# Introduction to

# Binary Logistic Regression

# using Python

# Contents

# Binary Logistic Regression

| DEPENDENT VARIABLE | INDEPENDENT VARIABLE |
| --- | --- |
| ↓ | ↓ |
| Categorical (Binary) | Categorical or Continuous |
| 0 or 1 | Gender |
| Death or Survival | Geographical Regions |
| Absence or Presence | Age |
| | Income |
| | Debt Ratio |
| ⋮ | ⋮ |

Binary logistic regression **models the dependent variable as a logit of p, where p is the probability that dependent variable takes the value 1 or 0**

# Statistical Model – For k Predictors

$$\log\left(\frac{p}{1-p}\right) = b_0 + b_1X_1 + \cdots + b_kX_k$$

where,

| | | |
|---|---|---|
| p | : | Probability that Y=1 given X |
| Y | : | Dependent Variable |
| $X_1, X_2, \ldots, X_k$ | : | Independent Variables |
| $b_0, b_1, \ldots, b_k$ | : | Parameters of Model |

Note that **LHS of the model can lie between - ∞ to ∞**

Parameters of the model are estimated by Maximum Likelihood Method

# Case Study – Modeling Loan Defaults

## Background

- A bank possesses demographic and transactional data of its loan customers. If the bank has a model to predict defaulters it can help in loan disbursal decision making.

## Objective

- To predict whether the customer applying for the loan will be a defaulter or not.

## Available Information

- Sample size is 700
- Independent Variables: Age group, Years at current address, Years at current employer, Debt to Income Ratio, Credit Card Debts, Other Debts. The information on predictors was collected at the time of loan application process.
- Dependent Variable: Defaulter (=1 if defaulter ,0 otherwise). The status is observed after loan is disbursed.

# Data Snapshot

Independent Variables

Dependent Variable

| SN | AGE | EMPLOY | ADDRESS | DEBTINC | CREDDEBT | OTHDEBT | DEFAULTE |
|---|---|---|---|---|---|---|---|
| 1 | 3 | 17 | 12 | 9.3 | 11.36 | 5.01 | 1 |
| 2 | 1 | 10 | 6 | 17.3 | 1.36 | 4 | 0 |

| Column | Description | Type | Measurement | Possible Values |
|---|---|---|---|---|
| SN | Serial Number | numeric | - | - |
| AGE | Age Groups | Categorical | 1(<28 years),<br>2(28-40 years),<br>3(>40 years) | 3 |
| EMPLOY | Number of years customer working at current employer | Continuous | - | Positive value |
| ADDRESS | Number of years customer staying at current address | Continuous | - | Positive value |
| DEBTINC | Debt to Income Ratio | Continuous | - | Positive value |
| CREDDEBT | Credit to Debit Ratio | Continuous | - | Positive value |
| OTHDEBT | Other Debt | Continuous | - | Positive value |
| DEFAULTER | Whether customer defaulted on loan | Binary | 1(Defaulter),<br>0(Non-Defaulter) | 2 |

# Binary Logistic Regression in Python

```
# Import data and check data structure before running model
```

```python
import pandas as pd
bankloan=pd.read_csv('BANK LOAN.csv')

bankloan.info()
```

```
# Output:
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 8 columns):
SN           700 non-null int64
AGE          700 non-null int64
EMPLOY       700 non-null int64
ADDRESS      700 non-null int64
DEBTINC      700 non-null float64
CREDDEBT     700 non-null float64
OTHDEBT      700 non-null float64
DEFAULTER    700 non-null int64
dtypes: float64(3), int64(5)
memory usage: 43.8 KB
```

# Binary Logistic Regression in Python

```
# Change ‘AGE’ variable into categorical
bankloan['AGE']=bankloan['AGE'].astype('category')

bankloan.info()
```

Age is an integer and we need to convert it into type “category” for modeling purposes.

```
# Output:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 8 columns):
SN            700 non-null int64
AGE           700 non-null category
EMPLOY        700 non-null int64
ADDRESS       700 non-null int64
DEBTINC       700 non-null float64
CREDDEBT      700 non-null float64
OTHDEBT       700 non-null float64
DEFAULTER     700 non-null int64
dtypes: category(1), float64(3), int64(4)
memory usage: 39.1 KB
```

# Binary Logistic Regression in Python

```
# Logistic Regression using logit function
import statsmodels.formula.api as smf

riskmodel = smf.logit(formula = 'DEFAULTER ~ AGE + EMPLOY + ADDRESS + DEBTINC + CREDDEBT + OTHDEBT', data = bankloan).fit()
```

- **logit()** fits a logistic regression model to the data.

```
# Model summary

riskmodel.summary()
```

**summary()** generates detailed summary of the model.

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              DEFAULTER   No. Observations:                  700
Model:                          Logit   Df Residuals:                      692
Method:                           MLE   Df Model:                            7
Date:                Tue, 23 Mar 2021   Pseudo R-squ.:                  0.3120
Time:                        11:41:05   Log-Likelihood:                -276.70
converged:                       True   LL-Null:                       -402.18
Covariance Type:            nonrobust   LLR p-value:                 1.733e-50
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.7882      0.264     -2.985      0.003      -1.306      -0.271
C(AGE)[T.2]    0.2520      0.267      0.946      0.344      -0.270       0.774
C(AGE)[T.3]    0.6271      0.361      1.739      0.082      -0.080       1.334
EMPLOY        -0.2617      0.032     -8.211      0.000      -0.324      -0.199
ADDRESS       -0.0996      0.022     -4.459      0.000      -0.143      -0.056
DEBTINC        0.0851      0.022      3.845      0.000       0.042       0.128
CREDDEBT       0.5634      0.089      6.347      0.000       0.389       0.737
OTHDEBT        0.0231      0.057      0.405      0.685      -0.089       0.135
==============================================================================
```

**Interpretation :**

- Since p-value is <0.05 for Employ, Address, Debtinc, Creddebt, these independent variables are significant.

# Re-run Model in Python

- Re-run the model with employ, address, debtinc, creddebt.

```
riskmodel = smf.logit(formula = 'DEFAULTER ~  EMPLOY + ADDRESS +
DEBTINC + CREDDEBT', data = bankloan).fit()

riskmodel.summary()
```

# Re-run Model in Python

```
# Output:
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              DEFAULTER   No. Observations:                  700
Model:                          Logit   Df Residuals:                      695
Method:                           MLE   Df Model:                            4
Date:                Tue, 23 Mar 2021   Pseudo R-squ.:                  0.3079
Time:                        11:36:38   Log-Likelihood:                -278.37
converged:                       True   LL-Null:                       -402.18
Covariance Type:            nonrobust   LLR p-value:                 2.114e-52
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.7911      0.252     -3.145      0.002      -1.284      -0.298
EMPLOY        -0.2426      0.028     -8.646      0.000      -0.298      -0.188
ADDRESS       -0.0812      0.020     -4.144      0.000      -0.120      -0.043
DEBTINC        0.0883      0.019      4.760      0.000       0.052       0.125
CREDDEBT       0.5729      0.087      6.566      0.000       0.402       0.744
==============================================================================
```

**Interpretation :**

- Since p-value is <0.05 for Employ, Address, Debtinc and Creddebt, these independent variables are significant.

# Odds Ratios In Python

- Final Model is :

$$\log\left(\frac{p}{1-p}\right) = -0.79107 - 0.24258 * (\text{EMPLOY}) - 0.08122 * (\text{ADDRESS}) + 0.08827 * (\text{DEBTINC}) + 0.57290\ (\text{CREDDEBT})$$

- This model is used for predicting the probabilities.

```
import numpy as np
conf = riskmodel.conf_int()
conf['OR'] = riskmodel.params
conf.columns = ['2.5%', '97.5%', 'OR']
print(np.exp(conf))
```

- **conf_int():** calculates confidence intervals for parameters
- **riskmodel.params:** identify the model parameter estimates

# Odds Ratios in Python

```
# Output:
```

| | 2.5% | 97.5% | OR |
|---|---|---|---|
| Intercept | 0.276905 | 0.742255 | 0.453359 |
| EMPLOY | 0.742617 | 0.828950 | 0.784597 |
| ADDRESS | 0.887246 | 0.958093 | 0.921989 |
| DEBTINC | 1.053295 | 1.132703 | 1.092278 |
| CREDDEBT | 1.494635 | 2.104150 | 1.773397 |

**Interpretation :**

- Note that, confidence interval for odds ratio does not include '1' for all variables retained in the model. Which means that all of these variables are significant.
- The odds ratio for CREDDEBT is approximately 1.77
- For one unit change CREDDEBT, the odds of being a defaulter will change by 1.77 folds.

# Predicting Probabilities in Python

```
# Predicting Probabilities
```

```python
bankloan = bankloan.assign(pred=pd.Series(riskmodel.predict()))
bankloan.head(10)
```

- **predict()** function calculates predicted probabilities which are saved in the same dataset ‘bankloan’ under new column ‘pred’.

```
# Output:
```

| | SN | AGE | EMPLOY | ADDRESS | DEBTINC | CREDDEBT | OTHDEBT | DEFAULTER | pred |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 3 | 17 | 12 | 9.3 | 11.36 | 5.01 | 1 | 0.808346726 |
| 1 | 2 | 1 | 10 | 6 | 17.3 | 1.36 | 4 | 0 | 0.198114704 |
| 2 | 3 | 2 | 15 | 14 | 5.5 | 0.86 | 2.17 | 0 | 0.010062815 |
| 3 | 4 | 3 | 15 | 14 | 2.9 | 2.66 | 0.82 | 0 | 0.022159721 |
| 4 | 5 | 1 | 2 | 0 | 17.3 | 1.79 | 3.06 | 1 | 0.781808095 |
| 5 | 6 | 3 | 5 | 5 | 10.2 | 0.39 | 2.16 | 0 | 0.21646839 |
| 6 | 7 | 2 | 20 | 9 | 30.6 | 3.83 | 16.67 | 0 | 0.185631512 |
| 7 | 8 | 3 | 12 | 11 | 3.6 | 0.13 | 1.24 | 0 | 0.014726159 |
| 8 | 9 | 1 | 3 | 4 | 24.4 | 1.36 | 3.28 | 1 | 0.748212503 |
| 9 | 10 | 2 | 0 | 13 | 19.7 | 2.78 | 2.15 | 0 | 0.815255803 |

**Interpretation :**

- Last column ‘pred’ gives predicted probabilities.

# Classification Table

- Based on **cut-off value** of p, Y is estimated to be either 1 or 0

  Ex. $p>0.5$ ; $Y=1$

  $p \leq 0.5$ ; $Y=0$

- **Cross tabulation** of observed values of Y and predicted values of Y is called as **Classification Table.**
- The predictive success of the logistic regression can be assessed by looking at the classification table, but classification table is not a good measure of goodness fit since it **varies with the cut off value set.**
- Accuracy Rate measures **how accurate a model is in predicting outcomes.**
- In the adjoining table, 479 times Y=0 was observed as well as predicted. Similarly, Y=1 was observed and predicted 92 times.

  **Accuracy Rate = 478+92/700 = 81.43 %**

  Here **misclassification rate is : (39 +91) / 700=18.57%**

| | | Expected | |
|---|---|---|---|
| | | 0 | 1 |
| Observed | 0 | 478 | 39 |
| | 1 | 91 | 92 |

# Classification Table Terminology

| | |
|---|---|
| Sensitivity | % of occurrences correctly predicted<br>P(Ypred=1/Y=1) |
| Specificity | % of non occurrences correctly predicted<br>P(Ypred=0/Y=0) |
| False Positive Rate<br>(1 – Specificity) | % of non occurrences which are incorrectly predicted.<br>P(Ypred=1/Y=0) |
| False Negative Rate<br>(1- Sensitivity) | % of occurrences which are incorrectly predicted.<br>P(Ypred=0/Y=1) |

| | | Predicted | |
|---|---|---|---|
| | | 0 | 1 |
| Observed | 0 | **Specificity** | **False Positive (1-Specificity)** |
| | 1 | **False Negative (1-Sensitivity)** | **Sensitivity** |

# Sensitivity and Specificity calculations

| Cut-off Value | | Accuracy | Sensitivity | Specificity |
|---|---|---|---|---|
| 0.1 | FALSE TRUE<br>0 245 272<br>1 12 171 | (245+171)/700<br>= 59.43% | 171/183=93.4% | 245/517=47.4% |
| 0.2 | FALSE TRUE<br>0 349 168<br>1 26 157 | (349+157)/700<br>= 72.29% | 157/183=85.8% | 349/517=67.5% |
| 0.3 | FALSE TRUE<br>0 415 102<br>1 45 138 | (415+138)/700<br>= 84.71% | 138/183=75.4% | 415/517=80.3% |
| 0.4 | FALSE TRUE<br>0 447 70<br>1 69 114 | (447+114)/700<br>= 80.14% | 114/183=62.3% | 447/517=86.5% |
| 0.5 | FALSE TRUE<br>0 478 39<br>1 91 92 | (478+92)/700<br>=81. 43% | 92/183=50.3% | 478/517=92.5% |

# Classification table in Python

```
# Predicting Probabilities
from sklearn.metrics import confusion_matrix

predicted_values1 = riskmodel.predict()
threshold=0.5
predicted_class1=np.zeros(predicted_values1.shape)
predicted_class1[predicted_values1>threshold]=1
cm1 = confusion_matrix(bankloan['DEFAULTER'],predict
print('Confusion Matrix : \n', cm1)
```

- **confusion_matrix** function creates a cross table of observed Y (defaulter)vs. predicted Y

```
# Output:
Confusion Matrix :
 [[478  39]
 [ 91  92]]
```

| | | Predicted | |
|---|---|---|---|
| | | 0 | 1 |
| Actual | 0 | TN | FP |
| | 1 | FN | TP |

- This is how the python output of the confusion matrix appears .

**Interpretation :**

- There are 478 correctly predicted non-defaulters and 92 correctly predicted defaulters.
- There are 39 wrongly predicted as defaulters and 91 wrongly predicted as non-defaulters.

# Sensitivity and Specificity in Python

```
# Sensitivity and Specificity
```

```
sensitivity = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Sensitivity : ', sensitivity)

specificity = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Specificity : ', specificity )
```

```
# Output:
```

```
Sensitivity :  0.5027322404371585
Specificity :  0.9245647969052224
```

**Interpretation :**

- The Sensitivity is at 50.27% and the Specificity is at 92.46%. Note that the threshold is set at 0.5

# Precision & Recall

- Precision : Precision tells us what percentage of predicted positive cases are correctly predicted.

$$Precision = \frac{TP}{TP + FP}$$

- Recall or Sensitivity : Recall tells us what percentage of  actual positive cases are correctly predicted.

$$Recall = \frac{TP}{TP + FN}$$

# Classification Report

#Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(bankloan['DEFAULTER'],predicted_class1))
```

# Output:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.92 | 0.88 | 517 |
| 1 | 0.70 | 0.50 | 0.59 | 183 |
| accuracy | | | 0.81 | 700 |
| macro avg | 0.77 | 0.71 | 0.73 | 700 |
| weighted avg | 0.80 | 0.81 | 0.80 | 700 |

classification_report() gives recall, precision and accuracy along with other measures.

**Interpretation :**

- Recall is 50% & Precision is 70%.
- Accuracy is 81%.

# Quick Recap

In this session, we learned about **Binary Logistic Regression** :

| | |
|---|---|
| Binary logistic regression | • Dependent variable is binary and independent variables are categorical or continuous or mix of both.<br>• Regression line is sigmoid curve.<br>• Parameters are estimated using MLE. |
| Classification table | • percentage of correctly predicted observations = accuracy.<br>• Percentage of wrongly predicted observations = misclassification rate |
| Sensitivity/True Positive rate | • % of occurrences correctly predicted |
| Specificity/True Negative rate | • % of non occurrences correctly predicted |
| False Positive Rate | • % of non occurrences which are incorrectly predicted |
| False Negative Rate | • % of occurrences which are incorrectly predicted |
| **Precision & Recall** | • Precision tells us what percentage of predicted positive cases are correctly predicted.<br>• Recall tells us what percentage of actual positive cases are correctly predicted. |