0x1cebdbd54c0>



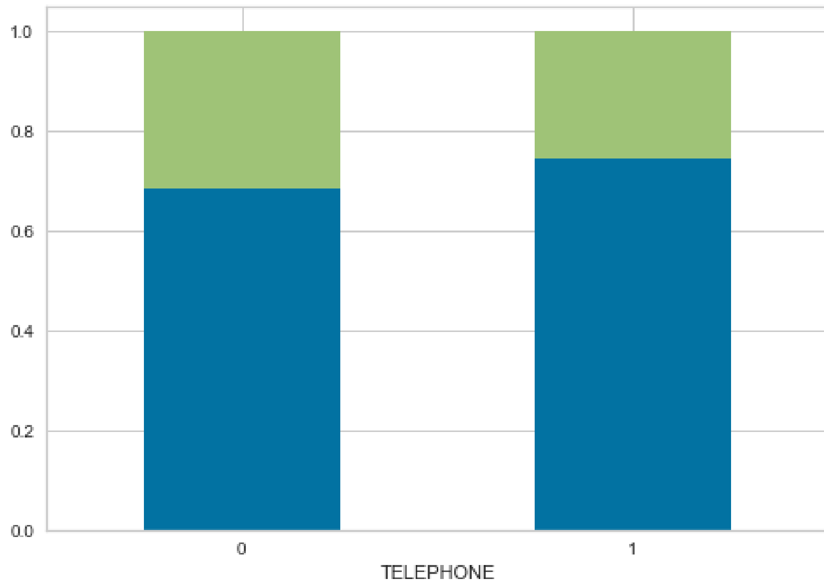
In [157]:
```
graf_func('TELEPHONE')
```

Out[157]: <seaborn.axisgrid.JointGrid at 0x1cebdb06700>



In [158]:
```
contingency('TELEPHONE')
```

Out[158]: <AxesSubplot:xlabel='TELEPHONE'>



In [159]:
```
data_tabla('TELEPHONE')
```

Out[159]:

	TELEPHONE
count	970.000000
mean	0.389691
std	0.487932
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	1.000000

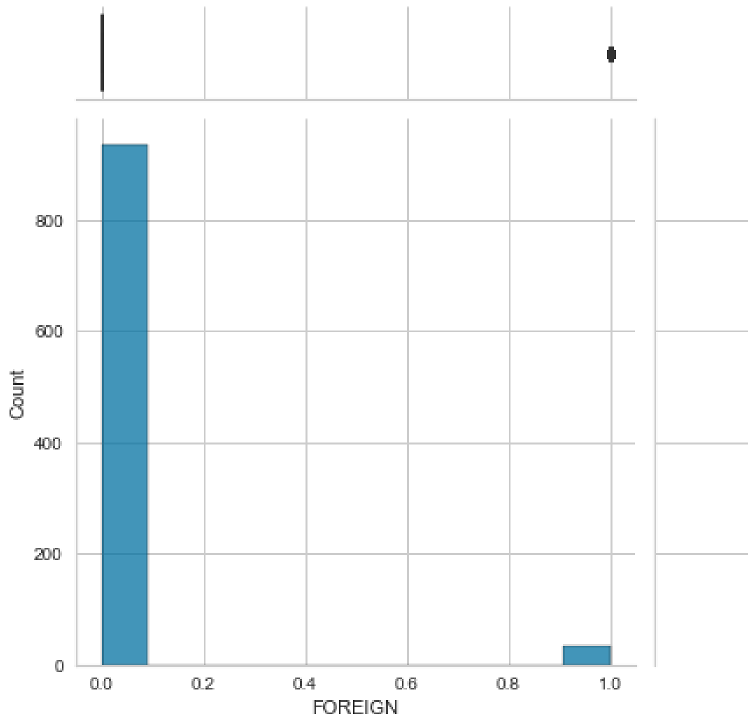
In [160]:
```
logit('TELEPHONE')
```

```
Optimization terminated successfully.
         Current function value: 0.601787
         Iterations 5
```

Out[160]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                              chi2                  P>chi2  df constraint
Intercept   [[77.72873179480068]]  1.1820927382996235e-18              1
x          [[3.6904500789542554]]     0.05472484726136939              1
```

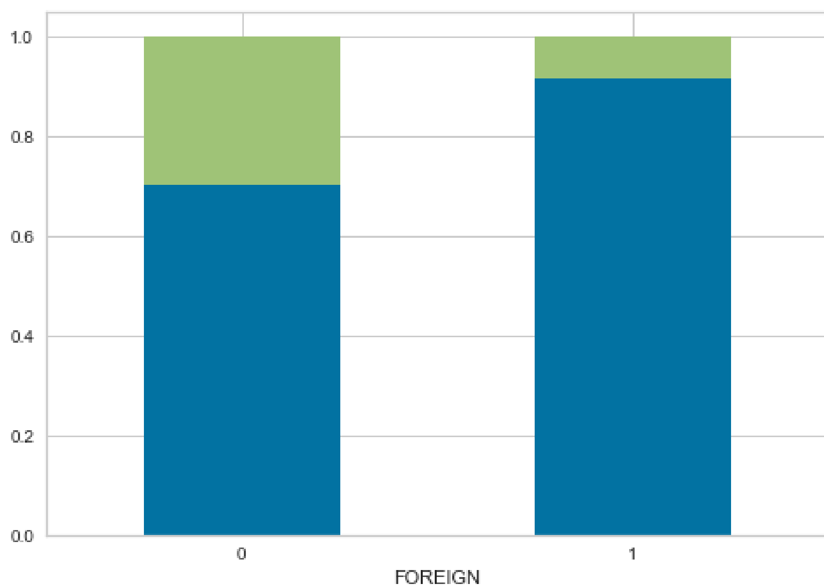
In [161]:
```
graf_func('FOREIGN')
```

Out[161]: <seaborn.axisgrid.JointGrid at 0x1cebed909a0>



In [162]:
```
contingency('FOREIGN')
```

Out[162]: <AxesSubplot:xlabel='FOREIGN'>



In [163]:
```
data_tabla('FOREIGN')
```

Out[163]:

	FOREIGN
count	970.000000
mean	0.036082
std	0.186592
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

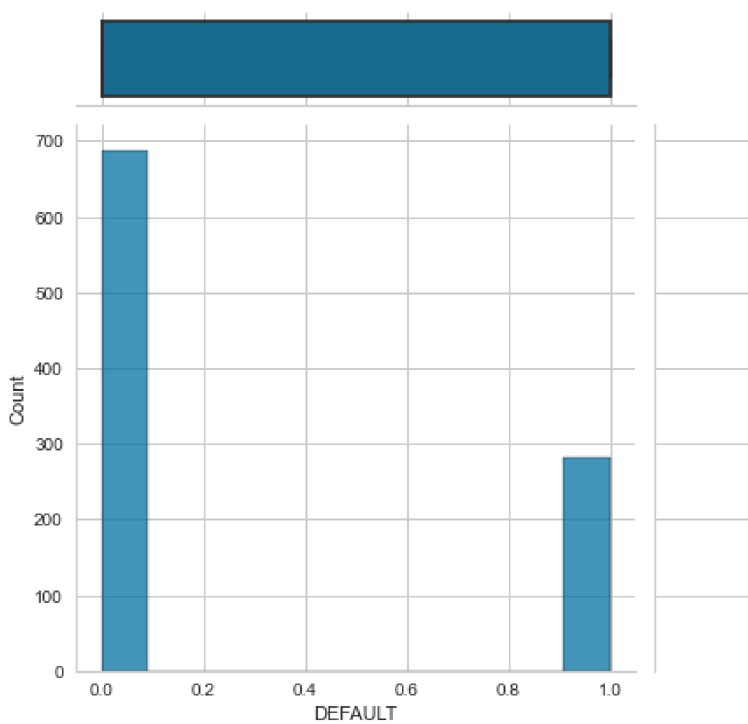
In [164]:
```
logit('FOREIGN')
```

```
Optimization terminated successfully.
         Current function value: 0.598940
         Iterations 6
```

Out[164]:
```
<class 'statsmodels.stats.contrast.WaldTestResults'>
                                chi2                   P>chi2  df constraint
Intercept  [[141.66671193641702]]  1.1501572465254038e-32              1
x            [[6.22734056625585]]    0.012579251950645912              1
```

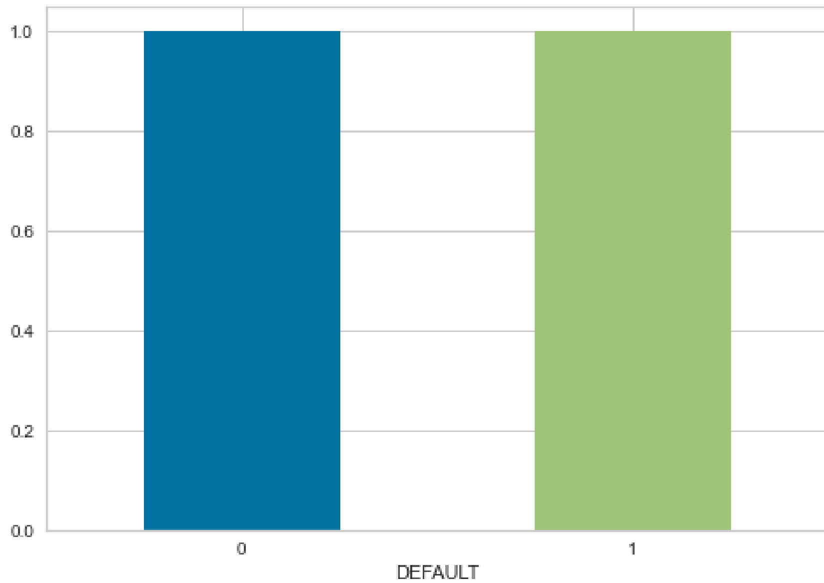
In [165]:
```
graf_func('DEFAULT')
```

Out[165]: <seaborn.axisgrid.JointGrid at 0x1cebecf0490>



In [166]:
```
contingency('DEFAULT')
```

Out[166]: <AxesSubplot:xlabel='DEFAULT'>



In [167]:
```
data_tabla('DEFAULT')
```

Out[167]:

	DEFAULT
count	970.000000
mean	0.291753
std	0.454804
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	1.000000

In [168]:
```
scaler= MinMaxScaler()

x_scaler= scaler.fit_transform(df.drop(columns='DEFAULT'))

x_scaler
```

Out[168]:
```
array([[0.        , 0.03571429, 1.        , ..., 0.        , 1.        ,
        0.        ],
       [0.33333333, 0.78571429, 0.5       , ..., 0.        , 0.        ,
        0.        ],
       [1.        , 0.14285714, 1.        , ..., 1.        , 0.        ,
        0.        ],
       ...,
       [1.        , 0.14285714, 0.5       , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.73214286, 0.5       , ..., 0.        , 1.        ,
        0.        ],
       [0.33333333, 0.73214286, 1.        , ..., 0.        , 0.        ,
        0.        ]])
```

In [169]:
```
data_scaler= pd.DataFrame(data=(x_scaler), columns=(df.drop(columns='DEFAULT').columns))
df_scal= pd.concat([data_scaler,df['DEFAULT']],axis=1)
df_scal
```

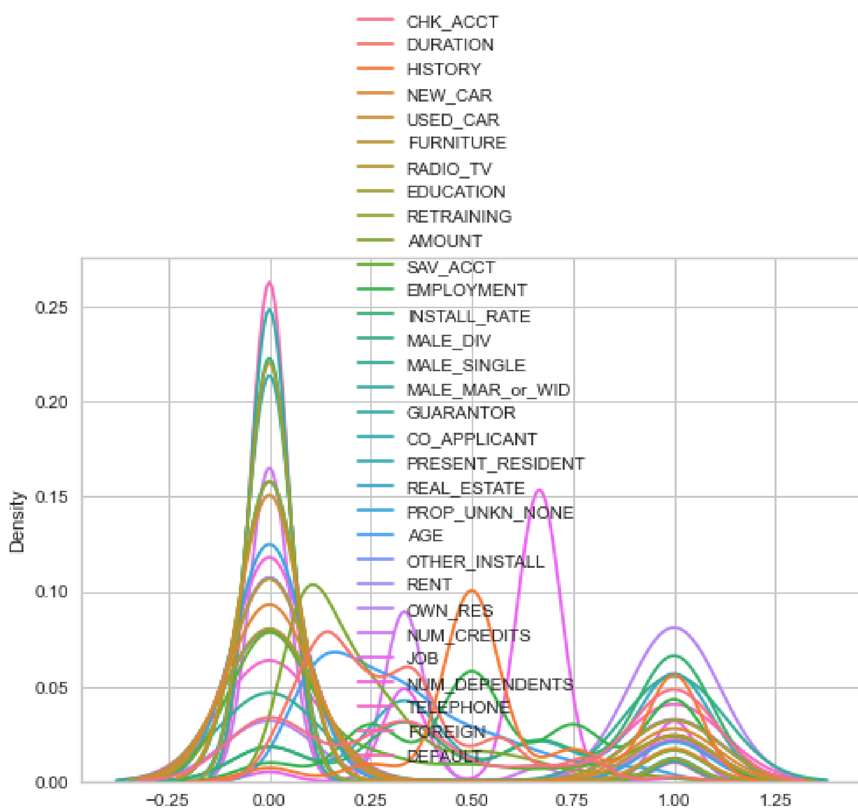
Out[169]:

	CHK_ACCT	DURATION	HISTORY	NEW_CAR	USED_CAR	FURNITURE	RADIO_TV	EDUCATION	RETRAINING
0	0.000000	0.035714	1.00	0.0	0.0	0.0	1.0	0.0	0.0
1	0.333333	0.785714	0.50	0.0	0.0	0.0	1.0	0.0	0.0
2	1.000000	0.142857	1.00	0.0	0.0	0.0	0.0	1.0	0.0
3	0.000000	0.678571	0.50	0.0	0.0	1.0	0.0	0.0	0.0
4	0.000000	0.357143	0.75	1.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...
965	1.000000	0.142857	0.50	0.0	0.0	1.0	0.0	0.0	0.0
966	0.000000	0.464286	0.50	0.0	1.0	0.0	0.0	0.0	0.0
967	1.000000	0.142857	0.50	0.0	0.0	0.0	1.0	0.0	0.0
968	0.000000	0.732143	0.50	0.0	0.0	0.0	1.0	0.0	0.0
969	0.333333	0.732143	1.00	0.0	1.0	0.0	0.0	0.0	0.0

970 rows × 31 columns

In [170]:
```
sns.kdeplot(data=df_scal)
```

Out[170]: <AxesSubplot:ylabel='Density'>



In [171]:
```
correlation,p= stats.spearmanr(df)
correlation=pd.DataFrame(data= correlation, columns= df.columns, index= df.columns)
correlation['DEFAULT']
```

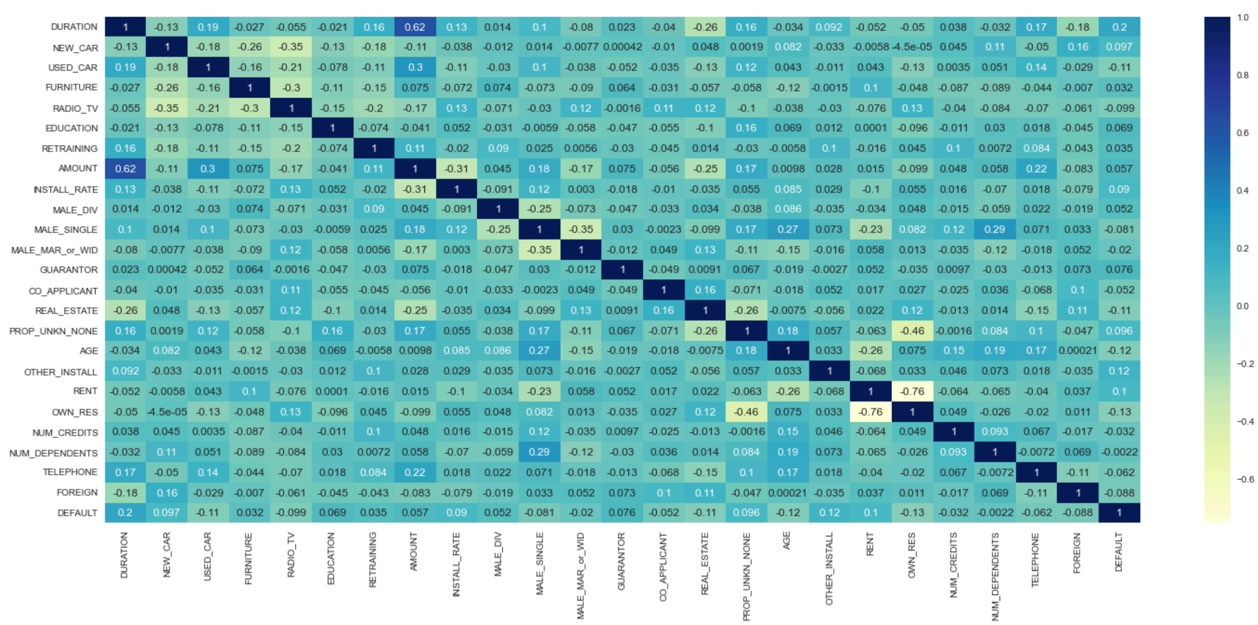
Out[171]:
```
CHK_ACCT           -0.350126
DURATION            0.199830
HISTORY            -0.207638
NEW_CAR             0.096692
USED_CAR           -0.111501
FURNITURE           0.031996
RADIO_TV           -0.099494
EDUCATION           0.069425
RETRAINING          0.034617
AMOUNT              0.056642
SAV_ACCT           -0.177027
EMPLOYMENT         -0.130025
INSTALL_RATE        0.089639
MALE_DIV            0.052243
MALE_SINGLE        -0.080831
MALE_MAR_or_WID    -0.019831
GUARANTOR           0.076447
CO_APPLICANT       -0.052067
PRESENT_RESIDENT    0.003448
REAL_ESTATE        -0.107214
PROP_UNKN_NONE      0.095685
AGE                -0.116987
OTHER_INSTALL       0.115619
RENT                0.100003
OWN_RES            -0.126240
NUM_CREDITS        -0.032425
JOB                 0.025611
NUM_DEPENDENTS     -0.002161
TELEPHONE          -0.061769
FOREIGN            -0.087695
DEFAULT             1.000000
Name: DEFAULT, dtype: float64
```

In [172]:
```
pvalue=pd.DataFrame(data= p, columns= df.columns, index= df.columns)
pvalue['DEFAULT']
```

Out[172]:
```
CHK_ACCT            2.355026e-29
DURATION            3.406040e-10
HISTORY             6.586220e-11
NEW_CAR             2.572897e-03
USED_CAR            5.031866e-04
FURNITURE           3.195091e-01
RADIO_TV            1.919355e-03
EDUCATION           3.061490e-02
RETRAINING          2.814411e-01
AMOUNT              7.785649e-02
SAV_ACCT            2.852359e-08
EMPLOYMENT          4.873238e-05
INSTALL_RATE        5.208762e-03
MALE_DIV            1.039278e-01
MALE_SINGLE         1.179053e-02
MALE_MAR_or_WID     5.373093e-01
GUARANTOR           1.724993e-02
CO_APPLICANT        1.050976e-01
PRESENT_RESIDENT    9.145948e-01
REAL_ESTATE         8.243022e-04
PROP_UNKN_NONE      2.853614e-03
AGE                 2.608398e-04
OTHER_INSTALL       3.080844e-04
RENT                1.818563e-03
OWN_RES             8.065620e-05
NUM_CREDITS         3.130538e-01
JOB                 4.255872e-01
NUM_DEPENDENTS      9.464070e-01
TELEPHONE           5.446139e-02
FOREIGN             6.276042e-03
DEFAULT             0.000000e+00
Name: DEFAULT, dtype: float64
```

In [173]:
```
correlation= df.corr(method='spearman')
plt.figure(figsize=(25,10))
sns.heatmap(correlation, annot=True, cmap= 'YlGnBu')
```

Out[173]: <AxesSubplot:>



In [174]:
```
df_var= VarClusHi(df_scal,maxclus=None,maxeigval2=0.7)
df_var.varclus()
```

Out[174]: <varclushi.varclushi.VarClusHi at 0x1cebf232be0>

In [175]:
```
df_var.info
```

Out[175]:

	Cluster	N_Vars	Eigval1	Eigval2	VarProp
0	0	2	1.635864	0.364136	0.817932
1	1	1	1.000000	0.000000	1.000000
2	2	2	1.755574	0.244426	0.877787
3	3	2	1.345434	0.654566	0.672717
4	4	2	1.350721	0.649279	0.675361
5	5	2	1.435437	0.564563	0.717719
6	6	1	1.000000	0.000000	1.000000
7	7	1	1.000000	0.000000	1.000000
8	8	1	1.000000	0.000000	1.000000
9	9	2	1.369402	0.630598	0.684701
10	10	1	1.000000	0.000000	1.000000
11	11	1	1.000000	0.000000	1.000000
12	12	1	1.000000	0.000000	1.000000
13	13	1	1.000000	0.000000	1.000000
14	14	1	1.000000	0.000000	1.000000
15	15	1	1.000000	0.000000	1.000000
16	16	1	1.000000	0.000000	1.000000
17	17	1	1.000000	0.000000	1.000000
18	18	1	1.000000	0.000000	1.000000
19	19	1	1.000000	0.000000	1.000000
20	20	1	1.000000	0.000000	1.000000
21	21	1	1.000000	0.000000	1.000000
22	22	1	1.000000	0.000000	1.000000
23	23	1	1.000000	0.000000	1.000000
24	24	1	1.000000	0.000000	1.000000

In [176]:
```
df_var.rsquare
```

Out[176]:

	Cluster	Variable	RS_Own	RS_NC	RS_Ratio
0	0	DURATION	0.817932	0.054053	1.924719e-01
1	0	AMOUNT	0.817932	0.101059	2.025362e-01
2	1	AGE	1.000000	0.081224	0.000000e+00
3	2	RENT	0.877787	0.052142	1.289362e-01
4	2	OWN_RES	0.877787	0.210672	1.548321e-01
5	3	NEW_CAR	0.672717	0.068019	3.511690e-01
6	3	RADIO_TV	0.672717	0.091073	3.600763e-01
7	4	CHK_ACCT	0.675361	0.052151	3.425012e-01
8	4	DEFAULT	0.675361	0.032215	3.354458e-01
9	5	HISTORY	0.717719	0.059328	3.000847e-01
10	5	NUM_CREDITS	0.717719	0.027227	2.901820e-01
11	6	OTHER_INSTALL	1.000000	0.010435	0.000000e+00
12	7	PRESENT_RESIDENT	1.000000	0.072989	0.000000e+00
13	8	NUM_DEPENDENTS	1.000000	0.085698	0.000000e+00
14	9	JOB	0.684701	0.059501	3.352462e-01
15	9	TELEPHONE	0.684701	0.040710	3.286792e-01
16	10	INSTALL_RATE	1.000000	0.016419	0.000000e+00
17	11	PROP_UNKN_NONE	1.000000	0.066970	0.000000e+00
18	12	CO_APPLICANT	1.000000	0.026315	0.000000e+00
19	13	GUARANTOR	1.000000	0.005493	0.000000e+00
20	14	FOREIGN	1.000000	0.017689	0.000000e+00
21	15	EDUCATION	1.000000	0.026275	0.000000e+00
22	16	MALE_DIV	1.000000	0.061932	0.000000e+00
23	17	FURNITURE	1.000000	0.025898	0.000000e+00
24	18	MALE_MAR_or_WID	1.000000	0.123157	0.000000e+00
25	19	RETRAINING	1.000000	0.023588	2.274088e-16
26	20	USED_CAR	1.000000	0.073342	0.000000e+00
27	21	SAV_ACCT	1.000000	0.061575	0.000000e+00
28	22	REAL_ESTATE	1.000000	0.067929	0.000000e+00
29	23	EMPLOYMENT	1.000000	0.081224	0.000000e+00
30	24	MALE_SINGLE	1.000000	0.123157	0.000000e+00

In [177]:
```
corr= stats.spearmanr(df_scal[['DEFAULT','RENT','OWN_RES']])
corr[0][0]
```

Out[177]:
```
array([ 1.        ,  0.10000259, -0.12624022])
```

In [178]:
```
corr2= stats.spearmanr(df_scal[['DEFAULT','AMOUNT','DURATION']])
corr2[0][0]
```

Out[178]:
```
array([1.        , 0.05664248, 0.19982967])
```

In [179]:
```
corr3= stats.spearmanr(df_scal[['DEFAULT','HISTORY','NUM_CREDITS']])
corr3[0][0]
```

Out[179]:
```
array([ 1.        , -0.20763757, -0.03242503])
```

In [180]:
```
corr4= stats.spearmanr(df_scal[['DEFAULT','NEW_CAR','RADIO_TV']])
corr4[0][0]
```

Out[180]:
```
array([ 1.        ,  0.09669185, -0.09949423])
```

In [181]:
```
corr5= stats.spearmanr(df_scal[['DEFAULT','TELEPHONE','JOB']])
corr5[0][0]
```

Out[181]:
```
array([ 1.        , -0.06176926,  0.02561143])
```

In [182]:
```
df.drop(columns=['DEFAULT']).astype('int64').sum().rank(method='average', ascending=False)
```

Out[182]:
```
CHK_ACCT             9.0
DURATION             3.0
HISTORY              5.0
NEW_CAR             18.0
USED_CAR            23.0
FURNITURE           19.0
RADIO_TV            17.0
EDUCATION           27.0
RETRAINING          24.5
AMOUNT               1.0
SAV_ACCT            12.0
EMPLOYMENT           6.0
INSTALL_RATE         4.0
MALE_DIV            28.0
MALE_SINGLE         14.0
MALE_MAR_or_WID     24.5
GUARANTOR           29.0
CO_APPLICANT        26.0
PRESENT_RESIDENT     8.0
REAL_ESTATE         16.0
PROP_UNKN_NONE      22.0
AGE                  2.0
OTHER_INSTALL       20.0
RENT                21.0
OWN_RES             13.0
NUM_CREDITS         10.0
JOB                  7.0
NUM_DEPENDENTS      11.0
TELEPHONE           15.0
FOREIGN             30.0
dtype: float64
```

In [183]:
```
def woe_iv(column, dtype,user_splits=None):
    x= df[column]
    y= df['DEFAULT']
    #y= default['N_DEFAULT'].values
    optb= OptimalBinning(name='y', dtype=dtype,monotonic_trend='auto',
                         user_splits=user_splits


                        )
    #method : 'uniform', 'quantile', 'cart', 'mdlp'
    #solver : str, optional (default="cp")
    #The optimizer to solve the optimal binning problem. Supported solversare "mip" to choose
    optb.fit(x,y)
    binning_table= optb.binning_table
    return binning_table.build()
```

In [184]:
```
def woe(column, dtype,bins,method):
    x= df[column]
    y= df['DEFAULT']    #y= default['N_DEFAULT'].values
    optb= OptimalBinning(name='y', dtype=dtype,monotonic_trend='auto',
                         max_n_prebins=bins,prebinning_method= method,
                        )
    #method : 'uniform', 'quantile', 'cart', 'mdlp'
    optb.fit(x,y)
    binning_table= optb.binning_table
    return binning_table.build()
```

In [187]:
```
woe_iv('DURATION','numerical',user_splits=[11,15,24,30])
```

Out[187]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	(-inf, 11.00)	164	0.169072	140	24	0.146341	0.876701	0.104309	0.012636
1	[11.00, 15.00)	196	0.202062	146	50	0.255102	0.184696	0.006619	0.000826
2	[15.00, 24.00)	217	0.223711	154	63	0.290323	0.00693	0.000011	0.000001
3	[24.00, 30.00)	198	0.204124	137	61	0.308081	-0.07778	0.001255	0.000157
4	[30.00, inf)	195	0.201031	110	85	0.435897	-0.629058	0.088217	0.010849
5	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
6	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.200411	0.024470

In [188]:
```
woe('AMOUNT','numerical',9,'quantile')
```

Out[188]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	(-inf, 958.67)	108	0.111340	73	35	0.324074	-0.151776	0.002643	0.000330
1	[958.67, 1288.33)	108	0.111340	75	33	0.305556	-0.065907	0.000490	0.000061
2	[1288.33, 3181.00)	430	0.443299	324	106	0.246512	0.230417	0.022364	0.002789
3	[3181.00, 4042.00)	107	0.110309	85	22	0.205607	0.464721	0.021372	0.002648
4	[4042.00, 6337.33)	109	0.112371	69	40	0.366972	-0.34166	0.013976	0.001739
5	[6337.33, inf)	108	0.111340	61	47	0.435185	-0.626161	0.048393	0.005952
6	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
7	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.109238	0.013519

In [189]:
```
woe('INSTALL_RATE', 'categorical',4,'uniform')
```

Out[189]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[1]	128	0.131959	100	28	0.218750	0.386078	0.017999	0.002236
1	[2]	221	0.227835	163	58	0.262443	0.14642	0.004732	0.000591
2	[3]	153	0.157732	111	42	0.274510	0.084973	0.001118	0.000140
3	[4]	468	0.482474	313	155	0.331197	-0.184109	0.016956	0.002117
4	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
5	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.040806	0.005083

In [190]:
```
woe('AGE', 'numerical',7,'quantile')
```

Out[190]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	(-inf, 24.00)	102	0.105155	62	40	0.392157	-0.448632	0.022923	0.002842
1	[24.00, 27.00)	133	0.137113	83	50	0.375940	-0.38007	0.021232	0.002638
2	[27.00, 35.00)	303	0.312371	209	94	0.310231	-0.087848	0.002454	0.000307
3	[35.00, 40.00)	149	0.153608	121	28	0.187919	0.576699	0.044514	0.005488
4	[40.00, inf)	283	0.291753	212	71	0.250883	0.207019	0.011946	0.001491
5	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
6	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.103069	0.012765

In [191]:
```
woe('NUM_CREDITS','categorical',4,'uniform')
```

Out[191]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[3]	27	0.027835	21	6	0.222222	0.365876	0.003427	0.000426
1	[2]	329	0.339175	238	91	0.276596	0.074524	0.001854	0.000232
2	[1]	608	0.626804	424	184	0.302632	-0.05209	0.001719	0.000215
3	[4]	6	0.006186	4	2	0.333333	-0.19374	0.000241	0.000030
4	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
5	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.007241	0.000903

In [192]:
```
woe('NUM_DEPENDENTS', 'categorical',2, 'uniform')
```

Out[192]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[2]	152	0.156701	108	44	0.289474	0.011054	0.000019	2.387983e-06
1	[1]	818	0.843299	579	239	0.292176	-0.002048	0.000004	4.425226e-07
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000e+00
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000e+00
Totals		970	1.000000	687	283	0.291753		0.000023	2.830505e-06

In [193]:
```python
woe('NEW_CAR', 'categorical',2, 'uniform')
```

Out[193]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[0]	747	0.770103	547	200	0.267738	0.119244	0.010673	0.001333
1	[1]	223	0.229897	140	83	0.372197	-0.364086	0.032586	0.004051
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.043259	0.005384

In [194]:
```python
woe('USED_CAR', 'categorical',2, 'uniform')
```

Out[194]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[1]	99	0.102062	85	14	0.141414	0.916707	0.068071	0.008223
1	[0]	871	0.897938	602	269	0.308840	-0.081341	0.006040	0.000755
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.074111	0.008978

In [195]:
```python
woe('FURNITURE', 'categorical',2, 'uniform')
```

Out[195]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[0]	790	0.814433	565	225	0.284810	0.033838	0.000926	0.000116
1	[1]	180	0.185567	122	58	0.322222	-0.143309	0.003921	0.000490
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.004847	0.000605

In [196]:
```python
woe('RADIO_TV', 'categorical',2, 'uniform')
```

Out[196]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[1]	277	0.285567	216	61	0.220217	0.377517	0.037322	0.004638
1	[0]	693	0.714433	471	222	0.320346	-0.134707	0.013317	0.001663
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.050640	0.006301

In [197]:
```python
woe('EDUCATION', 'categorical',2, 'uniform')
```

Out[197]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[0]	921	0.949485	659	262	0.284473	0.035492	0.001187	0.000148
1	[1]	49	0.050515	28	21	0.428571	-0.599205	0.020042	0.002468
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.021229	0.002617

In [198]:
```
woe('RETRAINING', 'categorical',2, 'uniform')
```

Out[198]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[0]	879	0.906186	627	252	0.286689	0.02463	0.000547	0.000068
1	[1]	91	0.093814	60	31	0.340659	-0.22653	0.005030	0.000627
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.005577	0.000696

In [199]:
```
woe('MALE_DIV', 'categorical',2, 'uniform')
```

Out[199]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[0]	922	0.950515	658	264	0.286334	0.026368	0.000657	0.000082
1	[1]	48	0.049485	29	19	0.395833	-0.464031	0.011566	0.001433
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.012223	0.001515

In [200]:
```
woe('MALE_SINGLE', 'categorical',2, 'uniform')
```

Out[200]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[1]	527	0.543299	391	136	0.258065	0.169165	0.014984	0.001871
1	[0]	443	0.456701	296	147	0.331828	-0.186961	0.016560	0.002067
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.031544	0.003938

In [201]:
```
woe('MALE_MAR_or_WID', 'categorical',2, 'uniform')
```

Out[201]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[1]	91	0.093814	67	24	0.263736	0.139751	0.001778	0.000222
1	[0]	879	0.906186	620	259	0.294653	-0.013996	0.000178	0.000022
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.001956	0.000244

In [202]:
```
woe('GUARANTOR', 'categorical',2, 'uniform')
```

Out[202]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[0]	931	0.959794	666	265	0.284640	0.034672	0.001145	0.000143
1	[1]	39	0.040206	21	18	0.461538	-0.732737	0.024207	0.002960
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.025353	0.003103

In [203]:
```
woe('CO_APPLICANT', 'categorical',2, 'uniform')
```

Out[203]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[1]	52	0.053608	42	10	0.192308	0.548197	0.014143	0.001746
1	[0]	918	0.946392	645	273	0.297386	-0.027109	0.000699	0.000087
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.014843	0.001834

In [204]:
```
woe('REAL_ESTATE', 'categorical',2, 'uniform')
```

Out[204]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[1]	279	0.287629	219	60	0.215054	0.40784	0.043542	0.005405
1	[0]	691	0.712371	468	223	0.322721	-0.145591	0.015544	0.001941
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.059086	0.007347

In [205]:
```
woe('PROP_UNKN_NONE', 'categorical',2, 'uniform')
```

Out[205]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[0]	832	0.857732	604	228	0.274038	0.087341	0.006422	0.000803
1	[1]	138	0.142268	83	55	0.398551	-0.47538	0.034955	0.004329
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.041378	0.005131

In [206]:
```
woe('OTHER_INSTALL', 'categorical',2, 'uniform')
```

Out[206]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[0]	791	0.815464	580	211	0.266751	0.124283	0.012263	0.001532
1	[1]	179	0.184536	107	72	0.402235	-0.490725	0.048418	0.005992
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.060681	0.007524

In [207]:
```
woe('RENT', 'categorical',2, 'uniform')
```

Out[207]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[0]	792	0.816495	578	214	0.270202	0.10671	0.009087	0.001135
1	[1]	178	0.183505	109	69	0.387640	-0.429646	0.036587	0.004538
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.045674	0.005674

In [208]:
```
woe('OWN_RES', 'categorical',2, 'uniform')
```

Out[208]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[1]	696	0.717526	518	178	0.255747	0.181304	0.022668	0.002830
1	[0]	274	0.282474	169	105	0.383212	-0.410949	0.051380	0.006378
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.074048	0.009207

In [209]:
```
woe('TELEPHONE', 'categorical',2, 'uniform')
```

Out[209]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[1]	378	0.389691	281	97	0.256614	0.176756	0.011713	0.001462
1	[0]	592	0.610309	406	186	0.314189	-0.106281	0.007043	0.000880
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.018756	0.002342

In [210]:
```
woe('FOREIGN', 'categorical',2, 'uniform')
```

Out[210]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[1]	35	0.036082	32	3	0.085714	1.480236	0.053257	0.006109
1	[0]	935	0.963918	655	280	0.299465	-0.037042	0.001333	0.000167
2	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
3	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.054590	0.006276

In [211]:
```
woe('CHK_ACCT', 'categorical',4, 'uniform')
```

Out[211]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[3]	388	0.400000	343	45	0.115979	1.144181	0.389321	0.046173
1	[2]	62	0.063918	48	14	0.225806	0.345256	0.007043	0.000876
2	[1]	252	0.259794	161	91	0.361111	-0.316343	0.027586	0.003434
3	[0]	268	0.276289	135	133	0.496269	-0.871962	0.238445	0.028896
4	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
5	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.662394	0.079379

In [212]:
```
woe('HISTORY', 'categorical',5, 'uniform')
```

Out[212]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[4]	287	0.295876	238	49	0.170732	0.693563	0.120187	0.014729
1	[2]	515	0.530928	357	158	0.306796	-0.071747	0.002773	0.000347
2	[3]	86	0.088660	58	28	0.325581	-0.158649	0.002303	0.000288
3	[1]	46	0.047423	20	26	0.565217	-1.149252	0.072128	0.008550
4	[0]	36	0.037113	14	22	0.611111	-1.338873	0.076798	0.008941
5	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
6	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.274188	0.032855

In [213]:
```
woe('SAV_ACCT', 'categorical',5, 'uniform')
```

Out[213]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[3]	48	0.049485	42	6	0.125000	1.059023	0.042291	0.005052
1	[4]	175	0.180412	146	29	0.165714	0.729423	0.080269	0.009817
2	[2]	62	0.063918	51	11	0.177419	0.647043	0.022884	0.002812
3	[1]	100	0.103093	69	31	0.310000	-0.086768	0.000790	0.000099
4	[0]	585	0.603093	379	206	0.352137	-0.277227	0.048859	0.006088
5	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
6	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.195093	0.023868

In [214]:
```
woe('EMPLOYMENT', 'categorical',5, 'uniform')
```

Out[214]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[3]	169	0.174227	131	38	0.224852	0.350724	0.019784	0.002460
1	[4]	246	0.253608	188	58	0.235772	0.289112	0.019864	0.002474
2	[2]	330	0.340206	232	98	0.296970	-0.025118	0.000216	0.000027
3	[0]	55	0.056701	34	21	0.381818	-0.405049	0.010011	0.001243
4	[1]	170	0.175258	102	68	0.400000	-0.481422	0.044200	0.005472
5	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
6	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.094074	0.011677

In [215]:
```
woe('PRESENT_RESIDENT', 'categorical',4, 'uniform')
```

Out[215]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[0]	127	0.130928	92	35	0.275591	0.079553	8.146767e-04	1.018077e-04
1	[3]	398	0.410309	282	116	0.291457	0.001429	8.381878e-07	1.047735e-07
2	[2]	147	0.151546	104	43	0.292517	-0.003697	2.072464e-06	2.590579e-07
3	[1]	298	0.307216	209	89	0.298658	-0.03319	3.407362e-04	4.259007e-05
4	Special	0	0.000000	0	0	0.000000	0.0	0.000000e+00	0.000000e+00
5	Missing	0	0.000000	0	0	0.000000	0.0	0.000000e+00	0.000000e+00
Totals		970	1.000000	687	283	0.291753		1.158324e-03	1.447616e-04

In [216]:
```
woe('JOB', 'categorical',4, 'uniform')
```

Out[216]:

	Bin	Count	Count (%)	Non-event	Event	Event rate	WoE	IV	JS
0	[1]	198	0.204124	143	55	0.277778	0.068624	0.000947	0.000118
1	[2]	623	0.642268	443	180	0.288925	0.013726	0.000121	0.000015
2	[0]	20	0.020619	14	6	0.300000	-0.03959	0.000033	0.000004
3	[3]	129	0.132990	87	42	0.325581	-0.158649	0.003454	0.000431
4	Special	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
5	Missing	0	0.000000	0	0	0.000000	0.0	0.000000	0.000000
Totals		970	1.000000	687	283	0.291753		0.004555	0.000569

PERFORMANCE MODEL TO VARIABLE SELECTION

In [217]:
```
x=  df.drop(columns=['DEFAULT'])
y= df['DEFAULT']
```

In [218]:
```
def forward_selection(data, target, significance_level=0.05):
    initial_features = data.columns.tolist()
    best_features = []
    while (len(initial_features)>0):
        remaining_features = list(set(initial_features)-set(best_features))
        new_pval = pd.Series(index=remaining_features)
        for new_column in remaining_features:
            model = sm.OLS(target, sm.add_constant(data[best_features+[new_column]])).fit()
            new_pval[new_column] = model.pvalues[new_column]
        min_p_value = new_pval.min()
        if(min_p_value<significance_level):
            best_features.append(new_pval.idxmin())
        else:
            break
    return best_features
```

In [219]:
```
forward_selection(x,y)
```

Out[219]:
```
['CHK_ACCT',
 'DURATION',
 'HISTORY',
 'USED_CAR',
 'SAV_ACCT',
 'CO_APPLICANT',
 'NEW_CAR',
 'EDUCATION',
 'OTHER_INSTALL',
 'RENT',
 'INSTALL_RATE',
 'EMPLOYMENT',
 'FOREIGN']
```

In [220]:
```
X1= sm.add_constant(x[['CHK_ACCT',
 'DURATION',
 'HISTORY',
 'USED_CAR',
 'SAV_ACCT',
 'CO_APPLICANT',
 'NEW_CAR',
 'EDUCATION',
 'OTHER_INSTALL',
 'RENT',
 'INSTALL_RATE',
 'EMPLOYMENT',
 'FOREIGN']])
Y1=y

logit= sm.OLS(Y1,X1,hasconst=True).fit()
print(logit.summary(),logit.wald_test_terms())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                DEFAULT   R-squared:                       0.253
Model:                            OLS   Adj. R-squared:                  0.243
Method:                 Least Squares   F-statistic:                     24.89
Date:                Tue, 23 Apr 2024   Prob (F-statistic):           3.26e-52
Time:                        00:43:13   Log-Likelihood:                -470.25
No. Observations:                 970   AIC:                             968.5
Df Residuals:                     956   BIC:                             1037.
Df Model:                          13
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const             0.3967      0.060      6.640      0.000       0.279       0.514
CHK_ACCT         -0.0958      0.011     -8.951      0.000      -0.117      -0.075
DURATION          0.0077      0.001      6.668      0.000       0.005       0.010
HISTORY          -0.0566      0.012     -4.590      0.000      -0.081      -0.032
USED_CAR         -0.1138      0.044     -2.565      0.010      -0.201      -0.027
SAV_ACCT         -0.0315      0.008     -3.739      0.000      -0.048      -0.015
CO_APPLICANT     -0.1806      0.058     -3.138      0.002      -0.293      -0.068
NEW_CAR           0.1230      0.032      3.884      0.000       0.061       0.185
EDUCATION         0.1587      0.059      2.685      0.007       0.043       0.275
OTHER_INSTALL     0.0982      0.033      2.956      0.003       0.033       0.163
RENT              0.1011      0.034      3.012      0.003       0.035       0.167
INSTALL_RATE      0.0323      0.012      2.752      0.006       0.009       0.055
EMPLOYMENT       -0.0308      0.011     -2.823      0.005      -0.052      -0.009
FOREIGN          -0.1604      0.070     -2.284      0.023      -0.298      -0.023
==============================================================================
Omnibus:                       78.376   Durbin-Watson:                   2.014
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               65.178
Skew:                           0.552   Prob(JB):                     7.03e-15
Kurtosis:                       2.373   Cond. No.                         135.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
F                      P>F  df constraint  df denom
const          [[44.08327982120954]]  5.2690710703504416e-11             1     956.0
CHK_ACCT       [[80.12925183688586]]  1.7913091594333816e-18             1     956.0
DURATION       [[44.45845489737213]]  4.3872021706735486e-11             1     956.0
HISTORY       [[21.072177079729162]]   5.012724648220772e-06             1     956.0
USED_CAR       [[6.580825065504512]]     0.010459859570977179             1     956.0
SAV_ACCT      [[13.979407495053643]]  0.00019580313169354605             1     956.0
CO_APPLICANT   [[9.846105823600695]]    0.0017540337241176412             1     956.0
NEW_CAR       [[15.083800074987131]]  0.00010991782783800691             1     956.0
EDUCATION      [[7.208363338583791]]     0.007382059966988308             1     956.0
OTHER_INSTALL  [[8.737257081782381]]    0.0031942955624720505             1     956.0
RENT            [[9.07262880834418]]    0.0026629284612881272             1     956.0
INSTALL_RATE  [[7.5726951923450025]]    0.0060380267090670405             1     956.0
EMPLOYMENT     [[7.966620537516356]]     0.004863619379985443             1     956.0
FOREIGN        [[5.215918007238329]]     0.022599600696540875             1     956.0
```

In [221]:
```
def backward_elimination(data, target,significance_level = 0.1):
    features = data.columns.tolist()
    while(len(features)>0):
        p_values = sm.OLS(target, sm.add_constant(data[features])).fit().pvalues[1:]
        max_p_value = p_values.max()
        if(max_p_value >= significance_level):
            excluded_feature = p_values.idxmax()
            features.remove(excluded_feature)
        else:
            break
    return features
```

In [222]:
```
backward_elimination(x,y)
```

Out[222]:
```
['CHK_ACCT',
 'DURATION',
 'HISTORY',
 'NEW_CAR',
 'USED_CAR',
 'EDUCATION',
 'SAV_ACCT',
 'EMPLOYMENT',
 'INSTALL_RATE',
 'MALE_DIV',
 'GUARANTOR',
 'CO_APPLICANT',
 'REAL_ESTATE',
 'OTHER_INSTALL',
 'RENT',
 'NUM_CREDITS',
 'TELEPHONE',
 'FOREIGN']
```

In [223]:
```
X3= sm.add_constant(x[['CHK_ACCT',
 'DURATION',
 'HISTORY',
 'NEW_CAR',
 'USED_CAR',
 'EDUCATION',
 'SAV_ACCT',
 'EMPLOYMENT',
 'INSTALL_RATE',
 'MALE_DIV',
 'GUARANTOR',
 'CO_APPLICANT',
 'REAL_ESTATE',
 'OTHER_INSTALL',
 'RENT',
 'NUM_CREDITS',
 'TELEPHONE',
 'FOREIGN']])
Y3=y

logit= sm.OLS(Y3,X3,hasconst=True).fit()
print(logit.summary(),logit.wald_test_terms())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                DEFAULT   R-squared:                       0.266
Model:                            OLS   Adj. R-squared:                  0.252
Method:                 Least Squares   F-statistic:                     19.12
Date:                Tue, 23 Apr 2024   Prob (F-statistic):           3.00e-52
Time:                        00:44:11   Log-Likelihood:                -461.82
No. Observations:                 970   AIC:                             961.6
Df Residuals:                     951   BIC:                             1054.
Df Model:                          18
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.3706      0.064      5.785      0.000       0.245       0.496
CHK_ACCT        -0.0924      0.011     -8.654      0.000      -0.113      -0.071
DURATION         0.0074      0.001      6.282      0.000       0.005       0.010
HISTORY         -0.0670      0.014     -4.926      0.000      -0.094      -0.040
NEW_CAR          0.1243      0.032      3.945      0.000       0.062       0.186
USED_CAR        -0.1012      0.045     -2.265      0.024      -0.189      -0.014
EDUCATION        0.1653      0.059      2.789      0.005       0.049       0.282
SAV_ACCT        -0.0295      0.008     -3.510      0.000      -0.046      -0.013
EMPLOYMENT      -0.0325      0.011     -2.985      0.003      -0.054      -0.011
INSTALL_RATE     0.0352      0.012      3.003      0.003       0.012       0.058
MALE_DIV         0.1173      0.059      1.989      0.047       0.002       0.233
GUARANTOR        0.1349      0.065      2.070      0.039       0.007       0.263
CO_APPLICANT    -0.1556      0.058     -2.678      0.008      -0.270      -0.042
REAL_ESTATE     -0.0508      0.030     -1.711      0.087      -0.109       0.007
OTHER_INSTALL    0.0913      0.033      2.744      0.006       0.026       0.157
RENT             0.1000      0.033      2.990      0.003       0.034       0.166
NUM_CREDITS      0.0488      0.025      1.984      0.048       0.001       0.097
TELEPHONE       -0.0513      0.027     -1.912      0.056      -0.104       0.001
FOREIGN         -0.1694      0.070     -2.406      0.016      -0.308      -0.031
==============================================================================
Omnibus:                       70.252   Durbin-Watson:                   2.007
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               62.552
Skew:                           0.552   Prob(JB):                     2.61e-14
Kurtosis:                       2.426   Cond. No.                         138.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
F                      P>F  df constraint  df denom
const           [[33.46237724847455]]    9.85380139440441e-09             1     951.0
CHK_ACCT        [[74.89125596396225]]  2.0970989860144393e-17             1     951.0
DURATION       [[39.461082214882865]]   5.085705164726036e-10             1     951.0
HISTORY        [[24.267786251067236]]   9.883173639658822e-07             1     951.0
NEW_CAR        [[15.566175421614338]]    8.55254676705085e-05             1     951.0
USED_CAR        [[5.132355673337669]]     0.02370801820139074             1     951.0
EDUCATION      [[7.7808677122579155]]    0.005385730847091965             1     951.0
SAV_ACCT       [[12.322566378758545]]  0.00046851445428218626             1     951.0
EMPLOYMENT       [[8.90916241763326]]   0.0029100375504461295             1     951.0
INSTALL_RATE    [[9.016358256132957]]    0.002745730574881345             1     951.0
MALE_DIV       [[3.9564939402577908]]     0.04697700506267998             1     951.0
GUARANTOR       [[4.285350778783514]]    0.038711620750494995             1     951.0
CO_APPLICANT    [[7.170913988853459]]    0.007537220248318708             1     951.0
REAL_ESTATE    [[2.9277672227615223]]     0.08739392714927105             1     951.0
OTHER_INSTALL   [[7.530528605727383]]    0.006180427223236352             1     951.0
RENT            [[8.940547551939613]]   0.0028609211900220057             1     951.0
NUM_CREDITS    [[3.9377615917769226]]     0.04750062565963171             1     951.0
TELEPHONE       [[3.656785070669066]]    0.056141481537621496             1     951.0
FOREIGN          [[5.78765510181665]]    0.016329113262855056             1     951.0
```

In [224]:
```
def stepwise_selection(data, target,SL_in=0.05,SL_out =0.1):
    initial_features = data.columns.tolist()
    best_features = []
    while (len(initial_features)>0):
        remaining_features = list(set(initial_features)-set(best_features))
        new_pval = pd.Series(index=remaining_features)
        for new_column in remaining_features:
            model = sm.OLS(target, sm.add_constant(data[best_features+[new_column]])).fit()
            new_pval[new_column] = model.pvalues[new_column]
        min_p_value = new_pval.min()
        if(min_p_value<SL_in):
            best_features.append(new_pval.idxmin())
            while(len(best_features)>0):
                best_features_with_constant = sm.add_constant(data[best_features])
                p_values = sm.OLS(target, best_features_with_constant).fit().pvalues[1:]
                max_p_value = p_values.max()
                if(max_p_value >= SL_out):
                    excluded_feature = p_values.idxmax()
                    best_features.remove(excluded_feature)
                else:
                    break
        else:
            break
    return best_features
```

In [225]:
```
stepwise_selection(x,y)
```

Out[225]:
```
['CHK_ACCT',
 'DURATION',
 'HISTORY',
 'USED_CAR',
 'SAV_ACCT',
 'CO_APPLICANT',
 'NEW_CAR',
 'EDUCATION',
 'OTHER_INSTALL',
 'RENT',
 'INSTALL_RATE',
 'EMPLOYMENT',
 'FOREIGN']
```

In [226]:
```
X4= sm.add_constant(x[['CHK_ACCT',
 'DURATION',
 'HISTORY',
 'USED_CAR',
 'SAV_ACCT',
 'CO_APPLICANT',
 'NEW_CAR',
 'EDUCATION',
 'OTHER_INSTALL',
 'RENT',
 'INSTALL_RATE',
 'EMPLOYMENT',
 'FOREIGN']])
Y4= y

logit= sm.OLS(Y4,X4).fit()
print(logit.summary(),logit.wald_test_terms())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                DEFAULT   R-squared:                       0.253
Model:                            OLS   Adj. R-squared:                  0.243
Method:                 Least Squares   F-statistic:                     24.89
Date:                Tue, 23 Apr 2024   Prob (F-statistic):           3.26e-52
Time:                        00:44:26   Log-Likelihood:                -470.25
No. Observations:                 970   AIC:                             968.5
Df Residuals:                     956   BIC:                             1037.
Df Model:                          13
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const             0.3967      0.060      6.640      0.000       0.279       0.514
CHK_ACCT         -0.0958      0.011     -8.951      0.000      -0.117      -0.075
DURATION          0.0077      0.001      6.668      0.000       0.005       0.010
HISTORY          -0.0566      0.012     -4.590      0.000      -0.081      -0.032
USED_CAR         -0.1138      0.044     -2.565      0.010      -0.201      -0.027
SAV_ACCT         -0.0315      0.008     -3.739      0.000      -0.048      -0.015
CO_APPLICANT     -0.1806      0.058     -3.138      0.002      -0.293      -0.068
NEW_CAR           0.1230      0.032      3.884      0.000       0.061       0.185
EDUCATION         0.1587      0.059      2.685      0.007       0.043       0.275
OTHER_INSTALL     0.0982      0.033      2.956      0.003       0.033       0.163
RENT              0.1011      0.034      3.012      0.003       0.035       0.167
INSTALL_RATE      0.0323      0.012      2.752      0.006       0.009       0.055
EMPLOYMENT       -0.0308      0.011     -2.823      0.005      -0.052      -0.009
FOREIGN          -0.1604      0.070     -2.284      0.023      -0.298      -0.023
==============================================================================
Omnibus:                       78.376   Durbin-Watson:                   2.014
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               65.178
Skew:                           0.552   Prob(JB):                     7.03e-15
Kurtosis:                       2.373   Cond. No.                         135.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
F                      P>F  df constraint  df denom
const           [[44.08327982120954]]  5.2690710703504416e-11              1     956.0
CHK_ACCT        [[80.12925183688586]]  1.7913091594333816e-18              1     956.0
DURATION        [[44.45845489737213]]  4.3872021706735486e-11              1     956.0
HISTORY        [[21.072177079729162]]   5.012724648220772e-06              1     956.0
USED_CAR        [[6.580825065504512]]    0.010459859570977179              1     956.0
SAV_ACCT       [[13.979407495053643]]  0.00019580313169354605              1     956.0
CO_APPLICANT    [[9.846105823600695]]   0.0017540337241176412              1     956.0
NEW_CAR        [[15.083800074987131]]  0.00010991782783800691              1     956.0
EDUCATION       [[7.208363338583791]]    0.007382059966988308              1     956.0
OTHER_INSTALL   [[8.737257081782381]]   0.0031942955624720505              1     956.0
RENT             [[9.07262880834418]]   0.0026629284612881272              1     956.0
INSTALL_RATE   [[7.5726951923450025]]   0.0060380267090670405              1     956.0
EMPLOYMENT      [[7.966620537516356]]    0.004863619379985443              1     956.0
FOREIGN         [[5.215918007238329]]    0.022599600696540875              1     956.0
```

Se crearon modelos para medir el rendimiento de las variables y seleccionar las mejores variables, posteriormente se analizarán los modelos

In [227]:
```
logistic= LogisticRegression(penalty= 'none',random_state=0)
logistic.fit(x,y)

print(x.columns,logistic.coef_)

kf= StratifiedKFold(n_splits=5)
scores= cross_val_score(LogisticRegression(penalty='none',random_state=0),x,y,cv=kf)
print(scores)
print(scores.mean())
```

```
Index(['CHK_ACCT', 'DURATION', 'HISTORY', 'NEW_CAR', 'USED_CAR', 'FURNITURE',
       'RADIO_TV', 'EDUCATION', 'RETRAINING', 'AMOUNT', 'SAV_ACCT',
       'EMPLOYMENT', 'INSTALL_RATE', 'MALE_DIV', 'MALE_SINGLE',
       'MALE_MAR_or_WID', 'GUARANTOR', 'CO_APPLICANT', 'PRESENT_RESIDENT',
       'REAL_ESTATE', 'PROP_UNKN_NONE', 'AGE', 'OTHER_INSTALL', 'RENT',
       'OWN_RES', 'NUM_CREDITS', 'JOB', 'NUM_DEPENDENTS', 'TELEPHONE',
       'FOREIGN'],
      dtype='object') [[-5.77265038e-01  3.32258217e-02 -4.96635821e-01  3.95964787e-01
  -2.91270997e-01 -3.02588158e-02 -2.40322606e-01  1.47715080e-01
   2.23715226e-02  6.02098442e-05 -1.21503149e-01 -2.75763880e-01
   3.48059215e-01  1.01842403e-01 -2.78464348e-01 -4.63175454e-02
   1.05525589e-01 -1.90685394e-01 -9.87515562e-03 -1.96233468e-01
   1.09083826e-01 -5.05028450e-03  3.11325414e-01  1.71149240e-01
  -1.80967919e-01  1.88816663e-01  4.64149351e-02  2.88983117e-02
  -1.75068988e-01 -1.18068912e-01]]
[0.76804124 0.7628866  0.78865979 0.75257732 0.77835052]
0.7701030927835051
```

In [230]:
```
rf = RandomForestClassifier(random_state=0)
rf.fit(x,y)

print(' ')

print(x.columns, rf.feature_importances_)

param_grid= {'n_estimators': [50,100,150, 200, 300],
             'max_depth': [1, 5, 10, 20]}

grid_search = GridSearchCV(RandomForestClassifier(random_state=0),param_grid, cv=5, scoring='a
grid_search.fit(x, y)

print(' ')

print(grid_search.best_params_)

rf1 = RandomForestClassifier(random_state=0, max_depth=10, n_estimators= 150)
rf1.fit(x,y)

print(' ')

for i, importance in enumerate(rf1.feature_importances_):
    print(f"Feature {i}: {importance}")

print(' ')

scores= cross_val_score(RandomForestClassifier(random_state=0, max_depth= 10, n_estimators= 15
print(scores)
print(scores.mean())
```

```
Index(['CHK_ACCT', 'DURATION', 'HISTORY', 'NEW_CAR', 'USED_CAR', 'FURNITURE',
       'RADIO_TV', 'EDUCATION', 'RETRAINING', 'AMOUNT', 'SAV_ACCT',
       'EMPLOYMENT', 'INSTALL_RATE', 'MALE_DIV', 'MALE_SINGLE',
       'MALE_MAR_or_WID', 'GUARANTOR', 'CO_APPLICANT', 'PRESENT_RESIDENT',
       'REAL_ESTATE', 'PROP_UNKN_NONE', 'AGE', 'OTHER_INSTALL', 'RENT',
       'OWN_RES', 'NUM_CREDITS', 'JOB', 'NUM_DEPENDENTS', 'TELEPHONE',
       'FOREIGN'],
      dtype='object') [0.10905052 0.09856478 0.06362732 0.02381895 0.0099288  0.01529811
 0.01559132 0.00877962 0.01065373 0.11994951 0.04529798 0.05150524
 0.04446937 0.00962088 0.01939761 0.01139307 0.01066511 0.00981126
 0.04085623 0.01896198 0.01326075 0.1008909  0.02300582 0.01429155
 0.01709138 0.02287261 0.03310841 0.01585528 0.01755988 0.00482204]

{'max_depth': 10, 'n_estimators': 150}

Feature 0: 0.12907408195959214
Feature 1: 0.10337586925561698
Feature 2: 0.06563126592007795
Feature 3: 0.020932519760713874
Feature 4: 0.01339411477163134
Feature 5: 0.012163340455143216
Feature 6: 0.013514535745251773
Feature 7: 0.010865358958459637
Feature 8: 0.010589421701536554
Feature 9: 0.11561966487761619
Feature 10: 0.05042225519964994
Feature 11: 0.04782700974455181
Feature 12: 0.037453159386138586
Feature 13: 0.008805109034590876
Feature 14: 0.0173733906517237
Feature 15: 0.008400099415900072
Feature 16: 0.010507054998127126
Feature 17: 0.011131029695031
Feature 18: 0.036549432652998405
Feature 19: 0.018716110991136556
Feature 20: 0.014335272552621486
Feature 21: 0.09704026686811038
Feature 22: 0.02314099472602032
Feature 23: 0.014631099419894943
Feature 24: 0.01641470646379743
Feature 25: 0.022506707798453637
Feature 26: 0.03249429977513278
Feature 27: 0.013640957053634679
Feature 28: 0.017997925262150497
Feature 29: 0.005452944904696196

[0.77319588 0.74742268 0.74742268 0.78865979 0.77835052]
0.7670103092783505
```

In [ ]:
```
xgb = XGBClassifier(enable_categorical=True,random_state=0)
xgb.fit(x, y)
print(' ')
print(x.columns,xgb.feature_importances_)

param_grid = {
    'max_depth': [3, 5, 7],
    'learning_rate': [0.1, 0.01, 0.001],
    'n_estimators': [20,50,100, 200],
    'gamma': [0, 0.1, 0.2],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'reg_alpha': [0, 0.1, 0.5],
    'reg_lambda': [0, 0.1, 0.5]
}

grid_search = GridSearchCV(xgb, param_grid, cv=5, scoring='accuracy')
grid_search.fit(x, y)

print(' ')
print(grid_search.best_params_)
```

In [ ]:
```
scores= cross_val_score(XGBClassifier(enable_categorical=True,random_state=0),x_train,y_train,
print(scores)
print(scores.mean())
```

In [ ]:
```
gbc= GradientBoostingClassifier(random_state=0)
gbc.fit(x,y)

print(x.columns,gbc.feature_importances_)

param_grid = {
    'n_estimators': [10,20,50,100],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Ejecutar GridSearchCV
grid_search = GridSearchCV(gbc, param_grid, cv=5, scoring='accuracy')
grid_search.fit(x, y)
print(grid_search.best_params_)
```

In [ ]:
```
scores= cross_val_score(GradientBoostingClassifier(random_state=0),x_train,y_train,cv=kf)
print(scores)
print(scores.mean())
```

In [233]:
```
dt= DecisionTreeClassifier(random_state=0)
dt.fit(x,y)
print(' ')
print(x.columns,dt.feature_importances_)

param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None,1,2, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Ejecutar GridSearchCV
grid_search = GridSearchCV(dt, param_grid, cv=5, scoring='accuracy')
grid_search.fit(x, y)
print(' ')
print(grid_search.best_params_)

dt1= DecisionTreeClassifier(criterion='entropy', max_depth=5, min_samples_leaf= 1, min_samples_
dt1.fit(x,y)
print(' ')
for i, importance in enumerate(dt1.feature_importances_):
    print(f"Feature {i}: {importance}")
```

```
Index(['CHK_ACCT', 'DURATION', 'HISTORY', 'NEW_CAR', 'USED_CAR', 'FURNITURE',
       'RADIO_TV', 'EDUCATION', 'RETRAINING', 'AMOUNT', 'SAV_ACCT',
       'EMPLOYMENT', 'INSTALL_RATE', 'MALE_DIV', 'MALE_SINGLE',
       'MALE_MAR_or_WID', 'GUARANTOR', 'CO_APPLICANT', 'PRESENT_RESIDENT',
       'REAL_ESTATE', 'PROP_UNKN_NONE', 'AGE', 'OTHER_INSTALL', 'RENT',
       'OWN_RES', 'NUM_CREDITS', 'JOB', 'NUM_DEPENDENTS', 'TELEPHONE',
       'FOREIGN'],
      dtype='object') [0.14594134 0.11318168 0.03597795 0.01505445 0.01602915 0.00901523
 0.02218673 0.00916803 0.00415764 0.1278756  0.04573627 0.04516139
 0.02153043 0.00447442 0.0031643  0.00666055 0.02063812 0.0206771
 0.02612084 0.02970806 0.01026344 0.14769313 0.01181658 0.00374188
 0.0175809  0.03083017 0.03662927 0.00396475 0.01502059 0.        ]

{'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 2}

Feature 0: 0.3308167722709448
Feature 1: 0.12638456324550915
Feature 2: 0.09120018812405575
Feature 3: 0.024949101899064713
Feature 4: 0.058228348293558496
Feature 5: 0.0
Feature 6: 0.0
Feature 7: 0.0
Feature 8: 0.01918157999606331
Feature 9: 0.05764707170434061
Feature 10: 0.03619114787144967
Feature 11: 0.01101947346865786
Feature 12: 0.0
Feature 13: 0.0
Feature 14: 0.0
Feature 15: 0.0
Feature 16: 0.026251032750079808
Feature 17: 0.0
Feature 18: 0.034230839722372085
Feature 19: 0.02080133697204199
Feature 20: 0.0
Feature 21: 0.11508015330783312
Feature 22: 0.04801839037402869
Feature 23: 0.0
Feature 24: 0.0
Feature 25: 0.0
Feature 26: 0.0
Feature 27: 0.0
Feature 28: 0.0
Feature 29: 0.0
```

In [ ]:
```
dt= DecisionTreeClassifier(criterion='gini', max_depth= 5, min_samples_leaf= 1, min_samples_sp
dt.fit(x_train,y_train)
y_pred= dt.predict(x_test)

print(accuracy_score(y_test,y_pred))
print(precision_score(y_test,y_pred))
print(roc_auc_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(x_train.columns,dt.feature_importances_)
```

In [ ]:
```
scores= cross_val_score(DecisionTreeClassifier(criterion='entropy', max_depth= 5, min_samples_
print(scores)
print(scores.mean())
```

In [241]:
```
param_grid = {
    'alpha': [0.0001, 0.001, 0.01,1.0,0.05,0.5],
    'l1_ratio': [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
}
```

In [242]:
```
lasso_model= SGDClassifier(loss='log',penalty='l1',shuffle=False,random_state=0)
lasso_model.fit(x,y)

print(x.columns,lasso_model.coef_)

grid_search = GridSearchCV(lasso_model, param_grid, cv=5, scoring='accuracy')
grid_search.fit(x, y)
print(' ')
print(grid_search.best_params_)

lasso_model1= SGDClassifier(loss='log',penalty='l1',shuffle=False,random_state=0,alpha=0.05,l1_
lasso_model1.fit(x,y)

for i, importance in enumerate(lasso_model1.coef_):
    print(f"Feature {i}: {importance}")
```

```
Index(['CHK_ACCT', 'DURATION', 'HISTORY', 'NEW_CAR', 'USED_CAR', 'FURNITURE',
       'RADIO_TV', 'EDUCATION', 'RETRAINING', 'AMOUNT', 'SAV_ACCT',
       'EMPLOYMENT', 'INSTALL_RATE', 'MALE_DIV', 'MALE_SINGLE',
       'MALE_MAR_or_WID', 'GUARANTOR', 'CO_APPLICANT', 'PRESENT_RESIDENT',
       'REAL_ESTATE', 'PROP_UNKN_NONE', 'AGE', 'OTHER_INSTALL', 'RENT',
       'OWN_RES', 'NUM_CREDITS', 'JOB', 'NUM_DEPENDENTS', 'TELEPHONE',
       'FOREIGN'],
      dtype='object') [[-1.06481466e+04  2.30136570e+04 -4.77846592e+03  1.02731644e+03
  -1.28730841e+03  1.97020421e+02 -8.82243003e+02  4.93636190e+02
   3.08634136e+02 -1.86464367e+02 -7.14888951e+03 -3.54505394e+03
   2.39266800e+03  2.10016860e+02 -1.08064427e+03  0.00000000e+00
   4.54511555e+02 -1.99721608e+02 -1.72345758e+02 -4.06070176e+02
   5.61582050e+02 -1.38014825e+04  1.03030731e+03  8.45278822e+02
  -8.83421283e+02  2.02358882e+01 -4.03524763e+02  0.00000000e+00
  -8.05110213e+02 -3.13610800e+02]]

{'alpha': 0.05, 'l1_ratio': 0.0}
Feature 0: [-27.91831689 133.70511059  -5.94174162   0.           0.
   0.           0.           0.           0.          -1.21306591
 -23.07071315  -9.15503383   3.63605649   0.           0.
   0.           0.           0.           0.           0.
   0.         -88.68992929   0.           0.           0.
   0.           0.           0.           0.           0.        ]
```

In [ ]:
```
scores= cross_val_score(SGDClassifier(loss='log',penalty='l1',shuffle=False,random_state=0),x_
print(scores)
print(scores.mean())
```

In [243]:
```
ridge_model= SGDClassifier(loss='log',penalty='l2',random_state=0)
ridge_model.fit(x,y)

print(x.columns,ridge_model.coef_)

grid_search = GridSearchCV(ridge_model, param_grid, cv=5, scoring='accuracy')
grid_search.fit(x, y)
print(' ')
print(grid_search.best_params_)

ridge_model1= SGDClassifier(loss='log',penalty='l2',shuffle=False,random_state=0,alpha=0.5,l1_
ridge_model1.fit(x,y)

print(x.columns,ridge_model1.coef_)
```

```
Index(['CHK_ACCT', 'DURATION', 'HISTORY', 'NEW_CAR', 'USED_CAR', 'FURNITURE',
       'RADIO_TV', 'EDUCATION', 'RETRAINING', 'AMOUNT', 'SAV_ACCT',
       'EMPLOYMENT', 'INSTALL_RATE', 'MALE_DIV', 'MALE_SINGLE',
       'MALE_MAR_or_WID', 'GUARANTOR', 'CO_APPLICANT', 'PRESENT_RESIDENT',
       'REAL_ESTATE', 'PROP_UNKN_NONE', 'AGE', 'OTHER_INSTALL', 'RENT',
       'OWN_RES', 'NUM_CREDITS', 'JOB', 'NUM_DEPENDENTS', 'TELEPHONE',
       'FOREIGN'],
      dtype='object') [[-2103.77987434  4840.29980404 -1309.42039233   176.16517505
   -189.49877775    12.52550557  -220.4084931     70.30445059
     10.9093113    116.26497505 -1413.56391038  -951.33235015
    163.63966949    41.81902665  -262.42954403   -31.31376391
     61.61740641   -56.9708479   -189.09472919  -218.28723813
    120.60849714 -7426.31164266   165.65991232   130.70971131
   -328.18844825  -238.48966646  -225.56011233  -149.90201822
   -193.13521485   -69.69837775]]

{'alpha': 0.5, 'l1_ratio': 0.0}
Index(['CHK_ACCT', 'DURATION', 'HISTORY', 'NEW_CAR', 'USED_CAR', 'FURNITURE',
       'RADIO_TV', 'EDUCATION', 'RETRAINING', 'AMOUNT', 'SAV_ACCT',
       'EMPLOYMENT', 'INSTALL_RATE', 'MALE_DIV', 'MALE_SINGLE',
       'MALE_MAR_or_WID', 'GUARANTOR', 'CO_APPLICANT', 'PRESENT_RESIDENT',
       'REAL_ESTATE', 'PROP_UNKN_NONE', 'AGE', 'OTHER_INSTALL', 'RENT',
       'OWN_RES', 'NUM_CREDITS', 'JOB', 'NUM_DEPENDENTS', 'TELEPHONE',
       'FOREIGN'],
      dtype='object') [[-0.46786578  0.89001446 -0.25210581  0.03570372 -0.05171711  0.006104
07
  -0.04378276  0.01834086  0.01064518 -0.10211618 -0.33003047 -0.18936819
   0.02553935  0.00775147 -0.0531075  -0.00199662  0.01600843 -0.00921198
  -0.04703448 -0.02907501  0.02004982 -1.4537172   0.03036131  0.0300779
  -0.05124824 -0.02788357 -0.06236159 -0.01722841 -0.04418297 -0.01346719]]
```

In [ ]:
```
scores= cross_val_score(SGDClassifier(loss='log',penalty='l2',shuffle=False,random_state=0),x_
print(scores)
print(scores.mean())
```

In [244]:
```
elasnet_model= SGDClassifier(loss='log',penalty='elasticnet',shuffle=False,random_state=0)
elasnet_model.fit(x,y)

print(x.columns,elasnet_model.coef_)

grid_search = GridSearchCV(elasnet_model, param_grid, cv=5, scoring='accuracy')
grid_search.fit(x, y)
print(' ')
print(grid_search.best_params_)

elasnet_model1= SGDClassifier(loss='log',penalty='elasticnet',shuffle=False,random_state=0,alp
elasnet_model1.fit(x,y)

print(x.columns,elasnet_model1.coef_)
```

```
Index(['CHK_ACCT', 'DURATION', 'HISTORY', 'NEW_CAR', 'USED_CAR', 'FURNITURE',
       'RADIO_TV', 'EDUCATION', 'RETRAINING', 'AMOUNT', 'SAV_ACCT',
       'EMPLOYMENT', 'INSTALL_RATE', 'MALE_DIV', 'MALE_SINGLE',
       'MALE_MAR_or_WID', 'GUARANTOR', 'CO_APPLICANT', 'PRESENT_RESIDENT',
       'REAL_ESTATE', 'PROP_UNKN_NONE', 'AGE', 'OTHER_INSTALL', 'RENT',
       'OWN_RES', 'NUM_CREDITS', 'JOB', 'NUM_DEPENDENTS', 'TELEPHONE',
       'FOREIGN'],
      dtype='object') [[-2683.1013486   5696.82374659 -1403.03909444   223.5692671
   -293.21122136    44.34056964  -256.16130133   106.72713251
     64.29985468  -539.25253227 -1907.20576123 -1026.90632618
    298.58489614    36.02713992  -285.28925199    -8.4091611
     89.89351786   -55.36738243  -180.21066208  -164.46475471
    117.28171661 -7662.21755489   184.24072673   183.67232229
   -287.82915339  -126.93174274  -306.17675034   -72.31815213
   -240.78200218   -77.39009442]]

{'alpha': 0.001, 'l1_ratio': 0.0}
Index(['CHK_ACCT', 'DURATION', 'HISTORY', 'NEW_CAR', 'USED_CAR', 'FURNITURE',
       'RADIO_TV', 'EDUCATION', 'RETRAINING', 'AMOUNT', 'SAV_ACCT',
       'EMPLOYMENT', 'INSTALL_RATE', 'MALE_DIV', 'MALE_SINGLE',
       'MALE_MAR_or_WID', 'GUARANTOR', 'CO_APPLICANT', 'PRESENT_RESIDENT',
       'REAL_ESTATE', 'PROP_UNKN_NONE', 'AGE', 'OTHER_INSTALL', 'RENT',
       'OWN_RES', 'NUM_CREDITS', 'JOB', 'NUM_DEPENDENTS', 'TELEPHONE',
       'FOREIGN'],
      dtype='object') [[-233.55011213  465.07667114 -131.85389019   17.44123549   -26.12869499
     3.10023168   -22.50615701     8.86982983     6.23374385    -5.86870794
  -165.24413285 -101.19907457     8.83647966     3.43187809   -28.27569118
    -1.19369673     7.56004155    -4.7747869    -26.55163396   -15.85793513
     9.40036171 -858.54347318    15.56776853    14.75538587   -26.6343959
   -18.17086532   -30.5388749    -11.82927266   -22.30720756    -6.76428144]]
```

In [ ]:
```
scores= cross_val_score(SGDClassifier(loss='log',penalty='elasticnet',random_state=0),x_train,
print(scores)
print(scores.mean())
```

AJUSTAR PARAMETROS PARA LOGISTIC REGRESSION

In [245]:
```
param_grid = {
    'penalty': ['l1', 'l2'],
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'solver': ['liblinear', 'saga','elasticnet']
}

grid_search = GridSearchCV(estimator=LogisticRegression(random_state=0,penalty='none'), param_g
grid_search.fit(x, y)


print("Mejores parámetros:", grid_search.best_params_)

print("Mejor puntuación (exactitud):", grid_search.best_score_)
```

```
Mejores parámetros: {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
Mejor puntuación (exactitud): 0.7721649484536082
```

In [246]:
```
final_model= df[['DURATION','OTHER_INSTALL','USED_CAR','MALE_SINGLE',
                 'GUARANTOR','OWN_RES','CHK_ACCT','SAV_ACCT','EMPLOYMENT',
                 'FOREIGN']]
```

In [247]:
```
x_train,x_test,y_train,y_test= train_test_split(final_model,y,train_size=.7,random_state=0)
```

In [250]:
```
x_train
```

Out[250]:

	DURATION	OTHER_INSTALL	USED_CAR	MALE_SINGLE	GUARANTOR	OWN_RES	CHK_ACCT	SAV_ACCT	EI
285	48	1	0	0	0	1	3	4	
71	42	1	0	0	0	1	1	0	
49	24	1	0	1	0	1	1	4	
491	15	0	1	1	0	1	2	2	
840	18	1	0	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	
835	18	0	0	0	0	1	3	0	
192	24	1	1	1	0	0	1	4	
629	12	0	0	1	0	0	0	0	
559	12	0	0	0	0	1	1	0	
684	15	0	0	0	0	0	3	2	

679 rows × 10 columns

In [253]:
```
logistic= LogisticRegression(C= 0.1, penalty= 'l2', solver= 'liblinear',random_state=0)
logistic.fit(x_train,y_train)
y_pred= logistic.predict(x_test)

print(accuracy_score(y_test,y_pred))
print(precision_score(y_test,y_pred))
print(roc_auc_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(final_model.columns,logistic.coef_)
```

```
0.7731958762886598
0.6226415094339622
0.6616909481729162
              precision    recall  f1-score   support

           0       0.81      0.91      0.85       212
           1       0.62      0.42      0.50        79

    accuracy                           0.77       291
   macro avg       0.71      0.66      0.68       291
weighted avg       0.76      0.77      0.76       291

Index(['DURATION', 'OTHER_INSTALL', 'USED_CAR', 'MALE_SINGLE', 'GUARANTOR',
       'OWN_RES', 'CHK_ACCT', 'SAV_ACCT', 'EMPLOYMENT', 'FOREIGN'],
      dtype='object') [[ 0.04100033  0.44206029 -0.54050599 -0.19690607  0.07563142 -0.369565
36
  -0.54077715 -0.17462533 -0.15461502 -0.35507341]]
```

Transform data to simplify

In [ ]:
```
fin_model['DURATION_11']= np.where(fin_model['DURATION']<=11,1,0)

fin_model['DURATION_11_15']= np.where((fin_model['DURATION']>11) & (fin_model['DURATION']<15),

#fin_model['DURATION_6_8']= np.where((fin_model['DURATION']>15)&(fin_model['DURATION']<24),1,0

fin_model['DURATION_15_30']= np.where((fin_model['DURATION']>15)&(fin_model['DURATION']<30),1,

fin_model['DURATION_30']= np.where(fin_model['DURATION']>30,1,0)


fin_model2['DURATION_11']= np.where(fin_model2['DURATION']<=11,1,0)

fin_model2['DURATION_11_15']= np.where((fin_model2['DURATION']>11) & (fin_model2['DURATION']<1

#fin_model2['DURATION_6_8']= np.where((fin_model2['DURATION']>15)&(fin_model2['DURATION']<24),

fin_model2['DURATION_15_30']= np.where((fin_model2['DURATION']>15)&(fin_model2['DURATION']<30)

fin_model2['DURATION_30']= np.where(fin_model2['DURATION']>30,1,0)
```

In [ ]:
```
fin_model= fin_model.drop(columns='DURATION')
fin_model2= fin_model2.drop(columns='DURATION')
```

In [ ]:
```
x_final= sm.add_constant(fin_model['CHK_ACCT_0'])
y_final= y_train
logit= sm.OLS(y_final,x_final).fit()
print(logit.summary2(),logit.wald_test_terms(), np.exp(logit.params))
```