BINARAY LOGISTIC REGRESSION MODEL CROSS VALIDATION IN PYTHON

DATA SCIENCE INSTITUTE

Contents

1. Cross Validation
2. Hold out validation
3. Performance Measures : Accuracy, Recall, Precision
4. K-fold validation

DATA SCIENCE INSTITUTE

Recap: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), 2(28-40 years), 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

DATA SCIENCE INSTITUTE

Binary Logistic Regression in Python

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

```
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
```

DATA SCIENCE INSTITUTE

Binary Logistic Regression in Python

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

bankloan.info()
```

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

```
# 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
```

DATA SCIENCE INSTITUTE

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,

DATA SCIENCE INSTITUTE

Re-run Model in Python

- Once the variables to be retained are finalized, re-run the model with these final variables and obtain revised coefficients for the model.
- Re-run the model with employ, address, debtinc, creddebt.

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

riskmodel.summary()
```

DATA SCIENCE INSTITUTE

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.

Cross Validation in Predictive Modeling

Cross Validation is a
process of evaluating the model on
'Out of Sample' data

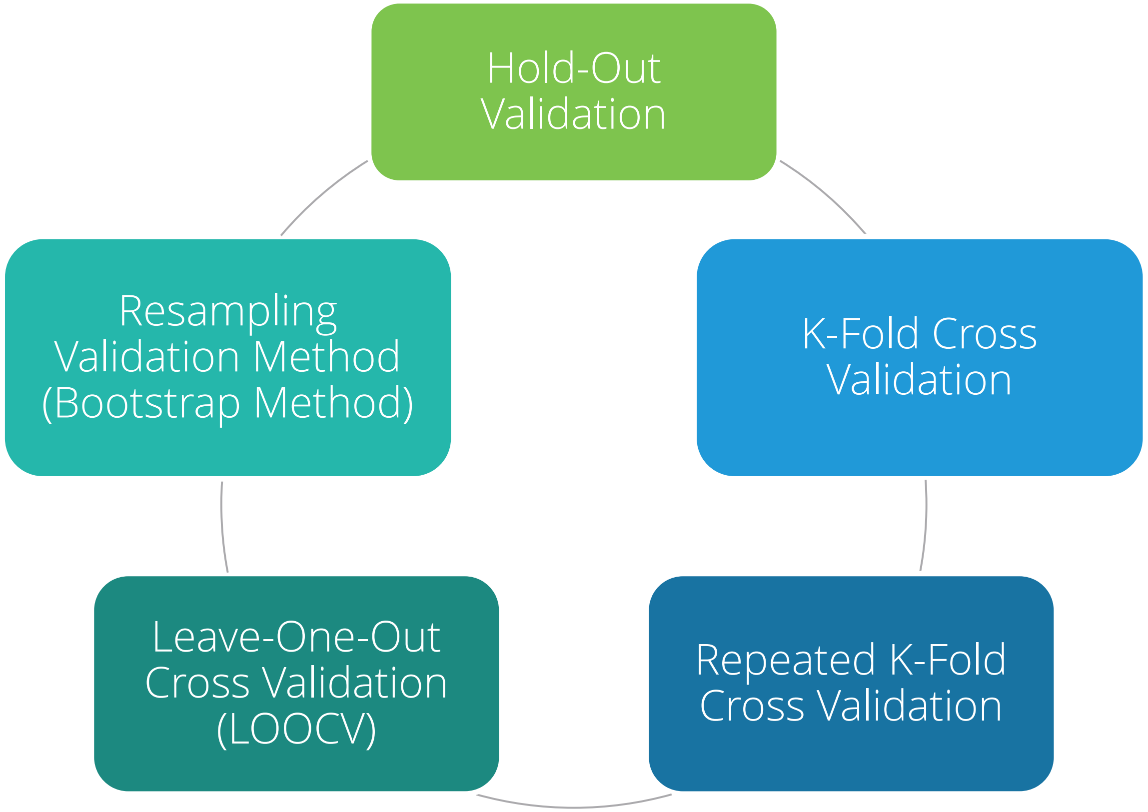
- **Model performance measures** for binary logistic regression such as Accuracy rate, Sensitivity, Specificity **tend to be optimistic on 'In Sample Data'**
- More realistic measures of model performance are calculated using "Out of Sample' data
- Cross-validation is a procedure for estimating the generalization performance in this context

Cross validation is important because although a model is built on historical data, ultimately it is to be used on future data. However good the model, if it fails on out of sample data then it defeats the purpose of predictive modeling

DATA SCIENCE
INSTITUTE

Cross Validation in Predictive Modeling

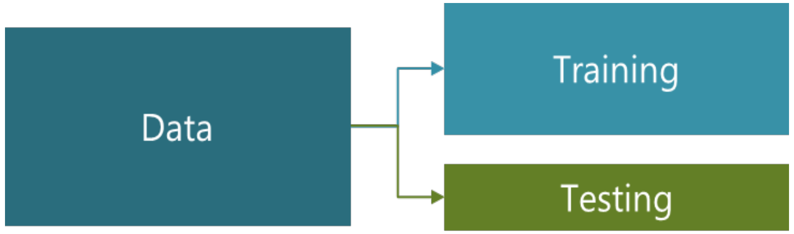
There are different approaches for cross validation. Five most significant of them are:



We will focus on **Hold Out** and **K-Fold** Cross validation methods.

DATA SCIENCE INSTITUTE

Hold-Out Validation



In Hold-Out validation method, available data is split into two non-overlapped parts: 'Training Data' and 'Testing Data'

- The model is
  - Developed using training data
  - Evaluated using testing data
- Training data should have more sample size. Typically 70%-80% data is used for model development

* Here we continue to use previous data of bank loan for our further analysis.

Hold Out Validation in Python

```
# Create 2 groups of the data: Training and Testing
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import statsmodels.formula.api as smf

bankloan=pd.read_csv('BANK LOAN.csv')

X_train, X_test = train_test_split(bankloan, test_size=0.3)
```

- Import **train_test_split** from sklearn.model_selection
- **train_test_split()** creates Training and Testing data sets
- **test_size=** is the percentage of data to be kept as test data

Hold Out Validation in Python

```
# Check the dimensions training and testing data
X_train.shape
```

```
# Output:
(490, 8)
```

```
X_test.shape
```

```
# Output:
(210, 8)
```

The data of 700 observations are partitioned into 2 parts:
With 490 observations in training (model development) data
and
remaining 210 observations in testing data (out of sample).

Hold Out Validation

- Model will be run on the training data and predicted probabilities will be generated.
- Same model will be applied to test data to get the predicted probabilities.
- Classification Report will be used to check the performance of the model in training and testing data.

Performance Measures : Accuracy, Precision, Recall

- Accuracy : Accuracy is defined as the ratio of correctly predicted cases by the total cases.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

- 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.

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

Performance Measures in Python

```
# Generate classification report for training data
```

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

predicted_values1=riskmodel.predict()
threshold=0.3
predicted_class1=np.zeros(predicted_values1.shape)
predicted_class1[predicted_values1>threshold]=1

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

```
# Output:
```

	precision	recall	f1-score	support
0	0.89	0.78	0.83	360
1	0.55	0.75	0.63	130
accuracy			0.77	490
macro avg	0.72	0.76	0.73	490
weighted avg	0.80	0.77	0.78	490

DATA SCIENCE INSTITUTE

Performance Measures in Python

```
# Generate classification report for test data

predicted_values1=riskmodel.predict(X_test)
threshold=0.3
predicted_class1=np.zeros(predicted_values1.shape)
predicted_class1[predicted_values1>threshold]=1

print(classification_report(X_test['DEFAULTER'],predicted_class1))
```

```
# Output:

              precision    recall  f1-score   support

           0       0.84      0.77      0.81       150
           1       0.53      0.63      0.58        60

    accuracy                           0.73       210
   macro avg       0.68      0.70      0.69       210
weighted avg       0.75      0.73      0.74       210
```

Interpretation :

- Accuracy & Sensitivity of test data is lower than that of train data. However, the values are still acceptable.

DATA SCIENCE INSTITUTE

K fold Cross Validation

- In k-fold cross-validation the data is first partitioned into k equally (or nearly equally) sized segments or folds.
- Then k iterations of training and testing are performed such that each time one fold is kept aside for testing and model is developed using k-1 folds.

K-fold Validation in Python

```
# Create k-folds
from sklearn import linear_model
lmreg = linear_model.LogisticRegression()

y=bankloan.DEFAULTER
X=bankloan[['EMPLOY', 'ADDRESS', 'DEBTINC', 'CREDDEBT']]

from sklearn.model_selection import cross_val_predict
from sklearn.metrics.classification import cohen_kappa_score

predicted_prob = cross_val_predict(lmreg, X, y, cv=4,
method='predict_proba')
threshold=0.3
predicted = predicted_prob[:,1]
predicted_class1=np.zeros(predicted.shape)
predicted_class1[predicted>threshold]=1
```

- **cross_val_predict()** generates cross-validated estimates for each input data point.
- **method='predict_proba'** calculates probabilities for both classes.
- cv=4 specifies 4 folds

K-fold Validation in Python

```
# Generate classification report for k-fold validation
print(classification_report(y,predicted_class1))
```

```
# Output:
              precision    recall  f1-score   support

           0       0.90      0.80      0.85       517
           1       0.57      0.75      0.65       183

    accuracy                           0.79       700
   macro avg       0.74      0.77      0.75       700
weighted avg       0.81      0.79      0.80       700
```

- **classification_report() :** gives accuracy, recall and precision values

Interpretation : accuracy of 0.79 and recall of 0.75 indicate that the model is performing good.

Quick Recap

In this session, we learnt about **Model Validation** :

Cross Validation	• Cross Validation is a process of evaluating the model on 'Out of Sample' data.
Hold out validation	• In Hold-Out validation method, available data is split into two non-overlapped parts: 'Training Data' and 'Testing Data'.
Performance Measures	• Performance measures like Accuracy, recall & precision are calculated to check model performance of train & test data. • `classification_report()` gives all these measures
K-fold validation	• In k-fold cross-validation the data is first partitioned into k equally (or nearly equally) sized segments or folds. • Then k iterations of training and testing are performed such that each time one fold is kept aside for testing and model is developed using k-1 folds.

DATA SCIENCE INSTITUTE

THANK YOU!!

DATA SCIENCE
INSTITUTE