Credit Risk Analysis: Modeling

Elie Diwambuena,

June 23, 2023

Part 3. Modelling

In Part 2, we found significant differences in loan amount across gender, education levels and employment types. No other significant differences in income and loan amount across other groups were found. Additionally, we found some relationships between civil status and gender, and between dependencies and gender. However, this information is not sufficient to predict which customer is likely to default or not. This is what we intend to do in this Part 3.

In the following lines, we construction a classification model using the logit analysis. In section 1, we plot refine some categorical variables in order to reduce the number of dummy variables which often lead to multicolinearity among predictors. In section 2, we imporve the model by eliminating variables that provide the least contribution to the prediction power of the model. Then, we conclude in section 3 with som notes.

- **Importing the libraries and data**

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
import random as rd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import *
from sklearn.preprocessing import StandardScaler
import statsmodels.api as sm
import warnings
```

[2]:
```
data = pd.read_excel("loan_data.xlsx")
```

[3]:
```
data.dropna(inplace=True)
```

* **Encoding categorical variables and spitting the data into train and test set**

[4]:
```
data = data.drop(columns=["CustrNo"])
```

[5]:
```
response_y = data.DefaultHist.values
predictors_x = data[[x for x in data if x !="DefaultHist"]]
```

[6]:
```
response_y = pd.get_dummies(response_y)
predictors_x = pd.get_dummies(predictors_x)
```

[7]:
```
predictors_x.info()
response_y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1555 entries, 0 to 1594
Data columns (total 22 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Age                 1555 non-null   int64
 1   AppIncome           1555 non-null   int64
 2   CoAppIncome         1555 non-null   float64
 3   LoanAmt             1555 non-null   int64
 4   LoanTerm            1555 non-null   float64
 5   Prop_Value          1555 non-null   int64
 6   Gender_Female       1555 non-null   uint8
 7   Gender_Male         1555 non-null   uint8
 8   Civil_Divorced      1555 non-null   uint8
 9   Civil_Married       1555 non-null   uint8
 10  Civil_Unmarried     1555 non-null   uint8
 11  Civil_Widow         1555 non-null   uint8
 12  Dependents_0        1555 non-null   uint8
 13  Dependents_1        1555 non-null   uint8
 14  Dependents_2        1555 non-null   uint8
 15  Dependents_3+       1555 non-null   uint8
 16  Educ_Graduate       1555 non-null   uint8
 17  Educ_Not Graduate   1555 non-null   uint8
 18  Self-emp_No         1555 non-null   uint8
 19  Self-emp_Yes        1555 non-null   uint8
 20  OthDebts_No         1555 non-null   uint8
 21  OthDebts_Yes        1555 non-null   uint8
dtypes: float64(2), int64(4), uint8(16)
memory usage: 109.3 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1555 entries, 0 to 1554
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   No      1555 non-null   uint8
 1   Yes     1555 non-null   uint8
dtypes: uint8(2)
memory usage: 3.2 KB
```

[8]:
```
# removing the extra dummies
predictors_x = predictors_x.
 ↪drop(columns=["Civil_Divorced","Civil_Unmarried","Civil_Widow","Dependents_1","Dependents_2
predictors_x = predictors_x.drop(columns=["Gender_Female","Educ_Not␣
 ↪Graduate","Self-emp_No","OthDebts_No"])
response_y = response_y.loc[:,"Yes"]
```

[9]:
```
predictors_x.info()
response_y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1555 entries, 0 to 1594
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Age            1555 non-null   int64
 1   AppIncome      1555 non-null   int64
 2   CoAppIncome    1555 non-null   float64
 3   LoanAmt        1555 non-null   int64
 4   LoanTerm       1555 non-null   float64
 5   Prop_Value     1555 non-null   int64
 6   Gender_Male    1555 non-null   uint8
 7   Civil_Married  1555 non-null   uint8
 8   Dependents_0   1555 non-null   uint8
 9   Educ_Graduate  1555 non-null   uint8
 10  Self-emp_Yes   1555 non-null   uint8
 11  OthDebts_Yes   1555 non-null   uint8
dtypes: float64(2), int64(4), uint8(6)
memory usage: 94.2 KB
<class 'pandas.core.series.Series'>
RangeIndex: 1555 entries, 0 to 1554
Series name: Yes
Non-Null Count  Dtype
--------------  -----
1555 non-null   uint8
dtypes: uint8(1)
memory usage: 1.6 KB
```

[10]:
```
X_train, X_test, y_train, y_test = train_test_split(predictors_x,␣
 ↪response_y,test_size=0.2, train_size=0.8)
```

1. Logistic Regression

To use a logistic regression, the following conditions should be met:

1. The dependent variable should be binary or dichotomous.

2. Independence of observations: Each data point should represent an independent sample or observation.
3. Linearity of predictors: The relationship between the independent variables (predictors) and the log-odds of the outcome variable should be linear. This assumption can be checked by examining plots of the predictors against the log-odds.
4. No multicollinearity: There should be little or no multicollinearity among the independent variables.
5. Absence of influential outliers: Extreme outliers or influential data points can have a significant impact on the estimated coefficients and the overall model.
6. Sufficient sample size: Logistic regression typically requires a sufficient sample size to ensure reliable estimation of coefficients and accurate inference. As a rule of thumb, it is recommended to have at least ten events (instances of the outcome) per predictor to avoid issues with overfitting.

The two first conditons are already satisfied. And we have verified conditions 4,5 and 6 in Part 1 and Part 2. However, we still need to test the second conditon. We will do it towards the end after having computed the log-odds.

We observe multicollinearity among variables related to having dependents and civil status in Part 1. More specifically, there is a relatively high correlation between married and divorced, married and unmarried, depend0 and depend1, and depend0 and depend3+. This is not a suprise since does variable come from the same variable. And this is also knwon as the **dummy variable trap**. We can reduce the number of dummy variables by redefining categories with more than 2 categories. Therefore, we redefine Civil and Dependents variables. For Civil, we redefine as married or not married nd for dependents we redefine as dependent or not dependents.

We also observe the pressence of influential outliers in income and loan amount. We can handle this in two ways: (1) by winsorizing the data or (2) by standardizing the data. Either approach are good at reducing the influence of outliers bu the second one is preferred when predictors are of different scale and unit of measure. Therefore, we choose to standardize the data.

* **Feature scaling**

[11]:
```
std_scale = StandardScaler()
```

[12]:
```
X_train = std_scale.fit_transform(X_train)
X_test = std_scale.fit_transform(X_test)
```

* **Fitting the models**

[13]:
```
lr_reg = LogisticRegression()
```

[14]:
```
lr_reg.fit(X=X_train, y=np.ravel(y_train))
```

[14]: LogisticRegression()

[15]:
```
# first trial
obs_num = rd.randint(0,len(X_train))
obs_x = X_train[obs_num,:]
```

[16]:
```
y_train.iloc[obs_num]
```

[16]: 0

[17]:
```
obs_x = np.array(obs_x)
lr_reg.predict([obs_x])
```

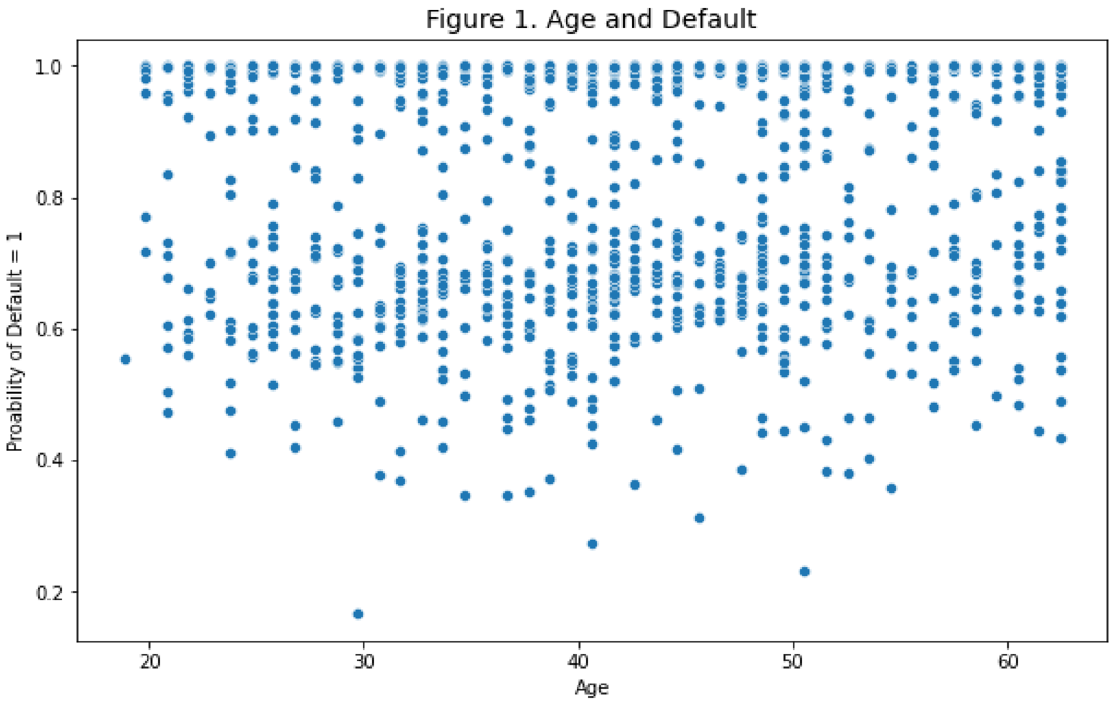
[17]: array([0], dtype=uint8)

* **Testing for linearity**

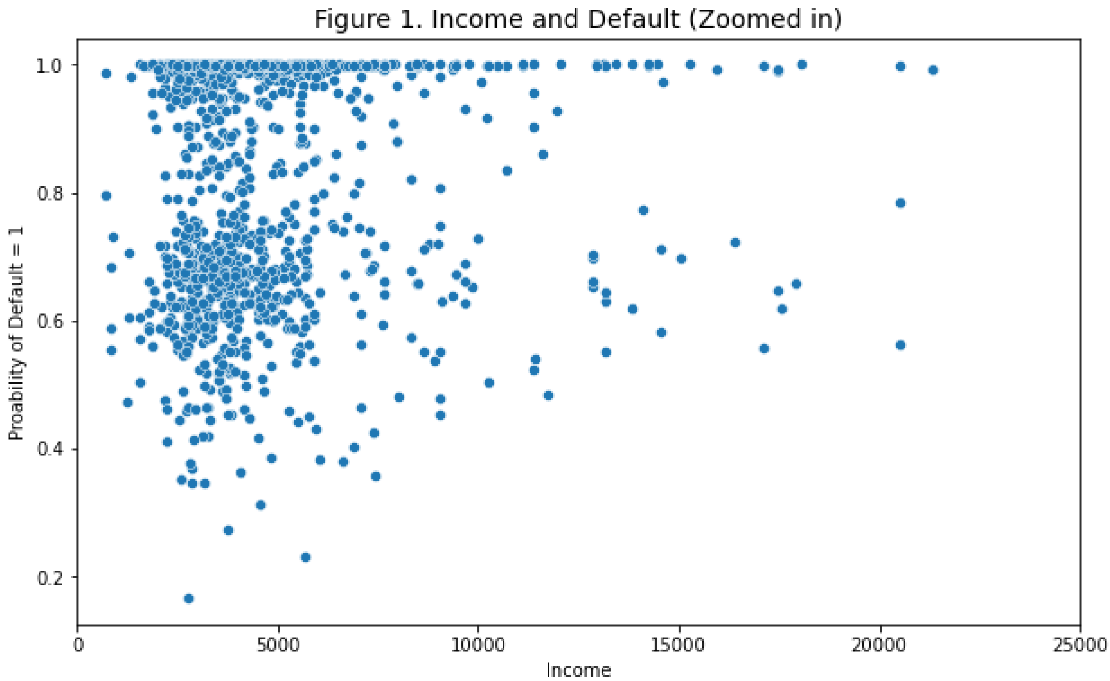
[18]:
```
pred_y = lr_reg.predict_proba(X_train)
```

[19]:
```
X_train = std_scale.inverse_transform(X_train)
```

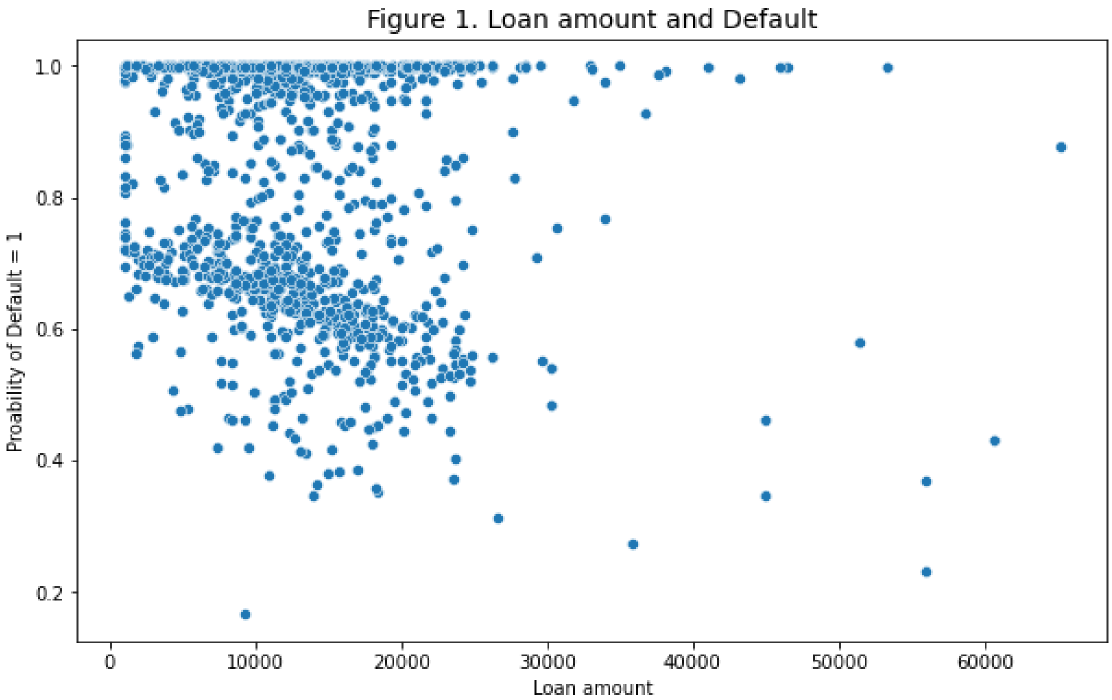
[20]:
```
# Age and Default
plt.figure(figsize=(10,6))
sb.scatterplot(y=pred_y[:,0], x=X_train[:,0])
plt.title("Figure 1. Age and Default", fontsize=14)
plt.xlabel("Age")
plt.ylabel("Proability of Default = 1")
plt.show()
```



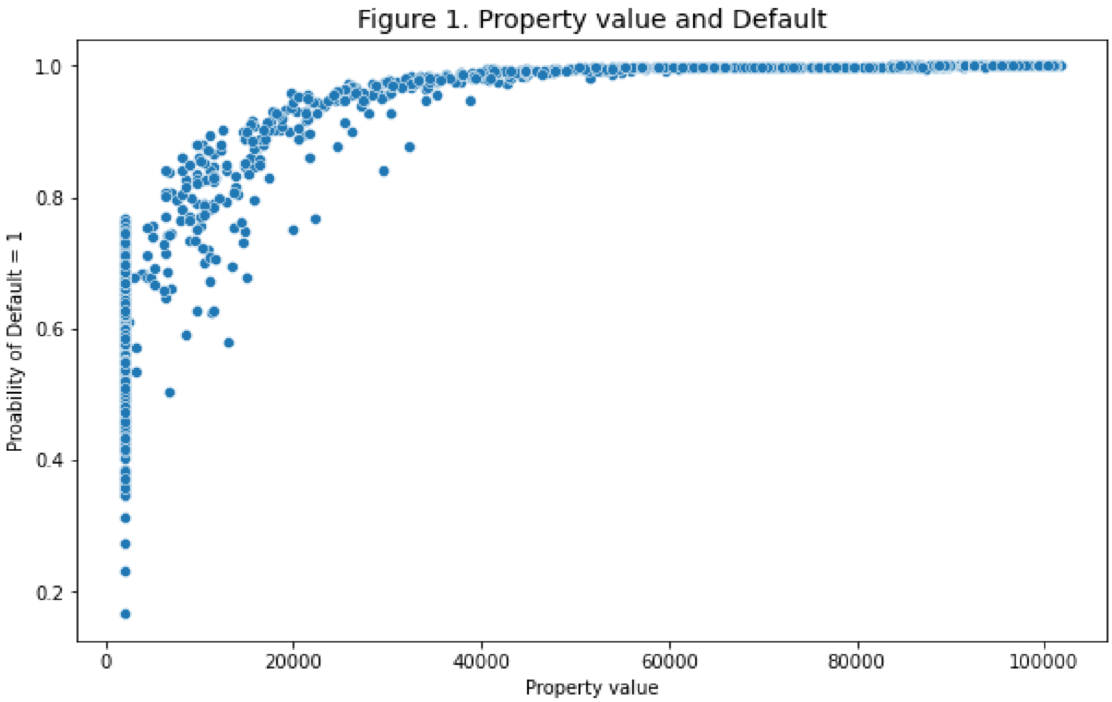
[21]:
```
plt.figure(figsize=(10,6))
sb.scatterplot(y=pred_y[:,0], x=X_train[:,1])
plt.title("Figure 1. Income and Default (Zoomed in)", fontsize=14)
plt.xlabel("Income")
plt.ylabel("Proability of Default = 1")
plt.xlim(0,25000)
plt.show()
```



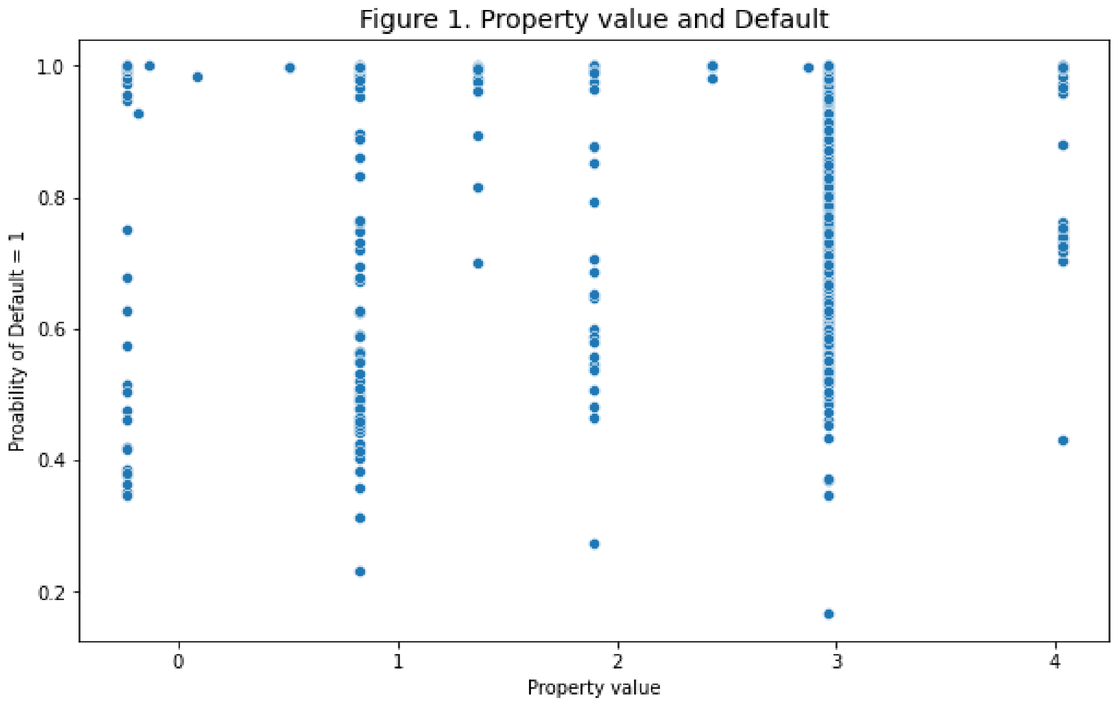
[22]:
```
plt.figure(figsize=(10,6))
sb.scatterplot(y=pred_y[:,0], x=X_train[:,3])
plt.title("Figure 1. Loan amount and Default", fontsize=14)
plt.xlabel("Loan amount")
plt.ylabel("Proability of Default = 1")
plt.show()
```



[23]:
```
plt.figure(figsize=(10,6))
sb.scatterplot(y=pred_y[:,0], x=X_train[:,5])
plt.title("Figure 1. Property value and Default", fontsize=14)
plt.xlabel("Property value")
plt.ylabel("Proability of Default = 1")
plt.show()
```



[24]:
```
plt.figure(figsize=(10,6))
sb.scatterplot(y=pred_y[:,0], x=X_train[:,4])
plt.title("Figure 1. Property value and Default", fontsize=14)
plt.xlabel("Property value")
plt.ylabel("Proability of Default = 1")
plt.show()
```



[25]:
```
df = pd.DataFrame(response_y,columns=["Yes"])
df.assign(count=df.Yes).groupby(by=["Yes"]).count()
```

[25]:

	count
Yes	
0	1292
1	263

When plotting the log-odds of the response variable being 1 for default against continuous predictors such as income, loan amount, age, property value and term periods, we find no clear linearity except for property value. Nonetheless, loan amount shows a slight negative linaar relationship with default. That is, it looks like the probability of defaulting decreases as the loan amount increases. Income shows few relationship with failure which is quite surprising.

Besides linearity issue, the sample size is quite large. It is higher than the recommended size of 110 (11 * 11 variables) per class. We have 1292 no-default and 263 yes-default classes. And, we have handled the extreme outliers by standardizing features.

* **Model performance**

[59]:
```
df = pd.DataFrame(y_test,columns=["Yes"])
df.assign(count=df.Yes).groupby(by=["Yes"]).count()
```

[59]:
```
      count
Yes
0       256
1        55
```

[26]:
```
pred_y = lr_reg.predict(X_test)
true_y = y_test
```

[27]:
```
confusion_matrix(true_y, pred_y, labels=[1,0])
```

[27]:
```
array([[ 11,  44],
       [  8, 248]])
```

[28]:
```
accuracy = round(accuracy_score(true_y, pred_y),4)*100
precision = round(precision_score(true_y, pred_y),4)*100
sensitivity = round(recall_score(true_y,pred_y),4)*100
print(f"Accuracy rate of {accuracy:.2f}%, precision rate of {precision:.2f}%␣
 ↪and sensitivity rate of {sensitivity:.2f}%")
```

```
Accuracy rate of 83.28%, precision rate of 57.89% and sensitivity rate of 20.00%
```

The model yields a very low accuracy. About 20% of variables in general were wrongly classify which we think is not desirable. The precision or true postive rate is about 50%. This means that out of the 100 individuals predicted as defaulted, only 50 were actually correct. Moreover, the sensitivity rate is about 15% which means out of the 100 actual defaulted individuals only 15 were correctly classified. This suggests that this model is suboptimal and that it should be improved. There are different ways we can improve it: (1)finding additional variables which can affect the likelihood of default or (2)eliminating variables that do not make a big contribution to the model.

The first option is of course time consuming and maybe not possible at all. Thus, we will implement the second option and eliminate unecessary predictors. We start by removing (1) age, (2)gender, (3)civil, (4)education, (5) self-employed and (6) loan amount. This is because this coefficients are not significantly different from zero in their contribution to the prediction of default likelihood. Put differently, these variables do not explain the change in default likelihood as shown in the regression results summary below.

[29]:
```
X_train = pd.DataFrame(std_scale.
 ↪fit_transform(X_train),columns=list(predictors_x.columns))
X_with_const = sm.add_constant(X_train)
```

[30]:
```
warnings.filterwarnings("ignore")
res = sm.Logit(y_train.reset_index().iloc[:,-1], X_with_const).fit()
```

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 10
```

[31]:
```
print(res.summary())
```

Logit Regression Results

Dep. Variable:	Yes	No. Observations:	1244
Model:	Logit	Df Residuals:	1231
Method:	MLE	Df Model:	12
Date:	Fri, 23 Jun 2023	Pseudo R-squ.:	inf
Time:	20:56:02	Log-Likelihood:	-inf
converged:	True	LL-Null:	0.0000
Covariance Type:	nonrobust	LLR p-value:	1.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-4.7567	0.534	-8.906	0.000	-5.804	-3.710
Age	-0.0898	0.095	-0.942	0.346	-0.277	0.097
AppIncome	0.0781	0.082	0.955	0.340	-0.082	0.238
CoAppIncome	0.1838	0.108	1.699	0.089	-0.028	0.396
LoanAmt	0.2032	0.091	2.229	0.026	0.025	0.382
LoanTerm	-0.2810	0.089	-3.158	0.002	-0.455	-0.107
Prop_Value	-4.4673	0.578	-7.733	0.000	-5.600	-3.335
Gender_Male	0.0628	0.100	0.629	0.530	-0.133	0.259
Civil_Married	-0.0115	0.095	-0.122	0.903	-0.198	0.175
Dependents_0	-0.0613	0.092	-0.665	0.506	-0.242	0.119
Educ_Graduate	0.0561	0.090	0.626	0.531	-0.120	0.232
Self-emp_Yes	0.0360	0.095	0.378	0.705	-0.150	0.222
OthDebts_Yes	0.0111	0.090	0.124	0.902	-0.165	0.187

Possibly complete quasi-separation: A fraction 0.24 of observations can be perfectly predicted. This might indicate that there is complete quasi-separation. In this case some parameters will not be identified.

* **Backward elimination**

Removing age

[32]:
```
X_train = X_train.iloc[:,1:]
```

[33]:
```
lr_reg.fit(X=X_train, y=np.ravel(y_train))
```

[33]:
```
LogisticRegression()
```

[34]:
```
X_test = pd.DataFrame(X_test,columns=list(predictors_x.columns))
X_test = X_test.iloc[:,1:]
```

[35]:
```
pred_y = lr_reg.predict(X_test)
true_y = y_test
```

[36]:
```
confusion_matrix(true_y, pred_y, labels=[1,0])
```

[36]:
```
array([[ 11,  44],
       [  8, 248]])
```

[37]:
```
X_with_const =sm.add_constant(X_train)
res = sm.Logit(y_train.reset_index().iloc[:,-1], X_with_const).fit()
print(res.summary())
```

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 10
                           Logit Regression Results
==============================================================================
Dep. Variable:                    Yes   No. Observations:                 1244
Model:                          Logit   Df Residuals:                     1232
Method:                           MLE   Df Model:                           11
Date:                Fri, 23 Jun 2023   Pseudo R-squ.:                     inf
Time:                        20:56:02   Log-Likelihood:                   -inf
converged:                       True   LL-Null:                        0.0000
Covariance Type:            nonrobust   LLR p-value:                     1.000
================================================================================
=
                   coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const           -4.7603      0.535     -8.905      0.000      -5.808
-3.713
AppIncome        0.0581      0.079      0.735      0.462      -0.097
```

```
0.213
CoAppIncome        0.1823      0.107       1.699      0.089      -0.028
0.393
LoanAmt            0.2029      0.091       2.224      0.026       0.024
0.382
LoanTerm          -0.2737      0.088      -3.092      0.002      -0.447
-0.100
Prop_Value        -4.4756      0.578      -7.741      0.000      -5.609
-3.342
Gender_Male        0.0667      0.100       0.669      0.504      -0.129
0.262
Civil_Married     -0.0108      0.095      -0.114      0.909      -0.197
0.175
Dependents_0      -0.0610      0.092      -0.662      0.508      -0.242
0.120
Educ_Graduate      0.0533      0.089       0.596      0.551      -0.122
0.229
Self-emp_Yes       0.0348      0.095       0.366      0.714      -0.151
0.221
OthDebts_Yes       0.0128      0.090       0.143      0.887      -0.163
0.188
================================================================================
=

Possibly complete quasi-separation: A fraction 0.24 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

[38]:

```
accuracy = round(accuracy_score(true_y, pred_y),4)*100
precision = round(precision_score(true_y, pred_y),4)*100
sensitivity = round(recall_score(true_y,pred_y),4)*100
print(f"Accuracy rate of {accuracy:.2f}%, precision rate of {precision:.2f}% and sensitivity rate of {sensitivity:.2f}%")
```

```
Accuracy rate of 83.28%, precision rate of 57.89% and sensitivity rate of 20.00%
```

Removing age seems not to be beneficial to the model as the performance actually worsen instead of improving. Nonetheless, we move to the elimination of other variables and postpone this discussion until the end.

Removing gender

[39]:

```
X_train = X_train.iloc[:,[0,1,2,3,4,6,7,8,9,10]]
X_test = X_test.iloc[:,[0,1,2,3,4,6,7,8,9,10]]
lr_reg.fit(X=X_train, y=np.ravel(y_train))
pred_y = lr_reg.predict(X_test)
true_y = y_test
```

[40]:

```
confusion_matrix(true_y, pred_y, labels=[1,0])
```

[40]:
```
array([[ 10,  45],
       [ 10, 246]])
```

[41]:
```
accuracy = round(accuracy_score(true_y, pred_y),4)*100
precision = round(precision_score(true_y, pred_y),4)*100
sensitivity = round(recall_score(true_y,pred_y),4)*100
print(f"Accuracy rate of {accuracy:.2f}%, precision rate of {precision:.2f}%␣
 ↪and sensitivity rate of {sensitivity:.2f}%")
```

```
Accuracy rate of 82.32%, precision rate of 50.00% and sensitivity rate of 18.18%
```

[42]:
```
X_with_const =sm.add_constant(X_train)
res = sm.Logit(y_train.reset_index().iloc[:,-1], X_with_const).fit()
print(res.summary())
```

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 10
                           Logit Regression Results
==============================================================================
Dep. Variable:                    Yes   No. Observations:                 1244
Model:                          Logit   Df Residuals:                     1233
Method:                           MLE   Df Model:                           10
Date:                Fri, 23 Jun 2023   Pseudo R-squ.:                     inf
Time:                        20:56:02   Log-Likelihood:                   -inf
converged:                       True   LL-Null:                        0.0000
Covariance Type:            nonrobust   LLR p-value:                     1.000
================================================================================
=
                    coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const            -4.7690      0.535     -8.919      0.000      -5.817
-3.721
AppIncome         0.0594      0.079      0.751      0.452      -0.095
0.214
CoAppIncome       0.1895      0.107      1.772      0.076      -0.020
0.399
LoanAmt           0.2076      0.091      2.279      0.023       0.029
0.386
LoanTerm         -0.2776      0.088     -3.141      0.002      -0.451
-0.104
Prop_Value       -4.4891      0.578     -7.764      0.000      -5.622
-3.356
Civil_Married     0.0025      0.093      0.027      0.978      -0.179
0.184
Dependents_0     -0.0680      0.091     -0.744      0.457      -0.247
```

```
0.111
Educ_Graduate      0.0514       0.089       0.575       0.565       -0.124
0.227
Self-emp_Yes       0.0346       0.095       0.363       0.716       -0.152
0.221
OthDebts_Yes       0.0141       0.090       0.158       0.875       -0.161
0.190
================================================================================
=

Possibly complete quasi-separation: A fraction 0.25 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

Removing civil status

[43]:
```python
X_train = X_train.iloc[:,[0,1,2,3,4,6,7,8,9]]
X_test = X_test.iloc[:,[0,1,2,3,4,6,7,8,9]]
lr_reg.fit(X=X_train, y=np.ravel(y_train))
pred_y = lr_reg.predict(X_test)
true_y = y_test
```

[44]:
```python
confusion_matrix(true_y, pred_y, labels=[1,0])
```

[44]:
```
array([[ 10,  45],
       [ 10, 246]])
```

[45]:
```python
accuracy = round(accuracy_score(true_y, pred_y),4)*100
precision = round(precision_score(true_y, pred_y),4)*100
sensitivity = round(recall_score(true_y,pred_y),4)*100
print(f"Accuracy rate of {accuracy:.2f}%, precision rate of {precision:.2f}%␣
 ↪and sensitivity rate of {sensitivity:.2f}%")
```

```
Accuracy rate of 82.32%, precision rate of 50.00% and sensitivity rate of 18.18%
```

[46]:
```python
X_with_const =sm.add_constant(X_train)
res = sm.Logit(y_train.reset_index().iloc[:,-1], X_with_const).fit()
print(res.summary())
```

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 10
                           Logit Regression Results
==============================================================================
Dep. Variable:                    Yes   No. Observations:                 1244
Model:                          Logit   Df Residuals:                     1234
Method:                           MLE   Df Model:                            9
Date:                Fri, 23 Jun 2023   Pseudo R-squ.:                     inf
Time:                        20:56:02   Log-Likelihood:                   -inf
```

```
converged:                       True   LL-Null:                        0.0000
Covariance Type:            nonrobust   LLR p-value:                     1.000
================================================================================
=
                    coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const            -4.7692      0.535     -8.920      0.000      -5.817
-3.721
AppIncome         0.0594      0.079      0.752      0.452      -0.095
0.214
CoAppIncome       0.1897      0.107      1.777      0.076      -0.020
0.399
LoanAmt           0.2076      0.091      2.279      0.023       0.029
0.386
LoanTerm         -0.2779      0.088     -3.157      0.002      -0.450
-0.105
Prop_Value       -4.4893      0.578     -7.765      0.000      -5.622
-3.356
Dependents_0     -0.0685      0.089     -0.766      0.443      -0.244
0.107
Educ_Graduate     0.0514      0.089      0.575      0.565      -0.124
0.226
Self-emp_Yes      0.0345      0.095      0.363      0.717      -0.152
0.221
OthDebts_Yes      0.0140      0.089      0.156      0.876      -0.161
0.189
================================================================================
=

Possibly complete quasi-separation: A fraction 0.25 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

Removing dependents

[47]:
```
X_train = X_train.iloc[:,[0,1,2,3,4,6,7,8]]
X_test = X_test.iloc[:,[0,1,2,3,4,6,7,8]]
lr_reg.fit(X=X_train, y=np.ravel(y_train))
pred_y = lr_reg.predict(X_test)
true_y = y_test
```

[48]:
```
confusion_matrix(true_y, pred_y, labels=[1,0])
```

[48]:
```
array([[ 11,  44],
       [ 10, 246]])
```

[49]:
```
accuracy = round(accuracy_score(true_y, pred_y),4)*100
precision = round(precision_score(true_y, pred_y),4)*100
sensitivity = round(recall_score(true_y,pred_y),4)*100
print(f"Accuracy rate of {accuracy:.2f}%, precision rate of {precision:.2f}%␣
↪and sensitivity rate of {sensitivity:.2f}%")
```

```
Accuracy rate of 82.64%, precision rate of 52.38% and sensitivity rate of 20.00%
```

[50]:
```
X_with_const =sm.add_constant(X_train)
res = sm.Logit(y_train.reset_index().iloc[:,-1], X_with_const).fit()
print(res.summary())
```

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 10
                           Logit Regression Results
==============================================================================
Dep. Variable:                    Yes   No. Observations:                 1244
Model:                          Logit   Df Residuals:                     1235
Method:                           MLE   Df Model:                            8
Date:                Fri, 23 Jun 2023   Pseudo R-squ.:                     inf
Time:                        20:56:02   Log-Likelihood:                   -inf
converged:                       True   LL-Null:                        0.0000
Covariance Type:            nonrobust   LLR p-value:                     1.000
================================================================================
=
                    coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const            -4.7592      0.534     -8.905      0.000      -5.807
-3.712
AppIncome         0.0604      0.079      0.763      0.445      -0.095
0.215
CoAppIncome       0.1876      0.107      1.753      0.080      -0.022
0.397
LoanAmt           0.2145      0.091      2.364      0.018       0.037
0.392
LoanTerm         -0.2789      0.088     -3.175      0.001      -0.451
-0.107
Prop_Value       -4.4781      0.578     -7.749      0.000      -5.611
-3.345
Educ_Graduate     0.0496      0.089      0.556      0.578      -0.125
0.224
Self-emp_Yes      0.0368      0.095      0.388      0.698      -0.149
0.223
OthDebts_Yes      0.0188      0.089      0.211      0.833      -0.156
0.193
```

```
==============================================================================
=

Possibly complete quasi-separation: A fraction 0.24 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

Removing education

```
[51]: X_train = X_train.iloc[:,[0,1,2,3,4,6,7]]
      X_test = X_test.iloc[:,[0,1,2,3,4,6,7]]
      lr_reg.fit(X=X_train, y=np.ravel(y_train))
      pred_y = lr_reg.predict(X_test)
      true_y = y_test
```

```
[52]: confusion_matrix(true_y, pred_y, labels=[1,0])
```

```
[52]: array([[ 10,  45],
             [ 10, 246]])
```

```
[53]: accuracy = round(accuracy_score(true_y, pred_y),4)*100
      precision = round(precision_score(true_y, pred_y),4)*100
      sensitivity = round(recall_score(true_y,pred_y),4)*100
      print(f"Accuracy rate of {accuracy:.2f}%, precision rate of {precision:.2f}%␣
       ↪and sensitivity rate of {sensitivity:.2f}%")
```

```
Accuracy rate of 82.32%, precision rate of 50.00% and sensitivity rate of 18.18%
```

```
[54]: X_with_const =sm.add_constant(X_train)
      res = sm.Logit(y_train.reset_index().iloc[:,-1], X_with_const).fit()
      print(res.summary())
```

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 10
                           Logit Regression Results
==============================================================================
Dep. Variable:                    Yes   No. Observations:                 1244
Model:                          Logit   Df Residuals:                     1236
Method:                           MLE   Df Model:                            7
Date:                Fri, 23 Jun 2023   Pseudo R-squ.:                     inf
Time:                        20:56:02   Log-Likelihood:                   -inf
converged:                       True   LL-Null:                        0.0000
Covariance Type:            nonrobust   LLR p-value:                     1.000
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const           -4.7650      0.535     -8.913      0.000      -5.813      -3.717
AppIncome        0.0580      0.079      0.731      0.465      -0.097       0.213
```

```
CoAppIncome       0.1873      0.107      1.753      0.080      -0.022       0.397
LoanAmt           0.2206      0.090      2.448      0.014       0.044       0.397
LoanTerm         -0.2739      0.087     -3.135      0.002      -0.445      -0.103
Prop_Value       -4.4835      0.578     -7.755      0.000      -5.617      -3.350
Self-emp_Yes      0.0364      0.095      0.383      0.701      -0.150       0.222
OthDebts_Yes      0.0172      0.089      0.193      0.847      -0.157       0.192
================================================================================

Possibly complete quasi-separation: A fraction 0.24 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

Removing self-employed

[55]:
```
X_train = X_train.iloc[:,[0,1,2,3,4,6]]
X_test = X_test.iloc[:,[0,1,2,3,4,6]]
lr_reg.fit(X=X_train, y=np.ravel(y_train))
pred_y = lr_reg.predict(X_test)
true_y = y_test
```

[56]:
```
confusion_matrix(true_y, pred_y, labels=[1,0])
```

[56]:
```
array([[ 10,  45],
       [ 10, 246]])
```

[57]:
```
accuracy = round(accuracy_score(true_y, pred_y),4)*100
precision = round(precision_score(true_y, pred_y),4)*100
sensitivity = round(recall_score(true_y,pred_y),4)*100
print(f"Accuracy rate of {accuracy:.2f}%, precision rate of {precision:.2f}%␣
 ↪and sensitivity rate of {sensitivity:.2f}%")
```

```
Accuracy rate of 82.32%, precision rate of 50.00% and sensitivity rate of 18.18%
```

[58]:
```
X_with_const =sm.add_constant(X_train)
res = sm.Logit(y_train.reset_index().iloc[:,-1], X_with_const).fit()
print(res.summary())
```

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 10
                           Logit Regression Results
==============================================================================
Dep. Variable:                    Yes   No. Observations:                 1244
Model:                          Logit   Df Residuals:                     1237
Method:                           MLE   Df Model:                            6
Date:                Fri, 23 Jun 2023   Pseudo R-squ.:                     inf
Time:                        20:56:02   Log-Likelihood:                   -inf
converged:                       True   LL-Null:                        0.0000
Covariance Type:            nonrobust   LLR p-value:                     1.000
```

```
==================================================================================
                  coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
const          -4.7511      0.532     -8.932      0.000      -5.794      -3.709
AppIncome       0.0576      0.079      0.726      0.468      -0.098       0.213
CoAppIncome     0.1863      0.107      1.742      0.082      -0.023       0.396
LoanAmt         0.2258      0.089      2.540      0.011       0.052       0.400
LoanTerm       -0.2752      0.087     -3.153      0.002      -0.446      -0.104
Prop_Value     -4.4669      0.575     -7.770      0.000      -5.594      -3.340
OthDebts_Yes    0.0168      0.089      0.189      0.850      -0.158       0.191
==================================================================================

Possibly complete quasi-separation: A fraction 0.24 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

3. Conclusion

In general, removing variables does not improve significantly the performance of the model. However, there are some minimal improve in metrics such sensitivity or precision rate. But the improvement in one metric is sometimes offset by the degradation of the other. Therefore, on average backward elimination seems not to make the model any better.

It is important to note that property value, loan amount and loan term have been consistently significant in all models. This suggests that these variables better explain the change in default likelihood among our customers than any other variable. This is not very surprising as loan amount is often linked to loan term. And having a collateral can reduce moral hazard and motivate individuals to pay back their loans.

[ ]:

```
```