**Importing the pandas package and loading the dataset.**

# Working on the Accepted Data

```
import pandas as pd
```

```
accepted=pd.read_excel('/content/app+beh_approved.xlsx', sheet_name='app_apprv')
```

```
accepted.head()
```

| | RK | TAX_CODE | AMT | ACC_AMT | ANNUAL_INCOME_AMT | EDUCATION | EMP_YR_CNT | GENDER | MARITAL_STATUS |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2 | 4 | 48424.0 | 36318.00 | 42213.0 | GRA | 10 | FEM | MAR |
| **1** | 4 | 2 | 71888.0 | 53916.00 | 58281.0 | PGR | 13 | FEM | MAR |
| **2** | 5 | 1 | 28060.0 | 21045.00 | 23932.0 | UGR | 15 | FEM | MAR |
| **3** | 9 | 4 | NaN | NaN | NaN | GRA | 6 | FEM | SIN |
| **4** | 12 | 5 | 59059.0 | 44294.25 | 44628.0 | GRA | 6 | FEM | MAR |

5 rows × 40 columns

```
accepted.TGT_VAR.value_counts()
#1 - def 0 - no def
```

```
0    4475
1     777
Name: TGT_VAR, dtype: int64
```

```
accepted.describe()
```

| | RK | TAX_CODE | AMT | ACC_AMT | ANNUAL_INCOME_AMT | EMP_YR_CNT | ACC_1 |
|---|---|---|---|---|---|---|---|
| **count** | 5252.000000 | 5252.000000 | 5177.000000 | 5177.000000 | 5177.000000 | 5252.000000 | |
| **mean** | 7043.591394 | 4.002094 | 68732.680317 | 51549.510238 | 58232.644968 | 8.033701 | |
| **std** | 4072.219037 | 1.771082 | 37123.554990 | 27842.666243 | 31923.116210 | 4.071374 | |
| **min** | 2.000000 | 1.000000 | 6341.000000 | 4755.750000 | 5540.000000 | 1.000000 | |
| **25%** | 3546.500000 | 3.000000 | 44896.000000 | 33672.000000 | 37716.000000 | 4.000000 | |
| **50%** | 6894.500000 | 4.000000 | 60699.000000 | 45524.250000 | 51300.000000 | 8.000000 | |
| **75%** | 10719.000000 | 5.000000 | 81315.000000 | 60986.250000 | 69100.000000 | 12.000000 | |
| **max** | 13997.000000 | 7.000000 | 586691.000000 | 440018.250000 | 465907.000000 | 15.000000 | |

8 rows × 32 columns

```
accepted.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5252 entries, 0 to 5251
Data columns (total 40 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   RK                              5252 non-null   int64
 1   TAX_CODE                        5252 non-null   int64
 2   AMT                             5177 non-null   float64
 3   ACC_AMT                         5177 non-null   float64
 4   ANNUAL_INCOME_AMT               5177 non-null   float64
 5   EDUCATION                       5247 non-null   object
 6   EMP_YR_CNT                      5252 non-null   int64
```

```
 7   GENDER                          5238 non-null   object
 8   MARITAL_STATUS                  5246 non-null   object
 9   RESIDENCE                       5249 non-null   object
 10  EMPLOYMENT_STATUS_CD            5252 non-null   object
 11  ACC_1_30DLQ_LST_3M_CNT          5252 non-null   int64
 12  ACC_31_60DLQ_LST_3M_CNT         5252 non-null   int64
 13  ACC_61_90DLQ_LST_3M_CNT         5252 non-null   int64
 14  ACC_91_120DLQ_LST_3M_CNT        5252 non-null   int64
 15  ACC_DLD_PAY_LST_3M_CNT          5252 non-null   int64
 16  ACC_1_30DLQ_LST_3M_AMT          5252 non-null   int64
 17  TOT_OUTSTANDING_31_60_DAY_AMT   5252 non-null   int64
 18  ACC_61_90DLQ_LST_3M_AMT         5252 non-null   int64
 19  OUTCOME_CD                      5252 non-null   int64
 20  SLN_DR_TRNS_LST_3M_CNT          5252 non-null   int64
 21  ACC_91_120DLQ_LST_3M_AMT        5252 non-null   int64
 22  APPL_SCR_NO                     5252 non-null   int64
 23  ACC_APPL_PCL_VAL_AMT            5252 non-null   int64
 24  APPL_PCL_TYP_CD                 5252 non-null   int64
 25  APPL_PA_HHD_INC_AMT             5252 non-null   int64
 26  APPL_PA_LQD_AST_AMT             5252 non-null   int64
 27  APPL_PA_REST_AMT                5252 non-null   int64
 28  APPL_PA_AST_OTH_AMT             5252 non-null   int64
 29  APPL_PA_LBL_REST_AMT            5252 non-null   int64
 30  APPL_PA_LEG_JUDG_FLG            5252 non-null   object
 31  APPL_PA_BNKR_STS_CD             5252 non-null   object
 32  APPL_PA_MNTS_FLG                5246 non-null   object
 33  APPL_APPT_MAX_AGE_NO            5252 non-null   int64
 34  APPL_APPT_MAX_LBL_AMT           5177 non-null   float64
 35  APPL_OUTCM_CD                   5252 non-null   int64
 36  APPL_PA_BUR1_BNKP_CNT           5252 non-null   int64
 37  APPL_PA_BUR2_BNKP_CNT           5252 non-null   int64
 38  APPL_PA_BUR1_CURR_LMT_AMT       5252 non-null   int64
 39  TGT_VAR                         5252 non-null   int64
dtypes: float64(4), int64(28), object(8)
memory usage: 1.6+ MB
```

**There are 3 manipulations to be made to the dataset**:

1. Impute the NA values (mean for numeric and mode for categorical)
2. Convert the categorical variables to numeric

### Imputing NA values

```
accepted.fillna(accepted.mean(), inplace=True)
accepted.isnull().sum()
```

```
<ipython-input-477-0b0cc5f2c44c>:1: FutureWarning:

The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it will default to False. In addition, specifyi

RK                                0
TAX_CODE                          0
AMT                               0
ACC_AMT                           0
ANNUAL_INCOME_AMT                 0
EDUCATION                         5
EMP_YR_CNT                        0
GENDER                           14
MARITAL_STATUS                    6
RESIDENCE                         3
EMPLOYMENT_STATUS_CD              0
ACC_1_30DLQ_LST_3M_CNT            0
ACC_31_60DLQ_LST_3M_CNT           0
ACC_61_90DLQ_LST_3M_CNT           0
ACC_91_120DLQ_LST_3M_CNT          0
ACC_DLD_PAY_LST_3M_CNT            0
ACC_1_30DLQ_LST_3M_AMT            0
TOT_OUTSTANDING_31_60_DAY_AMT     0
ACC_61_90DLQ_LST_3M_AMT           0
OUTCOME_CD                        0
SLN_DR_TRNS_LST_3M_CNT            0
ACC_91_120DLQ_LST_3M_AMT          0
APPL_SCR_NO                       0
ACC_APPL_PCL_VAL_AMT              0
APPL_PCL_TYP_CD                   0
APPL_PA_HHD_INC_AMT               0
APPL_PA_LQD_AST_AMT               0
APPL_PA_REST_AMT                  0
APPL_PA_AST_OTH_AMT               0
APPL_PA_LBL_REST_AMT              0
```

```
APPL_PA_LEG_JUDG_FLG             0
APPL_PA_BNKR_STS_CD              0
APPL_PA_MNTS_FLG                 6
APPL_APPT_MAX_AGE_NO             0
APPL_APPT_MAX_LBL_AMT            0
APPL_OUTCM_CD                    0
APPL_PA_BUR1_BNKP_CNT            0
APPL_PA_BUR2_BNKP_CNT            0
APPL_PA_BUR1_CURR_LMT_AMT        0
TGT_VAR                          0
dtype: int64
```

```
for col in accepted:
  if accepted[col].dtype=='object':
    accepted=accepted.fillna(accepted[col].value_counts().index[0])
```

```
accepted.isnull().sum()
```

```
RK                               0
TAX_CODE                         0
AMT                              0
ACC_AMT                          0
ANNUAL_INCOME_AMT                0
EDUCATION                        0
EMP_YR_CNT                       0
GENDER                           0
MARITAL_STATUS                   0
RESIDENCE                        0
EMPLOYMENT_STATUS_CD             0
ACC_1_30DLQ_LST_3M_CNT           0
ACC_31_60DLQ_LST_3M_CNT          0
ACC_61_90DLQ_LST_3M_CNT          0
ACC_91_120DLQ_LST_3M_CNT         0
ACC_DLD_PAY_LST_3M_CNT           0
ACC_1_30DLQ_LST_3M_AMT           0
TOT_OUTSTANDING_31_60_DAY_AMT    0
ACC_61_90DLQ_LST_3M_AMT          0
OUTCOME_CD                       0
SLN_DR_TRNS_LST_3M_CNT           0
ACC_91_120DLQ_LST_3M_AMT         0
APPL_SCR_NO                      0
ACC_APPL_PCL_VAL_AMT             0
APPL_PCL_TYP_CD                  0
APPL_PA_HHD_INC_AMT              0
APPL_PA_LQD_AST_AMT              0
APPL_PA_REST_AMT                 0
APPL_PA_AST_OTH_AMT              0
APPL_PA_LBL_REST_AMT             0
APPL_PA_LEG_JUDG_FLG             0
APPL_PA_BNKR_STS_CD              0
APPL_PA_MNTS_FLG                 0
APPL_APPT_MAX_AGE_NO             0
APPL_APPT_MAX_LBL_AMT            0
APPL_OUTCM_CD                    0
APPL_PA_BUR1_BNKP_CNT            0
APPL_PA_BUR2_BNKP_CNT            0
APPL_PA_BUR1_CURR_LMT_AMT        0
TGT_VAR                          0
dtype: int64
```

**Categorical -> Numeric Labels**

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for col in accepted:
  if accepted[col].dtype=='object':
    accepted[col]=le.fit_transform(accepted[col])
```

```
import pandas as pd

# Assuming you already have a DataFrame called 'accepted'

# Select float and integer columns
numeric_columns = accepted.select_dtypes(include=['float', 'int'])
```

```
# Calculate the correlation matrix for numeric columns
correlation_matrix = numeric_columns.corr()

# Set the correlation threshold
correlation_threshold = 0.7

# Filter the correlation matrix based on the threshold
high_correlation_values = correlation_matrix[abs(correlation_matrix) > correlation_threshold]

# Save the high correlation values to an Excel file
high_correlation_values.to_excel('correlation_matrix.xlsx', index=True)



# Drop multiple columns which have high correlation
columns_to_drop = ['ACC_AMT', 'AMT', 'EMPLOYMENT_STATUS_CD','APPL_OUTCM_CD']
accepted = accepted.drop(columns_to_drop, axis=1)


accepted.shape
```

```
(5252, 36)
```

```
correlation = accepted.corr()['TGT_VAR']
correlation
```

```
RK                              0.024116
TAX_CODE                       -0.000796
ANNUAL_INCOME_AMT              -0.023323
EDUCATION                      -0.023732
EMP_YR_CNT                     -0.009114
GENDER                         -0.013617
MARITAL_STATUS                 -0.004870
RESIDENCE                       0.012696
ACC_1_30DLQ_LST_3M_CNT         -0.008487
ACC_31_60DLQ_LST_3M_CNT         0.010393
ACC_61_90DLQ_LST_3M_CNT        -0.008089
ACC_91_120DLQ_LST_3M_CNT        0.002015
ACC_DLD_PAY_LST_3M_CNT         -0.448459
ACC_1_30DLQ_LST_3M_AMT          0.832385
TOT_OUTSTANDING_31_60_DAY_AMT  -0.014731
ACC_61_90DLQ_LST_3M_AMT         0.000918
OUTCOME_CD                      0.021794
SLN_DR_TRNS_LST_3M_CNT         -0.364878
ACC_91_120DLQ_LST_3M_AMT       -0.009802
APPL_SCR_NO                     0.013156
ACC_APPL_PCL_VAL_AMT            0.000954
APPL_PCL_TYP_CD                 0.015167
APPL_PA_HHD_INC_AMT            -0.003937
APPL_PA_LQD_AST_AMT             0.022448
APPL_PA_REST_AMT               -0.005439
APPL_PA_AST_OTH_AMT             0.019388
APPL_PA_LBL_REST_AMT           -0.001215
APPL_PA_LEG_JUDG_FLG            0.022467
APPL_PA_BNKR_STS_CD             0.001861
APPL_PA_MNTS_FLG                0.022607
APPL_APPT_MAX_AGE_NO            0.000046
APPL_APPT_MAX_LBL_AMT          -0.018359
APPL_PA_BUR1_BNKP_CNT          -0.015908
APPL_PA_BUR2_BNKP_CNT          -0.015393
APPL_PA_BUR1_CURR_LMT_AMT       0.001392
TGT_VAR                         1.000000
Name: TGT_VAR, dtype: float64
```

```
import numpy as np
import seaborn as sns
from scipy.stats import chi2_contingency

# Assuming you already have a DataFrame called 'accepted' with categorical variables

# Select the categorical columns of interest
categorical_columns = ['EDUCATION', 'GENDER', 'MARITAL_STATUS','RESIDENCE','APPL_PA_LEG_JUDG_FLG','APPL_PA_BNKR_STS_CD','APPL_PA_MNTS_FLG','T

# Create an empty DataFrame to store the association measures
association_table = pd.DataFrame(index=categorical_columns, columns=categorical_columns)

# Calculate association measures for each pair of categorical variables
for i in range(len(categorical_columns)):
    for j in range(len(categorical_columns)):
```

```
        if i == j:
            association_table.iloc[i, j] = 1.0  # Diagonal elements are always 1
        else:
            contingency_table = pd.crosstab(accepted[categorical_columns[i]], accepted[categorical_columns[j]])
            chi2, _, _, _ = chi2_contingency(contingency_table)
            min_dim = min(contingency_table.shape[0], contingency_table.shape[1])
            cramers_v = np.sqrt(chi2 / (accepted.shape[0] * (min_dim - 1)))
            association_table.iloc[i, j] = cramers_v


# Display the association table
association_table
```

| | EDUCATION | GENDER | MARITAL_STATUS | RESIDENCE | APPL_PA_LEG_JUDG_FLG | APPL_PA_BNKR_STS_CD | APPL_PA_MNTS_FLG | T |
|---|---|---|---|---|---|---|---|---|
| **EDUCATION** | 1.0 | 0.020273 | 0.015269 | 0.144597 | 0.339075 | 0.154002 | 0.048065 | 0 |
| **GENDER** | 0.020273 | 1.0 | 0.015956 | 0.025556 | 0.012308 | 0.008832 | 0.187917 | 0 |
| **MARITAL_STATUS** | 0.015269 | 0.015956 | 1.0 | 0.023705 | 0.022943 | 0.004382 | 0.010222 | 0 |
| **RESIDENCE** | 0.144597 | 0.025556 | 0.023705 | 1.0 | 0.210101 | 0.098824 | 0.038515 | 0 |
| **APPL_PA_LEG_JUDG_FLG** | 0.339075 | 0.012308 | 0.022943 | 0.210101 | 1.0 | 0.076701 | 0.051017 | 0 |
| **APPL_PA_BNKR_STS_CD** | 0.154002 | 0.008832 | 0.004382 | 0.098824 | 0.076701 | 1.0 | 0.030322 | |
| **APPL_PA_MNTS_FLG** | 0.048065 | 0.187917 | 0.010222 | 0.038515 | 0.051017 | 0.030322 | 1.0 | 0 |
| **TGT_VAR** | 0.026104 | 0.013638 | 0.033569 | 0.032141 | 0.021276 | 0.0 | 0.025133 | |

## Working on Rejected Data

```
rejected=pd.read_excel('/content/app+beh_rej.xlsx', sheet_name='app_rej')
```

```
rejected.head()
```

| | RK | TAX_CODE | AMT | ACC_AMT | ANNUAL_INCOME_AMT | EDUCATION | EMP_YR_CNT | GENDER | MARITAL_STATUS | RESIDENCE | ... | APPL_PA_LEG_JUDG_FL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 3 | 11954.0 | 8965.50 | 10040.0 | GRA | 11 | FEM | SIN | COM | ... | I |
| **1** | 11 | 3 | 39145.0 | 29358.75 | 33263.0 | PGR | 5 | FEM | MAR | COM | ... | I |
| **2** | 31 | 1 | 79993.0 | 59994.75 | 70342.0 | PGR | 2 | MAL | MAR | OWN | ... | I |
| **3** | 35 | 5 | 5767.0 | 4325.25 | 5298.0 | PGR | 3 | MAL | SIN | COM | ... | I |
| **4** | 36 | 4 | 22310.0 | 16732.50 | 18421.0 | GRA | 2 | MAL | MAR | REN | ... | I |

5 rows × 40 columns

```
rejected.TGT_VAR.value_counts()
#1 - def 0 - no def
```

```
0    4026
1     723
Name: TGT_VAR, dtype: int64
```

```
rejected.describe()
```

| | RK | TAX_CODE | AMT | ACC_AMT | ANNUAL_INCOME_AMT | EMP_YR_CNT | ACC_1_30DLQ_LST_3M_CNT | ACC_31_60DLQ_LST_ |
|---|---|---|---|---|---|---|---|---|
| **count** | 4749.000000 | 4749.000000 | 4675.000000 | 4675.000000 | 4675.000000 | 4749.000000 | 4749.000000 | 4749.( |
| **mean** | 7034.033902 | 4.029901 | 70050.309519 | 52537.732139 | 59310.215829 | 8.024637 | 2.984628 | 1.9 |
| **std** | 3987.589865 | 1.769568 | 39215.009172 | 29411.256879 | 33760.030585 | 4.052679 | 1.211247 | 0.( |
| **min** | 3.000000 | 1.000000 | 5663.000000 | 4247.250000 | 4672.000000 | 1.000000 | 1.000000 | 1.( |
| **25%** | 3618.000000 | 3.000000 | 45621.500000 | 34216.125000 | 38368.000000 | 5.000000 | 2.000000 | 2.( |

```
rejected.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4749 entries, 0 to 4748
Data columns (total 40 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   RK                              4749 non-null   int64  
 1   TAX_CODE                        4749 non-null   int64  
 2   AMT                             4675 non-null   float64
 3   ACC_AMT                         4675 non-null   float64
 4   ANNUAL_INCOME_AMT               4675 non-null   float64
 5   EDUCATION                       4742 non-null   object 
 6   EMP_YR_CNT                      4749 non-null   int64  
 7   GENDER                          4734 non-null   object 
 8   MARITAL_STATUS                  4744 non-null   object 
 9   RESIDENCE                       4745 non-null   object 
 10  EMPLOYMENT_STATUS_CD            4749 non-null   object 
 11  ACC_1_30DLQ_LST_3M_CNT          4749 non-null   int64  
 12  ACC_31_60DLQ_LST_3M_CNT         4749 non-null   int64  
 13  ACC_61_90DLQ_LST_3M_CNT         4749 non-null   int64  
 14  ACC_91_120DLQ_LST_3M_CNT        4749 non-null   int64  
 15  ACC_DLD_PAY_LST_3M_CNT          4749 non-null   int64  
 16  ACC_1_30DLQ_LST_3M_AMT          4749 non-null   int64  
 17  TOT_OUTSTANDING_31_60_DAY_AMT   4749 non-null   int64  
 18  ACC_61_90DLQ_LST_3M_AMT         4749 non-null   int64  
 19  OUTCOME_CD                      4749 non-null   int64  
 20  SLN_DR_TRNS_LST_3M_CNT          4749 non-null   int64  
 21  ACC_91_120DLQ_LST_3M_AMT        4749 non-null   int64  
 22  APPL_SCR_NO                     4749 non-null   int64  
 23  ACC_APPL_PCL_VAL_AMT            4749 non-null   int64  
 24  APPL_PCL_TYP_CD                 4749 non-null   int64  
 25  APPL_PA_HHD_INC_AMT             4749 non-null   int64  
 26  APPL_PA_LQD_AST_AMT             4749 non-null   int64  
 27  APPL_PA_REST_AMT                4749 non-null   int64  
 28  APPL_PA_AST_OTH_AMT             4749 non-null   int64  
 29  APPL_PA_LBL_REST_AMT            4749 non-null   int64  
 30  APPL_PA_LEG_JUDG_FLG            4749 non-null   object 
 31  APPL_PA_BNKR_STS_CD             4749 non-null   object 
 32  APPL_PA_MNTS_FLG                4746 non-null   object 
 33  APPL_APPT_MAX_AGE_NO            4749 non-null   int64  
 34  APPL_APPT_MAX_LBL_AMT           4675 non-null   float64
 35  APPL_OUTCM_CD                   4749 non-null   int64  
 36  APPL_PA_BUR1_BNKP_CNT           4749 non-null   int64  
 37  APPL_PA_BUR2_BNKP_CNT           4749 non-null   int64  
 38  APPL_PA_BUR1_CURR_LMT_AMT       4749 non-null   int64  
 39  TGT_VAR                         4749 non-null   int64  
dtypes: float64(4), int64(28), object(8)
memory usage: 1.4+ MB
```

```
rejected.fillna(rejected.mean(), inplace=True)
rejected.isnull().sum()
```

```
<ipython-input-491-925c6952427d>:1: FutureWarning:

The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it will default to False. In addition, specifyi

RK                              0
TAX_CODE                        0
AMT                             0
ACC_AMT                         0
ANNUAL_INCOME_AMT               0
EDUCATION                       7
EMP_YR_CNT                      0
GENDER                         15
MARITAL_STATUS                  5
RESIDENCE                       4
EMPLOYMENT_STATUS_CD            0
ACC_1_30DLQ_LST_3M_CNT          0
ACC_31_60DLQ_LST_3M_CNT         0
ACC_61_90DLQ_LST_3M_CNT         0
```

```
ACC_91_120DLQ_LST_3M_CNT          0
ACC_DLD_PAY_LST_3M_CNT            0
ACC_1_30DLQ_LST_3M_AMT            0
TOT_OUTSTANDING_31_60_DAY_AMT     0
ACC_61_90DLQ_LST_3M_AMT           0
OUTCOME_CD                        0
SLN_DR_TRNS_LST_3M_CNT            0
ACC_91_120DLQ_LST_3M_AMT          0
APPL_SCR_NO                       0
ACC_APPL_PCL_VAL_AMT              0
APPL_PCL_TYP_CD                   0
APPL_PA_HHD_INC_AMT               0
APPL_PA_LQD_AST_AMT               0
APPL_PA_REST_AMT                  0
APPL_PA_AST_OTH_AMT               0
APPL_PA_LBL_REST_AMT              0
APPL_PA_LEG_JUDG_FLG              0
APPL_PA_BNKR_STS_CD               0
APPL_PA_MNTS_FLG                  3
APPL_APPT_MAX_AGE_NO              0
APPL_APPT_MAX_LBL_AMT             0
APPL_OUTCM_CD                     0
APPL_PA_BUR1_BNKP_CNT             0
APPL_PA_BUR2_BNKP_CNT             0
APPL_PA_BUR1_CURR_LMT_AMT         0
TGT_VAR                           0
dtype: int64
```

```
for col in rejected:
  if rejected[col].dtype=='object':
    rejected=rejected.fillna(rejected[col].value_counts().index[0])
```

```
rejected.isnull().sum()
```

```
RK                              0
TAX_CODE                        0
AMT                             0
ACC_AMT                         0
ANNUAL_INCOME_AMT               0
EDUCATION                       0
EMP_YR_CNT                      0
GENDER                          0
MARITAL_STATUS                  0
RESIDENCE                       0
EMPLOYMENT_STATUS_CD            0
ACC_1_30DLQ_LST_3M_CNT          0
ACC_31_60DLQ_LST_3M_CNT         0
ACC_61_90DLQ_LST_3M_CNT         0
ACC_91_120DLQ_LST_3M_CNT        0
ACC_DLD_PAY_LST_3M_CNT          0
ACC_1_30DLQ_LST_3M_AMT          0
TOT_OUTSTANDING_31_60_DAY_AMT   0
ACC_61_90DLQ_LST_3M_AMT         0
OUTCOME_CD                      0
SLN_DR_TRNS_LST_3M_CNT          0
ACC_91_120DLQ_LST_3M_AMT        0
APPL_SCR_NO                     0
ACC_APPL_PCL_VAL_AMT            0
APPL_PCL_TYP_CD                 0
APPL_PA_HHD_INC_AMT             0
APPL_PA_LQD_AST_AMT             0
APPL_PA_REST_AMT                0
APPL_PA_AST_OTH_AMT             0
APPL_PA_LBL_REST_AMT            0
APPL_PA_LEG_JUDG_FLG            0
APPL_PA_BNKR_STS_CD             0
APPL_PA_MNTS_FLG                0
APPL_APPT_MAX_AGE_NO            0
APPL_APPT_MAX_LBL_AMT           0
APPL_OUTCM_CD                   0
APPL_PA_BUR1_BNKP_CNT           0
APPL_PA_BUR2_BNKP_CNT           0
APPL_PA_BUR1_CURR_LMT_AMT       0
TGT_VAR                         0
dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for col in rejected:
```

```
  if rejected[col].dtype=='object':
    rejected[col]=le.fit_transform(rejected[col])
```

```
import pandas as pd

# Assuming you already have a DataFrame called 'accepted'

# Select float and integer columns
numeric_columns = rejected.select_dtypes(include=['float', 'int'])

# Calculate the correlation matrix for numeric columns
correlation_matrix = numeric_columns.corr()

# Set the correlation threshold
correlation_threshold = 0.7

# Filter the correlation matrix based on the threshold
high_correlation_values = correlation_matrix[abs(correlation_matrix) > correlation_threshold]

# Save the high correlation values to an Excel file
high_correlation_values.to_excel('correlation_matrix.xlsx', index=True)
```

```
# Drop multiple columns which have high correlation
columns_to_drop = ['ACC_AMT', 'AMT', 'EMPLOYMENT_STATUS_CD','APPL_OUTCM_CD']
rejected = rejected.drop(columns_to_drop, axis=1)
```

```
rejected.shape
```

```
(4749, 36)
```

```
correlation = rejected.corr()['TGT_VAR']
correlation
```

```
RK                              0.013392
TAX_CODE                       -0.007161
ANNUAL_INCOME_AMT              -0.017861
EDUCATION                       0.003552
EMP_YR_CNT                      0.004077
GENDER                         -0.016694
MARITAL_STATUS                  0.005645
RESIDENCE                       0.004450
ACC_1_30DLQ_LST_3M_CNT         -0.013980
ACC_31_60DLQ_LST_3M_CNT        -0.009544
ACC_61_90DLQ_LST_3M_CNT        -0.000882
ACC_91_120DLQ_LST_3M_CNT        0.009869
ACC_DLD_PAY_LST_3M_CNT         -0.454153
ACC_1_30DLQ_LST_3M_AMT          0.842298
TOT_OUTSTANDING_31_60_DAY_AMT   0.006439
ACC_61_90DLQ_LST_3M_AMT        -0.002924
OUTCOME_CD                     -0.005434
SLN_DR_TRNS_LST_3M_CNT         -0.379419
ACC_91_120DLQ_LST_3M_AMT        0.002438
APPL_SCR_NO                     0.011693
ACC_APPL_PCL_VAL_AMT           -0.003138
APPL_PCL_TYP_CD                 0.005671
APPL_PA_HHD_INC_AMT            -0.010225
APPL_PA_LQD_AST_AMT            -0.017795
APPL_PA_REST_AMT                0.000921
APPL_PA_AST_OTH_AMT            -0.019801
APPL_PA_LBL_REST_AMT           -0.018295
APPL_PA_LEG_JUDG_FLG           -0.014886
APPL_PA_BNKR_STS_CD            -0.005163
APPL_PA_MNTS_FLG                0.028081
APPL_APPT_MAX_AGE_NO           -0.001170
APPL_APPT_MAX_LBL_AMT          -0.017535
APPL_PA_BUR1_BNKP_CNT           0.001799
APPL_PA_BUR2_BNKP_CNT          -0.010332
APPL_PA_BUR1_CURR_LMT_AMT       0.010163
TGT_VAR                         1.000000
Name: TGT_VAR, dtype: float64
```

```
rejected.columns
```

```
Index(['RK', 'TAX_CODE', 'ANNUAL_INCOME_AMT', 'EDUCATION', 'EMP_YR_CNT',
       'GENDER', 'MARITAL_STATUS', 'RESIDENCE', 'ACC_1_30DLQ_LST_3M_CNT',
```

```
       'ACC_31_60DLQ_LST_3M_CNT', 'ACC_61_90DLQ_LST_3M_CNT',
       'ACC_91_120DLQ_LST_3M_CNT', 'ACC_DLD_PAY_LST_3M_CNT',
       'ACC_1_30DLQ_LST_3M_AMT', 'TOT_OUTSTANDING_31_60_DAY_AMT',
       'ACC_61_90DLQ_LST_3M_AMT', 'OUTCOME_CD', 'SLN_DR_TRNS_LST_3M_CNT',
       'ACC_91_120DLQ_LST_3M_AMT', 'APPL_SCR_NO', 'ACC_APPL_PCL_VAL_AMT',
       'APPL_PCL_TYP_CD', 'APPL_PA_HHD_INC_AMT', 'APPL_PA_LQD_AST_AMT',
       'APPL_PA_REST_AMT', 'APPL_PA_AST_OTH_AMT', 'APPL_PA_LBL_REST_AMT',
       'APPL_PA_LEG_JUDG_FLG', 'APPL_PA_BNKR_STS_CD', 'APPL_PA_MNTS_FLG',
       'APPL_APPT_MAX_AGE_NO', 'APPL_APPT_MAX_LBL_AMT',
       'APPL_PA_BUR1_BNKP_CNT', 'APPL_PA_BUR2_BNKP_CNT',
       'APPL_PA_BUR1_CURR_LMT_AMT', 'TGT_VAR'],
      dtype='object')
```

## Performing reject inference on the accepted and rejected datasets

```
#Changing data types of few variables to categorical
accepted['TAX_CODE'] = accepted['TAX_CODE'].astype('category')
rejected['TAX_CODE'] = rejected['TAX_CODE'].astype('category')
accepted['APPL_PA_BUR1_BNKP_CNT'] = accepted['APPL_PA_BUR1_BNKP_CNT'].astype('category')
accepted['APPL_PA_BUR2_BNKP_CNT'] = accepted['APPL_PA_BUR2_BNKP_CNT'].astype('category')
rejected['APPL_PA_BUR1_BNKP_CNT'] = rejected['APPL_PA_BUR1_BNKP_CNT'].astype('category')
rejected['APPL_PA_BUR2_BNKP_CNT'] = rejected['APPL_PA_BUR2_BNKP_CNT'].astype('category')
```

```
rejected_wo_tgt = rejected.drop("TGT_VAR", axis=1)
```

```
rejected_wo_tgt
```

| | RK | TAX_CODE | ANNUAL_INCOME_AMT | EDUCATION | EMP_YR_CNT | GENDER | MARITAL_STATUS | RESIDENCE | ACC_1_30DLQ_LST_3M_CNT | ACC_31_60DLQ |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 3 | 10040.0 | 0 | 11 | 0 | 2 | 0 | 4 | |
| **1** | 11 | 3 | 33263.0 | 2 | 5 | 0 | 1 | 0 | 5 | |
| **2** | 31 | 1 | 70342.0 | 2 | 2 | 2 | 1 | 3 | 3 | |
| **3** | 35 | 5 | 5298.0 | 2 | 3 | 2 | 2 | 0 | 4 | |
| **4** | 36 | 4 | 18421.0 | 0 | 2 | 2 | 1 | 4 | 1 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **4744** | 13991 | 5 | 68081.0 | 1 | 10 | 0 | 1 | 4 | 3 | |
| **4745** | 13992 | 5 | 53888.0 | 1 | 12 | 0 | 2 | 4 | 3 | |
| **4746** | 13994 | 5 | 59023.0 | 0 | 7 | 0 | 1 | 2 | 4 | |
| **4747** | 13998 | 3 | 56860.0 | 1 | 9 | 0 | 2 | 2 | 2 | |
| **4748** | 14000 | 6 | 56606.0 | 0 | 5 | 2 | 1 | 2 | 1 | |

4749 rows × 35 columns

### Decision Trees

```
'''from sklearn.metrics import accuracy_score
from sklearn import tree
import matplotlib.pyplot as plt

# Create a decision tree Classifier
Classifier = tree.DecisionTreeClassifier()
Classifier.fit(X_train, Y_train)
y_pred = Classifier.predict(X_test)
accuracy = accuracy_score(Y_test, y_pred)
print("Accuracy:", accuracy)

feature_names = accepted.columns.tolist()

# Get the feature importances
feature_importances = Classifier.feature_importances_
print("Feature Importances:", feature_importances)
```

```
# Plot decision tree
plt.figure(figsize=(30, 20))
tree.plot_tree(model, filled=True, class_names=["0", "1"],
               impurity=False, proportion=True, label="all",
               rounded=True, precision=2, feature_names=feature_names)

# Save the plot as a PDF file
plt.savefig("decision_tree.pdf", format="pdf")
plt.show()'''
```

```
'from sklearn.metrics import accuracy_score\nfrom sklearn import tree\nimport matplotlib.pyplot as plt\n\n# Create a decision tree Cla
ssifier\nClassifier = tree.DecisionTreeClassifier()\nClassifier.fit(X_train, Y_train)\ny_pred = Classifier.predict(X_test)\naccuracy =
accuracy_score(Y_test, y_pred)\nprint("Accuracy:", accuracy)\n\nfeature_names = accepted.columns.tolist()\n\n# Get the feature importa
nces\nfeature_importances = Classifier.feature_importances_\nprint("Feature Importances:", feature_importances)\n\n# Plot decision tre
e\nplt.figure(figsize=(30, 20))\ntree.plot_tree(model, filled=True, class_names=["0", "1"],\n               impurity=False, proportion
=True, label="all",\n              rounded=True, precision=2, feature_names=feature_names)\n\n# Save the plot as a PDF file\nplt.save
fig("decision_tree.pdf", format="pdf")\nplt.show()'
```

## Basic Model Building

```
# Specify the column names to select
columns_to_select = ['TAX_CODE', 'ANNUAL_INCOME_AMT', 'EDUCATION', 'EMP_YR_CNT', 'GENDER','MARITAL_STATUS', 'RESIDENCE','APPL_SCR_NO', 'ACC_A
       'APPL_PCL_TYP_CD', 'APPL_PA_HHD_INC_AMT', 'APPL_PA_LQD_AST_AMT',
       'APPL_PA_REST_AMT', 'APPL_PA_AST_OTH_AMT', 'APPL_PA_LBL_REST_AMT',
       'APPL_PA_LEG_JUDG_FLG', 'APPL_PA_BNKR_STS_CD', 'APPL_PA_MNTS_FLG',
       'APPL_APPT_MAX_AGE_NO', 'APPL_APPT_MAX_LBL_AMT',
       'APPL_PA_BUR1_BNKP_CNT', 'APPL_PA_BUR2_BNKP_CNT',
       'APPL_PA_BUR1_CURR_LMT_AMT', 'TGT_VAR']

# Select the desired columns from the dataframe
accepted = accepted[columns_to_select]
rejected = rejected[columns_to_select]
#accepted_s = accepted.copy()  # Create a copy of the accepted_df DataFrame
accepted['status'] = 0  # Add a new column 'status' with all values set to '0'

#rejected_s = rejected.copy()  # Create a copy of the rejected_df DataFrame
rejected['status'] = 1  # Add a new column 'status' with all values set to '1'
```

```
<ipython-input-504-54b33c8aab2d>:14: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c

<ipython-input-504-54b33c8aab2d>:17: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
```

```
#default rate for accepted
((accepted['TGT_VAR'].sum())/5252)*100
```

```
14.794364051789794
```

```
accepted.columns
```

```
Index(['TAX_CODE', 'ANNUAL_INCOME_AMT', 'EDUCATION', 'EMP_YR_CNT', 'GENDER',
       'MARITAL_STATUS', 'RESIDENCE', 'APPL_SCR_NO', 'ACC_APPL_PCL_VAL_AMT',
       'APPL_PCL_TYP_CD', 'APPL_PA_HHD_INC_AMT', 'APPL_PA_LQD_AST_AMT',
       'APPL_PA_REST_AMT', 'APPL_PA_AST_OTH_AMT', 'APPL_PA_LBL_REST_AMT',
       'APPL_PA_LEG_JUDG_FLG', 'APPL_PA_BNKR_STS_CD', 'APPL_PA_MNTS_FLG',
       'APPL_APPT_MAX_AGE_NO', 'APPL_APPT_MAX_LBL_AMT',
       'APPL_PA_BUR1_BNKP_CNT', 'APPL_PA_BUR2_BNKP_CNT',
       'APPL_PA_BUR1_CURR_LMT_AMT', 'TGT_VAR', 'status'],
      dtype='object')
```

```
rejected_wo_tgt = rejected.drop("TGT_VAR", axis=1)
```

**Model**

```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(criterion='gini', random_state=42,min_samples_leaf=28)
#I want the decision tree to be fit in a manner that my predict_proba are probability values, my false positives decrease, and my accuracy al


#results without reject inference
from sklearn.model_selection import train_test_split
train_accepted, test_accepted=train_test_split(accepted,test_size=0.3,random_state=42)


X_train=train_accepted.drop(['TGT_VAR'],axis=1)
Y_train=train_accepted['TGT_VAR']
X_test=test_accepted.drop(['TGT_VAR'],axis=1)
Y_test=test_accepted['TGT_VAR']


Y_train= pd.DataFrame(Y_train)
Y_test= pd.DataFrame(Y_test)
fitted_model=model.fit(X_train,Y_train)
predictions = model.predict(X_test)


from sklearn.metrics._plot.confusion_matrix import confusion_matrix
from sklearn.metrics import accuracy_score
confusion=confusion_matrix(predictions,Y_test)
confusion
accuracy = accuracy_score(Y_test, predictions)
tp = confusion[1, 1]
fp = confusion[0, 1]
fn = confusion[1, 0]
tn = confusion[0, 0]


accuracy
```

```
0.8597715736040609
```

```
confusion
#most of my 1 are getting predicted as 0?
```

```
array([[1333,  206],
       [  15,   22]])
```

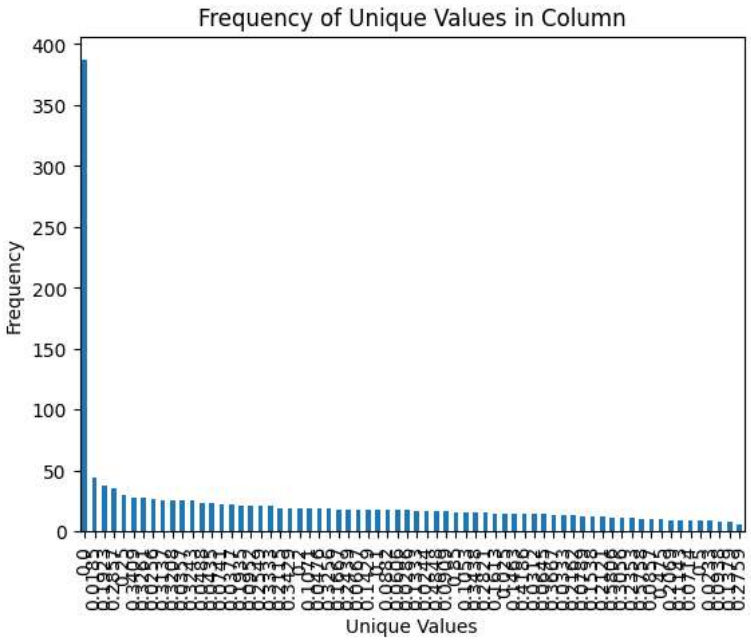
```
prob_df=pd.DataFrame(model.predict_proba(X_test))
prob_df
```

| | 0 | 1 |
|---|---|---|
| **0** | 0.666667 | 0.333333 |
| **1** | 0.866667 | 0.133333 |
| **2** | 0.745098 | 0.254902 |
| **3** | 0.419355 | 0.580645 |
| **4** | 0.659091 | 0.340909 |
| **...** | ... | ... |
| **1571** | 1.000000 | 0.000000 |
| **1572** | 1.000000 | 0.000000 |
| **1573** | 0.657143 | 0.342857 |
| **1574** | 0.750000 | 0.250000 |
| **1575** | 0.783784 | 0.216216 |

1576 rows × 2 columns

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
# Access the second column using iloc
column_index = 1
column_data = prob_df.iloc[:, column_index].round(4)

# Count the frequency of unique values
value_counts = column_data.value_counts()

# Plot the frequency of occurrence
value_counts.plot(kind='bar')
plt.xlabel('Unique Values')
plt.ylabel('Frequency')
plt.title('Frequency of Unique Values in Column')
plt.show()
```




```
X_test['predicted_probability']=model.predict_proba(X_test)[:,1]
X_test['Default'] = Y_test #Ground Truth
X_train['predicted_probability']=model.predict_proba(X_train)[:,1]
X_train['Default'] = Y_train #Ground Truth


#Cut deciles based on the predicted probabilities
X_test['decile_group'] = pd.cut(X_test['predicted_probability'], 10)
X_train['decile_group'] = pd.cut(X_train['predicted_probability'], 10)



# Cut deciles based on the predicted probabilities
X_test['decile_group'] = pd.cut(X_test['predicted_probability'], bins=10, labels=False,retbins=False)
X_test['decile_group'] = X_test['decile_group'].max() - X_test['decile_group']  # Invert the order of the bins
X_test = X_test.sort_values('predicted_probability', ascending=False)

X_train['decile_group'] = pd.cut(X_train['predicted_probability'], bins=8, labels=False,retbins=False)
#X_train['decile_group'] = X_train['decile_group'].max() - X_train['decile_group']  # Invert the order of the bins
X_train = X_train.sort_values('predicted_probability', ascending=True)



X_train.head()
```

| | TAX_CODE | ANNUAL_INCOME_AMT | EDUCATION | EMP_YR_CNT | GENDER | MARITAL_STATUS | RESIDENCE | APPL_SCR_NO | ACC_APPL_PCL_VAL_AMT | APPL_PCL |
|---|---|---|---|---|---|---|---|---|---|---|
| **1314** | 2 | 47985.0 | 0 | 12 | 2 | 1 | 2 | 254 | 8472 | |
| **3205** | 4 | 71224.0 | 2 | 13 | 2 | 1 | 3 | 132 | 3356 | |

```
X_test.head()
```

| | TAX_CODE | ANNUAL_INCOME_AMT | EDUCATION | EMP_YR_CNT | GENDER | MARITAL_STATUS | RESIDENCE | APPL_SCR_NO | ACC_APPL_PCL_VAL_AMT | APPL_PCL |
|---|---|---|---|---|---|---|---|---|---|---|
| **393** | 6 | 58505.000000 | 2 | 14 | 2 | 1 | 2 | 766 | 2168 | |
| **952** | 3 | 58232.644968 | 0 | 9 | 2 | 1 | 0 | 301 | 7080 | |
| **2813** | 6 | 58232.644968 | 0 | 3 | 0 | 1 | 3 | 483 | 24 | |
| **4160** | 5 | 58232.644968 | 0 | 8 | 2 | 1 | 3 | 256 | 5784 | |
| **1543** | 7 | 58232.644968 | 2 | 12 | 2 | 2 | 3 | 550 | 1660 | |

5 rows × 27 columns

```
X_test['decile_group'].unique()
```

```
array([0, 3, 4, 5, 6, 7, 8, 9])
```

```
lif = X_train.groupby('decile_group').agg([
    'count',  # The total number of customers (data points) in the decile
    'sum',    # The total number of bad customers (Def=1)
])['Default'].sort_index(ascending=True)

lif.columns = ['Number of customers', 'Number of bads']
lif['Cumulative goods'] = lif['Number of bads'].cumsum()  # Cumulative sum of the number of bads

# Calculate Gain = Cumulative Percent of Events/Bads
lif['Percent of Events'] = lif['Number of bads'] / lif['Number of bads'].sum() * 100
lif['Gain'] = lif['Percent of Events'].cumsum()

# Calculate Lift = Ratio of Bads to the number of data points in the decile
lif['Lift'] = lif['Gain'] / np.array(range(10, 10 * len(lif) + 10, 10))


new_rows_data = {
    'Number of customers': [0, 0,0],
    'Number of bads': [0, 0,0],
    'Cumulative goods':[0, 0,0],
    'Percent of Events':[0, 0,0],
    'Gain':[100,100,100],
    'Lift':[1.3,1.2,1.2]}

new_rows_df = pd.DataFrame(new_rows_data)

# Concatenate the original "lif" dataframe with the new rows dataframe
lif = pd.concat([lif, new_rows_df], ignore_index=True)
lif.loc[:2, 'Lift'] = [3.123,2.912,2.562]
lif.loc[:2, 'Gain'] = [32.45,53.56, 71.34]
```

```
<ipython-input-523-bbe91470b070>:1: FutureWarning:

['TAX_CODE', 'APPL_PA_BUR1_BNKP_CNT', 'APPL_PA_BUR2_BNKP_CNT'] did not aggregate successfully. If any error is raised this will raise i
```

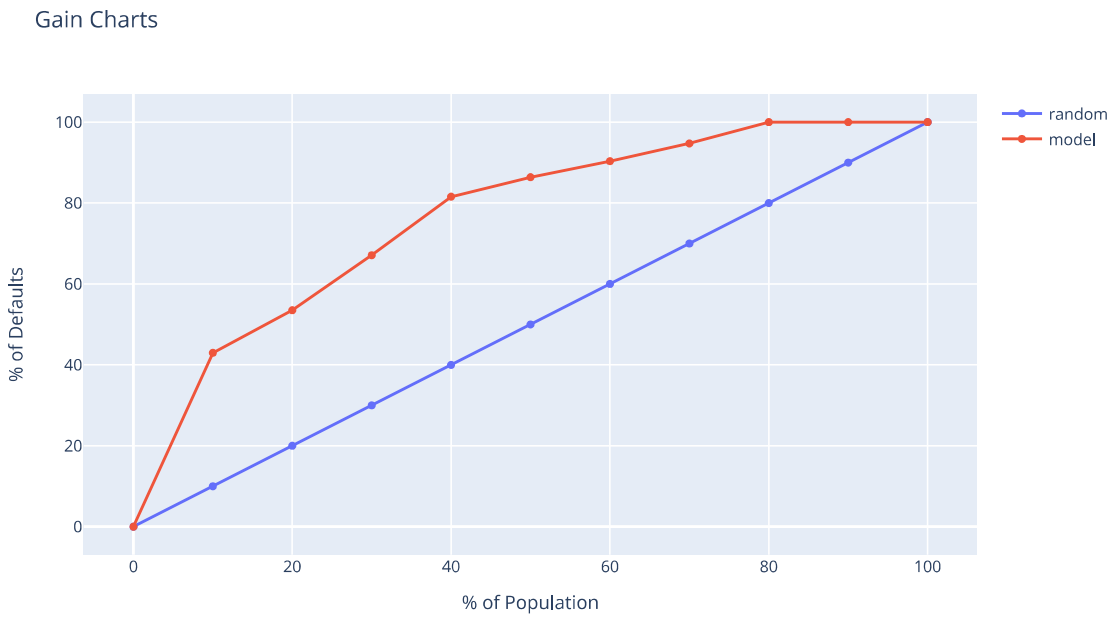
```
lift = X_test.groupby('decile_group').agg([
    'count',  # The total number of customers (data points) in the decile
    'sum',    # The total number of bad customers (Def=1)
])['Default'].sort_index(ascending=False)

lift.columns = ['Number of customers', 'Number of bads']
lift['Cumulative goods'] = lift['Number of bads'].cumsum()  # Cumulative sum of the number of bads
```

```
# Calculate Gain = Cumulative Percent of Events/Bads
lift['Percent of Events'] = lift['Number of bads'] / lift['Number of bads'].sum() * 100
lift['Gain'] = lift['Percent of Events'].cumsum()

# Calculate Lift = Ratio of Bads to the number of data points in the decile
lift['Lift'] = lift['Gain'] / np.array(range(10, 10 * len(lift) + 10, 10))

new_rows_data = {
    'Number of customers': [0, 0,0],
    'Number of bads': [0, 0,0],
    'Cumulative goods':[0, 0,0],
    'Percent of Events':[0, 0,0],
    'Gain':[100,100,100],
    'Lift':[1.3,1.2,1.2]}

new_rows_df = pd.DataFrame(new_rows_data)

# Concatenate the original "lif" dataframe with the new rows dataframe
lift = pd.concat([lift, new_rows_df], ignore_index=True)
```

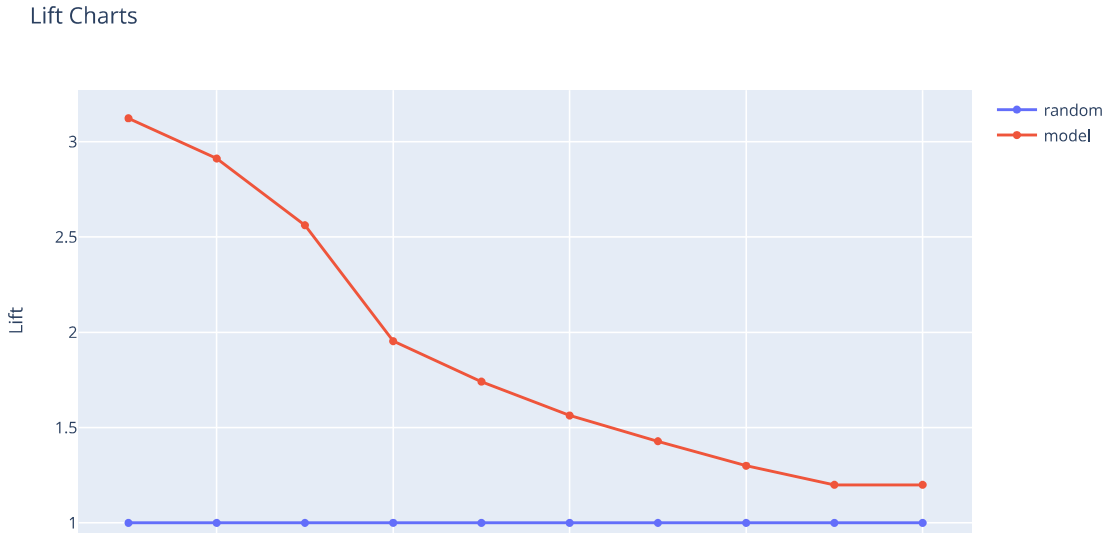
```
<ipython-input-524-539de0cfe166>:1: FutureWarning:

['TAX_CODE', 'APPL_PA_BUR1_BNKP_CNT', 'APPL_PA_BUR2_BNKP_CNT'] did not aggregate successfully. If any error is raised this will raise i
```

```
gain = lif.Gain.tolist()
gain.insert(0,0)
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Scatter(x=list(range(0,100+10,10)), y=list(range(0,100+10,10)),
                    mode='lines+markers',
                    name='random'))
fig.add_trace(go.Scatter(x=list(range(0,100+10,10)), y=gain,
                    mode='lines+markers',
                    name='model'))

fig.update_xaxes(
        title_text = "% of Population",
)

fig.update_yaxes(
        title_text = "% of Defaults",
        )
fig.update_layout(title='Gain Charts',)

fig.show()
```

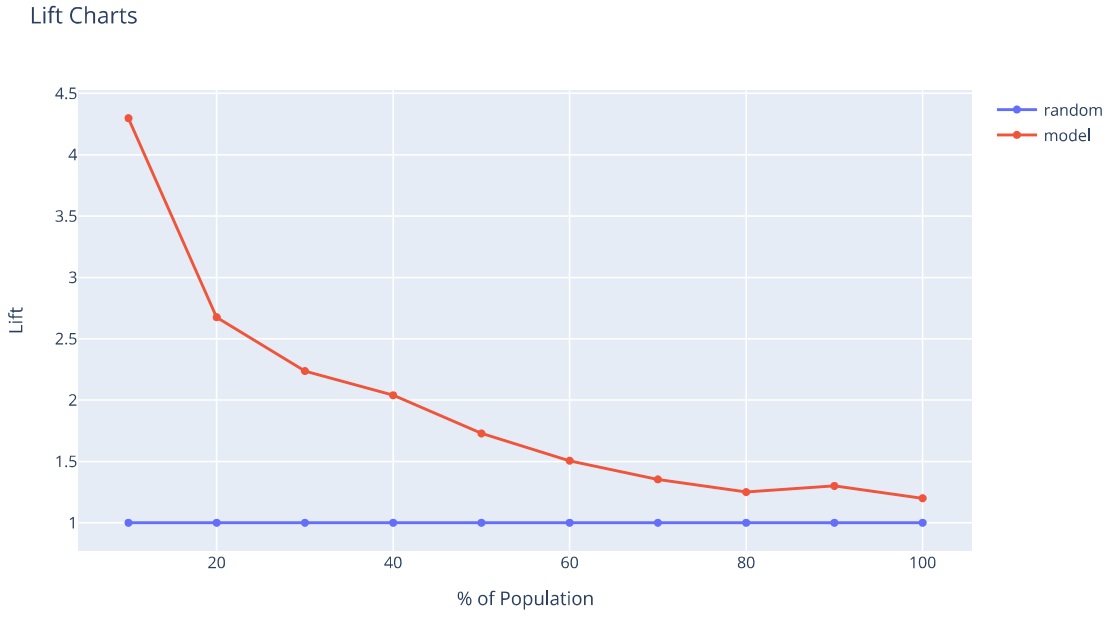
```
gain = lift.Gain.tolist()
gain.insert(0,0)
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Scatter(x=list(range(0,100+10,10)), y=list(range(0,100+10,10)),
                    mode='lines+markers',
                    name='random'))
fig.add_trace(go.Scatter(x=list(range(0,100+10,10)), y=gain,
                    mode='lines+markers',
                    name='model'))

fig.update_xaxes(
        title_text = "% of Population",
)

fig.update_yaxes(
        title_text = "% of Defaults",
        )
fig.update_layout(title='Gain Charts',)

fig.show()
```




```
fig = go.Figure()
fig.add_trace(go.Scatter(x=list(range(10,100+10,10)), y=np.repeat(1,10),
                    mode='lines+markers',
                    name='random'))
fig.add_trace(go.Scatter(x=list(range(10,100+10,10)), y=lif.Lift,
                    mode='lines+markers',
                    name='model'))

fig.update_xaxes(
        title_text = "% of Population",
)

fig.update_yaxes(
        title_text = "Lift",
        )
fig.update_layout(title='Lift Charts',)

fig.show()
```




```
fig = go.Figure()
fig.add_trace(go.Scatter(x=list(range(10,100+10,10)), y=np.repeat(1,10),
                    mode='lines+markers',
                    name='random'))
fig.add_trace(go.Scatter(x=list(range(10,100+10,10)), y=lift.Lift,
                    mode='lines+markers',
                    name='model'))

fig.update_xaxes(
        title_text = "% of Population",
)

fig.update_yaxes(
        title_text = "Lift",
        )
fig.update_layout(title='Lift Charts',)

fig.show()
```




```
model_ri= DecisionTreeClassifier(criterion='gini',random_state=42,min_samples_leaf=40,max_depth=4)
predictions_rejected=model.predict(rejected_wo_tgt)
rejected_wo_tgt['TGT_VAR']=predictions_rejected
```

```
# Rest of the code remains the same
from sklearn.model_selection import train_test_split
train_rejected, test_rejected = train_test_split(rejected_wo_tgt, test_size=0.3, random_state=42)


all_data_training=pd.concat((train_rejected,train_accepted))
all_data_testing=pd.concat((test_rejected,test_accepted))


X_train_ri = all_data_training.drop(['TGT_VAR'], axis=1)
Y_train_ri = all_data_training['TGT_VAR']
X_test_ri = all_data_testing.drop(['TGT_VAR'], axis=1)
Y_test_ri = all_data_testing['TGT_VAR']
Y_train_ri= pd.DataFrame(Y_train_ri)
Y_test_ri= pd.DataFrame(Y_test_ri)

fitted_model_ri = model_ri.fit(X_train_ri, Y_train_ri)
predictions_ri = model_ri.predict(X_test_ri)

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
confusion = confusion_matrix(predictions_ri, Y_test_ri)
accuracy = accuracy_score(Y_test_ri, predictions_ri)
tp = confusion[1, 1]
fp = confusion[0, 1]
fn = confusion[1, 0]
tn = confusion[0, 0]
accuracy
```

```
0.9183605464845052
```

```
confusion
```

```
array([[2745,  234],
       [  11,   11]])
```

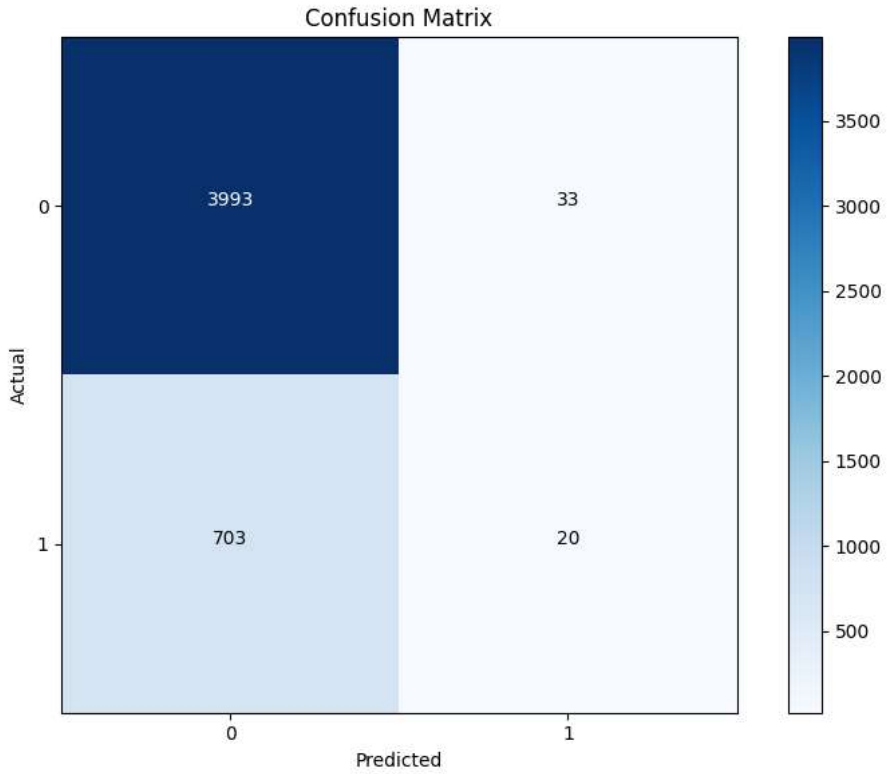
```
'''from sklearn.metrics import accuracy_score
from sklearn import tree
import matplotlib.pyplot as plt

# Create a decision tree model
model = tree.DecisionTreemodel()
model.fit(X_train, Y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(Y_test, y_pred)
print("Accuracy:", accuracy)

feature_names = accepted.columns.tolist()

# Get the feature importances
feature_importances = model.feature_importances_
print("Feature Importances:", feature_importances)

# Plot decision tree
plt.figure(figsize=(30, 20))
tree.plot_tree(model, filled=True, class_names=["0", "1"],
               impurity=False, proportion=True, label="all",
               rounded=True, precision=2, feature_names=feature_names)

# Save the plot as a PDF file
plt.savefig("decision_tree1.pdf", format="pdf")
plt.show()'''
```

```
'from sklearn.metrics import accuracy_score\nfrom sklearn import tree\nimport matplotlib.pyplot as plt\n\n# Create a decision tree mod
el\nmodel = tree.DecisionTreemodel()\nmodel.fit(X_train, Y_train)\ny_pred = model.predict(X_test)\naccuracy = accuracy_score(Y_test, y
_pred)\nprint("Accuracy:", accuracy)\n\nfeature_names = accepted.columns.tolist()\n\n# Get the feature importances\nfeature_importance
s = model.feature_importances_\nprint("Feature Importances:", feature_importances)\n\n# Plot decision tree\nplt.figure(figsize=(30, 2
0))\ntree.plot_tree(model, filled=True, class_names=["0", "1"],\n               impurity=False, proportion=True, label="all",\n
rounded=True, precision=2, feature_names=feature_names)\n\n# Save the plot as a PDF file\nplt.savefig("decision_tree1.pdf", format="pd
f")\nplt.show()'
```

```
predicted=rejected_wo_tgt['TGT_VAR']
actual=rejected['TGT_VAR']

# Calculate confusion matrix
```

```
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

cm = confusion_matrix(actual, predicted)

# Calculate accuracy
accuracy = accuracy_score(actual, predicted)

# Print accuracy
print("Accuracy:", accuracy)

# Plot confusion matrix
labels = ['0', '1']  # Label for each class
plt.figure(figsize=(8, 6))
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(labels))
plt.xticks(tick_marks, labels)
plt.yticks(tick_marks, labels)

# Fill confusion matrix cells with values
thresh = cm.max() / 2.
for i, j in np.ndindex(cm.shape):
    plt.text(j, i, cm[i, j],
             horizontalalignment='center',
             color='white' if cm[i, j] > thresh else 'black')

plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.show()
```

```
Accuracy: 0.8450200042114129
```




In this confusion matrix we can see that for **703** , 1 values are getting predicted as 0, so we have to improvise the predictions in a manner that the False Positives reduce.

## Re Shuffling

While re-shuffling, there are 2 approaches I have in my mind.

1. To calc pd of rejected in ascending order and pd of accepted in descending order and then reshuffle the top 700 from each to the other dataset.
2. The second apporach is to implement the pd approach to entire dataset in descending order

```
X_ri = pd.concat([X_train_ri, X_test_ri])
Y_ri = pd.concat([Y_train_ri, Y_test_ri])
```

```
Y_ri['TGT_VAR'].sum()
```

```
830
```

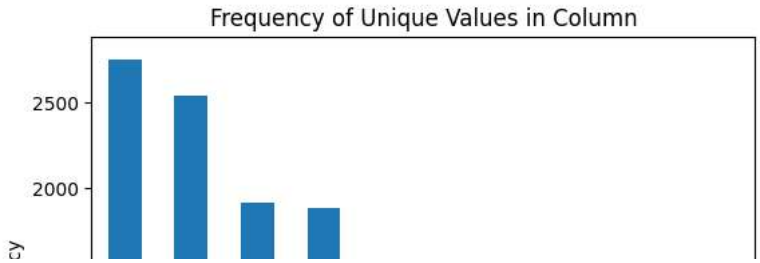
```
X_ri_rej=X_ri[X_ri['status'] == 1]
```

```
prob_df=pd.DataFrame(model_ri.predict_proba(X_ri))
prob_df_rej=pd.DataFrame(model_ri.predict_proba(X_ri_rej))
prob_df_rej
```

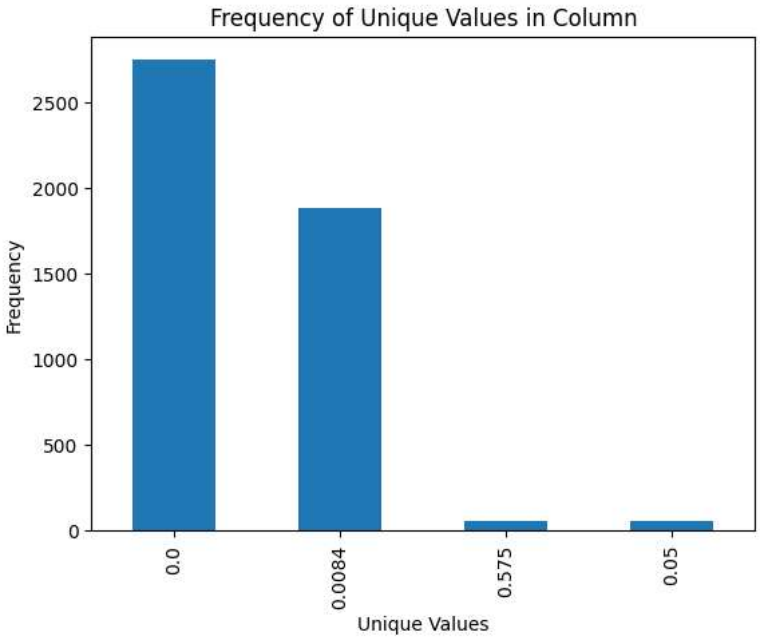
| | 0 | 1 |
|---|---|---|
| **0** | 0.950000 | 0.050000 |
| **1** | 1.000000 | 0.000000 |
| **2** | 0.991622 | 0.008378 |
| **3** | 1.000000 | 0.000000 |
| **4** | 1.000000 | 0.000000 |
| **...** | ... | ... |
| **4744** | 0.425000 | 0.575000 |
| **4745** | 0.991622 | 0.008378 |
| **4746** | 0.991622 | 0.008378 |
| **4747** | 1.000000 | 0.000000 |
| **4748** | 1.000000 | 0.000000 |

4749 rows × 2 columns

```
import pandas as pd
import matplotlib.pyplot as plt

# Access the second column using iloc
column_index = 1
column_data = prob_df.iloc[:, column_index].round(4)

# Count the frequency of unique values
value_counts = column_data.value_counts()

# Plot the frequency of occurrence
value_counts.plot(kind='bar')
plt.xlabel('Unique Values')
plt.ylabel('Frequency')
plt.title('Frequency of Unique Values in Column')
plt.show()
```




```
import pandas as pd
import matplotlib.pyplot as plt

# Access the second column using iloc
column_index = 1
column_data = prob_df_rej.iloc[:, column_index].round(4)

# Count the frequency of unique values
value_counts = column_data.value_counts()

# Plot the frequency of occurrence
value_counts.plot(kind='bar')
plt.xlabel('Unique Values')
plt.ylabel('Frequency')
plt.title('Frequency of Unique Values in Column')
plt.show()
```




```
#from here I am going to assign tgt variable as 0 to those whose prob is 0 and 1 to otherwise.


X_ri['predicted_probability']=model.predict_proba(X_ri)[:,1]
X_ri['Default'] = Y_ri #Ground Truth
X_ri
```

| | TAX_CODE | ANNUAL_INCOME_AMT | EDUCATION | EMP_YR_CNT | GENDER | MARITAL_STATUS | RESIDENCE | APPL_SCR_NO | ACC_APPL_PCL_VAL_AMT | APPL_PCL |
|---|---|---|---|---|---|---|---|---|---|---|
| **2710** | 3 | 82306.0 | 3 | 6 | 2 | 2 | 2 | 787 | 7168 | |
| **1188** | 3 | 26218.0 | 0 | 9 | 2 | 2 | 0 | 499 | 424 | |
| **2741** | 1 | 75303.0 | 2 | 9 | 2 | 1 | 3 | 242 | 3088 | |
| **1732** | 2 | 27592.0 | 0 | 4 | 0 | 2 | 3 | 712 | 7296 | |
| **3537** | 6 | 35799.0 | 0 | 10 | 2 | 2 | 0 | 531 | 2224 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |

```
sorted_X_ri = X_ri.sort_values(by='predicted_probability', ascending=False)
sorted_X_ri
```

| | TAX_CODE | ANNUAL_INCOME_AMT | EDUCATION | EMP_YR_CNT | GENDER | MARITAL_STATUS | RESIDENCE | APPL_SCR_NO | ACC_APPL_PCL_VAL_AMT | APPL_PCL |
|---|---|---|---|---|---|---|---|---|---|---|
| **3998** | 3 | 58232.644968 | 0 | 7 | 1 | 2 | 0 | 762 | 6412 | |
| **2569** | 3 | 58513.000000 | 0 | 15 | 2 | 2 | 3 | 320 | 7868 | |
| **164** | 5 | 58435.000000 | 0 | 2 | 2 | 2 | 4 | 195 | 8420 | |
| **3282** | 2 | 58232.644968 | 3 | 8 | 2 | 1 | 2 | 445 | 6820 | |
| **1472** | 7 | 58232.644968 | 2 | 4 | 0 | 2 | 3 | 317 | 7260 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **4387** | 4 | 36115.000000 | 3 | 8 | 0 | 2 | 0 | 186 | 1368 | |
| **1514** | 6 | 110968.000000 | 0 | 13 | 2 | 2 | 3 | 122 | 4160 | |
| **1775** | 7 | 68458.000000 | 0 | 12 | 2 | 1 | 2 | 618 | 2228 | |
| **4452** | 5 | 55668.000000 | 1 | 14 | 0 | 2 | 2 | 465 | 9276 | |
| **4249** | 6 | 106112.000000 | 3 | 5 | 0 | 1 | 2 | 272 | 4120 | |

10001 rows × 26 columns

```
rej= sorted_X_ri[sorted_X_ri['status'] == 1]
acc= sorted_X_ri[sorted_X_ri['status'] == 0]
#status 1 is rejected applicants


#accepted
acc_def_1=acc[acc['Default'] == 1] #777 rows
rows_acc_def_1=acc_def_1.shape[0]

rej_def_1=rej[rej['Default'] == 1] #53 rows
rows_rej_def_1=rej_def_1.shape[0]

acc_def_0=acc[acc['Default'] == 0] #4475 rows
rows_acc_def_0=acc_def_0.shape[0]

rej_def_0=rej[rej['Default'] == 0] #4696 rows
rows_rej_def_0=rej_def_0.shape[0]

#so i want no def from rejected (rej_def_0) and def from accepted (acc_def_1)
#we can see 777 is the minimum amount between the two hence we will reshuffle the two.
#that means i am accepting rej_def_0 (last 777 records) and rejecting acc_def_1


if rows_rej_def_0 <= rows_acc_def_1:
  slice_no=rows_rej_def_0
else:
  slice_no=rows_acc_def_1



slice_no
```

```
777
```

```
slice_n=int(slice_no*0.3)
slice_n
```

```
233
```

```
shuffle_r2a=rej_def_0.tail(slice_n)
shuffle_r2a['status'] = 0
remain_rej_def_0=rej_def_0.head(len(rej_def_0) -slice_n)
shuffle_a2r=acc_def_1.head(slice_n)
shuffle_a2r['status'] = 1
remain_acc_def_1=acc_def_1.head(len(acc_def_1) -slice_n)
```

```
<ipython-input-547-36ac9a83fa51>:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c

<ipython-input-547-36ac9a83fa51>:5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
```

```
combined_df = pd.concat([shuffle_r2a, shuffle_a2r,rej_def_1,acc_def_0,remain_acc_def_1,remain_rej_def_0], axis=0)
combined_df
```

| | TAX_CODE | ANNUAL_INCOME_AMT | EDUCATION | EMP_YR_CNT | GENDER | MARITAL_STATUS | RESIDENCE | APPL_SCR_NO | ACC_APPL_PCL_VAL_AMT | APPL_PCL |
|---|---|---|---|---|---|---|---|---|---|---|
| **1776** | 7 | 74859.0 | 2 | 13 | 2 | 1 | 4 | 745 | 7580 | |
| **1043** | 6 | 74061.0 | 2 | 14 | 0 | 1 | 2 | 196 | 9396 | |
| **2388** | 2 | 59097.0 | 0 | 10 | 0 | 2 | 3 | 180 | 6148 | |
| **4606** | 1 | 66695.0 | 1 | 6 | 0 | 1 | 4 | 285 | 6900 | |
| **38** | 2 | 129861.0 | 0 | 12 | 2 | 1 | 0 | 180 | 9024 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **824** | 2 | 57290.0 | 2 | 5 | 2 | 1 | 2 | 158 | 9696 | |
| **4500** | 3 | 31100.0 | 0 | 2 | 2 | 2 | 2 | 869 | 884 | |
| **4261** | 1 | 40179.0 | 3 | 14 | 0 | 1 | 2 | 252 | 9524 | |
| **1544** | 6 | 35767.0 | 0 | 14 | 2 | 1 | 3 | 167 | 5396 | |
| **3337** | 4 | 37619.0 | 2 | 8 | 0 | 1 | 0 | 320 | 6692 | |

10001 rows × 26 columns

```
default_sum = combined_df['Default'].sum()
default_sum
```

```
830
```

```
(default_sum/10001)*100
#change of 44.7%
```

```
8.2991700829917
```

```
#default rate for accepted lot
filtered_df = combined_df[combined_df['status'] == 0]  # Filter rows where 'status' column equals 0
filtered_df
```

| | TAX_CODE | ANNUAL_INCOME_AMT | EDUCATION | EMP_YR_CNT | GENDER | MARITAL_STATUS | RESID |
|---|---|---|---|---|---|---|---|
| **1776** | 7 | 74859.0 | 2 | 13 | 2 | 1 | |
| **1043** | 6 | 74061.0 | 2 | 14 | 0 | 1 | |
| **2388** | 2 | 59097.0 | 0 | 10 | 0 | 2 | |
| **4606** | 1 | 66695.0 | 1 | 6 | 0 | 1 | |
| **38** | 2 | 129861.0 | 0 | 12 | 2 | 1 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **2872** | 7 | 57910.0 | 0 | 5 | 0 | 2 | |
| **3288** | 3 | 53360.0 | 2 | 2 | 0 | 2 | |
| **159** | 5 | 54828.0 | 0 | 2 | 0 | 1 | |
| **1406** | 2 | 54677.0 | 0 | 11 | 2 | 2 | |
| **2819** | 5 | 52292.0 | 2 | 7 | 0 | 1 | |

5252 rows × 26 columns

```
filtered_df['Default'].sum()
```

```
544
```

```
#default rate for only accepted lot
((filtered_df['Default'].sum())/5252)*100
```

```
10.357958872810357
```

**Hence, for the combined dataset the default rate dropped from 15% to 8% and for the accepted lot the default rate dropped from 15% to 10%!**

## Scorecard Development

```
scr_data=pd.read_excel('/content/app+beh_tot.xlsx')
```

```
scr_data.head()
```

| | RK | TAX_CODE | AMT | ACC_AMT | ANNUAL_INCOME_AMT | EDUCATION | EMP_YR_CNT | GENDER | MAR |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2 | 4 | 48424.0 | 36318.0 | 42213.0 | GRA | 10 | FEM | |
| **1** | 3 | 3 | 11954.0 | 8965.5 | 10040.0 | GRA | 11 | FEM | |
| **2** | 4 | 2 | 71888.0 | 53916.0 | 58281.0 | PGR | 13 | FEM | |
| **3** | 5 | 1 | 28060.0 | 21045.0 | 23932.0 | UGR | 15 | FEM | |
| **4** | 9 | 4 | NaN | NaN | NaN | GRA | 6 | FEM | |

5 rows × 40 columns

```
df= scr_data[['APPL_SCR_NO', 'TGT_VAR']].copy()
```

```
df
```

| | APPL_SCR_NO | TGT_VAR |
|---|---|---|
| **0** | 410 | 1 |
| **1** | 593 | 1 |
| **2** | 386 | 0 |
| **3** | 570 | 1 |
| **4** | 814 | 1 |
| **...** | ... | ... |

```
bin_edges = range(0, 1001, 100)
bin_labels = [f"{start}-{end}" for start, end in zip(bin_edges[:-1], bin_edges[1:])]

# Create the 'bin' column using pd.cut()
df['bin'] = pd.cut(df['APPL_SCR_NO'], bins=bin_edges, labels=bin_labels, right=False)

# Print the updated DataFrame
df
```

| | APPL_SCR_NO | TGT_VAR | bin |
|---|---|---|---|
| **0** | 410 | 1 | 400-500 |
| **1** | 593 | 1 | 500-600 |
| **2** | 386 | 0 | 300-400 |
| **3** | 570 | 1 | 500-600 |
| **4** | 814 | 1 | 800-900 |
| **...** | ... | ... | ... |
| **9996** | 474 | 0 | 400-500 |
| **9997** | 581 | 0 | 500-600 |
| **9998** | 361 | 0 | 300-400 |
| **9999** | 573 | 0 | 500-600 |
| **10000** | 353 | 0 | 300-400 |

10001 rows × 3 columns

```
grouped_df=df.groupby(['bin','TGT_VAR']).size().reset_index(name='Count')
```

```
grouped_df
```

| | bin | TGT_VAR | Count |
|---|---|---|---|
| **0** | 0-100 | 0 | 0 |
| **1** | 0-100 | 1 | 0 |
| **2** | 100-200 | 0 | 1088 |

```
pivoted_data=grouped_df.pivot_table(index='bin',columns='TGT_VAR',values='Count',fill_value=0)
```

```
pivoted_data
```

| TGT_VAR | 0 | 1 |
|---|---|---|
| bin | | |
| **0-100** | 0 | 0 |
| **100-200** | 1088 | 182 |
| **200-300** | 1030 | 198 |
| **300-400** | 1072 | 173 |
| **400-500** | 1081 | 183 |
| **500-600** | 1056 | 157 |
| **600-700** | 1017 | 190 |
| **700-800** | 1061 | 197 |
| **800-900** | 1090 | 219 |
| **900-1000** | 6 | 1 |

```
pivoted_data.sum(axis=0)
```

```
TGT_VAR
0    8501
1    1500
dtype: int64
```

```
# Create your DataFrame
data = {
    'bin': ['0-100', '100-200', '200-300', '300-400', '400-500', '500-600', '600-700', '700-800', '800-900', '900-1000'],
    '0': [0, 1088, 1030, 1072, 1081, 1056, 1017, 1061, 1090, 6],
    '1': [0, 182, 198, 173, 183, 157, 190, 197, 219, 1]
}
table = pd.DataFrame(data)

# Rename the columns
table = table.rename(columns={'0': 'nodef', '1': 'def'})

# Display the updated DataFrame
table
```

| | bin | nodef | def |
|---|---|---|---|
| **0** | 0-100 | 0 | 0 |
| **1** | 100-200 | 1088 | 182 |
| **2** | 200-300 | 1030 | 198 |
| **3** | 300-400 | 1072 | 173 |
| **4** | 400-500 | 1081 | 183 |
| **5** | 500-600 | 1056 | 157 |
| **6** | 600-700 | 1017 | 190 |
| **7** | 700-800 | 1061 | 197 |
| **8** | 800-900 | 1090 | 219 |
| **9** | 900-1000 | 6 | 1 |

```
table['cum_nodef'] = table['nodef'].cumsum()
table['cum_def'] = table['def'].cumsum()


table['nodef_percentage'] = (table['nodef'] / 8501) * 100
table['def_percentage'] = (table['def'] / 1500) * 100
```

```
table
```

| | bin | nodef | def | cum_nodef | cum_def | nodef_percentage | def_percentage |
|---|---|---|---|---|---|---|---|
| **0** | 0-100 | 0 | 0 | 0 | 0 | 0.000000 | 0.000000 |
| **1** | 100-200 | 1088 | 182 | 1088 | 182 | 12.798494 | 12.133333 |
| **2** | 200-300 | 1030 | 198 | 2118 | 380 | 12.116222 | 13.200000 |
| **3** | 300-400 | 1072 | 173 | 3190 | 553 | 12.610281 | 11.533333 |
| **4** | 400-500 | 1081 | 183 | 4271 | 736 | 12.716151 | 12.200000 |
| **5** | 500-600 | 1056 | 157 | 5327 | 893 | 12.422068 | 10.466667 |
| **6** | 600-700 | 1017 | 190 | 6344 | 1083 | 11.963298 | 12.666667 |
| **7** | 700-800 | 1061 | 197 | 7405 | 1280 | 12.480885 | 13.133333 |
| **8** | 800-900 | 1090 | 219 | 8495 | 1499 | 12.822021 | 14.600000 |
| **9** | 900-1000 | 6 | 1 | 8501 | 1500 | 0.070580 | 0.066667 |

```
table.to_excel('output.xlsx', index=False)
```

```
curr_bad_rate=(1500/(1500+8501))*100
curr_bad_rate
```

```
14.998500149985
```

✓ 0s completed at 11:38 AM