Binary Logistic Regression

Model Validation

Contents

Cross Validation in Predictive Modeling

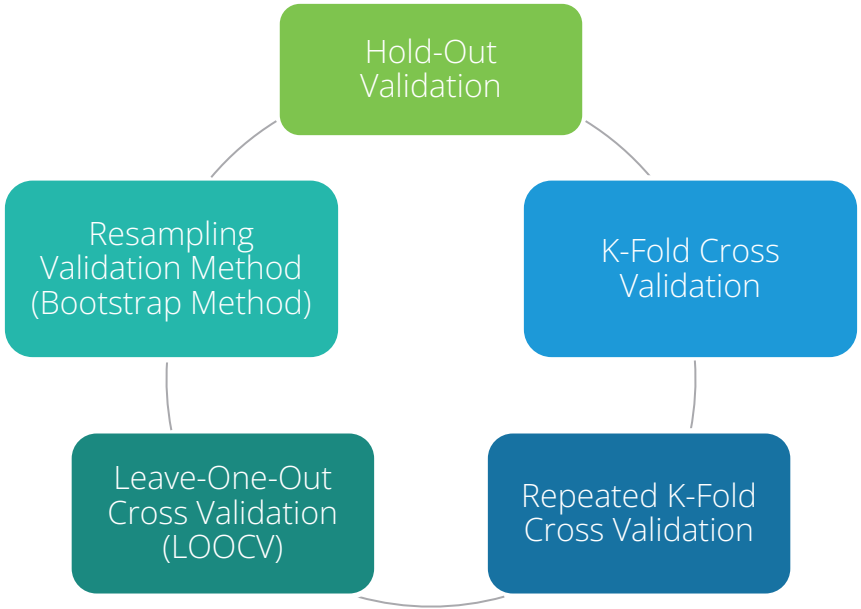
Cross Validation is a
process of evaluating the model on
'Out of Sample' data

- **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

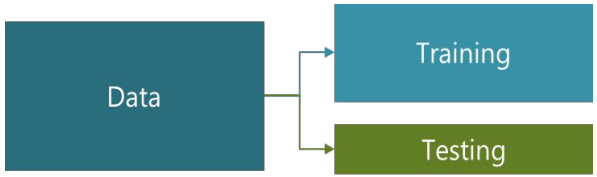
Cross Validation in Predictive Modeling

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.

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

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:
```

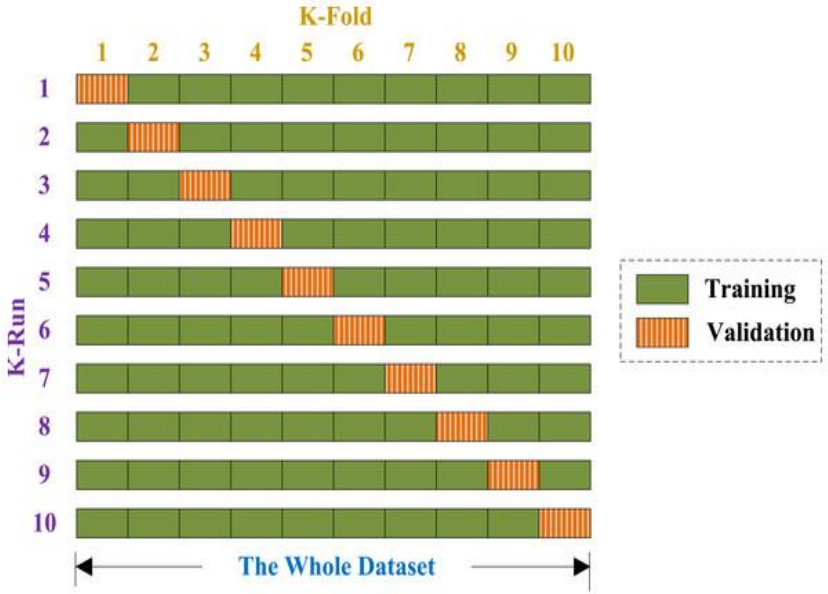
	precision	recall	f1-score	support
0	0.84	0.77	0.81	150
1	0.53	0.63	0.58	60
accuracy			0.73	210
macro avg	0.68	0.70	0.69	210
weighted avg	0.75	0.73	0.74	210

Interpretation :

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

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.