Introduction to Big Data

Lecture 11. Classification

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import seaborn as sns
```

Classification with Personal Loan Data

- Experience
- Income
- Famliy
- CCAvg: Average monthly card spent
- Education: Education level (1: undergrad; 2, Graduate; 3; Advance)
- Mortgage
- Personal Loan: Personal Loan (1:Yes, 0:No)
- Securities account: Securities (1:Yes, 0:No)
- CD account: CD account (1:Yes, 0:No)
- Online: Online account (1:Yes, 0:No)
- CreidtCard: Credit Card (1:Yes, 0:No)

```
PerLoan = pd.read_csv("/content/drive/MyDrive/[Lecture]/IntBigData/BigData_Python/11_Classification/personalLoa
PerLoan.head()
```

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
0	1	25	1	49	91107	4	1.6	1	0	0	1	0	0	0
1	2	45	19	34	90089	3	1.5	1	0	0	1	0	0	0
2	3	39	15	11	94720	1	1.0	1	0	0	0	0	0	0
3	4	35	9	100	94112	1	2.7	2	0	0	0	0	0	0

```
PerLoan.shape
```

```
(2500, 14)
```

```
PerLoan.columns
```

```
Index(['ID', 'Age', 'Experience', 'Income', 'ZIP Code', 'Family', 'CCAvg',
       'Education', 'Mortgage', 'Personal Loan', 'Securities Account',
       'CD Account', 'Online', 'CreditCard'],
      dtype='object')
```

```
PerLoan.rename(columns={'Personal Loan':'PersonalLoan'}, inplace=True)
```

```
PerLoan.columns
```

```
Index(['ID', 'Age', 'Experience', 'Income', 'ZIP Code', 'Family', 'CCAvg',
       'Education', 'Mortgage', 'PersonalLoan', 'Securities Account',
       'CD Account', 'Online', 'CreditCard'],
      dtype='object')
```

```
PerLoan.describe()
```

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	PersonalLoan	Securitie Accour
count	2500.00000	2500.000000	2500.000000	2500.0000	2500.000000	2500.00000	2500.000000	2500.00000	2500.000000	2500.000000	2500.00000
mean	1250.50000	45.346000	20.120800	74.4472	93135.691600	2.40800	1.951284	1.86560	57.388400	0.102400	0.10960
std	721.83216	11.519521	11.523824	46.6724	2420.763339	1.15986	1.795449	0.83655	100.816403	0.303234	0.31245

```
PerLoan.isnull().any()
```

```
ID                    False
Age                   False
Experience            False
Income                False
ZIP Code              False
Family                False
CCAvg                 False
Education             False
Mortgage              False
PersonalLoan          False
Securities Account    False
CD Account            False
Online                False
CreditCard            False
dtype: bool
```

```
PerLoan.count()
```

```
ID                    2500
Age                   2500
Experience            2500
Income                2500
ZIP Code              2500
Family                2500
CCAvg                 2500
Education             2500
Mortgage              2500
PersonalLoan          2500
Securities Account    2500
CD Account            2500
Online                2500
CreditCard            2500
dtype: int64
```

```
PL_X = PerLoan[['Age','CCAvg','Income','Education']]
PL_Y = PerLoan['PersonalLoan']
```

Logit Regression with statsmodels

```
from statsmodels.formula.api import logit

statsLogitModel = logit('PersonalLoan ~ Age + CCAvg + Income + Education',data=PerLoan)
statsLogitModel
```

```
<statsmodels.discrete.discrete_model.Logit at 0x7dbdf48699c0>
```

```
statsLogitModel_res = statsLogitModel.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.167135
         Iterations 9
```

```
print(statsLogitModel_res.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:           PersonalLoan   No. Observations:                 2500
Model:                          Logit   Df Residuals:                     2495
Method:                           MLE   Df Model:                            4
Date:                Thu, 23 Nov 2023   Pseudo R-squ.:                  0.4940
Time:                        13:14:44   Log-Likelihood:                -417.84
converged:                       True   LL-Null:                       -825.81
Covariance Type:            nonrobust   LLR p-value:                2.695e-175
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -11.2330      0.694    -16.179      0.000     -12.594      -9.872
Age            0.0113      0.008      1.401      0.161      -0.005       0.027
CCAvg          0.1187      0.046      2.606      0.009       0.029       0.208
Income         0.0472      0.003     16.648      0.000       0.042       0.053
Education      1.6128      0.135     11.938      0.000       1.348       1.878
==============================================================================
```

```
statsLogitModel_res.params
```

```
Intercept   -11.232994
Age           0.011319
CCAvg         0.118712
Income        0.047228
Education     1.612840
dtype: float64
```

```
np.exp(statsLogitModel_res.params)
```

```
Intercept    0.000013
Age          1.011383
CCAvg        1.126045
Income       1.048361
Education    5.017038
dtype: float64
```

Logit Regression with sklearn

Whole sample

```
from sklearn.linear_model import LogisticRegression

LogitModel0 = LogisticRegression()
```

```
LogitModel0_res = LogitModel0.fit(PL_X, PL_Y)
LogitModel0_res
```

```
▾ LogisticRegression
LogisticRegression()
```

```
LogitModel0_res.coef_
```

```
array([[0.01125087, 0.11722838, 0.04689883, 1.58410844]])
```

```
LogitModel0_res.intercept_
```

```
array([-11.12673563])
```

Test and train sample

```
from sklearn.model_selection import train_test_split

PL_X_train, PL_X_test, PL_Y_train, PL_Y_test = train_test_split(PL_X, PL_Y, test_size=0.3,
                                                                random_state=0)
```

```
# Practice of Random Sampling
PL_X_train1, PL_X_test1, PL_Y_train1, PL_Y_test1 = train_test_split(PL_X, PL_Y,
                                                                    test_size=0.3)
PL_X_train1.head()
```

	Age	CCAvg	Income	Education
2	39	1.0	11	1
851	41	1.4	23	2
2355	56	1.6	74	3
360	35	1.3	55	1
323	59	4.4	99	1

```
PL_X_train1, PL_X_test1, PL_Y_train1, PL_Y_test1 = train_test_split(PL_X, PL_Y,
                                                                    test_size=0.3,
                                                                    random_state=1)
PL_X_train1.head()
```

	Age	CCAvg	Income	Education
1181	25	0.2	65	1
372	56	0.7	44	2

```
PL_X_train1, PL_X_test1, PL_Y_train1, PL_Y_test1 = train_test_split(PL_X, PL_Y,
                                                                     test_size=0.3,
                                                                     random_state=1)
PL_X_train1.head()
```

	Age	CCAvg	Income	Education
1181	25	0.2	65	1
372	56	0.7	44	2
137	49	0.4	128	1
1831	47	0.4	30	2
2296	27	0.2	82	1

```
PL_X_train
```

	Age	CCAvg	Income	Education
1988	52	0.3	18	1

```
PL_X_test
```

	Age	CCAvg	Income	Education
53	50	2.10	190	3
2391	39	4.67	138	2
2310	32	0.30	32	1
728	45	4.40	114	2
850	46	0.20	39	1
...	...	...	...	...
2285	48	2.40	114	3
1165	43	1.70	113	1
2438	62	0.30	29	3
768	43	1.70	72	1
381	55	2.30	73	3

750 rows × 4 columns

```
PL_Y_train
```

```
1988    0
1969    0
1368    0
840     0
2214    0
       ..
1033    0
1731    1
763     0
835     0
1653    0
Name: PersonalLoan, Length: 1750, dtype: int64
```

```
PL_Y_test
```

```
53      1
2391    1
2310    0
728     1
850     0
       ..
2285    1
1165    0
2438    0
768     0
381     0
Name: PersonalLoan, Length: 750, dtype: int64
```

```
from sklearn.linear_model import LogisticRegression

LogitModel = LogisticRegression()
```

```
LogitModel.fit(PL_X_train, PL_Y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
LogitModel.coef_
```

```
array([[0.01166588, 0.07185039, 0.04623718, 1.44376302]])
```

```
LogitModel.intercept_
```

```
array([-10.5227233])
```

Validation

```
PL_Y_pred = LogitModel.predict(PL_X)
```

```
PL_Y_train_pred = LogitModel.predict(PL_X_train)
```

```
PL_Y_test_pred = LogitModel.predict(PL_X_test)
```

```
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score, confusion_matrix
```

Accuracy Score

```
accuracy_score(PL_Y_test, PL_Y_test_pred)
```

```
0.9546666666666667
```

Recall Score

```
recall_score(PL_Y_test, PL_Y_test_pred)
```

```
0.65625
```

Precision Score

```
precision_score(PL_Y_test, PL_Y_test_pred)
```

```
0.7777777777777778
```

Specificity

```
tn, fp, fn, tp = confusion_matrix(PL_Y_test, PL_Y_test_pred).ravel()
specificity = tn / (tn+fp)
specificity
```

```
0.9825072886297376
```