#code basic_packages.py

```
# 기본 패키지
import swat
import os
import sys

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

print(sys.version)
print(swat.__version__)
```

- 출력 폭 정의

```
pd.set_option('display.max_rows', 100)        # DataFrame 출력 시 최대 행 수
pd.set_option('display.max_columns', 100)     # DataFrame 출력 시 최대 열 수
pd.set_option('display.width', 512)         # 출력 폭을 1000 문자로 설정
```

```
import swat

server = "controller.sas-cas-server-default.sasviya4.svc.cluster.local"
port = 5570
user = "viyademo01"
password = "viya40!@"

cas = swat.CAS(server, port, user, password)

print(cas)
print(cas.caslibinfo())
```

- 카스 서버 연결

```
import swat;
cas =  swat.CAS('server.demo.sas.com', 30570, 'student', 'Metadata0')
```

- 카스 클래스와 카스 라이브러리 정보

```
print(cas)
print(cas.caslibinfo())
```

```
>>>
CAS('controller.sas-cas-server-default.sasviya4.svc.cluster.local', 5570, 'viyademo01', protocol='cas',
name='py-session-3', session='407ea76b-60f0-d348-b762-9087d6d39265')
[CASLibInfo]
                    Name  Type                                          Description
Path Definition  Subdirs  Local  Active  Personal  Hidden  Transient TableRedistUpPolicy
 0      CASUSER(student)  PATH                        Personal File System Caslib
/cas/data/caslibs/casuserlibraries/student/                 1.0    0.0     1.0       1.0     0.0        1.0
Not Specified
 1               Formats  PATH                       Stores user defined formats.
/cas/data/caslibs/formats/                0.0    0.0     0.0       0.0     0.0        0.0       Not
Specified
 2  ModelPerformanceData  PATH  Stores performance data output for the Model M...
/cas/data/caslibs/modelMonitorLibrary/                0.0    0.0     0.0       0.0     0.0        0.0
Not Specified
 3                Models  PATH  Stores models created by Visual Analytics for ...
/cas/data/caslibs/models/                0.0    0.0     0.0       0.0     0.0        0.0       Not
Specified
 4                Public  PATH  Shared and writeable caslib, accessible to all...
/cas/data/caslibs/public/                0.0    0.0     0.0       0.0     0.0        0.0       Not
Specified
 5               Samples  PATH               Stores sample data, supplied by SAS.
/cas/data/caslibs/samples/                0.0    0.0     0.0       0.0     0.0        0.0       Not
```

```
Specified
 6            SystemData  PATH  Stores application generated data, used for ge...
/cas/data/caslibs/sysData/                 0.0    0.0     0.0       0.0     0.0        0.0       Not
Specified
+ Elapsed: 0.000918s, user: 0.000887s, mem: 0.686mb
```

session 끊김 시간 정의하기

```
cas.session.timeout(time=3600)
```

CAS 인스턴스

- CAS객체는 SAS Viya와의 연결 및 분산 데이터 분석을 수행하기 위한 인터페이스

```
print(type(cas))
```

```
<class 'swat.cas.connection.CAS'>
```

- 활용 가능 메서드드

```
# 메서드
print(dir(cas)[:30])
```

```
['CASTable', 'SASFormatter', 'about', 'accesscontrol.accesspersonalcaslibs', 'accesscontrol.assumerole',
'accesscontrol.checkinallobjects', 'accesscontrol.checkoutobject', 'accesscontrol.committransaction',
'accesscontrol.completebackup',
'accesscontrol.createbackup', 'accesscontrol.deletedalist', 'accesscontrol.droprole',
'accesscontrol.isauthorized', 'accesscontrol.isauthorizedactions', 'accesscontrol.isauthorizedcolumns',
'accesscontrol.isauthorizedtables', 'accesscontrol.isinrole',
'accesscontrol.listacsactionset', 'accesscontrol.listacsdata', 'accesscontrol.listallprincipals',
'accesscontrol.listmetadata', 'accesscontrol.operactionmd', 'accesscontrol.operactionsetmd',
'accesscontrol.operadminmd', 'accesscontrol.opercolumnmd',
```

```
'accesscontrol.operdapaths', 'accesscontrol.opertablemd', 'accesscontrol.remallacsactionset',
'accesscontrol.remallacsdata', 'accesscontrol.repallacsaction']
```

CASResult 인스턴스

- `CASResult` 클래스는 **CAS 서버에 액션을 요청한 결과를 담고 있는 객체**입니다. 즉, SAS Viya의 CAS 서버에 특정 작업(액션)을 실행하도록 요청한 후, 그 작업의 결과 데이터, 상태 정보, 경고, 에러 메시지 등 다양한 메타데이터를 포함하여 반환되는 객체라고 할 수 있습니다.
- 예를 들어, Python의 SWAT 라이브러리를 이용하여 CAS 서버에 액션을 호출하면, 그 실행 결과가 `CASResult` 객체 형태로 반환되며, 이를 통해 반환된 데이터에 접근하거나 후속 처리를 할 수 있습니다.

```
r = cas.about()
print(r)
print(type(r))
```

```
NOTE: Grid node action status report: 1 nodes, 17 total actions executed.
[About]
 {'CAS': 'Cloud Analytic Services',
  'Version': '4.00',
  'VersionLong': 'V.04.00M0P03182024',
  'Viya Release': '20240322.1711070907331',
  'Viya Version': 'Stable 2024.03',
  'Copyright': 'Copyright © 2014-2024 SAS Institute Inc. All Rights Reserved.',
  'ServerTime': '2025-01-08T03:34:34Z',
  'System': {'Hostname': 'controller.sas-cas-server-default.sasviya4.svc.cluster.local',
   'OS Name': 'Linux',
```

```
   'OS Family': 'LIN X64',
   'OS Release': '4.18.0-348.7.1.el8_5.x86_64',
   'OS Version': '#1 SMP Wed Dec 22 13:25:12 UTC 2021',
   'Model Number': 'x86_64',
   'Linux Distribution': 'Red Hat Enterprise Linux release 8.9 (Ootpa)'},
  'license': {'site': 'KOLON BENIT-ALLIANCE PARTNER-VIYA VDS',
   'siteNum': 70278905,
   'expires': '30Jun2025:00:00:00',
   'gracePeriod': 0,
   'warningPeriod': 15},
  'CASHostAccountRequired': 'OPTIONAL',
  'Transferred': 'NO',
  'CASCacheLocation': 'CAS Disk Cache'}
[server]
 Server Status

    nodes  actions
 0      1       17
[nodestatus]
 Node Status

                                                 name        role   uptime  running  stalled
 0  controller.sas-cas-server-default.sasviya4.svc...  controller  634.654        0        0
```

```
+ Elapsed: 0.00236s, user: 0.000767s, sys: 0.000704s, mem: 0.304mb
<class 'swat.cas.results.CASResults'>
```

- CASResults 클래스는 사전 데이터 형

```
# CASResults 클래스의 정체
isinstance(r, dict)
```

```
True
```

SASDataFrame

```
# 데이터프레임형식
print(r['nodestatus'])
```

```
<class 'swat.dataframe.SASDataFrame'>
```

```
# SASDataFrame?
isinstance(r['nodestatus'], swat.SASDataFrame), isinstance(r['nodestatus'], pd.DataFrame)
```

카스 라이브러리

- SAS에서 라이브러리(LIBNAME)와 SAS Viya의 CAS 라이브러리(CASLIB)는 둘 다 데이터에 대한 접근 경로를 정의하는데 사용되지만, 이들 사이에는 몇 가지 주요 차이점이 있습니다.
- 분산 컴퓨팅 지원: CAS 라이브러리는 SAS Viya의 CAS(Cloud Analytic Services)에서 사용되며, 이는 분산 컴퓨팅 환경을 지원합니다. 따라서 CAS 라이브러리는 분산 데이터 소스(예: Hadoop)에 직접 접근할 수 있습니다. 반면에, 기존의 SAS 라이브러리는 일반적으로 단일 노드에서 실행되며, 분산 데이터에 대한 직접적인 접근을 지원하지 않습니다.
- 데이터 소스 유형: CAS 라이브러리는 다양한 데이터 소스 유형을 지원합니다. 이에는 로컬 및 분산 파일 시스템, Amazon S3, 데이터베이스 등이 포함됩니다. 반면에, SAS 라이브러리는 파일 시스템과 SAS 데이터 세트, 그리고 특정 데이터베이스에 대한 접근을 지원합니다.
- 인 메모리 처리: CAS는 인 메모리 분석을 지원하므로, CAS 라이브러리를 통해 로드된 데이터는 CAS 서버의 메모리에 저장됩니다. 이는 대량의 데이터를 빠르게 처리하는 데 유리합니다. 반면에, 기존의 SAS 라이브러리는 디스크 기반의 처리를 사용합니다.
- 데이터 접근 제어: CAS 라이브러리는 데이터 접근을 더 세밀하게 제어할 수 있습니다. 예를 들어, 특정 사용자가 특정 CAS 라이브러리만 접근할 수 있도록 설정할 수 있습니다. 반면에, SAS 라이브러리의 접근 제어는 보다 제한적입니다.
- 따라서, CAS 라이브러리와 SAS 라이브러리는 비슷한 목적을 가지고 있지만, 분산 컴퓨팅 지원, 데이터 소스 유형, 데이터 처리 방식, 접근 제어 등의 면에서 차이점을 가지고 있습니다.

```
# 카스 라이브러리
r = cas.table.caslibinfo()
```

```
print(r)
```

```
[CASLibInfo]

                    Name  Type                                        Description
Path
Definition  Subdirs  Local  Active  Personal  Hidden  Transient TableRedistUpPolicy
 0      CASUSER(student)  PATH                        Personal File System Caslib
/cas/data/caslibs/casuserlibraries/student/
           1.0    0.0     1.0       1.0     0.0        1.0       Not Specified
 1               Formats  PATH                       Stores user defined formats.
/cas/data/caslibs/formats/
           0.0    0.0     0.0       0.0     0.0        0.0       Not Specified
 2  ModelPerformanceData  PATH  Stores performance data output for the Model M...
/cas/data/caslibs/modelMonitorLibrary/
           0.0    0.0     0.0       0.0     0.0        0.0       Not Specified
 3                Models  PATH  Stores models created by Visual Analytics for ...
/cas/data/caslibs/models/
           0.0    0.0     0.0       0.0     0.0        0.0       Not Specified
 4                Public  PATH  Shared and writeable caslib, accessible to all...
/cas/data/caslibs/public/
           0.0    0.0     0.0       0.0     0.0        0.0       Not Specified
 5               Samples  PATH               Stores sample data, supplied by SAS.
/cas/data/caslibs/samples/
```

```
           0.0    0.0     0.0       0.0     0.0        0.0       Not Specified
 6            SystemData  PATH  Stores application generated data, used for ge...
/cas/data/caslibs/sysData/
           0.0    0.0     0.0       0.0     0.0        0.0       Not Specified
+ Elapsed: 0.000998s, user: 0.000719s, sys: 0.000242s, mem: 0.685mb
```

- Name: CAS 라이브러리의 이름
- Type: CAS 라이브러리의 유형. 'PATH'는 파일 시스템에 위치한 라이브러리
- Description: CAS 라이브러리에 대한 설명
- Path: CAS 라이브러리의 파일 시스템 경로
- Definition: CAS 라이브러리가 정의된 방법
- Subdirs: 하위 디렉토리를 스캔할지 여부
- Local: CAS 라이브러리가 로컬에 있는지 여부
- Active: CAS 라이브러리가 활성 상태인지 여부
- Personal: CAS 라이브러리가 개인 라이브러리인지 여부
- Hidden: CAS 라이브러리가 숨겨진 라이브러리인지 여부
- Transient: CAS 라이브러리가 일시적인 라이브러리인지 여부

카스 라이브러리의 몇가지 메서드

```
# 활동 라이브러리를 지정해주기
cas.sessionProp.setSessOpt(caslib='casuser')
```

```
# 활동 라이브러리의 의미
print(cas.table.tableinfo())

# 라이브러리 안에 있는 파일 보기
print(cas.fileinfo())
```

카스 테이블

데이터 읽기

- 클라이언트 사이드의 데이터를 읽고 서버로 올림

```
# 데이터 읽기: hmeq
url = "https://github.com/bong-ju-kang/data/raw/master/hmeq.csv"
cas.read_csv(url)

# 데이터 확인
print(cas.table.tableinfo()['TableInfo']['Name'])
```

```
0    TMPCB4B5O3G
Name: TableInfo, dtype: object
```

- 이름이 무작위로 표현됨

```
# 데이터 이름 명시
df = cas.CASTable(name='hmeq', caslib='casuser')
# df = cas.CASTable('hmeq')
```

```
# 데이터 확인
print(cas.table.tableinfo()['TableInfo']['Name'])
```

```
0    TMPCB4B5O3G
1           HMEQ
Name: TableInfo, dtype: object
```

카스 테이블

- SAS Viya의 **CAS 테이블**은 CAS 서버 내에서 관리되고 처리되는 분산 메모리 기반 데이터 구조를 의미합니다.
- **분산 메모리 테이블**: CAS 테이블은 여러 노드에 걸쳐 메모리 내에서 분산되어 저장되므로, 대용량 데이터를 빠르게 읽고 처리할 수 있습니다.
- **고속 분석 처리**: CAS의 병렬 처리 능력을 활용하여 데이터를 신속하게 로딩, 쿼리, 집계, 통계 분석 등을 수행할 수 있습니다.
- **데이터 관리**: CAS 테이블은 SAS Viya 환경에서 일시적 또는 영구적으로 데이터를 저장할 수 있는 객체로 활용되며, 데이터셋을 CAS 서버로 올리거나 내릴 때 중심 역할을 합니다.
- **Python 인터페이스**: `swat` 라이브러리 등을 통해 Python 코드 내에서도 CAS 테이블을 생성, 조회, 조작할 수 있으므로, Python과 SAS Viya 간의 효율적인 연동이 가능합니다.

```
# 카스 테이블 인터페이스
df = cas.CASTable({'name':'hmeq', 'caslib':'casuser'})
# df = cas.CASTable('hmeq')
print(type(df))
```

```
<class 'swat.cas.table.CASTable'>
```

- 테이블 정보 조회

```
# 예시: CAS 서버에 존재하는 테이블 목록 확인
result = cas.table.tableInfo(caslib='casuser')
print(result)
```

- 테이블 생성 또는 데이터 업로드

```
# 파일을 CAS 테이블로 업로드
cas.upload_file('data.csv', casout={'name': 'my_table', 'caslib': 'casuser'})
```

- 테이블 삭제(drop)

```
# 특정 테이블 삭제
cas.table.dropTable(caslib='casuser', name='my_table')
```

- 테이블 정보(컬럼 정보) 확인

```
# 테이블의 컬럼 정보 조회
result = cas_conn.table.columnInfo(caslib='casuser', name='my_table')
print(result)
```

#실습 카스 테이블에 대한 다양한 메서드를 적용해보자.

- 변수 유형(dtypes)
- 변수 정보(info)
- 넘파이 배열 전환(values)
- 차원(ndim)
- 크기(size)
- 모양(shape)
- 데이터 일부 보기(head, fetch, sample)
- 데이터 선택(loc, iloc): iloc 적용시 행 선택은 지원되지 않음
- 데이터 정렬(sort_values)
- 데이터 요약(describe)
- 건수(count)
- 통계량(mean, median)
- 결측값 건수(nmiss)
- 분위수(quantile)

#실습 판다스의 dtypes와 카스 테이블의 dtypes를 비교해보자

카스 컬럼

#실습 카스 컬럼의 다양한 속성들을 확인해보자.

- 유일값(unique)
- 유일값 건수(nunique)
- 1차 빈도표(value_counts)
- 2차 빈도표(cas.crosstab)

액션셋(action set)

- https://documentation.sas.com/doc/en/pgmsascdc/v_053/allprodsactions/actionSetsByProduct.htm
- SAS Viya의 CAS 환경에서는 **액션 셋(Action Set)** 이라는 개념을 사용하여 다양한 데이터 관리, 분석, 모델링 작업을 실행합니다. 액션 셋은 CAS 서버에서 미리 정의된 일련의 액션(함수)들을 모아둔 집합으로, 각 액션은 특정 작업(예: 데이터 로딩, 집계, 통계분석, 모델 학습 등)을 수행하도록 설계되어 있습니다.
- 액션셋 정보 보기

```
# 액션셋 = 패키지 또는 모듈
print(cas.builtins.actionsetinfo())
```

- 액션셋의 액션 보기

```
# builtins 액션셋의 액션들
print(cas.help(actionset='builtins'))

# table 액션셋의 액션들
print(cas.help(actionset='table'))
```

- 데이터 요약

```
# 카스테이블 지정
df = cas.CASTable('hmeq')
# 데이터 요약
print(df.simple.summary())
```

- 액션셋 추가

```
# 액션셋 메모리 로드
print(cas.loadactionset('dataSciencePilot'))
```

데이터 탐색

dataSciencePlot 액션셋

- 액션셋 불러오기

```
cas.loadactionset('dataSciencePilot')
```

exploreData 액션

- 탐색 기준표

정책	적용 변수 유형	기본값	비고
missing	all	5, 25	결측값 비율(%)
cardinality	nominal	lowMediumCutoff=32, mediumHighCutoff=64	작은 경계값(low-medium), 큰 경계값(medium-high)
entropy	nominal	0.25, 0.75	표준화 엔트로피 기준으로 정의. 표준화 엔트로피는 0과 1사이의 값이며, 값이 크다는 것은 랜덤 값에 가까움을 의미함
IQV(index of qualitative variation)	nominal	highVariationRatio=0.5, highTopBottom=100, highTopTwo=10	변동비, 빈도 최상/최하 비율, 빈도 상위/차상위 비율. 디폴트는 변동비로 판정

정책	적용 변수 유형	기본값	비고
cv	interval	lowMoment=1, lowRobust=1	기본값은 경계값(low-high)의 백분위수를 의미함. 해당 값보다 크면 모두 3등급으로 처리 |CV|*100 의 경계값. 값이 클수록 평균대비 편차가 크다는 것을 의미
skewness	Interval	적률(2, 10), 분위수(0.75, 2)	적률기준, 분위수 기준 작은/큰 경계값. 정규분포는 0에 가까움. 양쪽 모두 평가 후 높은 등급으로 판정
kurtosis	interval	적률(5, 10), 분위수(2, 3)	적률기준, 분위수 기준 작은/높은 경계값. 정규분포는 0에 가까움. . 양쪽 모두 평가 후 높은 등급으로 판정
outlier	interval	z(1, 2.5), IQR(1, 2.5)	z, IQR기준 이상값 백분위수. 양쪽 모두 평가 후 높은 등급으로 판정

```
# 데이터 탐색 액션(exploreData)
df.exploreData(
    casout={'name': 'explore_out', 'replace': True},
    target = 'BAD'
)

# 결과 확인
print(cas.CASTable('explore_out').columns)
# print(cas.CASTable('explore_out').head(999))
```

```
Index(['Variable', 'VarType', 'MissingRated', 'CardinalityRated', 'EntropyRated', 'IQVRated', 'CVRated',
'SkewnessRated', 'KurtosisRated', 'OutlierRated', 'N', 'NMiss', 'MissRate', 'Cardinality',
'CardinalityRatio', 'Shannon', 'Gini', 'VariationRatio', 'Top1Top2FreqRatio', 'Top1Bottom1FreqRatio',
'Top1', 'Top2', 'Top3', 'Bottom1', 'Bottom2', 'Bottom3', 'Min', 'Max', 'Mean', 'Median', 'Std', 'IQR',
'MomentCVPer', 'RobustCVPer', 'MomentSkewness', 'RobustSkewness', 'MomentKurtosis', 'RobustKurtosis',
'LowerOutlierMomentPer', 'UpperOutlierMomentPer', 'LowerOutlierRobustPer', 'UpperOutlierRobustPer'],
      dtype='object')
```

#실습 다음의 각각에 대하여 설명해보자.

1. 결측값 기준으로 2등급인 변수들이 왜 그런지 이유를 설명해보자. nmiss 메서드를 통하여 결과를 확인해보자
2. 유일 건수 기준으로 3등급인 'LOAN' 변수에 대한 이유를 설명해보자.
3. 엔트로피 기준 3등급인 'REASON' 변수에 대한 이유를 설명해보자.
4. 변동비 기준 3등급인 'JOB' 변수에 대한 이유를 설명해보자.
5. 변동계수 기준 3등급인 'MORTDUE' 변수에 대한 이유를 설명해보자.
6. 첨도 기준 3등급인 'VALUE' 변수에 대한 이유를 설명해보자.
7. 이상값 기준 3등급인 'MORTDUE' 변수에 대한 이유를 설명해보자.

#실습 주어진 데이터에 대하여 각 질문을 수행해보자.

```
url = 'https://github.com/bong-ju-kang/data/raw/master/gcr.csv'
```

- 카스 서버로 데이터 올리기 및 확인하기

- 데이터 일부 추출하여 확인하기
- 변수와 변수 유형 확인하기
- 변수 유형이 'double'인 변수이름 추출하기
- 데이터를 탐색하여 결측값에 문제 있는 변수가 있는지 확인 (2등급 이상 추출)
- ['history', 'purpose'] 변수에 대한 분포 확인

데이터 전처리

데이터 분할: sampling 액션셋

- 액션셋 불러오기

```
# sampling actionset 적재
cas.loadactionset('sampling')
```

층화 분할: sampling.stratified

- https://documentation.sas.com/doc/en/pgmsascdc/v_059/casactstat/cas-sampling-stratified.htm

```
# 데이터 분할
cas.sampling.stratified(

    # 데이터 대상
    table = {'name':df, 'groupby':'BAD'},

    # 비율 지정
    samppct = 70,

        # 시드값 지정
        seed = 42,
```

```
    # 분할 변수 출력 여부 결정
    partInd = True,

    # 출력 테이블 지정
    output = {'casout':{'name':'df_partn', 'replace':True}, 'copyVars':"ALL"}
)

# 결과 확인
print(cas.CASTable('df_partn')['_PartInd_'].value_counts(normalize=True))
```

```
NOTE: Stratified sampling is in effect.
NOTE: Using SEED=42 for sampling.
1.0    0.7
0.0    0.3
dtype: float64
```

- 훈련 데이터 정의

```
# 훈련 데이터 지정
train = cas.CASTable('df_partn').query('_PartInd_=1')

# 확인
print(train.shape)
```

```
(4172, 14)
```

- 변수 정의

```
# 일반적인 방법: 목표변수와 이벤트, 설명변수 정의
target = 'BAD'
event = '1'
partvar = '_PartInd_'
xvars = [x for x in train.columns.tolist() if x not in [target,partvar ]]
print(xvars)
```

```
['LOAN', 'MORTDUE', 'VALUE', 'REASON', 'JOB', 'YOJ', 'DEROG', 'DELINQ', 'CLAGE', 'NINQ', 'CLNO', 'DEBTINC']
```

```
# 범주 변수 정의
cats = [x for x in df.select_dtypes('varchar').columns.tolist() if x in xvars]
print(f"범주변수:\n{cats}")

# 숫자변수 정의
nums = list(set(xvars)-set(cats))
print(f"숫자변수:\n{nums}")
```

```
범주변수:
['REASON', 'JOB']
```

```
숫자변수:
['DEBTINC', 'CLNO', 'CLAGE', 'LOAN', 'YOJ', 'MORTDUE', 'NINQ', 'DELINQ', 'VALUE', 'DEROG']
```

결측값 영향도 분석: dataPreprocess.analyzeMissingPatterns

- 액션셋 불러오기

```
cas.loadactionset('dataSciencePilot.analyzeMissingPatterns')
```

- 결측값 영향도 분석

```
# 결측값 영향도
train.analyzeMissingPatterns(
    casout={'replace':True, 'name':'missing_patterns_out', 'replace':True},
    target = target,
    inputs = [xvars]
)
# 결과 확인
print(cas.CASTable('missing_patterns_out').sort_values('NormMI', ascending=False).head(999))
```

```
Selected Rows from Table MISSING_PATTERNS_OUT
   FirstVariable SecondVariable  Type        MI    NormMI        SU  EntropyPerChange
0         DEBTINC           BAD  _mt_  0.184663  0.555698  0.253300         25.620089
1           VALUE           BAD  _mt_  0.035338  0.261222  0.083080          4.902821
2           DEROG           BAD  _mt_  0.005747  0.106900  0.009207          0.797287
3             JOB           BAD  _mt_  0.003950  0.088701  0.008039          0.547956
```

4	DELINQ	BAD	_mt_	0.003828	0.087326	0.006481	0.531032
5	YOJ	BAD	_mt_	0.002962	0.076854	0.005139	0.410944
6	NINQ	BAD	_mt_	0.001975	0.062781	0.003470	0.273957
7	CLAGE	BAD	_mt_	0.000358	0.026762	0.000718	0.049700
8	REASON	BAD	_mt_	0.000158	0.017770	0.000329	0.021909
9	CLNO	BAD	_mt_	0.000082	0.012787	0.000174	0.011344
10	MORTDUE	BAD	_mt_	0.000003	0.002524	0.000006	0.000442
11	LOAN	BAD	_mt_	0.000000	0.000000	0.000000	0.000000

결측값 처리: dataPreprocess.impute

- 액션셋 불러오기

```
cas.loadactionset('dataPreprocess')
```

- 결측값 처리

```
# 결측값 처리
train.dataPreprocess.impute(

    # 결측값 처리 방식
    methodInterval='mean',
    methodNominal = 'mode',

    # 처리 변수
    inputs = xvars,

    # 결과 테이블
    casout={'name':'train_impute_out', 'replace':True},

    # 복사 변수
    copyVars = target,

    # 모델 저장
```

```
    code = {'casout':{'name':'impute_code', 'replace':True}}
)

# 결과 확인
print(f"결과 데이터:")
print(cas.CASTable('train_impute_out').head())

# 모델 확인
print(f"모델:")
print(cas.CASTable('impute_code').head())
```

```
결과 데이터:
Selected Rows from Table TRAIN_IMPUTE_OUT
   BAD   IMP_CLAGE  IMP_CLNO  IMP_DEBTINC  IMP_DELINQ  IMP_DEROG  IMP_LOAN  IMP_MORTDUE  IMP_NINQ  IMP_VALUE
IMP_YOJ IMP_JOB IMP_REASON
0  1.0   94.366667       9.0    33.849881         0.0        0.0    1100.0      25860.0       1.0    39025.0
10.5   Other    HomeImp
1  1.0  121.833333      14.0    33.849881         2.0        0.0    1300.0      70053.0       0.0    68400.0
7.0   Other    HomeImp
2  1.0  149.466667      10.0    33.849881         0.0        0.0    1500.0      13500.0       1.0    16700.0
4.0   Other    HomeImp
3  0.0   93.333333      14.0    33.849881         0.0        0.0    1700.0      97800.0       0.0   112000.0
3.0  Office    HomeImp
4  1.0   88.766030       8.0    36.884894         0.0        0.0    1800.0      28502.0       0.0    43034.0
```

```
11.0   Other    HomeImp
모델:
Selected Rows from Table IMPUTE_CODE
  ModelName                                          DataStepSrc DS2Src FormatXML FormatItemStore VarXML
AStore AStoreKey ModelUUID Notes
0               _ngbys_ = 1;\n   _igby_ = 0;\n   _tnn_ntran...
```

- 모델 저장

```
# 모델 저장
cas.table.save(
    # 대상
    table='impute_code',

    # 저장명
    name='impute_code', caslib='casuser', replace=True
)

# 저장 모델 확인
print(cas.fileinfo())
```

```
[FileInfo]
    Permission Owner Group                 Name   Size Encryption                      Time       ModTime
```

```
 0  -rwxr-xr-x   sas   sas  impute_code.sashdat  12168       NONE  2025-01-19T07:42:00+00:00  2.052892e+09
+ Elapsed: 0.00152s, sys: 0.00125s, mem: 0.685mb
```

- 모델 호출

```python
# 모델 호출
cas.table.loadtable(
    # 대상 정의
    path = 'impute_code.sashdat', caslib='casuser',
    # 목적지 정의
    casout={'name': 'impute_code', 'replace':True}
)

# 호출 결과 확인
print(cas.tableinfo())
```

```
[TableInfo]
                Name  Rows  Columns  IndexedColumns Encoding        CreateTimeFormatted
ModTimeFormatted        AccessTimeFormatted JavaCharSet    CreateTime       ModTime    AccessTime  Global
Repeated  View  MultiPart           SourceName
 SourceCaslib  Compressed  Creator Modifier     SourceModTimeFormatted  SourceModTime TableRedistUpPolicy
 0       TMPTE581N3F  5960       13               0    utf-8  2025-01-19T05:28:23+00:00  2025-01-
19T05:28:23+00:00  2025-01-19T05:28:23+00:00        UTF8  2.052884e+09  2.052884e+09  2.052884e+09       0
0     0          0
```

```
                        0  student           2025-01-19T05:28:23+00:00   2.052884e+09       Not Specified
 1              HMEQ  5960       13              0    utf-8  2025-01-19T05:28:23+00:00  2025-01-
19T05:28:23+00:00  2025-01-19T07:31:21+00:00        UTF8  2.052884e+09  2.052884e+09  2.052891e+09       0
0     0          0
                        0  student           2025-01-19T05:28:23+00:00   2.052884e+09       Not Specified
 2       EXPLORE_OUT    13       42              0    utf-8  2025-01-19T06:06:03+00:00  2025-01-
19T06:06:03+00:00  2025-01-19T06:59:55+00:00        UTF8  2.052886e+09  2.052886e+09  2.052889e+09       0
0     0          0
                        0  student                                                NaN       Not Specified
 3          DF_PARTN  5960       14              0    utf-8  2025-01-19T07:22:36+00:00  2025-01-
19T07:22:36+00:00  2025-01-19T07:39:37+00:00        UTF8  2.052891e+09  2.052891e+09  2.052892e+09       0
0     0          0
                        0  student                                                NaN       Not Specified
 4  TRAIN_IMPUTE_OUT  4172       13              0    utf-8  2025-01-19T07:39:37+00:00  2025-01-
19T07:39:37+00:00  2025-01-19T07:39:37+00:00        UTF8  2.052892e+09  2.052892e+09  2.052892e+09       0
0     0          0
                        0  student                                                NaN       Not Specified
 5       IMPUTE_CODE     1       10              0    utf-8  2025-01-19T07:44:18+00:00  2025-01-
19T07:44:18+00:00  2025-01-19T07:44:18+00:00        UTF8  2.052892e+09  2.052892e+09  2.052892e+09       0
0     0          0  impute_code.sashdat
CASUSER(student)           0  student           2025-01-19T07:42:00+00:00   2.052892e+09       Not Specified
+ Elapsed: 0.00159s, user: 0.000777s, sys: 0.000421s, mem: 0.697mb
```

- 모델 적용: dataStep.runCodeTable

```
# 액션셋 불러오기
cas.loadactionset('datastep')
```

```
# 적용 테이블 생성: runCdoeTable 에는 where 절이 적용되지 않음
code = """
data casuser.df_test;
   set casuser.df_partn;
   if _PartInd_ = 0 then output;
run;
"""
result = cas.runCode(code)
print(result)
```

```
# 결측값 처리
cas.dataStep.runCodeTable(
    # 대상
        table = cas.CASTable('df_test'),
    #  출력
    casout={'name':'test_impute_out'},
    # 변수
    dropvars = xvars,
    # 모델 테이블
    codeTable = 'impute_code'
```

```
)

# 결과 확인
print(cas.CASTable('test_impute_out').head())
print(cas.CASTable('test_impute_out').drop(partvar, axis=1).nmiss())
```

```
Selected Rows from Table TEST_IMPUTE_OUT
   BAD  _PartInd_  IMP_LOAN   IMP_MORTDUE      IMP_VALUE    IMP_YOJ  IMP_DEROG  IMP_DELINQ   IMP_CLAGE
IMP_NINQ   IMP_CLNO  IMP_DEBTINC IMP_REASON IMP_JOB
0  1.0        0.0    1500.0  74398.298219  102561.451682   8.929064    0.25558    0.456984  180.926512
1.171204  21.410377    33.849881    DebtCon   Other
1  1.0        0.0    1700.0  30548.000000   40320.000000   9.000000    0.00000    0.000000  101.466002
1.000000   8.000000    37.113614    HomeImp   Other
2  1.0        0.0    1800.0  48649.000000   57037.000000   5.000000    3.00000    2.000000   77.100000
1.000000  17.000000    33.849881    HomeImp   Other
3  1.0        0.0    2000.0  32700.000000   46740.000000   3.000000    0.00000    2.000000  216.933333
1.000000  12.000000    33.849881    HomeImp   Other
4  1.0        0.0    2000.0  74398.298219   62250.000000  16.000000    0.00000    0.000000  115.800000
0.000000  13.000000    33.849881    HomeImp   Sales
BAD            0
IMP_LOAN       0
IMP_MORTDUE    0
IMP_VALUE      0
IMP_YOJ        0
```

```
IMP_DEROG      0
IMP_DELINQ     0
IMP_CLAGE      0
IMP_NINQ       0
IMP_CLNO       0
IMP_DEBTINC    0
IMP_REASON     0
IMP_JOB        0
dtype: int64
```

- 결측값 처리가 된 변수 이름에는 모두 'IMP_' 의 접두사를 갖고 있음
- 변수 재 지정

```
# 변수 재 정의
imp_xvars = ['IMP_'+x for x in xvars]
imp_cats = ['IMP_'+x for x in cats]
imp_nums = ['IMP_'+x for x in nums]
print(imp_xvars)
print(imp_cats)
print(imp_nums)
```

```
['IMP_LOAN', 'IMP_MORTDUE', 'IMP_VALUE', 'IMP_REASON', 'IMP_JOB', 'IMP_YOJ', 'IMP_DEROG', 'IMP_DELINQ',
'IMP_CLAGE', 'IMP_NINQ', 'IMP_CLNO', 'IMP_DEBTINC']
['IMP_REASON', 'IMP_JOB']
```

```
['IMP_LOAN', 'IMP_MORTDUE', 'IMP_CLNO', 'IMP_VALUE', 'IMP_YOJ', 'IMP_DEBTINC', 'IMP_NINQ', 'IMP_CLAGE',
'IMP_DEROG', 'IMP_DELINQ']
```

변수 선별 (variable screening): dataSciencePilot.screenVariables

- https://documentation.sas.com/doc/en/pgmsascdc/v_059/casactml/cas-datasciencepilot-screenvariables.htm

```
train_impute = cas.CASTable('train_impute_out')
# 변수 스크리닝
train_impute.screenVariables(
    casout={'name':'screen_out', 'replace':True},
    target = target
)

# 결과 확인
print(cas.CASTable('screen_out').head(999))
```

```
Selected Rows from Table SCREEN_OUT
   Variable Recommendation                      Reason
0    REASON           keep  passed all screening tests
1       JOB           keep  passed all screening tests
2      LOAN           keep  passed all screening tests
3   MORTDUE           keep  passed all screening tests
4     VALUE           keep  passed all screening tests
5       YOJ           keep  passed all screening tests
6     DEROG           keep  passed all screening tests
```

```
7    DELINQ          keep  passed all screening tests
8     CLAGE          keep  passed all screening tests
9       NINQ         keep  passed all screening tests
10     CLNO          keep  passed all screening tests
11  DEBTINC          keep  passed all screening tests
```

변수 선택 (variable selection): dataSciencePilot.exploreCorrelation

- 상호 정보 기준 선택

```
# 상관관계 분석: 상호정보
train_impute.exploreCorrelation (
    casout={'name':'corr_out', 'replace':True},
    target=target,
    inputs=imp_xvars,
    nominals=imp_cats+[target],
    stats={
        'nominalNominal':['MI', 'NORMMI'],
        'nominalInterval':['MI', 'NORMMI']
    }
)

# 결과 확인: 특정 통계량값 기준 내림 차순 정렬
print(cas.CASTable('corr_out').sort_values('MI', ascending=False).head(999))
```

```
Selected Rows from Table CORR_OUT
   FirstVariable SecondVariable  Type        MI    NormMI
0    IMP_DEBTINC            BAD  _it_  0.231112  0.608374
1     IMP_DELINQ            BAD  _it_  0.077426  0.378759
2      IMP_VALUE            BAD  _it_  0.050293  0.309343
```

```
3      IMP_DEROG           BAD  _it_  0.047813  0.301987
4       IMP_LOAN           BAD  _it_  0.043087  0.287342
5      IMP_CLAGE           BAD  _it_  0.036812  0.266419
6       IMP_NINQ           BAD  _it_  0.022484  0.209693
7       IMP_CLNO           BAD  _it_  0.020666  0.201219
8    IMP_MORTDUE           BAD  _it_  0.018509  0.190634
9        IMP_YOJ           BAD  _it_  0.015662  0.175606
10       IMP_JOB           BAD  _nt_  0.009866  0.139778
11    IMP_REASON           BAD  _nt_  0.001280  0.050570
```

- 결정나무 기반 중요도: decisionTree.dtreeTrain

```
cas.loadactionset('decisionTree')
```

```
# 결정 나무 모델 적합 및 변수 중요도 계산
r = cas.decisionTree.dtreeTrain(
    # 입력 데이터
    table=train_impute,

    # 변수 역할 정의
    target=target,
    inputs=imp_xvars,
    nominals=imp_cats+[target],
```

```
    # 결정나무 모델: aStore 형식
    savestate={'name':'dtree_model_astore', 'caslib':'casuser', 'replace': True},

    # 분기 옵션
    crit='GINI',
    maxLevel=10,

    # 가지 치기 옵션
    prune=True,

    # 출력 옵션
    varimp=True
)

# 출력 결과 키 확인
print(r.keys())

# 변수 중요도 확인
print(r['DTreeVarImpInfo'])
```

```
odict_keys(['ModelInfo', 'DTreeVarImpInfo', 'OutputCasTables'])
Decision Tree for TRAIN_IMPUTE_OUT
       Variable  Importance  Std  Count
0   IMP_DEBTINC  415.073278  NaN   11.0
1    IMP_DELINQ  158.798241  NaN    6.0
```

```
2     IMP_CLAGE   45.428891  NaN    8.0
3   IMP_MORTDUE   43.193404  NaN   11.0
4      IMP_CLNO   29.960780  NaN   10.0
5       IMP_YOJ   28.372136  NaN   10.0
6       IMP_JOB   28.332807  NaN    8.0
7      IMP_LOAN   26.403597  NaN    9.0
8      IMP_NINQ   24.658321  NaN    6.0
9     IMP_DEROG   23.696715  NaN    6.0
10    IMP_VALUE   11.759707  NaN    4.0
11   IMP_REASON    2.571429  NaN    1.0
```

구간화 (binning): dataPreprocess.binning

- 비지도학습 구간화

```
# 숫자형 변수 구간화
cas.dataPreprocess.binning(
   # 대상 테이블
   table = train_impute,

    # 입력 변수
    vars = imp_nums,

    # 구간화 방법
    method = 'quantile', #bucket, cutpts, quantile  중 택일

    # 구간화 변수 이름 관련
    outVarsNamePrefix = 'BIN_', # 구간화 변수 이름 접두어

    # 결과 테이블
    casout = dict(name='train_binned', replace=True),

    # 복사 변수
    copyvars = [target] + imp_cats
)
```

```python
# 결과 확인
print(cas.CASTable('train_binned').head())
```

```
Selected Rows from Table TRAIN_BINNED
   BAD IMP_REASON IMP_JOB  BIN__IMP_CLAGE  BIN__IMP_CLNO  BIN__IMP_DEBTINC  BIN__IMP_DELINQ  BIN__IMP_DEROG
BIN__IMP_LOAN  BIN__IMP_MORTDUE  BIN__IMP_NINQ  BIN__IMP_VALUE  BIN__IMP_YOJ
0  1.0    HomeImp   Other             1.0            1.0               3.0              4.0              4.0
1.0               1.0            4.0             1.0           4.0
1  1.0    HomeImp   Other             2.0            2.0               3.0              5.0              4.0
1.0               3.0            3.0             2.0           3.0
2  1.0    HomeImp   Other             2.0            1.0               3.0              4.0              4.0
1.0               1.0            4.0             1.0           2.0
3  0.0    HomeImp  Office             1.0            2.0               3.0              4.0              4.0
1.0               5.0            3.0             4.0           2.0
4  1.0    HomeImp   Other             1.0            1.0               4.0              4.0              4.0
1.0               1.0            3.0             1.0           4.0
```

구간화 (discretize): dataPreprocess.discretize

- 비지도/지도 학습에 의한 구간화

```
# 숫자형 변수 구간화(지도학습)
cas.dataPreprocess.discretize(
   # 대상 테이블
   table = train_impute,

    # 입력 변수
    vars = imp_nums,

    # 목표 변수
    target = target,

    # 구간화 방법
    # "BUCKET" ¦ "CACC" ¦ "CAIM" ¦ "CHIMERGE" ¦ "CUTPTS" ¦ "DTREE" ¦ "MDLP" ¦ "QUANTILE" ¦ "RTREE" ¦ "WOE"
    method = 'dtree',

    # 구간화 방법의 추가 옵션
    arguments = dict(
        # 최대 구간 수
        maxNBins=10
    ),

    # 구간화 변수 이름 접두사
```

```
    outVarsNamePrefix="DIS_",

    # 결과 테이블
    casout = dict(name='train_discretize', replace=True),

    # 복사 변수
    copyvars = [target] + imp_cats,

    # 구간화 정보
    casoutbindetails = dict(name='discretize_info', replace=True),

    # 모델
    code = dict(casout=dict(name='discretize_model', replace=True))

)

# 결과 확인
print(cas.CASTable('train_discretize').head())

# 구간화 정보 확인
print(cas.CASTable('discretize_info').head(12))

# 구간화 모델 확인
print(cas.CASTable('discretize_model').head(999))
```

Selected Rows from Table TRAIN_DISCRETIZE

	BAD	IMP_REASON	IMP_JOB	DIS__IMP_CLAGE	DIS__IMP_CLNO	DIS__IMP_DEBTINC	DIS__IMP_DELINQ	DIS__IMP_DEROG	DIS__IMP_LOAN	DIS__IMP_MORTDUE	DIS__IMP_NINQ	DIS__IMP_VALUE	DIS__IMP_YOJ
0	1.0	HomeImp	Other	2.0	2.0	5.0	1.0	1.0	1.0	1.0	2.0	1.0	8.0
1	1.0	HomeImp	Other	4.0	3.0	5.0	3.0	1.0	1.0	5.0	1.0	4.0	6.0
2	1.0	HomeImp	Other	4.0	2.0	5.0	1.0	1.0	1.0	1.0	2.0	1.0	4.0
3	0.0	HomeImp	Office	2.0	3.0	5.0	1.0	1.0	1.0	7.0	1.0	7.0	4.0
4	1.0	HomeImp	Other	2.0	1.0	6.0	1.0	1.0	1.0	2.0	1.0	2.0	8.0

Selected Rows from Table DISCRETIZE_INFO

	Variable	_BinId_	_BinLowerBnd_	_BinUpperBnd_	_BinWidth_	_NInBin_	_Mean_	_Std_	_Min_	_Max_
0	IMP_CLAGE	1.0	0.000000	81.776349	81.776349	345.0	61.115710	19.507484	0.000000	81.625971
1	IMP_CLAGE	2.0	81.776349	105.141020	23.364671	409.0	93.872694	6.770500	81.840250	105.120709
2	IMP_CLAGE	3.0	105.141020	116.823356	11.682336	243.0	111.217133	3.458833	105.219297	116.796685
3	IMP_CLAGE	4.0	116.823356	151.870363	35.047007	676.0	131.836611	10.022303		

```
116.863449   151.851166
4   IMP_CLAGE      5.0     151.870363     175.235034   23.364671     350.0  164.734592    6.944572
151.933333   175.200960
5   IMP_CLAGE      6.0     175.235034     186.917370   11.682336     452.0  180.794516    2.475257
175.339376   186.805330
6   IMP_CLAGE      7.0     186.917370     210.282041   23.364671     419.0  198.876837    6.829726
186.983043   210.275476
7   IMP_CLAGE      8.0     210.282041     245.329048   35.047007     459.0  227.582186   10.555231
210.435534   245.295163
8   IMP_CLAGE      9.0     245.329048     315.423061   70.094014     577.0  279.078390   21.331432
245.523255   315.374391
9   IMP_CLAGE     10.0     315.423061    1168.233561  852.810499     242.0  376.054402  103.035643
315.466566  1168.233561
10   IMP_CLNO      1.0       0.000000       8.520000    8.520000     282.0    5.021277    2.683727
0.000000     8.000000
11   IMP_CLNO      2.0       8.520000      12.070000    3.550000     405.0   10.629630    1.128440
9.000000    12.000000
Selected Rows from Table DISCRETIZE_MODEL
  ModelName                                          DataStepSrc DS2Src FormatXML FormatItemStore VarXML
AStore AStoreKey ModelUUID Notes
0               _ngbys_ = 1;\n   _igby_ = 0;\n   _tnn_ntran...
```

범주형 변수 구간화: dataPreprocess.catTrans

- 비지도/지도 학습에 의한 범주형 변수 구간화화

```
cas.dataPreprocess.catTrans(
   # 대상 테이블
   table = train_impute,

    # 입력 변수
    vars = imp_cats,

    # 목표 변수
    target = target,

    # 구간화 방법
    # "DTREE" ¦ "GROUPRARE" ¦ "ONEHOT" ¦ "RTREE" ¦ "WOE"
    method = 'GROUPRARE',

    # 구간화 방법의 추가 옵션
    arguments = dict(
        # 최대 구간 수
        maxNBins=10,

        # 희귀 비율
        rareThresholdPercent=5
    ),
```

```
    # 구간화 변수 이름 접두사
    # outVarsNamePrefix="CAT_",

    # 결과 테이블
    casout = dict(name='train_cattrans', replace=True),

    # 복사 변수
    copyvars = [target] + imp_nums,

    # 구간화 정보
    casoutbindetails = dict(name='cattrans_info', replace=True),

    # 모델
    code = dict(casout=dict(name='cattrans_model', replace=True))
)
```

```
Selected Rows from Table TRAIN_CATTRANS
   BAD  IMP_LOAN  IMP_MORTDUE  IMP_CLNO  IMP_VALUE  IMP_YOJ  IMP_DEBTINC  IMP_NINQ   IMP_CLAGE  IMP_DEROG
IMP_DELINQ  CAT_IMP_JOB  CAT_IMP_REASON
0  1.0    1100.0      25860.0       9.0    39025.0     10.5    33.849881       1.0   94.366667        0.0
0.0          1.0             2.0
1  1.0    1300.0      70053.0      14.0    68400.0      7.0    33.849881       0.0  121.833333        0.0
2.0          1.0             2.0
2  1.0    1500.0      13500.0      10.0    16700.0      4.0    33.849881       1.0  149.466667        0.0
```

```
0.0          1.0             2.0
3  0.0    1700.0      97800.0      14.0   112000.0      3.0    33.849881       0.0   93.333333        0.0
0.0          3.0             2.0
4  1.0    1800.0      28502.0       8.0    43034.0     11.0    36.884894       0.0   88.766030        0.0
0.0          1.0             2.0
Selected Rows from Table CATTRANS_INFO
   _Variable_  _BinId_  _NLevelsInBin_  _NInBin_
0     IMP_JOB      1.0             1.0    1835.0
1     IMP_JOB      2.0             1.0     910.0
2     IMP_JOB      3.0             1.0     657.0
3     IMP_JOB      4.0             1.0     551.0
4     IMP_JOB      5.0             1.0     142.0
5     IMP_JOB      6.0             1.0      77.0
6  IMP_REASON      1.0             1.0    2911.0
7  IMP_REASON      2.0             1.0    1261.0
Selected Rows from Table CATTRANS_MODEL
  ModelName                                      DataStepSrc DS2Src FormatXML FormatItemStore VarXML
AStore AStoreKey ModelUUID Notes
0               _ngbys_ = 1;\n   _igby_ = 0;\n   _tcn_ntran...
```

- 최종적인 결과는 범주형 변수를 적절히 변환 후 다시 구간화를 진행함

모델 구성

그래디언트 부스팅 모델: lightGradBoost.lgbmTrain

- 액션셋 불러오기

```
# 액션셋 불러오기
cas.loadactionset('lightGradBoost')
```

- 모델 적합 및 저장

```
test_impute = cas.CASTable('test_impute_out')
# Light GBM 적합
r = cas.lightGradBoost.lgbmTrain(
    table = train_impute,

    inputs = imp_xvars,
    nominals = imp_cats + [target],
    target = target,

   # 모델 옵션: "DART" ¦ "GBDT" ¦ "GOSS" ¦ "RF" 중 택일
    boosting ='GOSS',

    # 이진 모델 지정
    objective = 'BINARY',
```

```
    maxIters = 300,

    # 검증 테이블
    validTable = test_impute,

    #  모델 저장
    saveState = {'name':'lgb_model_astore', 'replace':True}
)

# 결과 확인
print(r.keys())
print(r['IterHistory'])
```

```
odict_keys(['ModelInfo', 'NObs', 'IterHistory', 'PredName'])
     numberOfTrees  trainingAccuracyMetric  validationAccuracyMetric
0                1                0.452361                  0.455040
1                2                0.418618                  0.422725
2                3                0.391937                  0.398761
3                4                0.370340                  0.378856
4                5                0.352573                  0.363430
..             ...                     ...                       ...
295            296                0.007895                  0.230570
296            297                0.007828                  0.230816
297            298                0.007735                  0.231254
298            299                0.007665                  0.231793
```

```
299            300                0.007585                 0.231939
[300 rows x 3 columns]
```

#실습

1. IterHistory를 이용하여 모델이 충분히 학습되었는지 확인해보자.

- 모델 테이블

```
print(cas.CASTable('lgb_model_astore').head(999))
```

```
Selected Rows from Table LGB_MODEL_ASTORE
   _index_                                              _state_
0        0  b'\x18\x1f\x10\x113"\x0033\x01\x021\x013\x01#3...
```

- 이진 파일 형식임
- 모델 저장

```
# 모델 저장
cas.table.save(
    # 저장할 데이터
    table={'name': 'lgb_model_astore', 'caslib':'casuser'},

    # 저장 이름 등
```

```
    name='lgb_model_astore',
    caslib='casuser',
    replace=True,
)
```

#실습

1. 저장된 모델을 확인해보자.(fileinfo)
2. 모델을 다시 호출해보자. (loadtable)

- 모델 호출 및 스코어링: aStore.score

```
cas.loadactionset('aStore')
```

```
# 점수 산출출: ASTRE 모델 이용
cas.astore.score(
    # 점수 산출 대상 파일 지정
    table=test_impute,

    # ATORE 모델 지정
    rstore='lgb_model_astore',

    # 복제할 변수 지정
    copyvars=[target],
```

```
    # 점수 산출 저장 테이블 지정
    casout={'name': 'lgb_score_out', 'replace': True}
)

# 결과 확인
print(cas.CASTable('lgb_score_out').head())
```

```
Selected Rows from Table LGB_SCORE_OUT
  I_BAD    P_BAD0    P_BAD1  BAD
0     1  0.175871  0.824129  1.0
1     1  0.015671  0.984329  1.0
2     1  0.001966  0.998034  1.0
3     1  0.000311  0.999689  1.0
4     1  0.305604  0.694396  1.0
```

#실습

1. 예측된 확률은 어떤 변수에 저장되어 있는지 확인해보자.
2. 예측 변수 이름 규칙이 있는지 확인해보자.

- 모델 평가: percentile.assess

```
cas.loadactionset('percentile')
```

```python
# 모델 평가
r_lgb = cas.percentile.assess(
    # 평가를 위한 데이터 지정
    table='lgb_score_out',

    # 실제 목표변수 지정
    response=target,

    # 예측 변수 지정
    inputs=[{"name": 'P_' + target + event}],

    # 이벤트 값 지정
    event=event,
)

# 결과 확인
print(r_lgb.keys())
print(r_lgb['LIFTInfo'].head())
print(r_lgb['ROCInfo'].head())
```

```
odict_keys(['LIFTInfo', 'ROCInfo'])
Lift Information for LGB_SCORE_OUT
  Variable Event  Depth     Value  NObs  NEvents  NEventsBest      Resp   RespBest      Lift  LiftBest
CumResp  CumRespBest   CumLift  CumLiftBest     PctResp  PctRespBest  CumPctResp  CumPctRespBest      Gain
GainBest
```

```
0   P_BAD1     1    5.0  0.994631  90.0     90.0         90.0  25.210084  25.210084  5.042017  5.042017
25.210084    25.210084  5.042017     5.042017  100.000000   100.000000  100.000000      100.000000  4.042017
4.042017
1   P_BAD1     1   10.0  0.944185  90.0     86.0         90.0  24.089636  25.210084  4.817927  5.042017
49.299720    50.420168  4.929972     5.042017   95.555556   100.000000   97.777778      100.000000  3.929972
4.042017
2   P_BAD1     1   15.0  0.707454  90.0     67.0         90.0  18.767507  25.210084  3.753501  5.042017
68.067227    75.630252  4.537815     5.042017   74.444444   100.000000   90.000000      100.000000  3.537815
4.042017
3   P_BAD1     1   20.0  0.294185  90.0     42.0         87.0  11.764706  24.369748  2.352941  4.873950
79.831933   100.000000  3.991597     5.000000   46.666667    96.666667   79.166667       99.166667  2.991597
4.000000
4   P_BAD1     1   25.0  0.083615  90.0     33.0          0.0   9.243697   0.000000  1.848739  0.000000
89.075630   100.000000  3.563025     4.000000   36.666667     0.000000   70.666667       79.333333  2.563025
3.000000
ROC Information for LGB_SCORE_OUT
  Variable Event  CutOff     TP      FP    FN      TN  Sensitivity  Specificity   KS       KS2    F_HALF
FPR       ACC       FDR        F1         C      Gini     Gamma       Tau  MISCEVENT       FNR
0   P_BAD1     1    0.00  357.0  1431.0   0.0     0.0     1.000000     0.000000  0.0  0.000000  0.237715
1.000000  0.199664  0.800336  0.332867  0.946649  0.893297  0.936176  0.285655   0.800336  0.000000
1   P_BAD1     1    0.01  337.0   307.0  20.0  1124.0     0.943978     0.785465  0.0  0.729442  0.574497
0.214535  0.817114  0.476708  0.673327  0.946649  0.893297  0.936176  0.285655   0.182886  0.056022
2   P_BAD1     1    0.02  333.0   221.0  24.0  1210.0     0.932773     0.845563  0.0  0.778336  0.647105
```

```
0.154437  0.862975  0.398917  0.731065  0.946649  0.893297  0.936176  0.285655   0.137025  0.067227
3   P_BAD1     1    0.03  332.0   193.0  25.0  1238.0     0.929972     0.865129  0.0  0.795101  0.675621
0.134871  0.878076  0.367619  0.752834  0.946649  0.893297  0.936176  0.285655   0.121924  0.070028
4   P_BAD1     1    0.04  329.0   177.0  28.0  1254.0     0.921569     0.876310  0.0  0.797879  0.690886
0.123690  0.885347  0.349802  0.762457  0.946649  0.893297  0.936176  0.285655   0.114653  0.078431
```

- 모델 평가 기준

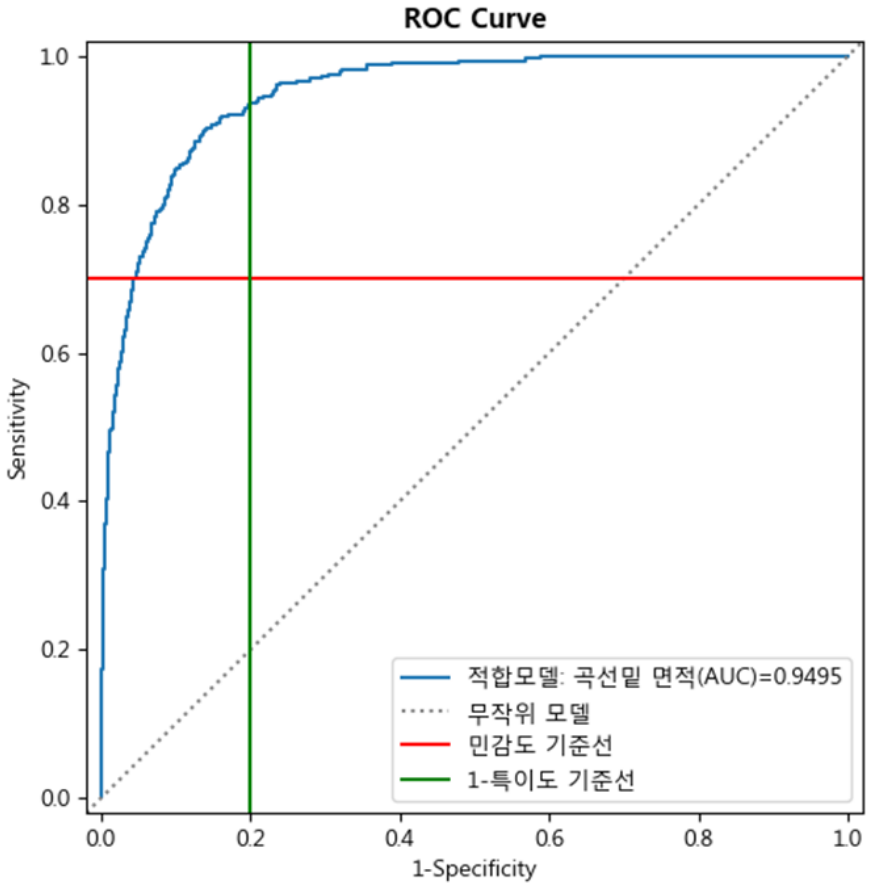
`혼동 행렬(confusion matrix)`

		실제 값(observed)	
예측 값(predicted)	Positive	Negative	합계
Positive	TP (True Positive)	FP (False Positive) 잘못 이벤트예측	TP+FP
Negative	FN (False Negative) 잘못 비이벤트예측	TN (True Negative)	FN+TN
합계	TP+FN	FP+TN	TP+FP+FN+TN

`평가 측도`

기준	계산식
오분류율 (misclassification rate)	$\frac{FP+FN}{TotalCount}$
정분류율 (accuracy)	$\frac{TP+TN}{TotalCount}$
정밀도 (precision)	$\frac{TP}{TP+FP}$
민감도, 재현율 (True Positive Rate (TPR, sensitivity, recall))	$\frac{TP}{TP+FN}$
특이도 (True Negative Rate (TNR, specificity))	$\frac{TN}{FP+TN}$
1-특이도 (False Positive Rate (FPR))	$\frac{FP}{FP+TN}$
F1 점수	$\frac{2}{1/Recall+1/Precision} = 2 \times \frac{Precision \times Recall}{Precision+Recall}$

ROC 곡선



#실습

1. ROC 곡선 밑의 면적(AUC)의 의미를 생각해보자.