

Data Import and Cleaning

```
In [1]: # import libraries and read in the excel file
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_excel("default of credit card clients.xls")
df.head()
```

```
Out[1]:      Unnamed:  
          0      X1      X2      X3      X4      X5      X6      X7      X8  
---  
 0      ID LIMIT_BAL SEX EDUCATION MARRIAGE AGE PAY_0 PAY_2 PAY_3 PAY_4  
 1      1    20000   2       2       1     24     2     2     -1  
 2      2    120000  2       2       2     26    -1     2     0  
 3      3    90000   2       2       2     34     0     0     0  
 4      4    50000   2       2       2     37     0     0     0  
 5      5    40000   2       2       2     33     1     1     0  
 6      6    30000   2       2       2     30     0     0     0  
 7      7    20000   2       2       2     27     1     1     0  
 8      8    10000   2       2       2     24     0     0     0  
 9      9     5000   2       2       2     23     0     0     0  
 10    10    4000   2       2       2     22     0     0     0  
 11    11    3000   2       2       2     21     0     0     0  
 12    12    2000   2       2       2     20     0     0     0  
 13    13    1000   2       2       2     19     0     0     0  
 14    14     500   2       2       2     18     0     0     0  
 15    15     200   2       2       2     17     0     0     0  
 16    16     100   2       2       2     16     0     0     0  
 17    17      50   2       2       2     15     0     0     0  
 18    18      20   2       2       2     14     0     0     0  
 19    19      10   2       2       2     13     0     0     0  
 20    20      5   2       2       2     12     0     0     0  
 21    21      2   2       2       2     11     0     0     0  
 22    22      1   2       2       2     10     0     0     0  
 23    23      0   2       2       2      9     0     0     0  
 24    24      0   2       2       2      8     0     0     0  
 25    25      0   2       2       2      7     0     0     0  
 26    26      0   2       2       2      6     0     0     0  
 27    27      0   2       2       2      5     0     0     0  
 28    28      0   2       2       2      4     0     0     0  
 29    29      0   2       2       2      3     0     0     0  
 30    30      0   2       2       2      2     0     0     0  
 31    31      0   2       2       2      1     0     0     0  
 32    32      0   2       2       2      0     0     0     0  
 33    33      0   2       2       2      0     0     0     0  
 34    34      0   2       2       2      0     0     0     0  
 35    35      0   2       2       2      0     0     0     0  
 36    36      0   2       2       2      0     0     0     0  
 37    37      0   2       2       2      0     0     0     0  
 38    38      0   2       2       2      0     0     0     0  
 39    39      0   2       2       2      0     0     0     0  
 40    40      0   2       2       2      0     0     0     0  
 41    41      0   2       2       2      0     0     0     0  
 42    42      0   2       2       2      0     0     0     0  
 43    43      0   2       2       2      0     0     0     0  
 44    44      0   2       2       2      0     0     0     0  
 45    45      0   2       2       2      0     0     0     0  
 46    46      0   2       2       2      0     0     0     0  
 47    47      0   2       2       2      0     0     0     0  
 48    48      0   2       2       2      0     0     0     0  
 49    49      0   2       2       2      0     0     0     0  
 50    50      0   2       2       2      0     0     0     0  
 51    51      0   2       2       2      0     0     0     0  
 52    52      0   2       2       2      0     0     0     0  
 53    53      0   2       2       2      0     0     0     0  
 54    54      0   2       2       2      0     0     0     0  
 55    55      0   2       2       2      0     0     0     0  
 56    56      0   2       2       2      0     0     0     0  
 57    57      0   2       2       2      0     0     0     0  
 58    58      0   2       2       2      0     0     0     0  
 59    59      0   2       2       2      0     0     0     0  
 60    60      0   2       2       2      0     0     0     0  
 61    61      0   2       2       2      0     0     0     0  
 62    62      0   2       2       2      0     0     0     0  
 63    63      0   2       2       2      0     0     0     0  
 64    64      0   2       2       2      0     0     0     0  
 65    65      0   2       2       2      0     0     0     0  
 66    66      0   2       2       2      0     0     0     0  
 67    67      0   2       2       2      0     0     0     0  
 68    68      0   2       2       2      0     0     0     0  
 69    69      0   2       2       2      0     0     0     0  
 70    70      0   2       2       2      0     0     0     0  
 71    71      0   2       2       2      0     0     0     0  
 72    72      0   2       2       2      0     0     0     0  
 73    73      0   2       2       2      0     0     0     0  
 74    74      0   2       2       2      0     0     0     0  
 75    75      0   2       2       2      0     0     0     0  
 76    76      0   2       2       2      0     0     0     0  
 77    77      0   2       2       2      0     0     0     0  
 78    78      0   2       2       2      0     0     0     0  
 79    79      0   2       2       2      0     0     0     0  
 80    80      0   2       2       2      0     0     0     0  
 81    81      0   2       2       2      0     0     0     0  
 82    82      0   2       2       2      0     0     0     0  
 83    83      0   2       2       2      0     0     0     0  
 84    84      0   2       2       2      0     0     0     0  
 85    85      0   2       2       2      0     0     0     0  
 86    86      0   2       2       2      0     0     0     0  
 87    87      0   2       2       2      0     0     0     0  
 88    88      0   2       2       2      0     0     0     0  
 89    89      0   2       2       2      0     0     0     0  
 90    90      0   2       2       2      0     0     0     0  
 91    91      0   2       2       2      0     0     0     0  
 92    92      0   2       2       2      0     0     0     0  
 93    93      0   2       2       2      0     0     0     0  
 94    94      0   2       2       2      0     0     0     0  
 95    95      0   2       2       2      0     0     0     0  
 96    96      0   2       2       2      0     0     0     0  
 97    97      0   2       2       2      0     0     0     0  
 98    98      0   2       2       2      0     0     0     0  
 99    99      0   2       2       2      0     0     0     0  
 100  100     0   2       2       2      0     0     0     0  
 101  101     0   2       2       2      0     0     0     0  
 102  102     0   2       2       2      0     0     0     0  
 103  103     0   2       2       2      0     0     0     0  
 104  104     0   2       2       2      0     0     0     0  
 105  105     0   2       2       2      0     0     0     0  
 106  106     0   2       2       2      0     0     0     0  
 107  107     0   2       2       2      0     0     0     0  
 108  108     0   2       2       2      0     0     0     0  
 109  109     0   2       2       2      0     0     0     0  
 110  110     0   2       2       2      0     0     0     0  
 111  111     0   2       2       2      0     0     0     0  
 112  112     0   2       2       2      0     0     0     0  
 113  113     0   2       2       2      0     0     0     0  
 114  114     0   2       2       2      0     0     0     0  
 115  115     0   2       2       2      0     0     0     0  
 116  116     0   2       2       2      0     0     0     0  
 117  117     0   2       2       2      0     0     0     0  
 118  118     0   2       2       2      0     0     0     0  
 119  119     0   2       2       2      0     0     0     0  
 120  120     0   2       2       2      0     0     0     0  
 121  121     0   2       2       2      0     0     0     0  
 122  122     0   2       2       2      0     0     0     0  
 123  123     0   2       2       2      0     0     0     0  
 124  124     0   2       2       2      0     0     0     0  
 125  125     0   2       2       2      0     0     0     0  
 126  126     0   2       2       2      0     0     0     0  
 127  127     0   2       2       2      0     0     0     0  
 128  128     0   2       2       2      0     0     0     0  
 129  129     0   2       2       2      0     0     0     0  
 130  130     0   2       2       2      0     0     0     0  
 131  131     0   2       2       2      0     0     0     0  
 132  132     0   2       2       2      0     0     0     0  
 133  133     0   2       2       2      0     0     0     0  
 134  134     0   2       2       2      0     0     0     0  
 135  135     0   2       2       2      0     0     0     0  
 136  136     0   2       2       2      0     0     0     0  
 137  137     0   2       2       2      0     0     0     0  
 138  138     0   2       2       2      0     0     0     0  
 139  139     0   2       2       2      0     0     0     0  
 140  140     0   2       2       2      0     0     0     0  
 141  141     0   2       2       2      0     0     0     0  
 142  142     0   2       2       2      0     0     0     0  
 143  143     0   2       2       2      0     0     0     0  
 144  144     0   2       2       2      0     0     0     0  
 145  145     0   2       2       2      0     0     0     0  
 146  146     0   2       2       2      0     0     0     0  
 147  147     0   2       2       2      0     0     0     0  
 148  148     0   2       2       2      0     0     0     0  
 149  149     0   2       2       2      0     0     0     0  
 150  150     0   2       2       2      0     0     0     0  
 151  151     0   2       2       2      0     0     0     0  
 152  152     0   2       2       2      0     0     0     0  
 153  153     0   2       2       2      0     0     0     0  
 154  154     0   2       2       2      0     0     0     0  
 155  155     0   2       2       2      0     0     0     0  
 156  156     0   2       2       2      0     0     0     0  
 157  157     0   2       2       2      0     0     0     0  
 158  158     0   2       2       2      0     0     0     0  
 159  159     0   2       2       2      0     0     0     0  
 160  160     0   2       2       2      0     0     0     0  
 161  161     0   2       2       2      0     0     0     0  
 162  162     0   2       2       2      0     0     0     0  
 163  163     0   2       2       2      0     0     0     0  
 164  164     0   2       2       2      0     0     0     0  
 165  165     0   2       2       2      0     0     0     0  
 166  166     0   2       2       2      0     0     0     0  
 167  167     0   2       2       2      0     0     0     0  
 168  168     0   2       2       2      0     0     0     0  
 169  169     0   2       2       2      0     0     0     0  
 170  170     0   2       2       2      0     0     0     0  
 171  171     0   2       2       2      0     0     0     0  
 172  172     0   2       2       2      0     0     0     0  
 173  173     0   2       2       2      0     0     0     0  
 174  174     0   2       2       2      0     0     0     0  
 175  175     0   2       2       2      0     0     0     0  
 176  176     0   2       2       2      0     0     0     0  
 177  177     0   2       2       2      0     0     0     0  
 178  178     0   2       2       2      0     0     0     0  
 179  179     0   2       2       2      0     0     0     0  
 180  180     0   2       2       2      0     0     0     0  
 181  181     0   2       2       2      0     0     0     0  
 182  182     0   2       2       2      0     0     0     0  
 183  183     0   2       2       2      0     0     0     0  
 184  184     0   2       2       2      0     0     0     0  
 185  185     0   2       2       2      0     0     0     0  
 186  186     0   2       2       2      0     0     0     0  
 187  187     0   2       2       2      0     0     0     0  
 188  188     0   2       2       2      0     0     0     0  
 189  189     0   2       2       2      0     0     0     0  
 190  190     0   2       2       2      0     0     0     0  
 191  191     0   2       2       2      0     0     0     0  
 192  192     0   2       2       2      0     0     0     0  
 193  193     0   2       2       2      0     0     0     0  
 194  194     0   2       2       2      0     0     0     0  
 195  195     0   2       2       2      0     0     0     0  
 196  196     0   2       2       2      0     0     0     0  
 197  197     0   2       2       2      0     0     0     0  
 198  198     0   2       2       2      0     0     0     0  
 199  199     0   2       2       2      0     0     0     0  
 200  200     0   2       2       2      0     0     0     0  
 201  201     0   2       2       2      0     0     0     0  
 202  202     0   2       2       2      0     0     0     0  
 203  203     0   2       2       2      0     0     0     0  
 204  204     0   2       2       2      0     0     0     0  
 205  205     0   2       2       2      0     0     0     0  
 206  206     0   2       2       2      0     0     0     0  
 207  207     0   2       2       2      0     0     0     0  
 208  208     0   2       2       2      0     0     0     0  
 209  209     0   2       2       2      0     0     0     0  
 210  210     0   2       2       2      0     0     0     0  
 211  211     0   2       2       2      0     0     0     0  
 212  212     0   2       2       2      0     0     0     0  
 213  213     0   2       2       2      0     0     0     0  
 214  214     0   2       2       2      0     0     0     0  
 215  215     0   2       2       2      0     0     0     0  
 216  216     0   2       2       2      0     0     0     0  
 217  217     0   2       2       2      0     0     0     0  
 218  218     0   2       2       2      0     0     0     0  
 219  219     0   2       2       2      0     0     0     0  
 220  220     0   2       2       2      0     0     0     0  
 221  221     0   2       2       2      0     0     0     0  
 222  222     0   2       2       2      0     0     0     0  
 223  223     0   2       2       2      0     0     0     0  
 224  224     0   2       2       2      0     0     0     0  
 225  225     0   2       2       2      0     0     0     0  
 226  226     0   2       2       2      0     0     0     0  
 227  227     0   2       2       2      0     0     0     0  
 228  228     0   2       2       2      0     0     0     0  
 229  229     0   2       2       2      0     0     0     0  
 230  230     0   2       2       2      0     0     0     0  
 231  231     0   2       2       2      0     0     0     0  
 232  232     0   2       2       2      0     0     0     0  
 233  233     0   2       2       2      0     0     0     0  
 234  234     0   2       2       2      0     0     0     0  
 235  235     0   2       2       2      0     0     0     0  
 236  236     0   2       2       2      0     0     0     0  
 237  237     0   2       2       2      0     0     0     0  
 238  238     0   2       2       2      0     0     0     0  
 239  239     0   2       2       2      0     0     0     0  
 240  240     0   2       2       2      0     0     0     0  
 241  241     0   2       2       2      0     0     0     0  
 242  242     0   2       2       2      0     0     0     0  
 243  243     0   2       2       2      0     0     0     0  
 244  244     0   2       2       2      0     0     0     0  
 245  245     0   2       2       2      0     0     0     0  
 246  246     0   2       2       2      0     0     0     0  
 247  247     0   2       2       2      0     0     0     0  
 248  248     0   2       2       2      0     0     0     0  
 249  249     0   2       2       2      0     0     0     0  
 250  250     0   2       2       2      0     0     0     0  
 251  251     0   2       2       2      0     0     0     0  
 252  252     0   2       2       2      0     0     0     0  
 253  253     0   2       2       2      0     0     0     0  
 254  254     0   2       2       2      0     0     0     0  
 255  255     0   2       2       2      0     0     0     0  
 256  256     0   2       2       2      0     0     0     0  
 257  257     0   2       2       2      0     0     0     0  
 258  258     0   2       2       2      0     0     0     0  
 259  259     0   2       2       2      0     0     0     0  
 260  260     0   2       2       2      0     0     0     0  
 261  261     0   2       2       2      0     0     0     0  
 262  262     0   2       2       2      0     0     0     0  
 263  263     0   2       2       2      0     0     0     0  
 264  264     0   2       2       2      0     0     0     0  
 265  265     0   2       2       2      0     0     0     0  
 266  266     0   2       2       2      0     0     0     0  
 267  267     0   2       2       2      0     0     0     0  
 268  268     0   2       2       2      0     0     0     0  
 269  269     0   2       2       2      0     0     0     0  
 270  270     0   2       2       2      0     0     0     0  
 271  271     0   2       2       2      0     0     0     0  
 272  272     0   2       2       2      0     0     0     0  
 273  273     0   2       2       2      0     0     0     0  
 274  274     0   2       2       2      0     0     0     0  
 275  275     0   2       2       2      0     0     0     0  
 276  276     0   2       2       2      0     0     0     0  
 277  277     0   2       2       2      0     0     0     0  
 278  278     0   2       2       2      0     0     0     0  
 279  279     0   2       2       2      0     0     0     0  
 280  280     0   2       2       2      0     0     0     0  
 281  281     0   2       2       2      0     0     0     0  
 282  282     0   2       2       2      0     0     0     0  
 283  283     0   2       2       2      0     0     0     0  
 284  284     0   2       2       2      0     0     0     0  
 285  285     0   2       2       2      0     0     0     0  
 286  286     0   2       2       2      0     0     0     0  
 287  287     0   2       2       2      0     0     0     0  
 288  288     0   2       2       2      0     0     0     0  
 289  289     0   2       2       2      0     0     0     0  
 290  290     0   2       2       2      0     0     0     0  
 291  291     0   2       2       2      0     0     0     0  
 292  292     0   2       2       2      0     0     0     0  
 293  293     0   2       2       2      0     0     0     0  
 294  294     0   2       2       2      0     0     0     0  
 295  295     0   2       2       2      0     0     0     0  
 296  296     0   2       2       2      0     0     0     0  
 297  297     0   2       2       2      0     0     0     0  
 298  298     0   2       2       2      0     0     0     0  
 299  299     0   2       2       2      0     0     0     0  
 300  300     0   2       2       2      0     0     0     0  
 301  301     0   2       2       2      0     0     0     0  
 302  302     0   2       2       2      0     0     0     0  
 303  303     0   2       2       2      0     0     0     0  
 304  304     0   2       2       2      0     0     0     0  
 305  305     0   2       2       2      0     0     0     0  
 306  306     0   2       2       2      0     0     0     0  
 307  307     0   2       2       2      0     0     0     0  
 308  308     0   2       2       2      0     0     0     0  
 309  309     0   2       2       2      0     0     0     0  
 310  310     0   2       2       2      0     0     0     0  
 311  311     0   2       2       2      0     0     0     0  
 312  312     0   2       2       2      0     0     0     0  
 313  313     0   2       2       2      0     0     0     0  
 314  314     0   2       2       2      0     0     0     0  
 315  315     0   2       2       2      0     0     0     0  
 316  316     0   2       2       2      0     0     0     0  
 317  317     0   2       2       2      0     0     0     0  
 318  318     0   2       2       2      0     0     0     0  
 319  319     0   2       2       2      0     0     0     0  
 320  320     0   2       2       2      0     0     0     0  
 
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30001 entries, 0 to 30000
Data columns (total 25 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Unnamed: 0   30001 non-null   object 
 1   X1          30001 non-null   object 
 2   X2          30001 non-null   object 
 3   X3          30001 non-null   object 
 4   X4          30001 non-null   object 
 5   X5          30001 non-null   object 
 6   X6          30001 non-null   object 
 7   X7          30001 non-null   object 
 8   X8          30001 non-null   object 
 9   X9          30001 non-null   object 
 10  X10         30001 non-null   object 
 11  X11         30001 non-null   object 
 12  X12         30001 non-null   object 
 13  X13         30001 non-null   object 
 14  X14         30001 non-null   object 
 15  X15         30001 non-null   object 
 16  X16         30001 non-null   object 
 17  X17         30001 non-null   object 
 18  X18         30001 non-null   object 
 19  X19         30001 non-null   object 
 20  X20         30001 non-null   object 
 21  X21         30001 non-null   object 
 22  X22         30001 non-null   object 
 23  X23         30001 non-null   object 
 24  Y           30001 non-null   object 
dtypes: object(25)
memory usage: 5.7+ MB
```

In [3]: `# Getting simple stats of the data
df.describe()`

Out[3]:

	Unnamed: 0	X1	X2	X3	X4	X5	X6	X7	X8	X9
count	30001	30001	30001	30001	30001	30001	30001	30001	30001	30001
unique	30001	82	3	8	5	57	12	12	12	12
top	ID	50000	2	2	2	29	0	0	0	0
freq	1	3365	18112	14030	15964	1605	14737	15730	15764	16455

4 rows × 25 columns

In [4]: `# Look at our columns again, so we can change them
df.columns`

```
Out[4]: Index(['Unnamed: 0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9',  
              'X10', 'X11', 'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X1  
              9',  
              'X20', 'X21', 'X22', 'X23', 'Y'],  
             dtype='object')
```

```
In [5]: # Drop the unnamed column  
df = df.drop(columns=["Unnamed: 0"])
```

```
In [6]: # Change our coulumn names  
df = df.rename(columns={  
    "X1": "credit_limit",  
    "X2": "gender",  
    "X3": "education",  
    "X4": "marital_status",  
    "X5": "age",  
  
    "X6": "pay_sep",  
    "X7": "pay_aug",  
    "X8": "pay_jul",  
    "X9": "pay_jun",  
    "X10": "pay_may",  
    "X11": "pay_apr",  
  
    "X12": "bill_sep",  
    "X13": "bill_aug",  
    "X14": "bill_jul",  
    "X15": "bill_jun",  
    "X16": "bill_may",  
    "X17": "bill_apr",  
  
    "X18": "paid_sep",  
    "X19": "paid_aug",  
    "X20": "paid_jul",  
    "X21": "paid_jun",  
    "X22": "paid_may",  
    "X23": "paid_apr",  
  
    "Y": "default"  
})
```

```
In [7]: # Make sure it worked  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30001 entries, 0 to 30000
Data columns (total 24 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   credit_limit     30001 non-null   object  
 1   gender           30001 non-null   object  
 2   education        30001 non-null   object  
 3   marital_status   30001 non-null   object  
 4   age              30001 non-null   object  
 5   pay_sep          30001 non-null   object  
 6   pay_aug          30001 non-null   object  
 7   pay_jul          30001 non-null   object  
 8   pay_jun          30001 non-null   object  
 9   pay_may          30001 non-null   object  
 10  pay_apr          30001 non-null   object  
 11  bill_sep         30001 non-null   object  
 12  bill_aug         30001 non-null   object  
 13  bill_jul         30001 non-null   object  
 14  bill_jun         30001 non-null   object  
 15  bill_may         30001 non-null   object  
 16  bill_apr         30001 non-null   object  
 17  paid_sep         30001 non-null   object  
 18  paid_aug         30001 non-null   object  
 19  paid_jul         30001 non-null   object  
 20  paid_jun         30001 non-null   object  
 21  paid_may         30001 non-null   object  
 22  paid_apr         30001 non-null   object  
 23  default          30001 non-null   object  
dtypes: object(24)
memory usage: 5.5+ MB
```

```
In [8]: # Get the possible values of the default column and see the frequency of the
# We can now see that there is an unwanted row
df["default"].value_counts(normalize=True)
```

```
Out[8]: default
0                  0.778774
1                  0.221193
default payment next month 0.000033
Name: proportion, dtype: float64
```

```
In [9]: # Find the row to make sure it is a constant problem and the right row
df[df["default"].astype(str).str.contains("default", case=False)]
```

```
Out[9]: credit_limit  gender  education  marital_status  age  pay_sep  pay_aug  pay_jul
0    LIMIT_BAL      SEX    EDUCATION      MARRIAGE    AGE    PAY_0    PAY_2    PAY_3
```

1 rows × 24 columns

```
In [10]: # Drop the row and set the column as a numeric
df = df[df["default"].isin([0, 1])]
df["default"] = df["default"].astype(int)
```

```
In [11]: # recheck the distribution to make sure it worked
df["default"].value_counts(normalize=True)
```

```
Out[11]: default
0    0.7788
1    0.2212
Name: proportion, dtype: float64
```

```
In [12]: # See that the row is deleted
df
```

```
Out[12]:   credit_limit  gender  education  marital_status  age  pay_sep  pay_aug  pay_jul
1      20000        2         2            1     24        2        2
2     120000        2         2            2     26       -1        2
3      90000        2         2            2     34        0        0
4      50000        2         2            1     37        0        0
5      50000        1         2            1     57       -1        0
...
29996   220000        1         3            1     39        0        0
29997   150000        1         3            2     43       -1       -1
29998   30000        1         2            2     37        4        3
29999   80000        1         3            1     41        1       -1
30000   50000        1         2            1     46        0        0
```

30000 rows × 24 columns

```
In [13]: # I want to group the columns for our model later on so that we can get more
behavior_features = [
    "pay_sep", "pay_aug", "pay_jul", "pay_may", "pay_apr",
    "bill_sep", "bill_aug", "bill_jul", "bill_jun", "bill_may", "bill_apr",
    "paid_sep", "paid_aug", "paid_jul", "paid_jun", "paid_may", "paid_apr"
]

capacity_features = [
    "credit_limit",
    "age"
]

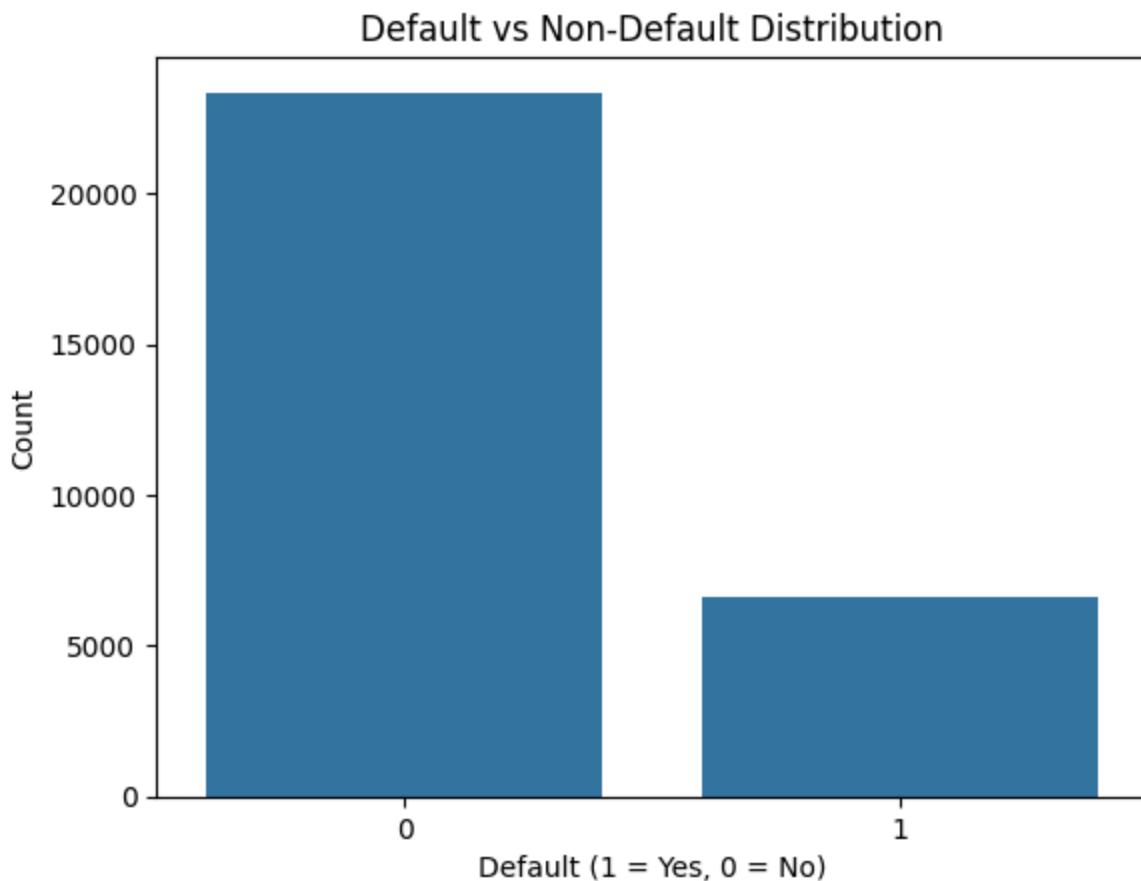
demographic_features = [
    "gender",
```

```
"education",
"marital_status"
]
```

Exploratory Data Analysis (EDA)

This section explores target balance, feature distributions, and relationships to default. EDA is used to validate data quality and identify signal before modeling.

```
In [14]: # Look at the distribution of defaults and non-defaults
sns.countplot(x="default", data=df)
plt.title("Default vs Non-Default Distribution")
plt.xlabel("Default (1 = Yes, 0 = No)")
plt.ylabel("Count")
plt.show()
```



EDA Summary

- Default rate is moderately imbalanced (~22%), justifying class-weighted models.
- Several financial features show clear distributional differences between defaulters and non-defaulters.
- Feature correlations suggest non-linear relationships, motivating tree-based models.

- Predicted probabilities show good spread, supporting percentile-based risk banding.

Logistic Regression

```
In [15]: # Define what our target variable is and our predictors
X = df[behavior_features + capacity_features + demographic_features]
y = df["default"]
```

```
In [16]: # Train our data
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.25,
    random_state=42,
    stratify=y
)
```

```
In [17]: # Scale the data based on our predictor variables
scale_features = [
    "credit_limit", "age",
    "bill_sep", "bill_aug", "bill_jul", "bill_jun", "bill_may", "bill_apr",
    "paid_sep", "paid_aug", "paid_jul", "paid_jun", "paid_may", "paid_apr"
]

scaler = StandardScaler()

X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()

X_train_scaled[scale_features] = scaler.fit_transform(X_train[scale_features])
X_test_scaled[scale_features] = scaler.transform(X_test[scale_features])
```

```
In [18]: # Run our logistic regression and make it balanced because the default column has two classes
log_model = LogisticRegression(
    max_iter=1000,
    class_weight="balanced",
    random_state=42
)

log_model.fit(X_train_scaled, y_train)
```

Out[18]:

 LogisticRegression


```
LogisticRegression(class_weight='balanced', max_iter=1000, random_state=42)
```

```
In [19]: # Get the predictions in probability format so that we get the risk for each
# Get our AUC score for the model
```

```
y_pred_prob = log_model.predict_proba(X_test_scaled)[:,1]

roc_auc_score(y_test, y_pred_prob)
```

Out[19]: 0.7164013527454849

An AUC score of 0.72 indicates my model has fair or acceptable discriminatory ability, meaning it's better than random guessing (0.5) but not excellent, suggesting a 72% probability it can distinguish a random positive case from a random negative one.

In [20]: *# I am taking the X_test that we got when we trained the data, copying it, and then making arrays of the y_test and the y_pred_prob within the results. This attaches the predictions to the test set*

```
results = X_test.copy()
results["actual_default"] = y_test.values
results["default_prob"] = y_pred_prob
```

In [21]: *# I got a pretty good AUC score, so I want to move on to categorizing the risk. I want to have risk groups or bands that can tell whether the loan is low, medium, or high risk.*

```
results["risk_band"] = pd.qcut(
    results["default_prob"],
    q=[0, 0.3, 0.7, 1],
    labels=["Low Risk", "Medium Risk", "High Risk"]
)
```

In [22]: *# Get a summary of the default rate by the bands we created*

```
results.groupby("risk_band")["actual_default"].mean()
```

```
/var/folders/sn/16t8g_2d0ml3r257zrrn1k2m0000gn/T/ipykernel_41552/1397559019.py:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
```

```
results.groupby("risk_band")["actual_default"].mean()
```

Out[22]: risk_band

risk_band	actual_default
Low Risk	0.123556
Medium Risk	0.141333
High Risk	0.425333

Name: actual_default, dtype: float64

In [23]: *# Check the volume*

```
results["risk_band"].value_counts(normalize=True)
```

Out[23]: risk_band

risk_band	proportion
Medium Risk	0.4
Low Risk	0.3
High Risk	0.3

Name: proportion, dtype: float64

Using percentile-based risk bands creates a balanced approval policy. The Low Risk group shows a substantially lower default rate, while the High Risk group concentrates a large share of defaults, indicating effective risk segmentation.

```
In [24]: # I need to make a profit array that calculates the profit per borrower base
# I then create a summary table that tells us how much we are gonna profit c
results["profit"] = np.where(
    results["actual_default"] == 1,
    -5000,
    1000
)
results.groupby("risk_band") ["profit"].mean()
```

```
/var/folders/sn/16t8g_2d0ml3r257zrrn1k2m0000gn/T/ipykernel_41552/1605992384.
py:8: FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to retain
current behavior or observed=True to adopt the future default and silence th
is warning.
```

```
results.groupby("risk_band") ["profit"].mean()
```

```
Out[24]: risk_band
Low Risk      258.666667
Medium Risk   152.000000
High Risk     -1552.000000
Name: profit, dtype: float64
```

XGBoost Model

Now that we have had success with a logistic regression, I want to try an XGBoost model to see if I can get a better result.

```
In [25]: # We need to make all of the columns into numeric values so that they are al
feature_cols = behavior_features + capacity_features + demographic_features

X_train_num = X_train.copy()
X_test_num = X_test.copy()

X_train_num[feature_cols] = X_train_num[feature_cols].apply(pd.to_numeric, e
X_test_num[feature_cols] = X_test_num[feature_cols].apply(pd.to_numeric, err
```

```
In [26]: # no nulls
X_train_num[feature_cols].isnull().sum().sort_values(ascending=False).head(1)
```

```
Out[26]: pay_sep      0
pay_aug       0
education     0
gender        0
age           0
credit_limit  0
paid_apr      0
paid_may      0
paid_jun      0
paid_jul      0
dtype: int64
```

```
In [27]: y_train_num = y_train
y_test_num = y_test
```

```
In [28]: from xgboost import XGBClassifier

xgb_model = XGBClassifier(
    n_estimators=200,
    max_depth=4,
    learning_rate=0.05,
    subsample=0.8,
    colsample_bytree=0.8,
    objective="binary:logistic",
    eval_metric="auc",
    random_state=42
)

xgb_model.fit(X_train_num, y_train_num)
```

Out[28]:

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.8, device=None, early_stopping_
rounds=None,
              enable_categorical=False, eval_metric='auc', featu
re_types=None,
              feature_weights=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None)
```

```
In [29]: # Get the AUC score
y_pred_prob_xgb = xgb_model.predict_proba(X_test_num)[:, 1]
roc_auc_score(y_test_num, y_pred_prob_xgb)
```

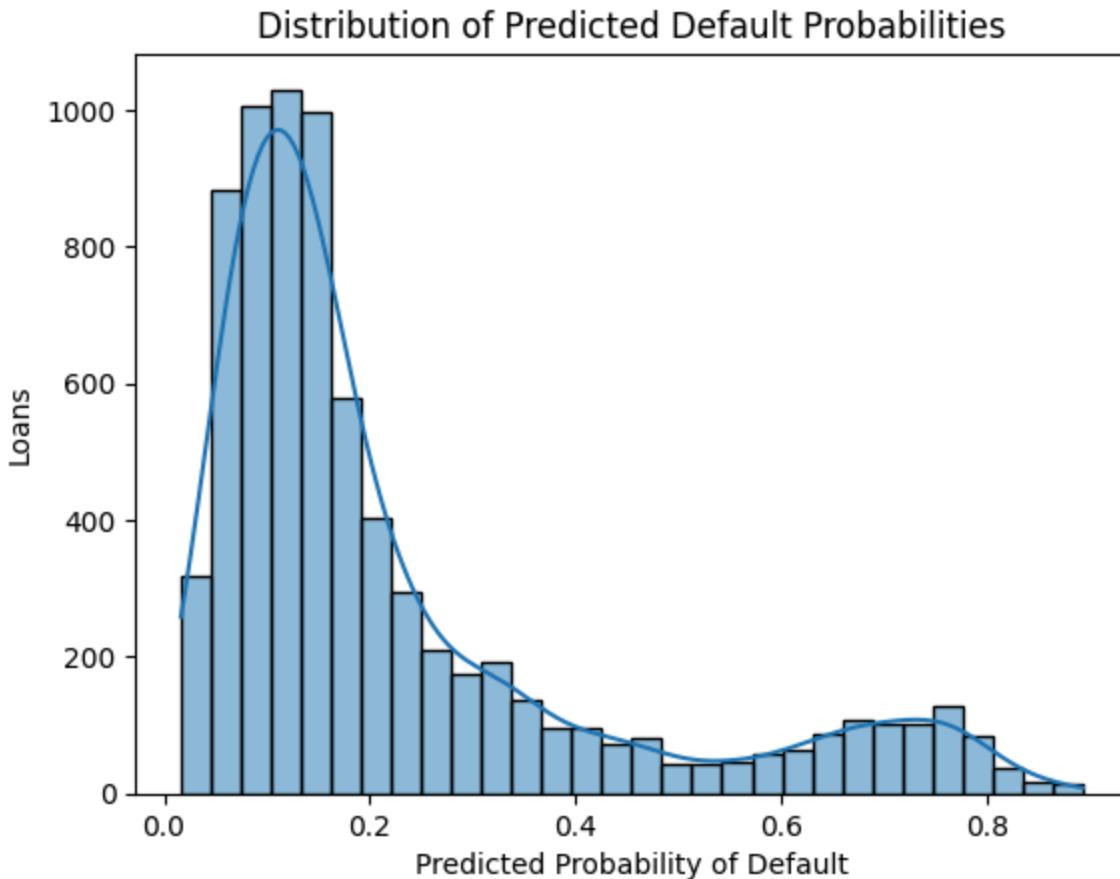
Out[29]: 0.7803783381985484

```
In [30]: # Rebuild the risk bands
results_xgb = X_test_num.copy()
results_xgb["actual_default"] = y_test_num.values
results_xgb["default_prob"] = y_pred_prob_xgb

results_xgb["risk_band"] = pd.qcut(
    results_xgb["default_prob"],
    q=[0, 0.3, 0.7, 1],
    labels=["Low Risk", "Medium Risk", "High Risk"]
)
```

```
In [31]: sns.histplot(results_xgb["default_prob"], bins=30, kde=True)
plt.title("Distribution of Predicted Default Probabilities")
plt.xlabel("Predicted Probability of Default")
```

```
plt.ylabel("Loans")
plt.show()
```



In [32]: `results_xgb.groupby("risk_band")["actual_default"].mean()`

```
/var/folders/sn/16t8g_2d0ml3r257zrrn1k2m000gn/T/ipykernel_41552/406419658.py:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
```

```
results_xgb.groupby("risk_band")["actual_default"].mean()
```

Out[32]: risk_band

Low Risk	0.068444
Medium Risk	0.152667
High Risk	0.465333
Name: actual_default, dtype: float64	

In [33]: `results_xgb["risk_band"].value_counts(normalize=True)`

Out[33]: risk_band

Medium Risk	0.4
Low Risk	0.3
High Risk	0.3
Name: proportion, dtype: float64	

In [34]: `results_xgb["profit"] = np.where(results_xgb["actual_default"] == 1, -5000,`

```
1000
)

results_xgb.groupby("risk_band")["profit"].mean()

/var/folders/sn/16t8g_2d0ml3r257zrrn1k2m0000gn/T/ipykernel_41552/538988647.p
y:7: FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to retain
current behavior or observed=True to adopt the future default and silence th
is warning.
results_xgb.groupby("risk_band")["profit"].mean()

Out[34]: risk_band
Low Risk      589.333333
Medium Risk    84.000000
High Risk     -1792.000000
Name: profit, dtype: float64
```

Final decision

Approval decisions are based on XGBoost-predicted default probabilities segmented into percentile-based risk bands. The model improves risk separation relative to logistic regression, leading to higher expected profitability under the same approval policy.

```
In [35]: results_xgb.to_csv("credit_risk_results.csv", index=False)
```