1 System Environment

1) Mount this file on Google drive

In [ ]:

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

Mounted at /content/drive

2) Data preparation : [Important] Change the pass for renaldata.csv

In [ ]:

```
# Loading modules
import numpy as np
import pandas as pd
%matplotlib inline
import seaborn as sns

# Change the pass: 'drive/My Drive/program/data/renaldata.clsv' to a new pass where renaldata.csv
 is located in your enviromnent
df = pd.read_csv('drive/My Drive/program/data/renaldata.csv')
# print('Number of colums and rows')
# print(df.shape)
# print()
print('Number of empty cells')
pd.set_option('display.max_rows', 1000)
df.isnull().sum()
```

Number of empty cells

Out[ ]:

```
RMale                     0
Rage                      0
InfectionCount            0
FeverOnly                 0
Pyrexia                   0
Inflammation              0
VirusInfection            0
CMV                       0
anemia                    0
HeartDisease              0
RespiratoryInfection      0
UpperRespiratoryInfection 0
UpperDigestivetract       0
Diarrhea                  0
UTI                       0
WBCinUrine                0
WBCpeakover10             0
urology                   0
Skin                      0
WoundInfection            0
HerpesZoster              0
Orthopedics               0
Ascites                   0
Surgery                   0
AerobicGPC                0
AerobicGNR                0
candida                   0
staphylococcusaureus      0
streptococcusaureus       0
enterobacteraerogenes     0
enterobactereclacue       0
enterococcusfaecalis      0
citobacterdiversus        0
pseudomonas               0
inflammationdatefirst     0
infectiondatelast         0
asthma                    0
pastanemia                0
infarctionhemohorrage     0
calcification             0
digestiveorgan            0
appendicitis              0
polyp                     0
ulcer                     0
GERD                      0
pastheart                 0
kidney                    0
pastliverbilialy          0
HBV                       0
HCV                       0
stone                     0
hypothyroidism            0
gynecology                0
ocular                    0
allergy                   0
hypertention              0
type2DM                   0
BTF                       1
timeoftransplantation     0
```

Regraft	0
ABOI	0
HLAABmm	0
HLADRmm	0
HLAmm	0
PRAclass1pre	0
PRAclass2pre	0
PRAclass1after	0
PRAclass2after	0
MFImax	0
twinpeak	1
DSAclass1	0
DSAclass1number	1
DSAclass2	0
DSAclass2number	0
preDSA	1
denovoDSA	1
antiHLAclass1	0
HSAclass1number	0
antiHLAclass2	0
HSAclass2number	0
denovoDSAandHSAclass2	0
denovoDSAandHSAclass1	0
preformedDSAandHLAclass2	0
preformedDSAandHLAclass1	0
A23	0
A25	0
A26	0
A32	0
A34	0
A66	0
B13	0
B18	0
B27	0
B35	0
B37	0
B38	0
B42	0
B44	0
B45	0
B47	0
B49	0
B50	0
B51	0
B52	0
B53	0
B54	0
B55	0
B56	0
B57	0
B58	0
B59	0
B61	0
B62	0
B63	0
B7	0
B71	0
B73	0
B75	0
B77	0
B78	0

B8	0
B82	0
Cw17	0
Cw6	0
Cw9	0
DP10	0
DP11	0
DP13	0
DP14	0
DP15	0
DP17	0
DP18	0
DP19	0
DP20	0
DP3	0
DP4	0
DP5	0
DP6	0
DP9	0
DQ11	0
DQ2	0
DQ4	0
DQ5	0
DQ6	0
DQ7	0
DQ8	0
DQ9	0
DR1	0
DR10	0
DR103	0
DR11	0
DR12	0
DR13	0
DR14	0
DR15	0
DR16	0
DR17	0
DR18	0
DR4	0
DR51	0
DR52	0
DR53	0
DR7	0
DR8	0
DR9	0
pregnancyhistory	0
birthhistory	0
NaturalAbortion	0
ArtificialAbortion	0
HDperiod	0
CGN	0
IgA	0
NS	0
hypoplastickidney	0
MalignantHypertention	0
Banfi	0
Banft	0
Banfg	0
Banfv	0
Banfci	0
Banfct	0

```
Banfcv                    0
Banfcg                    0
Banfptc                   0
Banfptcbm                 0
Banfah                    0
Banfaah                   0
InterstitialHemorrhage    0
CellInvasion              0
lymphinvasion             0
thrombusformation         0
coaglationnecrosis        0
IgA.1                     0
IgM                       0
IgG                       0
SABC1q                    0
C3                        0
C4d                       0
C5b                       0
bulbarsclerosis           0
CRPpreRej                 0
CRPpostRej                0
WBCpeakover5              0
MaxCRP                    0
WBCpreRej                 0
WBCpostKTx                0
WBCpeakover9postRej       0
MaxWBC                    0
MMFpostRej                0
MMFatRej                  7
CNIpostRej                2
GraftLoss                 0
dtype: int64
```

In [ ]:

```
# Interpolation of missing data    df2: pandas data、    data: numpy data
df2 = df.fillna(df.median())
data = df2.values

df2.isnull().sum()
```

Out[ ]:

```
RMale                     0
Rage                      0
InfectionCount            0
FeverOnly                 0
Pyrexia                   0
Inflammation              0
VirusInfection            0
CMV                       0
anemia                    0
HeartDisease              0
RespiratoryInfection      0
UpperRespiratoryInfection 0
UpperDigestivetract       0
Diarrhea                  0
UTI                       0
WBCinUrine                0
WBCpeakover10             0
urology                   0
Skin                      0
WoundInfection            0
HerpesZoster              0
Orthopedics               0
Ascites                   0
Surgery                   0
AerobicGPC                0
AerobicGNR                0
candida                   0
staphylococcusaureus      0
streptococcusaureus       0
enterobacteraerogenes     0
enterobactereclacue       0
enterococcusfaecalis      0
citobacterdiversus        0
pseudomonas               0
inflammationdatefirst     0
infectiondatelast         0
asthma                    0
pastanemia                0
infarctionhemohorrage     0
calcification             0
digestiveorgan            0
appendicitis              0
polyp                     0
ulcer                     0
GERD                      0
pastheart                 0
kidney                    0
pastliverbilialy          0
HBV                       0
HCV                       0
stone                     0
hypothyroidism            0
gynecology                0
ocular                    0
allergy                   0
hypertention              0
type2DM                   0
BTF                       0
timeoftransplantation     0
```

```
Regraft                 0
ABOI                    0
HLAABmm                 0
HLADRmm                 0
HLAmm                   0
PRAclass1pre            0
PRAclass2pre            0
PRAclass1after          0
PRAclass2after          0
MFImax                  0
twinpeak                0
DSAclass1               0
DSAclass1number         0
DSAclass2               0
DSAclass2number         0
preDSA                  0
denovoDSA               0
antiHLAclass1           0
HSAclass1number         0
antiHLAclass2           0
HSAclass2number         0
denovoDSAandHSAclass2   0
denovoDSAandHSAclass1   0
preformedDSAandHLAclass2  0
preformedDSAandHLAclass1  0
A23                     0
A25                     0
A26                     0
A32                     0
A34                     0
A66                     0
B13                     0
B18                     0
B27                     0
B35                     0
B37                     0
B38                     0
B42                     0
B44                     0
B45                     0
B47                     0
B49                     0
B50                     0
B51                     0
B52                     0
B53                     0
B54                     0
B55                     0
B56                     0
B57                     0
B58                     0
B59                     0
B61                     0
B62                     0
B63                     0
B7                      0
B71                     0
B73                     0
B75                     0
B77                     0
B78                     0
```

B8	0
B82	0
Cw17	0
Cw6	0
Cw9	0
DP10	0
DP11	0
DP13	0
DP14	0
DP15	0
DP17	0
DP18	0
DP19	0
DP20	0
DP3	0
DP4	0
DP5	0
DP6	0
DP9	0
DQ11	0
DQ2	0
DQ4	0
DQ5	0
DQ6	0
DQ7	0
DQ8	0
DQ9	0
DR1	0
DR10	0
DR103	0
DR11	0
DR12	0
DR13	0
DR14	0
DR15	0
DR16	0
DR17	0
DR18	0
DR4	0
DR51	0
DR52	0
DR53	0
DR7	0
DR8	0
DR9	0
pregnancyhistory	0
birthhistory	0
NaturalAbortion	0
ArtificialAbortion	0
HDperiod	0
CGN	0
IgA	0
NS	0
hypoplastickidney	0
MalignantHypertention	0
Banfi	0
Banft	0
Banfg	0
Banfv	0
Banfci	0
Banfct	0

```
Banfcv                  0
Banfcg                  0
Banfptc                 0
Banfptcbm               0
Banfah                  0
Banfaah                 0
InterstitialHemorrhage  0
CellInvasion            0
lymphinvasion           0
thrombusformation       0
coaglationnecrosis      0
IgA.1                   0
IgM                     0
IgG                     0
SABC1q                  0
C3                      0
C4d                     0
C5b                     0
bulbarsclerosis         0
CRPpreRej               0
CRPpostRej              0
WBCpeakover5            0
MaxCRP                  0
WBCpreRej               0
WBCpostKTx              0
WBCpeakover9postRej     0
MaxWBC                  0
MMFpostRej              0
MMFatRej                0
CNIpostRej              0
GraftLoss               0
dtype: int64
```

In [ ]:

```
# Dividing explanatory variables and target variable
x = df2.iloc[:, :-1].values
t = df2.iloc[:, -1].values

# splitting the data
from sklearn.model_selection import train_test_split
x_train, x_test, t_train, t_test = train_test_split(x, t, test_size=0.3, random_state=0)
```

3) Installation of Optuna and XGBoost

In [ ]:

```
# Installation of Optuna
!pip install optuna
```

```
Collecting optuna
  Downloading https://files.pythonhosted.org/packages/9f/b1/a5f0574fa0d769bf0
a5629722c84bb0af015967191a37401fe8508c5fb6a/optuna-2.1.0.tar.gz (232kB)
     |████████████████████████████████████████████████████████████████| 235kB 4.7MB/
s
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
    Preparing wheel metadata ... done
Collecting alembic
  Downloading https://files.pythonhosted.org/packages/12/aa/c261dfd7f4ba6ce47
01846a2689a46e2a172e012171de4378fc2926e3bf0/alembic-1.4.3-py2.py3-non
e-any.whl (159kB)
     |████████████████████████████████████████████████████████████████| 163kB 11.2M
B/s
Requirement already satisfied: joblib in /usr/local/lib/python3.6/dist-packages (from
optuna) (0.16.0)
Requirement already satisfied: sqlalchemy>=1.1.0 in /usr/local/lib/python3.6/dist-pa
ckages (from optuna) (1.3.19)
Collecting cmaes>=0.6.0
  Downloading https://files.pythonhosted.org/packages/ee/03/5d15a78ca92ac2bf0
9f466c54c48bb92979dfe19add2dfed415133ba9792/cmaes-0.6.1-py3-none-any.
whl
Requirement already satisfied: scipy!=1.4.0 in /usr/local/lib/python3.6/dist-packages
(from optuna) (1.4.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from
optuna) (4.41.1)
Collecting cliff
  Downloading https://files.pythonhosted.org/packages/71/06/03b1f92d46546a18
eabf33ff7f37ef422c18c93d5a926bf590fee32ebe75/cliff-3.4.0-py3-none-any.whl
(76kB)
     |████████████████████████████████████████████████████████████████| 81kB 5.4MB/s
Collecting colorlog
  Downloading https://files.pythonhosted.org/packages/2a/81/12d77537c82c5d46
aa2721dfee25a0e873ef5920ebd0827152f411effb57/colorlog-4.2.1-py2.py3-none
-any.whl
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (fro
m optuna) (1.18.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.6/dist-pac
kages (from optuna) (20.4)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-packa
ges (from alembic->optuna) (2.8.1)
Collecting Mako
  Downloading https://files.pythonhosted.org/packages/a6/37/0e706200d22172eb
8fa17d68a7ae22dec7631a0a92266634fb518a88a5b2/Mako-1.1.3-py2.py3-none-
any.whl (75kB)
     |████████████████████████████████████████████████████████████████| 81kB 6.7MB/s
Collecting python-editor>=0.3
  Downloading https://files.pythonhosted.org/packages/c6/d3/201fc3abe391bbae6
606e6f1d598c15d367033332bd54352b12f35513717/python_editor-1.0.4-py3-n
one-any.whl
Collecting pbr!=2.1.0,>=2.0.0
  Downloading https://files.pythonhosted.org/packages/c1/a3/d439f338aa90edd5a
d9096cd56564b44882182150e92148eb14ceb7488ba/pbr-5.5.0-py2.py3-none-a
ny.whl (106kB)
     |████████████████████████████████████████████████████████████████| 112kB 14.8M
B/s
Collecting cmd2!=0.8.3,>=0.8.0
  Downloading https://files.pythonhosted.org/packages/b2/28/0d604c8ccbf2d677
d3d2c8025724cfa60f10496c0e417e0ab5a22d23c869/cmd2-1.3.10-py3-none-an
y.whl (132kB)
     |████████████████████████████████████████████████████████████████| 133kB 12.9M
```

```
B/s
Requirement already satisfied: PyYAML>=3.12 in /usr/local/lib/python3.6/dist-packa
ges (from cliff->optuna) (3.13)
Requirement already satisfied: pyparsing>=2.1.0 in /usr/local/lib/python3.6/dist-pac
kages (from cliff->optuna) (2.4.7)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages
(from cliff->optuna) (1.15.0)
Collecting stevedore>=2.0.1
  Downloading https://files.pythonhosted.org/packages/b8/a1/004f04ba411a8002
b02aadb089fd6868116c12ddc9f6d576175e89d07587/stevedore-3.2.2-py3-none
-any.whl (42kB)
     |████████████████████████████████| 51kB 4.5MB/s
Requirement already satisfied: PrettyTable<0.8,>=0.7.2 in /usr/local/lib/python3.6/d
ist-packages (from cliff->optuna) (0.7.2)
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.6/dist-pa
ckages (from Mako->alembic->optuna) (1.1.1)
Collecting pyperclip>=1.6
  Downloading https://files.pythonhosted.org/packages/f6/5b/55866e1cde0f86f5e
ec59dab5de8a66628cb0d53da74b8dbc15ad8dabda3/pyperclip-1.8.0.tar.gz
Requirement already satisfied: setuptools>=34.4 in /usr/local/lib/python3.6/dist-pac
kages (from cmd2!=0.8.3,>=0.8.0->cliff->optuna) (50.3.0)
Requirement already satisfied: attrs>=16.3.0 in /usr/local/lib/python3.6/dist-packag
es (from cmd2!=0.8.3,>=0.8.0->cliff->optuna) (20.2.0)
Requirement already satisfied: importlib-metadata>=1.6.0; python_version < "3.8" in
/usr/local/lib/python3.6/dist-packages (from cmd2!=0.8.3,>=0.8.0->cliff->optuna)
(1.7.0)
Requirement already satisfied: wcwidth>=0.1.7 in /usr/local/lib/python3.6/dist-pack
ages (from cmd2!=0.8.3,>=0.8.0->cliff->optuna) (0.2.5)
Collecting colorama>=0.3.7
  Downloading https://files.pythonhosted.org/packages/c9/dc/45cdef1b4d119eb96
316b3117e6d5708a08029992b2fee2c143c7a0a5cc5/colorama-0.4.3-py2.py3-no
ne-any.whl
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-packages (f
rom importlib-metadata>=1.6.0; python_version < "3.8"->cmd2!=0.8.3,>=0.8.0->clif
f->optuna) (3.1.0)
Building wheels for collected packages: optuna
  Building wheel for optuna (PEP 517) ... done
  Created wheel for optuna: filename=optuna-2.1.0-cp36-none-any.whl size=321090
sha256=c0504c895410e56740b4b4dda5d4f24e2700a1340d06e1391777a8918
2a15d00
  Stored in directory: /root/.cache/pip/wheels/9f/25/24/a165483933b5eefbf4f93c
85f3188dc696cbb38620b73ad713
Successfully built optuna
Building wheels for collected packages: pyperclip
  Building wheel for pyperclip (setup.py) ... done
  Created wheel for pyperclip: filename=pyperclip-1.8.0-cp36-none-any.whl size=8693
sha256=e7b553fae2875673e98b458a3470fc475033f60cf3f5b44ff44f29b465ce
6c59
  Stored in directory: /root/.cache/pip/wheels/b2/ac/0a/b784f0afe26eaf52e88a7e
15c7369090deea0354fa1c6fc689
Successfully built pyperclip
Installing collected packages: Mako, python-editor, alembic, cmaes, pbr, pyperclip, col
orama, cmd2, stevedore, cliff, colorlog, optuna
Successfully installed Mako-1.1.3 alembic-1.4.3 cliff-3.4.0 cmaes-0.6.1 cmd2-1.3.10
colorama-0.4.3 colorlog-4.2.1 optuna-2.1.0 pbr-5.5.0 pyperclip-1.8.0 python-editor-
1.0.4 stevedore-3.2.2
```

In [ ]:

```
# Installation of XGBoost
!pip3 install xgboost
!pip3 install -q pydot
!pip3 install graphviz
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.6/dist-packages (0.
90)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (fro
m xgboost) (1.18.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from
xgboost) (1.4.1)
Requirement already satisfied: graphviz in /usr/local/lib/python3.6/dist-packages (0.
10.1)
```

2. Leave-One-Out cross validation with machine learning models

1) Simple Linear Regression ; ACU 0.498

In [ ]:

```
def linear():
    import numpy as np
    import pandas as pd
    from sklearn.linear_model import LinearRegression
    from sklearn.model_selection import LeaveOneOut

    # Statistical functions
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import r2_score

    # Rounding of fractional point
    import math
    def my_round(val, digit=0):
        p = 10 ** digit
        return (val * p * 2 + 1) // 2 / p


    x = []
    t = []

    # Predicted value
    pred = []
    round = []
    pairs = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)

    x = np.array(x)
    t = np.array(t)


    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)
    lr = LinearRegression()

    for train_index, test_index in loo.split(x):
        x_train, x_test = x[train_index], x[test_index]
        t_train, t_test = t[train_index], t[test_index]
        lr.fit(x_train, t_train)
        result = lr.predict(x_test)
        pred.append(result[0])

        # Category of prediction
        if (result[0]<0.5):
            round.append(int(0))
        else:
            round.append(int(1))

    for i in range(len(pred)):
        pairs.append([pred[i], t[i]])

    output = pd.DataFrame(pairs)
    # output.to_csv('drive/My Drive/program/data/cstatdata.csv', index=False)
```

```
    # Coefficient of Determination
    r2 = r2_score(t, pred)
    # AUC
    auc = roc_auc_score(t, pred)

    # Confusion Matrix
    TP = 0
    FP = 0
    FN = 0
    TN = 0

    for i in range(len(round)):
        if (round[i]==1) & (t[i]==1):
            TP += 1
        elif (round[i]==1) & (t[i]==0):
            FP += 1
        elif (round[i]==0) & (t[i]==1):
            FN += 1
        elif (round[i]==0) & (t[i]==0):
            TN += 1

    confmat =  [[TP, FP], [FN, TN]]

    acc = (TP+TN)/(TP+FP+FN+TN)

    if TP + FP != 0:
        prec = TP/(TP+FP)
    elif TP + FP == 0:
        prec = 'N/A'
    if TP + FN != 0:
        rec = TP/(TP+FN)
    elif TP + FP ==0:
        rec = 'N/A'
    if TN + FP != 0:
        spec = TN/(TN + FP)
    elif TN + FP == 0:
        spec = 'N/A'
    if TP + FP/2 + FN/2 != 0:
        f1 = TP/(TP + FP/2 + FN/2)
    elif TP + FP/2 + FN/2 == 0:
        f1 = 'N/A'

    res = np.array([auc, acc, prec, rec, f1, spec, [[TP, FP], [FN, TN]], r2])

    # Output : ( 1. AUC, 2, Accuracy, 3. Precision 4. Recall 5. f1-score 6. Specificity, 7. Confusion Matri
x, 8. R^2 in LOO)

    print('■ Leave-One-Out Cross Validation with Simple Linear Regression')
    print('')
    print('Sample size of all dataset: ' + str(len(x_train) + len(x_test)) )
    print('Sample size of training data: ' + str(len(x_train)) + '    Explanatory variables: ' + str(x_train.
shape) + '   Target value: ' + str(t_train.shape))
    print('Sample size of test data: ' + str(len(x_test)) + '    Explanatory variables: ' + str(x_test.shap
e) + '   Target value: ' + str(t_test.shape))
    print('')
    print('AUC: ' + str(my_round(auc, 5)) + '   Accuracy: ' + str(my_round(acc,5)) + '   R2: ' + str(my_
round(r2,5)))
    print('')
    print(pd.DataFrame(np.array(confmat), index=['Predict True', 'Predict False'], columns=['Actual T
rue', 'Actual False']))
    print('')
```

```
    print('Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confution matr
ix 8. R^2 in LOO')
    print('')


    return res
```

In [ ]:

```
linear()
```

■ Leave-One-Out Cross Validation with Simple Linear Regression

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.49773 Accuracy: 0.62745 R2: -2.04515

	Actual True	Actual False
Predict True	3	11
Predict False	8	29

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.49772727272727274, 0.6274509803921569, 0.21428571428571427,
       0.2727272727272727, 0.24, 0.725, list([[3, 11], [8, 29]]),
       -2.04515491157451], dtype=object)
```

2) Lasso Regression ; AUC 0.734

In [ ]:

```
def lasso(a):
    import numpy as np
    import pandas as pd
    from sklearn.linear_model import Lasso
    from sklearn.model_selection import LeaveOneOut

    # Statistical functions
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import r2_score

    # Rounding of fractional point
    import math
    def my_round(val, digit=0):
        p = 10 ** digit
        return (val * p * 2 + 1) // 2 / p

    x = []
    t = []

    # Predicted value
    pred = []
    round = []
    pairs = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)

    x = np.array(x)
    t = np.array(t)

    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)
    lr = Lasso(alpha=a)

    for train_index, test_index in loo.split(x):
        x_train, x_test = x[train_index], x[test_index]
        t_train, t_test = t[train_index], t[test_index]
        lr.fit(x_train, t_train)
        result = lr.predict(x_test)
        pred.append(result[0])

            # Category of prediction
        if (result[0]<0.5):
            round.append(int(0))
        else:
            round.append(int(1))

    for i in range(len(pred)):
        pairs.append([pred[i], t[i]])

    output = pd.DataFrame(pairs)
    # output.to_csv('drive/My Drive/program/data/cstatdata.csv', index=False)

    # Coefficient of Determination
    r2 = r2_score(t, pred)
```

```
    # AUC
    auc = roc_auc_score(t, pred)

    # Confusion Matrix
    TP = 0
    FP = 0
    FN = 0
    TN = 0

    for i in range(len(round)):
        if (round[i]==1) & (t[i]==1):
            TP += 1
        elif (round[i]==1) & (t[i]==0):
            FP += 1
        elif (round[i]==0) & (t[i]==1):
            FN += 1
        elif (round[i]==0) & (t[i]==0):
            TN += 1

    confmat = [[TP, FP], [FN, TN]]

    acc = (TP+TN)/(TP+FP+FN+TN)

    if TP + FP != 0:
        prec = TP/(TP+FP)
    elif TP + FP == 0:
        prec = 'N/A'
    if TP + FN != 0:
        rec = TP/(TP+FN)
    elif TP + FP ==0:
        rec = 'N/A'
    if TN + FP != 0:
        spec = TN/(TN + FP)
    elif TN + FP == 0:
        spec = 'N/A'
    if TP + FP/2 + FN/2 != 0:
        f1 = TP/(TP + FP/2 + FN/2)
    elif TP + FP/2 + FN/2 == 0:
        f1 = 'N/A'

    res = np.array([auc, acc, prec, rec, f1, spec, confmat, r2])

    # Output : ( 1. AUC, 2, Accuracy, 3. Precision 4. Recall 5. f1-score 6. Specificity, 7. Confusion Matri
x, 8. R^2 in LOO)

    print('■ Leave-One-Out Cross Validation with Lasso Regression')
    print('')
    print('Sample size of all dataset: ' + str(len(x_train) + len(x_test)) )
    print('Sample size of training data: ' + str(len(x_train)) + '    Explanatory variables: ' + str(x_train.
shape) + '   Target value: ' + str(t_train.shape))
    print('Sample size of test data: ' + str(len(x_test)) + '    Explanatory variables: ' + str(x_test.shap
e) + '   Target value: ' + str(t_test.shape))
    print('')
    print('AUC: ' + str(my_round(auc, 5)) + '   Accuracy: ' + str(my_round(acc,5)) + '   R2: ' + str(my_
round(r2,5)))
    print('')
    print(pd.DataFrame(np.array(confmat), index=['Predict True', 'Predict False'], columns=['Actual T
rue', 'Actual False']))
    print('')
    print('Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confution matr
ix 8. R^2 in LOO')
```

```
    print('')

    return res
```

In [ ]:

```
lasso(0.1)
```

■ Leave-One-Out Cross Validation with Lasso Regression

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.66591 Accuracy: 0.72549 R2: -0.19725

	Actual True	Actual False
Predict True	2	5
Predict False	9	35

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.665909090909091, 0.7254901960784313, 0.2857142857142857,
       0.18181818181818182, 0.2222222222222222, 0.875,
       list([[2, 5], [9, 35]]), -0.19725016235878678], dtype=object)
```

In [ ]:

```
lasso(0.01)
```

■ Leave-One-Out Cross Validation with Lasso Regression

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.69318 Accuracy: 0.66667 R2: -0.72191

	Actual True	Actual False
Predict True	4	10
Predict False	7	30

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.6931818181818181, 0.6666666666666666, 0.2857142857142857,
       0.36363636363636365, 0.32, 0.75, list([[4, 10], [7, 30]]),
       -0.7219084222358687], dtype=object)
```

In [ ]:

```
lasso(10)
```

■ Leave-One-Out Cross Validation with Lasso Regression

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (50,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.43409 Accuracy: 0.76471 R2: -0.06329

	Actual True	Actual False
Predict True	0	1
Predict False	11	39

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confution matrix 8. R^2 in LOO

Out[ ]:

```
array([0.434090909090909, 0.7647058823529411, 0.0, 0.0, 0.0, 0.975,
       list([[0, 1], [11, 39]]), -0.06329137947570573], dtype=object)
```

Optimization with Optuna

In [ ]:

```
def lassoOptuna(trial):
    import numpy as np
    import pandas as pd
    from sklearn.linear_model import Lasso
    from sklearn.model_selection import LeaveOneOut

    import optuna

    # Statistical functions
    from sklearn.metrics import roc_auc_score

    alpha = trial.suggest_uniform('alpha', 0.01, 1)

    x = []
    t = []

    # Rounding of fractional point
    pred = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)
    x = np.array(x)
    t = np.array(t)

    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)
    lr = Lasso(alpha=alpha)

    for train_index, test_index in loo.split(x):
        x_train, x_test = x[train_index], x[test_index]
        t_train, t_test = t[train_index], t[test_index]
        lr.fit(x_train, t_train)
        result = lr.predict(x_test)
        pred.append(result[0])

    # AUC
    auc = roc_auc_score(t, pred)

    res = auc
    # minimize 1 - AUC
    return 1/res


def lassoTrial(n_trials):
    import optuna
    study = optuna.create_study()
    study.optimize(lassoOptuna, n_trials)
    # result
    print()
    print('hyper parameter:', study.best_params)
    print('AUC:', 1/study.best_value)
```

In [ ]:

```
lassoTrial(100)
```

```
[I 2020-09-19 09:36:57,374] A new study created in memory with name: no-name-8
a3b1555-260d-4b9d-81c8-b6b0f7863917
[I 2020-09-19 09:36:57,446] Trial 0 finished with value: 1.7322834645669292 and
parameters: {'alpha': 0.5468973635867508}. Best is trial 0 with value: 1.73228346
45669292.
[I 2020-09-19 09:36:57,522] Trial 1 finished with value: 1.8181818181818181 and
parameters: {'alpha': 0.3164828966134761}. Best is trial 0 with value: 1.73228346
45669292.
[I 2020-09-19 09:36:57,594] Trial 2 finished with value: 1.8410041841004183 and
parameters: {'alpha': 0.2863582888630547}. Best is trial 0 with value: 1.73228346
45669292.
[I 2020-09-19 09:36:57,674] Trial 3 finished with value: 1.685823754789272 and
parameters: {'alpha': 0.6339916873793024}. Best is trial 3 with value: 1.68582375
4789272.
[I 2020-09-19 09:36:57,738] Trial 4 finished with value: 1.7054263565891472 and
parameters: {'alpha': 0.7962267259138708}. Best is trial 3 with value: 1.68582375
4789272.
[I 2020-09-19 09:36:57,806] Trial 5 finished with value: 1.7886178861788617 and
parameters: {'alpha': 0.9279092526335121}. Best is trial 3 with value: 1.68582375
4789272.
[I 2020-09-19 09:36:57,880] Trial 6 finished with value: 1.7600000000000002 and
parameters: {'alpha': 0.8849235907263094}. Best is trial 3 with value: 1.68582375
4789272.
[I 2020-09-19 09:36:57,958] Trial 7 finished with value: 1.7391304347826089 and
parameters: {'alpha': 0.417398291343736}. Best is trial 3 with value: 1.685823754
789272.
[I 2020-09-19 09:36:58,114] Trial 8 finished with value: 1.414790996784566 and
parameters: {'alpha': 0.069938124148563 3}. Best is trial 8 with value: 1.41479099
6784566.
[I 2020-09-19 09:36:58,198] Trial 9 finished with value: 1.752988047808765 and
parameters: {'alpha': 0.3788302209126961}. Best is trial 8 with value: 1.41479099
6784566.
[I 2020-09-19 09:36:58,413] Trial 10 finished with value: 1.4102564102564101 a
nd parameters: {'alpha': 0.01580298336902012}. Best is trial 10 with value: 1.4102
564102564101.
[I 2020-09-19 09:36:58,531] Trial 11 finished with value: 1.4012738853503186 a
nd parameters: {'alpha': 0.05869581248257437}. Best is trial 11 with value: 1.4012
738853503186.
[I 2020-09-19 09:36:58,739] Trial 12 finished with value: 1.4102564102564101 a
nd parameters: {'alpha': 0.015688143198711365}. Best is trial 11 with value: 1.401
2738853503186.
[I 2020-09-19 09:36:58,829] Trial 13 finished with value: 1.6923076923076925 a
nd parameters: {'alpha': 0.1394144993040486}. Best is trial 11 with value: 1.40127
38853503186.
[I 2020-09-19 09:36:58,917] Trial 14 finished with value: 1.71875 and parameters:
{'alpha': 0.16037498000763906}. Best is trial 11 with value: 1.401273885350318
6.
[I 2020-09-19 09:36:59,197] Trial 15 finished with value: 1.4102564102564104 a
nd parameters: {'alpha': 0.012113662316912584}. Best is trial 11 with value: 1.401
2738853503186.
[I 2020-09-19 09:36:59,284] Trial 16 finished with value: 1.7670682730923695 a
nd parameters: {'alpha': 0.20987889778274932}. Best is trial 11 with value: 1.4012
738853503186.
[I 2020-09-19 09:36:59,513] Trial 17 finished with value: 1.4193548387096773 a
nd parameters: {'alpha': 0.01382135345707276}. Best is trial 11 with value: 1.4012
738853503186.
[I 2020-09-19 09:36:59,597] Trial 18 finished with value: 1.8106995884773662 a
nd parameters: {'alpha': 0.21872009577382415}. Best is trial 11 with value: 1.4012
738853503186.
[I 2020-09-19 09:36:59,675] Trial 19 finished with value: 1.673003802281369 and
parameters: {'alpha': 0.7017219717329499}. Best is trial 11 with value: 1.4012738
```

853503186.
[I 2020-09-19 09:36:59,771] Trial 20 finished with value: 1.4379084967320261 a
nd parameters: {'alpha': 0.09129684047293109}. Best is trial 11 with value: 1.4012
738853503186.
[I 2020-09-19 09:36:59,965] Trial 21 finished with value: 1.414790996784566 and
parameters: {'alpha': 0.015335904046128316}. Best is trial 11 with value: 1.40127
38853503186.
[I 2020-09-19 09:37:00,066] Trial 22 finished with value: 1.4239482200647249 a
nd parameters: {'alpha': 0.0832103116610729 1}. Best is trial 11 with value: 1.4012
738853503186.
[I 2020-09-19 09:37:00,198] Trial 23 finished with value: 1.4012738853503186 a
nd parameters: {'alpha': 0.028973694745649363}. Best is trial 11 with value: 1.401
2738853503186.
[I 2020-09-19 09:37:00,280] Trial 24 finished with value: 1.8565400843881859 a
nd parameters: {'alpha': 0.24116158964313691}. Best is trial 11 with value: 1.4012
738853503186.
[I 2020-09-19 09:37:00,368] Trial 25 finished with value: 1.673003802281369 and
parameters: {'alpha': 0.12796382139306975}. Best is trial 11 with value: 1.401273
8853503186.
[I 2020-09-19 09:37:00,440] Trial 26 finished with value: 1.7322834645669292 a
nd parameters: {'alpha': 0.4451993548281571}. Best is trial 11 with value: 1.40127
38853503186.
[I 2020-09-19 09:37:00,516] Trial 27 finished with value: 1.752988047808765 and
parameters: {'alpha': 0.3338800403133423}. Best is trial 11 with value: 1.4012738
853503186.
[I 2020-09-19 09:37:00,606] Trial 28 finished with value: 1.7600000000000002 a
nd parameters: {'alpha': 0.19038686991209153}. Best is trial 11 with value: 1.4012
738853503186.
[I 2020-09-19 09:37:00,690] Trial 29 finished with value: 1.7322834645669294 a
nd parameters: {'alpha': 0.5217593894252353}. Best is trial 11 with value: 1.40127
38853503186.
[I 2020-09-19 09:37:00,808] Trial 30 finished with value: 1.38801261829653 and
parameters: {'alpha': 0.05000043102696057}. Best is trial 30 with value: 1.388012
61829653.
[I 2020-09-19 09:37:00,919] Trial 31 finished with value: 1.4057507987220446 a
nd parameters: {'alpha': 0.06358078025661738}. Best is trial 30 with value: 1.3880
1261829653.
[I 2020-09-19 09:37:01,028] Trial 32 finished with value: 1.423948220064725 and
parameters: {'alpha': 0.07855420242715934}. Best is trial 30 with value: 1.388012
61829653.
[I 2020-09-19 09:37:01,115] Trial 33 finished with value: 1.8487394957983194 a
nd parameters: {'alpha': 0.2754940352808024}. Best is trial 30 with value: 1.38801
261829653.
[I 2020-09-19 09:37:01,208] Trial 34 finished with value: 1.6793893129770991 a
nd parameters: {'alpha': 0.129788829132368}. Best is trial 30 with value: 1.388012
61829653.
[I 2020-09-19 09:37:01,280] Trial 35 finished with value: 1.7054263565891472 a
nd parameters: {'alpha': 0.5987456410544998}. Best is trial 30 with value: 1.38801
261829653.
[I 2020-09-19 09:37:01,389] Trial 36 finished with value: 1.4012738853503186 a
nd parameters: {'alpha': 0.061142791643547}. Best is trial 30 with value: 1.388012
61829653.
[I 2020-09-19 09:37:01,469] Trial 37 finished with value: 1.8644067796610169 a
nd parameters: {'alpha': 0.2704129315089426 3}. Best is trial 30 with value: 1.3880
1261829653.
[I 2020-09-19 09:37:01,569] Trial 38 finished with value: 1.71875 and parameters:
{'alpha': 0.1575037622176327 4}. Best is trial 30 with value: 1.38801261829653.
[I 2020-09-19 09:37:01,645] Trial 39 finished with value: 1.7600000000000002 a
nd parameters: {'alpha': 0.3302637445658586}. Best is trial 30 with value: 1.38801
261829653.
[I 2020-09-19 09:37:01,762] Trial 40 finished with value: 1.4012738853503186 a

nd parameters: {'alpha': 0.05828904229169433}. Best is trial 30 with value: 1.3880
1261829653.
[I 2020-09-19 09:37:01,867] Trial 41 finished with value: 1.4012738853503186 a
nd parameters: {'alpha': 0.06091954252337558 6}. Best is trial 30 with value: 1.388
01261829653.
[I 2020-09-19 09:37:01,953] Trial 42 finished with value: 1.5827338129496402 a
nd parameters: {'alpha': 0.11856182573744838}. Best is trial 30 with value: 1.3880
1261829653.
[I 2020-09-19 09:37:02,066] Trial 43 finished with value: 1.370716510903427 and
parameters: {'alpha': 0.03642667621574977}. Best is trial 43 with value: 1.370716
510903427.
[I 2020-09-19 09:37:02,163] Trial 44 finished with value: 1.7322834645669294 a
nd parameters: {'alpha': 0.1699626889034376}. Best is trial 43 with value: 1.37071
6510903427.
[I 2020-09-19 09:37:02,479] Trial 45 finished with value: 1.414790996784566 and
parameters: {'alpha': 0.011602792407860475}. Best is trial 43 with value: 1.37071
6510903427.
[I 2020-09-19 09:37:02,593] Trial 46 finished with value: 1.4012738853503186 a
nd parameters: {'alpha': 0.0555840301904161 7}. Best is trial 43 with value: 1.3707
16510903427.
[I 2020-09-19 09:37:02,698] Trial 47 finished with value: 1.5714285714285714 a
nd parameters: {'alpha': 0.11450103198782796}. Best is trial 43 with value: 1.3707
16510903427.
[I 2020-09-19 09:37:02,792] Trial 48 finished with value: 1.7600000000000002 a
nd parameters: {'alpha': 0.1896858898781643}. Best is trial 43 with value: 1.37071
6510903427.
[I 2020-09-19 09:37:02,907] Trial 49 finished with value: 1.3750000000000002 a
nd parameters: {'alpha': 0.04917600441396306}. Best is trial 43 with value: 1.3707
16510903427.
[I 2020-09-19 09:37:02,991] Trial 50 finished with value: 1.8565400843881859 a
nd parameters: {'alpha': 0.2462873072825028}. Best is trial 43 with value: 1.37071
6510903427.
[I 2020-09-19 09:37:03,103] Trial 51 finished with value: 1.370716510903427 and
parameters: {'alpha': 0.043945715265744985}. Best is trial 43 with value: 1.37071
6510903427.
[I 2020-09-19 09:37:03,241] Trial 52 finished with value: 1.375 and parameters: {'al
pha': 0.03369476493670899}. Best is trial 43 with value: 1.370716510903427.
[I 2020-09-19 09:37:03,338] Trial 53 finished with value: 1.456953642384106 and
parameters: {'alpha': 0.095314083471463}. Best is trial 43 with value: 1.37071651
0903427.
[I 2020-09-19 09:37:03,571] Trial 54 finished with value: 1.414790996784566 and
parameters: {'alpha': 0.013602797296579677}. Best is trial 43 with value: 1.37071
6510903427.
[I 2020-09-19 09:37:03,682] Trial 55 finished with value: 1.3664596273291925 a
nd parameters: {'alpha': 0.03893487770440182}. Best is trial 55 with value: 1.3664
596273291925.
[I 2020-09-19 09:37:03,758] Trial 56 finished with value: 1.6988416988416988 a
nd parameters: {'alpha': 0.7635445155985258}. Best is trial 55 with value: 1.36645
96273291925.
[I 2020-09-19 09:37:03,868] Trial 57 finished with value: 1.375 and parameters: {'al
pha': 0.03725837901260439}. Best is trial 55 with value: 1.3664596273291925.
[I 2020-09-19 09:37:03,954] Trial 58 finished with value: 1.7254901960784312 a
nd parameters: {'alpha': 0.15176837449457234}. Best is trial 55 with value: 1.3664
596273291925.
[I 2020-09-19 09:37:04,045] Trial 59 finished with value: 1.5492957746478873 a
nd parameters: {'alpha': 0.1098218202002442}. Best is trial 55 with value: 1.36645
96273291925.
[I 2020-09-19 09:37:04,228] Trial 60 finished with value: 1.4239482200647249 a
nd parameters: {'alpha': 0.016557298000796035}. Best is trial 55 with value: 1.366
4596273291925.
[I 2020-09-19 09:37:04,343] Trial 61 finished with value: 1.3664596273291925 a

nd parameters: {'alpha': 0.0388950040567291 3}. Best is trial 55 with value: 1.3664 596273291925.
[I 2020-09-19 09:37:04,467] Trial 62 finished with value: 1.3793103448275863 a nd parameters: {'alpha': 0.0325197983686166 8}. Best is trial 55 with value: 1.3664 596273291925.
[I 2020-09-19 09:37:04,535] Trial 63 finished with value: 1.8181818181818188 a nd parameters: {'alpha': 0.957666891654493}. Best is trial 55 with value: 1.366459 6273291925.
[I 2020-09-19 09:37:04,640] Trial 64 finished with value: 1.4715719063545152 a nd parameters: {'alpha': 0.0974860953880078 7}. Best is trial 55 with value: 1.3664 596273291925.
[I 2020-09-19 09:37:04,775] Trial 65 finished with value: 1.375 and parameters: {'al pha': 0.03549396262686037}. Best is trial 55 with value: 1.3664596273291925.
[I 2020-09-19 09:37:05,029] Trial 66 finished with value: 1.414790996784566 and parameters: {'alpha': 0.013648975026239737}. Best is trial 55 with value: 1.36645 96273291925.
[I 2020-09-19 09:37:05,142] Trial 67 finished with value: 1.4379084967320261 a nd parameters: {'alpha': 0.0910382043318106 2}. Best is trial 55 with value: 1.3664 596273291925.
[I 2020-09-19 09:37:05,245] Trial 68 finished with value: 1.7054263565891472 a nd parameters: {'alpha': 0.141600435825322 9}. Best is trial 55 with value: 1.36645 96273291925.
[I 2020-09-19 09:37:05,342] Trial 69 finished with value: 1.746031746031746 and parameters: {'alpha': 0.18463468482244685}. Best is trial 55 with value: 1.366459 6273291925.
[I 2020-09-19 09:37:05,495] Trial 70 finished with value: 1.3622291021671828 a nd parameters: {'alpha': 0.0419134815744176 06}. Best is trial 70 with value: 1.362 2291021671828.
[I 2020-09-19 09:37:05,614] Trial 71 finished with value: 1.3664596273291927 a nd parameters: {'alpha': 0.0425343530340154 06}. Best is trial 70 with value: 1.362 2291021671828.
[I 2020-09-19 09:37:05,723] Trial 72 finished with value: 1.4193548387096775 a nd parameters: {'alpha': 0.0741978526276395 7}. Best is trial 70 with value: 1.3622 291021671828.
[I 2020-09-19 09:37:05,862] Trial 73 finished with value: 1.4012738853503186 a nd parameters: {'alpha': 0.0318719679350325 25}. Best is trial 70 with value: 1.362 2291021671828.
[I 2020-09-19 09:37:05,954] Trial 74 finished with value: 1.6666666666666667 a nd parameters: {'alpha': 0.125553143608913 3}. Best is trial 70 with value: 1.36222 91021671828.
[I 2020-09-19 09:37:06,282] Trial 75 finished with value: 1.4285714285714286 a nd parameters: {'alpha': 0.010541908486259 32}. Best is trial 70 with value: 1.3622 291021671828.
[I 2020-09-19 09:37:06,373] Trial 76 finished with value: 1.8106995884773662 a nd parameters: {'alpha': 0.222125246398808 68}. Best is trial 70 with value: 1.3622 291021671828.
[I 2020-09-19 09:37:06,476] Trial 77 finished with value: 1.4379084967320261 a nd parameters: {'alpha': 0.0866922554108536 1}. Best is trial 70 with value: 1.3622 291021671828.
[I 2020-09-19 09:37:06,607] Trial 78 finished with value: 1.3664596273291927 a nd parameters: {'alpha': 0.0435093134639270 16}. Best is trial 70 with value: 1.362 2291021671828.
[I 2020-09-19 09:37:06,714] Trial 79 finished with value: 1.4193548387096775 a nd parameters: {'alpha': 0.0756023850981092 9}. Best is trial 70 with value: 1.3622 291021671828.
[I 2020-09-19 09:37:06,826] Trial 80 finished with value: 1.7254901960784312 a nd parameters: {'alpha': 0.155506145168472 4}. Best is trial 70 with value: 1.36222 91021671828.
[I 2020-09-19 09:37:06,948] Trial 81 finished with value: 1.3664596273291925 a nd parameters: {'alpha': 0.038338421831898 5}. Best is trial 70 with value: 1.36222 91021671828.

[I 2020-09-19 09:37:07,076] Trial 82 finished with value: 1.3664596273291927 a
nd parameters: {'alpha': 0.04624863388496574}. Best is trial 70 with value: 1.3622
291021671828.
[I 2020-09-19 09:37:07,169] Trial 83 finished with value: 1.5438596491228072 a
nd parameters: {'alpha': 0.10823107447737366}. Best is trial 70 with value: 1.3622
291021671828.
[I 2020-09-19 09:37:07,280] Trial 84 finished with value: 1.4012738853503186 a
nd parameters: {'alpha': 0.05563198974182235}. Best is trial 70 with value: 1.3622
291021671828.
[I 2020-09-19 09:37:07,347] Trial 85 finished with value: 1.7322834645669292 a
nd parameters: {'alpha': 0.43943800733807536}. Best is trial 70 with value: 1.3622
291021671828.
[I 2020-09-19 09:37:07,645] Trial 86 finished with value: 1.414790996784566 and
parameters: {'alpha': 0.011176676456483275}. Best is trial 70 with value: 1.36222
91021671828.
[I 2020-09-19 09:37:07,714] Trial 87 finished with value: 1.7529880478087647 a
nd parameters: {'alpha': 0.8593298466992645}. Best is trial 70 with value: 1.36222
91021671828.
[I 2020-09-19 09:37:07,828] Trial 88 finished with value: 1.4193548387096775 a
nd parameters: {'alpha': 0.07559292460228047}. Best is trial 70 with value: 1.3622
291021671828.
[I 2020-09-19 09:37:07,913] Trial 89 finished with value: 1.673003802281369 and
parameters: {'alpha': 0.1306665756260311 8}. Best is trial 70 with value: 1.362229
1021671828.
[I 2020-09-19 09:37:08,030] Trial 90 finished with value: 1.3793103448275863 a
nd parameters: {'alpha': 0.04723134178284924}. Best is trial 70 with value: 1.3622
291021671828.
[I 2020-09-19 09:37:08,141] Trial 91 finished with value: 1.370716510903427 and
parameters: {'alpha': 0.036323669312752854}. Best is trial 70 with value: 1.36222
91021671828.
[I 2020-09-19 09:37:08,235] Trial 92 finished with value: 1.5277777777777777 a
nd parameters: {'alpha': 0.10347908682490842}. Best is trial 70 with value: 1.3622
291021671828.
[I 2020-09-19 09:37:08,339] Trial 93 finished with value: 1.4193548387096775 a
nd parameters: {'alpha': 0.07368367320766124}. Best is trial 70 with value: 1.3622
291021671828.
[I 2020-09-19 09:37:08,495] Trial 94 finished with value: 1.38801261829653 and
parameters: {'alpha': 0.04991869890213535}. Best is trial 70 with value: 1.362229
1021671828.
[I 2020-09-19 09:37:08,779] Trial 95 finished with value: 1.4102564102564104 a
nd parameters: {'alpha': 0.012516180288603657}. Best is trial 70 with value: 1.362
2291021671828.
[I 2020-09-19 09:37:08,912] Trial 96 finished with value: 1.3793103448275863 a
nd parameters: {'alpha': 0.03490155629452168}. Best is trial 70 with value: 1.3622
291021671828.
[I 2020-09-19 09:37:09,004] Trial 97 finished with value: 1.746031746031746 and
parameters: {'alpha': 0.1747305656751487}. Best is trial 70 with value: 1.3622291
021671828.
[I 2020-09-19 09:37:09,104] Trial 98 finished with value: 1.4379084967320261 a
nd parameters: {'alpha': 0.0878019670850466}. Best is trial 70 with value: 1.36222
91021671828.
[I 2020-09-19 09:37:09,199] Trial 99 finished with value: 1.685823754789272 and
parameters: {'alpha': 0.13782148193563742}. Best is trial 70 with value: 1.362229
1021671828.

hyper parameter: {'alpha': 0.0419134815744176 06}
AUC: 0.734090909090909

In [ ]:

```
# This alpha value is one example of hyperparameter optimization. There are other values that yield a
 similar AUC value.
lasso(0.041913481574417606)
```

■ Leave-One-Out Cross Validation with Lasso Regression

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.73409 Accuracy: 0.72549 R2: -0.16791

	Actual True	Actual False
Predict True	3	6
Predict False	8	34

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.734090909090909, 0.7254901960784313, 0.3333333333333333,
       0.2727272727272727, 0.3, 0.85, list([[3, 6], [8, 34]]),
       -0.1679078135148202], dtype=object)
```

3) Ridge Regression ; AUC 0.632

In [ ]:

```
def ridge(a):
    import numpy as np
    import pandas as pd
    from sklearn.linear_model import Ridge
    from sklearn.model_selection import LeaveOneOut

    # Statistical functions
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import r2_score

    # Rounding of fractional point
    import math
    def my_round(val, digit=0):
        p = 10 ** digit
        return (val * p * 2 + 1) // 2 / p

    x = []
    t = []

    # Predicted value
    pred = []
    round = []
    pairs = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)

    x = np.array(x)
    t = np.array(t)

    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)
    lr = Ridge(alpha=a)

    for train_index, test_index in loo.split(x):
        x_train, x_test = x[train_index], x[test_index]
        t_train, t_test = t[train_index], t[test_index]
        lr.fit(x_train, t_train)
        result = lr.predict(x_test)
        pred.append(result[0])

        # Category of prediction
        if (result[0]<0.5):
            round.append(int(0))
        else:
            round.append(int(1))

    for i in range(len(pred)):
        pairs.append([pred[i], t[i]])

    output = pd.DataFrame(pairs)
    # output.to_csv('drive/My Drive/program/data/cstatdata.csv', index=False)

    # Coefficient of Determination
    r2 = r2_score(t, pred)
```

```
    # AUC
    auc = roc_auc_score(t, pred)

    # Confusion Matrix
    TP = 0
    FP = 0
    FN = 0
    TN = 0

    for i in range(len(round)):
        if (round[i]==1) & (t[i]==1):
            TP += 1
        elif (round[i]==1) & (t[i]==0):
            FP += 1
        elif (round[i]==0) & (t[i]==1):
            FN += 1
        elif (round[i]==0) & (t[i]==0):
            TN += 1

    confmat = [[TP, FP], [FN, TN]]

    acc = (TP+TN)/(TP+FP+FN+TN)

    if TP + FP != 0:
        prec = TP/(TP+FP)
    elif TP + FP == 0:
        prec = 'N/A'
    if TP + FN != 0:
        rec = TP/(TP+FN)
    elif TP + FP ==0:
        rec = 'N/A'
    if TN + FP != 0:
        spec = TN/(TN + FP)
    elif TN + FP == 0:
        spec = 'N/A'
    if TP + FP/2 + FN/2 != 0:
        f1 = TP/(TP + FP/2 + FN/2)
    elif TP + FP/2 + FN/2 == 0:
        f1 = 'N/A'

    res = np.array([auc, acc, prec, rec, f1, spec, confmat, r2])

    # Output : ( 1. AUC, 2, Accuracy, 3. Precision 4. Recall 5. f1-score 6. Specificity, 7. Confusion Matri
x, 8. R^2 in LOO)

    print('■ Leave-One-Out Cross Validation with Ridge Regression')
    print('')
    print('Sample size of all dataset: ' + str(len(x_train) + len(x_test)) )
    print('Sample size of training data: ' + str(len(x_train)) + '    Explanatory variables: ' + str(x_train.
shape) + '   Target value: ' + str(t_train.shape))
    print('Sample size of test data: ' + str(len(x_test)) + '    Explanatory variables: ' + str(x_test.shap
e) + '   Target value: ' + str(t_test.shape))
    print('')
    print('AUC: ' + str(my_round(auc, 5)) + '   Accuracy: ' + str(my_round(acc,5)) + '   R2: ' + str(my_
round(r2,5)))
    print('')
    print(pd.DataFrame(np.array(confmat), index=['Predict True', 'Predict False'], columns=['Actual T
rue', 'Actual False']))
    print('')
    print('Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confution matr
ix 8. R^2 in LOO')
```

```
    print('')

    return res
```

In [ ]:

```
ridge(1)
```

■ Leave-One-Out Cross Validation with Ridge Regression

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.51818 Accuracy: 0.62745 R2: -1.91292

	Actual True	Actual False
Predict True	3	11
Predict False	8	29

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confusio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.5181818181818182, 0.6274509803921569, 0.21428571428571427,
       0.2727272727272727, 0.24, 0.725, list([[3, 11], [8, 29]]),
       -1.9129247728164027], dtype=object)
```

In [ ]:

```
ridge(100)
```

■ Leave-One-Out Cross Validation with Ridge Regression

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.62727 Accuracy: 0.70588 R2: -0.46392

	Actual True	Actual False
Predict True	3	7
Predict False	8	33

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confusio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.6272727272727272, 0.7058823529411765, 0.3, 0.2727272727272727,
       0.2857142857142857, 0.825, list([[3, 7], [8, 33]]),
       -0.4639196454021983], dtype=object)
```

In [ ]:

```
ridge(1000)
```

■ Leave-One-Out Cross Validation with Ridge Regression

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.59091 Accuracy: 0.72549 R2: -0.21508

	Actual True	Actual False
Predict True	2	5
Predict False	9	35

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.5909090909090909, 0.7254901960784313, 0.2857142857142857,
       0.1818181818181818, 0.2222222222222222, 0.875,
       list([[2, 5], [9, 35]]), -0.2150790729071388], dtype=object)
```

In [ ]:

```
ridge(0.1)
```

■ Leave-One-Out Cross Validation with Ridge Regression

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.5 Accuracy: 0.62745 R2: -2.02895

	Actual True	Actual False
Predict True	3	11
Predict False	8	29

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.49999999999999994, 0.6274509803921569, 0.21428571428571427,
       0.2727272727272727, 0.24, 0.725, list([[3, 11], [8, 29]]),
       -2.028951181637043], dtype=object)
```

Optimization with Optuna

In [ ]:

```
def ridgeOptuna(trial):
    import numpy as np
    import pandas as pd
    from sklearn.linear_model import Ridge
    from sklearn.model_selection import LeaveOneOut

    import optuna

    # Statistical functions
    from sklearn.metrics import roc_auc_score

    alpha = trial.suggest_uniform('alpha', 1, 1000)

    x = []
    t = []

    # Predicted value
    pred = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)

    x = np.array(x)
    t = np.array(t)

    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)
    lr = Ridge(alpha=alpha)

    for train_index, test_index in loo.split(x):
        x_train, x_test = x[train_index], x[test_index]
        t_train, t_test = t[train_index], t[test_index]
        lr.fit(x_train, t_train)
        result = lr.predict(x_test)
        pred.append(result[0])

    auc = roc_auc_score(t, pred)

    res = auc
    # minimize 1/AUC
    return 1/res


def ridgeTrial(n_trials):
    import optuna
    study = optuna.create_study()
    study.optimize(ridgeOptuna, n_trials)
    # result
    print()
    print('hpyterparameter:', study.best_params)
    print('AUC:', 1/study.best_value)
    print()
    return study.best_params['alpha']
```

In [ ]:

```
ridgeTrial(100)
```

```
[I 2020-09-19 09:45:37,718] A new study created in memory with name: no-name-a
964ed54-ccb5-4261-ae12-0f194c6c19ae
[I 2020-09-19 09:45:37,807] Trial 0 finished with value: 1.6541353383458648 and
parameters: {'alpha': 502.08382376876386}. Best is trial 0 with value: 1.65413533
83458648.
[I 2020-09-19 09:45:38,028] Trial 1 finished with value: 1.660377358490566 and
parameters: {'alpha': 619.328346700925}. Best is trial 0 with value: 1.6541353383
458648.
[I 2020-09-19 09:45:38,099] Trial 2 finished with value: 1.6236162361623618 and
parameters: {'alpha': 320.54271531147975}. Best is trial 2 with value: 1.62361623
61623618.
[I 2020-09-19 09:45:38,170] Trial 3 finished with value: 1.8106995884773667 and
parameters: {'alpha': 4.673798212382571}. Best is trial 2 with value: 1.623616236
1623618.
[I 2020-09-19 09:45:38,233] Trial 4 finished with value: 1.6236162361623618 and
parameters: {'alpha': 225.60989331247978}. Best is trial 2 with value: 1.62361623
61623618.
[I 2020-09-19 09:45:38,300] Trial 5 finished with value: 1.6176470588235294 and
parameters: {'alpha': 211.23193416128123}. Best is trial 5 with value: 1.61764705
88235294.
[I 2020-09-19 09:45:38,389] Trial 6 finished with value: 1.660377358490566 and
parameters: {'alpha': 573.2863878412443}. Best is trial 5 with value: 1.617647058
8235294.
[I 2020-09-19 09:45:38,455] Trial 7 finished with value: 1.660377358490566 and
parameters: {'alpha': 601.8535689642396}. Best is trial 5 with value: 1.617647058
8235294.
[I 2020-09-19 09:45:38,518] Trial 8 finished with value: 1.685823754789272 and
parameters: {'alpha': 888.9027397426358}. Best is trial 5 with value: 1.617647058
8235294.
[I 2020-09-19 09:45:38,579] Trial 9 finished with value: 1.685823754789272 and
parameters: {'alpha': 898.8927893355691}. Best is trial 5 with value: 1.617647058
8235294.
[I 2020-09-19 09:45:38,648] Trial 10 finished with value: 1.7054263565891472 a
nd parameters: {'alpha': 14.700773600893115}. Best is trial 5 with value: 1.617647
0588235294.
[I 2020-09-19 09:45:38,714] Trial 11 finished with value: 1.6236162361623618 a
nd parameters: {'alpha': 288.74950748867013}. Best is trial 5 with value: 1.617647
0588235294.
[I 2020-09-19 09:45:38,781] Trial 12 finished with value: 1.6176470588235292 a
nd parameters: {'alpha': 253.66021211293764}. Best is trial 12 with value: 1.61764
70588235292.
[I 2020-09-19 09:45:38,851] Trial 13 finished with value: 1.6296296296296295 a
nd parameters: {'alpha': 166.64586709137822}. Best is trial 12 with value: 1.61764
70588235292.
[I 2020-09-19 09:45:38,926] Trial 14 finished with value: 1.6356877323420076 a
nd parameters: {'alpha': 393.8477394799787}. Best is trial 12 with value: 1.617647
0588235292.
[I 2020-09-19 09:45:38,994] Trial 15 finished with value: 1.6356877323420072 a
nd parameters: {'alpha': 176.26439897519035}. Best is trial 12 with value: 1.61764
70588235292.
[I 2020-09-19 09:45:39,060] Trial 16 finished with value: 1.5942028985507248 a
nd parameters: {'alpha': 97.6567889440346}. Best is trial 16 with value: 1.5942028
985507248.
[I 2020-09-19 09:45:39,127] Trial 17 finished with value: 1.6117216117216115 a
nd parameters: {'alpha': 49.32759945431692}. Best is trial 16 with value: 1.594202
8985507248.
[I 2020-09-19 09:45:39,201] Trial 18 finished with value: 1.588447653429603 and
parameters: {'alpha': 70.53955737531226}. Best is trial 18 with value: 1.58844765
3429603.
[I 2020-09-19 09:45:39,267] Trial 19 finished with value: 1.5942028985507248 a
nd parameters: {'alpha': 89.23054095829855}. Best is trial 18 with value: 1.588447
```

653429603.
[I 2020-09-19 09:45:39,336] Trial 20 finished with value: 1.6479400749063673 a
nd parameters: {'alpha': 408.7139909067889}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:39,417] Trial 21 finished with value: 1.5942028985507248 a
nd parameters: {'alpha': 111.77046786896135}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:39,484] Trial 22 finished with value: 1.5942028985507248 a
nd parameters: {'alpha': 99.3298020350109}. Best is trial 18 with value: 1.5884476
53429603.
[I 2020-09-19 09:45:39,550] Trial 23 finished with value: 1.5942028985507248 a
nd parameters: {'alpha': 87.46543395658568}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:39,621] Trial 24 finished with value: 1.6666666666666667 a
nd parameters: {'alpha': 771.8179828853912}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:39,700] Trial 25 finished with value: 1.6058394160583942 a
nd parameters: {'alpha': 144.38866265245946}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:39,770] Trial 26 finished with value: 1.8965517241379313 a
nd parameters: {'alpha': 2.8695791932072865}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:39,838] Trial 27 finished with value: 1.6236162361623618 a
nd parameters: {'alpha': 312.6880540777735}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:39,903] Trial 28 finished with value: 1.6356877323420076 a
nd parameters: {'alpha': 381.94223541578117}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:39,971] Trial 29 finished with value: 1.8803418803418805 a
nd parameters: {'alpha': 3.192063251883667}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:40,035] Trial 30 finished with value: 1.6541353383458648 a
nd parameters: {'alpha': 489.7930069050548}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:40,099] Trial 31 finished with value: 1.5942028985507246 a
nd parameters: {'alpha': 68.39219009842311}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:40,170] Trial 32 finished with value: 1.5942028985507246 a
nd parameters: {'alpha': 67.66983309557591}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:40,238] Trial 33 finished with value: 1.6117216117216115 a
nd parameters: {'alpha': 60.99807139733077}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:40,304] Trial 34 finished with value: 1.6236162361623618 a
nd parameters: {'alpha': 162.08096589115175}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:40,383] Trial 35 finished with value: 1.6117216117216115 a
nd parameters: {'alpha': 49.97011886952558}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:40,451] Trial 36 finished with value: 1.6236162361623618 a
nd parameters: {'alpha': 240.34524296677412}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:40,518] Trial 37 finished with value: 1.6296296296296295 a
nd parameters: {'alpha': 205.15656134309933}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:40,585] Trial 38 finished with value: 1.673003802281369 and
parameters: {'alpha': 697.3388274567179}. Best is trial 18 with value: 1.58844765
3429603.
[I 2020-09-19 09:45:40,659] Trial 39 finished with value: 1.5999999999999996 a
nd parameters: {'alpha': 133.85913573829123}. Best is trial 18 with value: 1.58844
7653429603.

[I 2020-09-19 09:45:40,730] Trial 40 finished with value: 1.6923076923076923 a
nd parameters: {'alpha': 989.6983573227985}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:40,808] Trial 41 finished with value: 1.6000000000000003 a
nd parameters: {'alpha': 113.98417749222337}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:40,904] Trial 42 finished with value: 1.7813765182186234 a
nd parameters: {'alpha': 5.6073292479673}. Best is trial 18 with value: 1.58844765
3429603.
[I 2020-09-19 09:45:40,982] Trial 43 finished with value: 1.5942028985507246 a
nd parameters: {'alpha': 66.89241936357244}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:41,055] Trial 44 finished with value: 1.6117216117216115 a
nd parameters: {'alpha': 53.71987260668536}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:41,127] Trial 45 finished with value: 1.6176470588235294 a
nd parameters: {'alpha': 209.29862319630593}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:41,210] Trial 46 finished with value: 1.6236162361623618 a
nd parameters: {'alpha': 274.88911401441385}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:41,294] Trial 47 finished with value: 1.6356877323420076 a
nd parameters: {'alpha': 35.45388725765309}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:41,372] Trial 48 finished with value: 1.6417910447761193 a
nd parameters: {'alpha': 188.35901638379738}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:41,453] Trial 49 finished with value: 1.5942028985507248 a
nd parameters: {'alpha': 79.66365132435199}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:41,531] Trial 50 finished with value: 1.6236162361623618 a
nd parameters: {'alpha': 163.63124630988162}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:41,614] Trial 51 finished with value: 1.5999999999999996 a
nd parameters: {'alpha': 119.07208097518517}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:41,692] Trial 52 finished with value: 1.588447653429603 and
parameters: {'alpha': 70.9846244638232}. Best is trial 18 with value: 1.588447653
429603.
[I 2020-09-19 09:45:41,777] Trial 53 finished with value: 1.7254901960784317 a
nd parameters: {'alpha': 9.400404414903946}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:41,854] Trial 54 finished with value: 1.6356877323420076 a
nd parameters: {'alpha': 36.64684183702276}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:41,930] Trial 55 finished with value: 1.5942028985507248 a
nd parameters: {'alpha': 79.55141159545691}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:42,009] Trial 56 finished with value: 1.6117216117216115 a
nd parameters: {'alpha': 148.74568884666968}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:42,086] Trial 57 finished with value: 1.5999999999999996 a
nd parameters: {'alpha': 118.17055067747586}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:42,158] Trial 58 finished with value: 1.6236162361623618 a
nd parameters: {'alpha': 240.13472791574642}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:42,239] Trial 59 finished with value: 1.6296296296296295 a
nd parameters: {'alpha': 340.3662409456555}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:42,317] Trial 60 finished with value: 1.6 and parameters: {'alph

a': 84.91950929489813}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:42,417] Trial 61 finished with value: 1.588447653429603 and parameters: {'alpha': 72.02117543350688}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:42,494] Trial 62 finished with value: 1.647940074906367 and parameters: {'alpha': 32.11969962421211}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:42,575] Trial 63 finished with value: 1.5942028985507246 and parameters: {'alpha': 68.04948133593533}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:42,653] Trial 64 finished with value: 1.904761904761905 and parameters: {'alpha': 1.4083581045015734}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:42,731] Trial 65 finished with value: 1.6176470588235294 and parameters: {'alpha': 56.569815344101684}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:42,821] Trial 66 finished with value: 1.6356877323420072 and parameters: {'alpha': 180.32306445142933}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:42,903] Trial 67 finished with value: 1.6058394160583942 and parameters: {'alpha': 142.43662015433648}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:42,982] Trial 68 finished with value: 1.5942028985507248 and parameters: {'alpha': 90.49351052497707}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:43,068] Trial 69 finished with value: 1.6793893129770991 and parameters: {'alpha': 24.110379337179317}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:43,148] Trial 70 finished with value: 1.6176470588235294 and parameters: {'alpha': 57.297032504114284}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:43,222] Trial 71 finished with value: 1.6 and parameters: {'alpha': 102.01882453250784}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:43,293] Trial 72 finished with value: 1.5942028985507248 and parameters: {'alpha': 77.24510026645001}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:43,374] Trial 73 finished with value: 1.5999999999999996 and parameters: {'alpha': 130.26533193966517}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:43,462] Trial 74 finished with value: 1.7813765182186234 and parameters: {'alpha': 6.088226007067249}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:43,534] Trial 75 finished with value: 1.6417910447761193 and parameters: {'alpha': 197.18702016710074}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:43,606] Trial 76 finished with value: 1.588447653429603 and parameters: {'alpha': 73.63087786866201}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:43,682] Trial 77 finished with value: 1.6541353383458648 and parameters: {'alpha': 480.6544731992191}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:43,758] Trial 78 finished with value: 1.647940074906367 and parameters: {'alpha': 31.68539645262848}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:43,834] Trial 79 finished with value: 1.6236162361623618 and parameters: {'alpha': 154.84939865824504}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:43,907] Trial 80 finished with value: 1.588447653429603 and parameters: {'alpha': 70.89895314356173}. Best is trial 18 with value: 1.588447653429603.
[I 2020-09-19 09:45:43,988] Trial 81 finished with value: 1.6176470588235294 a

```
nd parameters: {'alpha': 60.03702244724072}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:44,062] Trial 82 finished with value: 1.5942028985507248 a
nd parameters: {'alpha': 109.99829628422555}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:44,135] Trial 83 finished with value: 1.5942028985507246 a
nd parameters: {'alpha': 68.41820997568823}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:44,208] Trial 84 finished with value: 1.6730038022813687 a
nd parameters: {'alpha': 25.972023243022313}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:44,280] Trial 85 finished with value: 1.5942028985507248 a
nd parameters: {'alpha': 100.80881534821464}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:44,356] Trial 86 finished with value: 1.5999999999999996 a
nd parameters: {'alpha': 135.04766771926205}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:44,451] Trial 87 finished with value: 1.8965517241379313 a
nd parameters: {'alpha': 2.2790000600854086}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:44,556] Trial 88 finished with value: 1.6356877323420072 a
nd parameters: {'alpha': 179.46692360826006}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:44,641] Trial 89 finished with value: 1.5942028985507246 a
nd parameters: {'alpha': 67.79161309709357}. Best is trial 18 with value: 1.588447
653429603.
[I 2020-09-19 09:45:44,708] Trial 90 finished with value: 1.6117216117216115 a
nd parameters: {'alpha': 48.759728295520205}. Best is trial 18 with value: 1.58844
7653429603.
[I 2020-09-19 09:45:44,779] Trial 91 finished with value: 1.5827338129496402 a
nd parameters: {'alpha': 71.79566461970718}. Best is trial 91 with value: 1.582733
8129496402.
[I 2020-09-19 09:45:44,847] Trial 92 finished with value: 1.588447653429603 and
parameters: {'alpha': 98.37679264127507}. Best is trial 91 with value: 1.58273381
29496402.
[I 2020-09-19 09:45:44,917] Trial 93 finished with value: 1.5999999999999996 a
nd parameters: {'alpha': 128.427435113456}. Best is trial 91 with value: 1.5827338
129496402.
[I 2020-09-19 09:45:44,990] Trial 94 finished with value: 1.6793893129770991 a
nd parameters: {'alpha': 24.069693900895324}. Best is trial 91 with value: 1.58273
38129496402.
[I 2020-09-19 09:45:45,061] Trial 95 finished with value: 1.6236162361623618 a
nd parameters: {'alpha': 221.59272207075009}. Best is trial 91 with value: 1.58273
38129496402.
[I 2020-09-19 09:45:45,129] Trial 96 finished with value: 1.5942028985507248 a
nd parameters: {'alpha': 95.11421554206957}. Best is trial 91 with value: 1.582733
8129496402.
[I 2020-09-19 09:45:45,198] Trial 97 finished with value: 1.5999999999999996 a
nd parameters: {'alpha': 118.16221612366098}. Best is trial 91 with value: 1.58273
38129496402.
[I 2020-09-19 09:45:45,269] Trial 98 finished with value: 1.5942028985507248 a
nd parameters: {'alpha': 78.82828121104427}. Best is trial 91 with value: 1.582733
8129496402.
[I 2020-09-19 09:45:45,339] Trial 99 finished with value: 1.6296296296296295 a
nd parameters: {'alpha': 44.18414032831724}. Best is trial 91 with value: 1.582733
8129496402.

hpyterparameter: {'alpha': 71.79566461970718}
AUC: 0.6318181818181818
```

Out[ ]:

71.79566461970718

In [ ]:

```
# This alpha value is one example of hyperparameter optimization. There are other values that yield a
 similar AUC value.
ridge(71.79566461970718)
```

■ Leave-One-Out Cross Validation with Ridge Regression

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.63182 Accuracy: 0.72549 R2: -0.54674

	Actual True	Actual False
Predict True	3	6
Predict False	8	34

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.6318181818181818, 0.7254901960784313, 0.3333333333333333,
       0.2727272727272727, 0.3, 0.85, list([[3, 6], [8, 34]]),
       -0.5467390446019575], dtype=object)
```

4) Logistic Regression ; AUC 0.553

In [ ]:

```
# Scaling with Normalization using MaxMin

def logisticN():
    import numpy as np
    import pandas as pd
    from sklearn.linear_model import LogisticRegression
    from sklearn.model_selection import LeaveOneOut

    from sklearn.preprocessing import MinMaxScaler

    # Statistical functions
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import r2_score

    # Rounding of fractional point
    import math
    def my_round(val, digit=0):
        p = 10 ** digit
        return (val * p * 2 + 1) // 2 / p

    x = []
    t = []

    # Predicted value
    pred = []
    round = []
    pairs = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)

    x = np.array(x)
    scaler = MinMaxScaler()
    scaler.fit(x)
    x = scaler.transform(x)
    t = np.array(t)

    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)
    lr = LogisticRegression()


    for train_index, test_index in loo.split(x):
        x_train, x_test = x[train_index], x[test_index]
        t_train, t_test = t[train_index], t[test_index]
        lr.fit(x_train, t_train)
        result = lr.predict(x_test)
        pred.append(result[0])

        # Category of prediction
        if (result[0]<0.5):
            round.append(int(0))
        else:
            round.append(int(1))
```

```
    for i in range(len(pred)):
        pairs.append([pred[i], t[i]])

    output = pd.DataFrame(pairs)
    # output.to_csv('drive/My Drive/program/data/cstatdata.csv', index=False)

    # Coefficient of Determination
    r2 = r2_score(t, pred)

    # AUC
    auc = roc_auc_score(t, pred)

    # Confusion Matrix
    TP = 0
    FP = 0
    FN = 0
    TN = 0

    for i in range(len(round)):
        if (round[i]==1) & (t[i]==1):
            TP += 1
        elif (round[i]==1) & (t[i]==0):
            FP += 1
        elif (round[i]==0) & (t[i]==1):
            FN += 1
        elif (round[i]==0) & (t[i]==0):
            TN += 1

    confmat = [[TP, FP], [FN, TN]]

    acc = (TP+TN)/(TP+FP+FN+TN)

    if TP + FP != 0:
        prec = TP/(TP+FP)
    elif TP + FP == 0:
        prec = 'N/A'
    if TP + FN != 0:
        rec = TP/(TP+FN)
    elif TP + FP ==0:
        rec = 'N/A'
    if TN + FP != 0:
        spec = TN/(TN + FP)
    elif TN + FP == 0:
        spec = 'N/A'
    if TP + FP/2 + FN/2 != 0:
        f1 = TP/(TP + FP/2 + FN/2)
    elif TP + FP/2 + FN/2 == 0:
        f1 = 'N/A'

    res = np.array([auc, acc, prec, rec, f1, spec, [[TP, FP], [FN, TN]], r2]  )

    # Output : ( 1. AUC, 2, Accuracy, 3. Precision 4. Recall 5. f1-score 6. Specificity, 7. Confusion Matri
x, 8. R^2 in LOO)

    print('■ Leave-One-Out Cross Validation with Logistic Regression with Normalization')
    print('')
    print('Sample size of all dataset: ' + str(len(x_train) + len(x_test)) )
    print('Sample size of training data: ' + str(len(x_train)) + '    Explanatory variables: ' + str(x_train.
shape) + '   Target value: ' + str(t_train.shape))
    print('Sample size of test data: ' + str(len(x_test)) + '    Explanatory variables: ' + str(x_test.shap
```

```
e) + '   Target value: ' + str(t_test.shape))
    print('')
    print('AUC: ' + str(my_round(auc, 5)) + '   Accuracy: ' + str(my_round(acc,5)) + '   R2: ' + str(my_
round(r2,5)))
    print('')
    print(pd.DataFrame(np.array(confmat), index=['Predict True', 'Predict False'], columns=['Actual T
rue', 'Actual False']))
    print('')
    print('Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confution matr
ix 8. R^2 in LOO')
    print('')

    return res



# Scaling with Standardization

def logisticS():
    import numpy as np
    import pandas as pd
    from sklearn.linear_model import LogisticRegression
    from sklearn.model_selection import LeaveOneOut

    from sklearn.preprocessing import StandardScaler

    # Statistical functions
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import r2_score

    # Rounding of fractional point
    import math
    def my_round(val, digit=0):
        p = 10 ** digit
        return (val * p * 2 + 1) // 2 / p

    x = []
    t = []

    # Predicted value
    pred = []
    round = []
    pairs = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)

    x = np.array(x)
    scaler = StandardScaler()
    scaler.fit(x)
    x = scaler.transform(x)
    t = np.array(t)

    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)
    lr = LogisticRegression()
```

```
for train_index, test_index in loo.split(x):
    x_train, x_test = x[train_index], x[test_index]
    t_train, t_test = t[train_index], t[test_index]
    lr.fit(x_train, t_train)
    result = lr.predict(x_test)
    pred.append(result[0])

    # Category of prediction
    if (result[0]<0.5):
        round.append(int(0))
    else:
        round.append(int(1))

for i in range(len(pred)):
    pairs.append([pred[i], t[i]])

output = pd.DataFrame(pairs)
# output.to_csv('drive/My Drive/program/data/cstatdata.csv', index=False)

# Coefficient of Determination
r2 = r2_score(t, pred)
# AUC
auc = roc_auc_score(t, pred)


# Confusion Matrix
TP = 0
FP = 0
FN = 0
TN = 0

for i in range(len(round)):
    if (round[i]==1) & (t[i]==1):
        TP += 1
    elif (round[i]==1) & (t[i]==0):
        FP += 1
    elif (round[i]==0) & (t[i]==1):
        FN += 1
    elif (round[i]==0) & (t[i]==0):
        TN += 1

confmat = [[TP, FP], [FN, TN]]

acc = (TP+TN)/(TP+FP+FN+TN)

if TP + FP != 0:
    prec = TP/(TP+FP)
elif TP + FP == 0:
    prec = 'N/A'
if TP + FN != 0:
    rec = TP/(TP+FN)
elif TP + FP ==0:
    rec = 'N/A'
if TN + FP != 0:
    spec = TN/(TN + FP)
elif TN + FP == 0:
    spec = 'N/A'
if TP + FP/2 + FN/2 != 0:
```

```
    f1 = TP/(TP + FP/2 + FN/2)
  elif TP + FP/2 + FN/2 == 0:
    f1 = 'N/A'

  res = np.array([auc, acc, prec, rec, f1, spec, [[TP, FP], [FN, TN]], r2] )

  # Output : ( 1. AUC, 2, Accuracy, 3. Precision 4. Recall 5. f1-score 6. Specificity, 7. Confusion Matri
x, 8. R^2 in LOO)

  print('■ Leave-One-Out Cross Validation with Logistic Regression with Standardization')
  print('')
  print('Sample size of all dataset: ' + str(len(x_train) + len(x_test)) )
  print('Sample size of training data: ' + str(len(x_train)) + '    Explanatory variables: ' + str(x_train.
shape) + '   Target value: ' + str(t_train.shape))
  print('Sample size of test data: ' + str(len(x_test)) + '    Explanatory variables: ' + str(x_test.shap
e) + '   Target value: ' + str(t_test.shape))
  print('')
  print('AUC: ' + str(my_round(auc, 5)) + '   Accuracy: ' + str(my_round(acc,5)) + '   R2: ' + str(my_
round(r2,5)))
  print('')
  print(pd.DataFrame(np.array(confmat), index=['Predict True', 'Predict False'], columns=['Actual T
rue', 'Actual False']))
  print('')
  print('Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confution matr
ix 8. R^2 in LOO')
  print('')

  return res
```

In [ ]:

```
logisticN()
```

```
■ Leave-One-Out Cross Validation with Logistic Regression with Normalization

Sample size of all dataset: 51
Sample size of training data: 50    Explanatory variables: (50, 211)   Target value: (5
0,)
Sample size of test data: 1    Explanatory variables: (1, 211)   Target value: (1,)

AUC: 0.55341   Accuracy: 0.76471   R2: -0.39091

              Actual True  Actual False
Predict True            2             3
Predict False           9            37

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO
```

Out[ ]:

```
array([0.553409090909091, 0.7647058823529411, 0.4, 0.1818181818181818
2,
       0.25, 0.925, list([[2, 3], [9, 37]]), -0.3909090909090913],
      dtype=object)
```

In [ ]:

```
logisticS()
```

■ Leave-One-Out Cross Validation with Logistic Regression with Standardization

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.52045 Accuracy: 0.76471 R2: -0.39091

	Actual True	Actual False
Predict True	1	2
Predict False	10	38

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.5204545454545454, 0.7647058823529411, 0.3333333333333333,
       0.09090909090909091, 0.14285714285714285, 0.95,
       list([[1, 2], [10, 38]]), -0.3909090909090913], dtype=object)
```

5) Support Vector Machine ; AUC 0.500

In [ ]:

```
# Normalization

def SVMn():
    import numpy as np
    import pandas as pd
    from sklearn.svm import SVC
    from sklearn.model_selection import LeaveOneOut

    from sklearn.preprocessing import MinMaxScaler

    # Statistical functions
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import r2_score

    # Rounding of fractional point
    import math
    def my_round(val, digit=0):
        p = 10 ** digit
        return (val * p * 2 + 1) // 2 / p


    x = []
    t = []

    # Predicted value
    pred = []
    round = []
    pairs = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)

    x = np.array(x)
    scaler = MinMaxScaler()
    scaler.fit(x)
    x = scaler.transform(x)
    t = np.array(t)

    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)
    svm = SVC()


    for train_index, test_index in loo.split(x):
        x_train, x_test = x[train_index], x[test_index]
        t_train, t_test = t[train_index], t[test_index]
        svm.fit(x_train, t_train)
        result = svm.predict(x_test)
        pred.append(result[0])


        # Category of prediction
        if (result[0]<0.5):
            round.append(int(0))
        else:
```

```
        round.append(int(1))

    for i in range(len(pred)):
        pairs.append([pred[i], t[i]])

    output = pd.DataFrame(pairs)
    # output.to_csv('drive/My Drive/program/data/cstatdata.csv', index=False)

    # Coefficient of Determination
    r2 = r2_score(t, pred)
    # AUC
    auc = roc_auc_score(t, pred)

    # Confusion Matrix
    TP = 0
    FP = 0
    FN = 0
    TN = 0

    for i in range(len(round)):
        if (round[i]==1) & (t[i]==1):
            TP += 1
        elif (round[i]==1) & (t[i]==0):
            FP += 1
        elif (round[i]==0) & (t[i]==1):
            FN += 1
        elif (round[i]==0) & (t[i]==0):
            TN += 1

    confmat = [[TP, FP], [FN, TN]]

    acc = (TP+TN)/(TP+FP+FN+TN)

    if TP + FP != 0:
        prec = TP/(TP+FP)
    elif TP + FP == 0:
        prec = 'N/A'
    if TP + FN != 0:
        rec = TP/(TP+FN)
    elif TP + FP ==0:
        rec = 'N/A'
    if TN + FP != 0:
        spec = TN/(TN + FP)
    elif TN + FP == 0:
        spec = 'N/A'
    if TP + FP/2 + FN/2 != 0:
        f1 = TP/(TP + FP/2 + FN/2)
    elif TP + FP/2 + FN/2 == 0:
        f1 = 'N/A'


    res = np.array([auc, acc, prec, rec, f1, spec, [[TP, FP], [FN, TN]], r2])

    # Output : ( 1. AUC, 2, Accuracy, 3. Precision 4. Recall 5. f1-score 6. Specificity, 7. Confusion Matri
x, 8. R^2 in LOO)

    print('■ Leave-One-Out Cross Validation with Support Vector Machine with Normalizatioin')
    print('')
    print('Sample size of all dataset: ' + str(len(x_train) + len(x_test)) )
    print('Sample size of training data: ' + str(len(x_train)) + '    Explanatory variables: ' + str(x_train.
shape) + '   Target value: ' + str(t_train.shape))
```

```
    print('Sample size of test data: ' + str(len(x_test)) + '    Explanatory variables: ' + str(x_test.shape) + '   Target value: ' + str(t_test.shape))
    print('')
    print('AUC: ' + str(my_round(auc, 5)) + '   Accuracy: ' + str(my_round(acc,5)) + '   R2: ' + str(my_round(r2,5)))
    print('')
    print(pd.DataFrame(np.array(confmat), index=['Predict True', 'Predict False'], columns=['Actual True', 'Actual False']))
    print('')
    print('Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confution matrix 8. R^2 in LOO')
    print('')

    return res


# Standardization

def SVMs():
    import numpy as np
    import pandas as pd
    from sklearn.svm import SVC
    from sklearn.model_selection import LeaveOneOut

    from sklearn.preprocessing import StandardScaler

    # Statistical functions
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import r2_score

    # Rounding of fractional point
    import math
    def my_round(val, digit=0):
        p = 10 ** digit
        return (val * p * 2 + 1) // 2 / p

    x = []
    t = []

    # Predicted value
    pred = []
    round = []
    pairs = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)

    x = np.array(x)
    scaler = StandardScaler()
    scaler.fit(x)
    x = scaler.transform(x)
    t = np.array(t)

    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)
    svm = SVC()
```

```
for train_index, test_index in loo.split(x):
    x_train, x_test = x[train_index], x[test_index]
    t_train, t_test = t[train_index], t[test_index]
    svm.fit(x_train, t_train)
    result = svm.predict(x_test)
    pred.append(result[0])

    # Category of prediction
    if (result[0]<0.5):
        round.append(int(0))
    else:
        round.append(int(1))

for i in range(len(pred)):
    pairs.append([pred[i], t[i]])

output = pd.DataFrame(pairs)
# output.to_csv('drive/My Drive/program/data/cstatdata.csv', index=False)

# Coefficient of Determination
r2 = r2_score(t, pred)
# AUC
auc = roc_auc_score(t, pred)

# Confusion Matrix
TP = 0
FP = 0
FN = 0
TN = 0

for i in range(len(round)):
    if (round[i]==1) & (t[i]==1):
        TP += 1
    elif (round[i]==1) & (t[i]==0):
        FP += 1
    elif (round[i]==0) & (t[i]==1):
        FN += 1
    elif (round[i]==0) & (t[i]==0):
        TN += 1

confmat = [[TP, FP], [FN, TN]]

acc = (TP+TN)/(TP+FP+FN+TN)

if TP + FP != 0:
    prec = TP/(TP+FP)
elif TP + FP == 0:
    prec = 'N/A'
if TP + FN != 0:
    rec = TP/(TP+FN)
elif TP + FP ==0:
    rec = 'N/A'
if TN + FP != 0:
    spec = TN/(TN + FP)
elif TN + FP == 0:
    spec = 'N/A'
if TP + FP/2 + FN/2 != 0:
    f1 = TP/(TP + FP/2 + FN/2)
elif TP + FP/2 + FN/2 == 0:
```

```
        f1 = 'N/A'

    res = np.array([auc, acc, prec, rec, f1, spec, [[TP, FP], [FN, TN]], r2])

    # Output : ( 1. AUC, 2, Accuracy, 3. Precision 4. Recall 5. f1-score 6. Specificity, 7. Confusion Matri
x, 8. R^2 in LOO)

    print('■ Leave-One-Out Cross Validation with Support Vector Machine with Standardizatioin')
    print('')
    print('Sample size of all dataset: ' + str(len(x_train) + len(x_test)) )
    print('Sample size of training data: ' + str(len(x_train)) + '    Explanatory variables: ' + str(x_train.
shape) + '   Target value: ' + str(t_train.shape))
    print('Sample size of test data: ' + str(len(x_test)) + '    Explanatory variables: ' + str(x_test.shap
e) + '   Target value: ' + str(t_test.shape))
    print('')
    print('AUC: ' + str(my_round(auc, 5)) + '   Accuracy: ' + str(my_round(acc,5)) + '   R2: ' + str(my_
round(r2,5)))
    print('')
    print(pd.DataFrame(np.array(confmat), index=['Predict True', 'Predict False'], columns=['Actual T
rue', 'Actual False']))
    print('')
    print('Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confution matr
ix 8. R^2 in LOO')
    print('')

    return res
```

In [ ]:

```
SVMn()
```

```
■ Leave-One-Out Cross Validation with Support Vector Machine with Normalizatioin

Sample size of all dataset: 51
Sample size of training data: 50    Explanatory variables: (50, 211)   Target value: (5
0,)
Sample size of test data: 1    Explanatory variables: (1, 211)   Target value: (1,)

AUC: 0.5   Accuracy: 0.78431   R2: -0.275

              Actual True  Actual False
Predict True          0          0
Predict False        11         40

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO
```

Out[ ]:

```
array([0.5, 0.7843137254901961, 'N/A', 0.0, 0.0, 1.0,
       list([[0, 0], [11, 40]]), -0.27500000000000036], dtype=object)
```

In [ ]:

```
SVMs()
```

■ Leave-One-Out Cross Validation with Support Vector Machine with Standardizatioin

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.5 Accuracy: 0.78431 R2: -0.275

	Actual True	Actual False
Predict True	0	0
Predict False	11	40

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.5, 0.7843137254901961, 'N/A', 0.0, 0.0, 1.0,
       list([[0, 0], [11, 40]]), -0.27500000000000036], dtype=object)
```

6) Random Forest ; AUC 0.545

In [ ]:

```
def randomF(random_state):
    import numpy as np
    import pandas as pd
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import LeaveOneOut

    # Statistical functions
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import r2_score

    # Rounding of fractional point
    import math
    def my_round(val, digit=0):
        p = 10 ** digit
        return (val * p * 2 + 1) // 2 / p

    x = []
    t = []

    # Predicted value
    pred = []
    round = []
    pairs = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)

    x = np.array(x)
    t = np.array(t)

    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)
    rf = RandomForestClassifier(random_state=random_state)


    for train_index, test_index in loo.split(x):
        x_train, x_test = x[train_index], x[test_index]
        t_train, t_test = t[train_index], t[test_index]
        rf.fit(x_train, t_train)
        result = rf.predict(x_test)
        pred.append(result[0])

        # Category of prediction
        if (result[0]<0.5):
            round.append(int(0))
        else:
            round.append(int(1))

    for i in range(len(pred)):
        pairs.append([pred[i], t[i]])

    output = pd.DataFrame(pairs)
    # output.to_csv('drive/My Drive/program/data/cstatdata.csv', index=False)

    # Coefficient of Determination
```

```
    r2 = r2_score(t, pred)
    # AUC
    auc = roc_auc_score(t, pred)

    # Confusion Matrix
    TP = 0
    FP = 0
    FN = 0
    TN = 0

    for i in range(len(round)):
        if (round[i]==1) & (t[i]==1):
            TP += 1
        elif (round[i]==1) & (t[i]==0):
            FP += 1
        elif (round[i]==0) & (t[i]==1):
            FN += 1
        elif (round[i]==0) & (t[i]==0):
            TN += 1

    confmat = [[TP, FP], [FN, TN]]

    acc = (TP+TN)/(TP+FP+FN+TN)

    if TP + FP != 0:
        prec = TP/(TP+FP)
    elif TP + FP == 0:
        prec = 'N/A'
    if TP + FN != 0:
        rec = TP/(TP+FN)
    elif TP + FP ==0:
        rec = 'N/A'
    if TN + FP != 0:
        spec = TN/(TN + FP)
    elif TN + FP == 0:
        spec = 'N/A'
    if TP + FP/2 + FN/2 != 0:
        f1 = TP/(TP + FP/2 + FN/2)
    elif TP + FP/2 + FN/2 == 0:
        f1 = 'N/A'

    res = np.array([auc, acc, prec, rec, f1, spec, [[TP, FP], [FN, TN]], r2])

    # Output : ( 1. AUC, 2, Accuracy, 3. Precision 4. Recall 5. f1-score 6. Specificity, 7. Confusion Matri
x, 8. R^2 in LOO)

    print('■ Leave-One-Out Cross Validation with Random Forest')
    print('')
    print('Sample size of all dataset: ' + str(len(x_train) + len(x_test)) )
    print('Sample size of training data: ' + str(len(x_train)) + '    Explanatory variables: ' + str(x_train.
shape) + '   Target value: ' + str(t_train.shape))
    print('Sample size of test data: ' + str(len(x_test)) + '    Explanatory variables: ' + str(x_test.shap
e) + '   Target value: ' + str(t_test.shape))
    print('')
    print('AUC: ' + str(my_round(auc, 5)) + '   Accuracy: ' + str(my_round(acc,5)) + '   R2: ' + str(my_
round(r2,5)))
    print('')
    print(pd.DataFrame(np.array(confmat), index=['Predict True', 'Predict False'], columns=['Actual T
rue', 'Actual False']))
    print('')
    print('Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confution matr
```

```
ix 8. R^2 in LOO')
    print('')

#    from pprint import pprint
#    print('◆ The defalut settings of the hyperparameters')
#    pprint(rf.get_params())
    # n_estimators, max_features should be optimized first
    # then, max_depth, min_sample_split, min_samples_leaf, bootstrap

    return res
```

In [ ]:

```
randomF(1)
```

■ Leave-One-Out Cross Validation with Random Forest

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.53295 Accuracy: 0.78431 R2: -0.275

	Actual True	Actual False
Predict True	1	1
Predict False	10	39

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.5329545454545453, 0.7843137254901961, 0.5, 0.090909090909090
91,
       0.15384615384615385, 0.975, list([[1, 1], [10, 39]]),
       -0.27500000000000036], dtype=object)
```

Optimization with RandomizedSearchCV & GridSearchCV

A. RandomSearch

In [ ]:

```
def RandomSearch(variables, label):
    from sklearn.model_selection import RandomizedSearchCV
    from pprint import pprint

    # Number of trees in random forest
    n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
    # Number of features to consider at every split
    max_features = ['auto', 'sqrt']
    # Maximum number of levels in tree
    max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
    max_depth.append(None)
    # Minimum number of samples required to split a node
    min_samples_split = [2, 5, 10]
    # Minimum number of samples required at each leaf node
    min_samples_leaf = [1, 2, 4]
    # Method of selecting samples for training each tree
    bootstrap = [True, False]

    # Create the random grid
    random_grid = {'n_estimators': n_estimators,
                   'max_features': max_features,
                   'max_depth': max_depth,
                   'min_samples_split': min_samples_split,
                   'min_samples_leaf': min_samples_leaf,
                   'bootstrap': bootstrap}

    print('◆ Random Hypterparameter Grid')
    pprint(random_grid)
    print('')

    # Use the random grid to search for best hyperparameters
    # First create the base model to tune
    from sklearn.ensemble import RandomForestClassifier
    rf = RandomForestClassifier()
    # Random search of parameters, using 3 fold cross validation,
    # search across 100 different combinations, and use all available cores
    rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid, n_iter = 1
00, cv = 3, verbose=2, random_state=42, n_jobs = -1)
    # Fit the random search model
    rf_random.fit(variables, label)
    print('')
    print('◆ Best Parameters using RandomizedSearchCV')
    pprint(rf_random.best_params_)
```

In [ ]:

```
RandomSearch(x, t)
```

◆ Random Hypterparameter Grid
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]}

Fitting 3 folds for each of 100 candidates, totalling 300 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done 37 tasks | elapsed: 45.1s
[Parallel(n_jobs=-1)]: Done 158 tasks | elapsed: 3.0min
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed: 5.9min finished

◆ Best Parameters using RandomizedSearchCV
{'bootstrap': False,
 'max_depth': 30,
 'max_features': 'sqrt',
 'min_samples_leaf': 4,
 'min_samples_split': 5,
 'n_estimators': 800}

B. GridSearch

In [ ]:

```
def GridSearch(variables, label):
  from sklearn.model_selection import GridSearchCV
  from pprint import pprint
  # Create the parameter grid based on the results of random search
  param_grid = {
     'bootstrap': [True],
     'max_depth': [20,30,40],
     'max_features': ['sqrt'],
     'min_samples_leaf': [1,2,3],
     'min_samples_split': [4,5,6],
     'n_estimators': [300,400,500]
  }
  # Create a based model
  from sklearn.ensemble import RandomForestClassifier
  rf = RandomForestClassifier()
  # Instantiate the grid search model
  grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                             cv = 3, n_jobs = -1, verbose = 2)
  # Fit the grid search to the data
  grid_search.fit(variables, label)
  print('')
  print('◆ Best Parameters using GridSearchCV')
  pprint(grid_search.best_params_)
```

In [ ]:

```
GridSearch(x, t)
```

```
Fitting 3 folds for each of 81 candidates, totalling 243 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:   18.5s
[Parallel(n_jobs=-1)]: Done 158 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 243 out of 243 | elapsed:  1.9min finished

◆ Best Parameters using GridSearchCV
{'bootstrap': True,
 'max_depth': 20,
 'max_features': 'sqrt',
 'min_samples_leaf': 1,
 'min_samples_split': 4,
 'n_estimators': 300}
```

In [ ]:

```
def randomforest(random_state, bootstrap, max_depth, max_features, min_samples_leaf, min_sam
ples_split, n_estimators):
    import numpy as np
    import pandas as pd
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import LeaveOneOut

    # Statistical functions
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import r2_score

    # Rounding of fractional point
    import math
    def my_round(val, digit=0):
        p = 10 ** digit
        return (val * p * 2 + 1) // 2 / p

    x = []
    t = []

    # Predicted value
    pred = []
    round = []
    pairs = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)

    x = np.array(x)
    t = np.array(t)

    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)
    rf = RandomForestClassifier(random_state=random_state,
                                bootstrap = True,
                                max_depth = 20,
                                max_features = 'sqrt',
                                min_samples_leaf = 1,
                                min_samples_split = 4,
                                n_estimators = 300
                                )


    for train_index, test_index in loo.split(x):
        x_train, x_test = x[train_index], x[test_index]
        t_train, t_test = t[train_index], t[test_index]
        rf.fit(x_train, t_train)
        result = rf.predict(x_test)
        pred.append(result[0])

        # Category of prediction
        if (result[0]<0.5):
            round.append(int(0))
        else:
            round.append(int(1))
```

```
    for i in range(len(pred)):
        pairs.append([pred[i], t[i]])

    output = pd.DataFrame(pairs)
    # output.to_csv('drive/My Drive/practice/data/cstatdata.csv', index=False)

    # Coefficient of Determination
    r2 = r2_score(t, pred)
    # AUC
    auc = roc_auc_score(t, pred)

    # Confusion Matrix
    TP = 0
    FP = 0
    FN = 0
    TN = 0

    for i in range(len(round)):
        if (round[i]==1) & (t[i]==1):
            TP += 1
        elif (round[i]==1) & (t[i]==0):
            FP += 1
        elif (round[i]==0) & (t[i]==1):
            FN += 1
        elif (round[i]==0) & (t[i]==0):
            TN += 1

    confmat = [[TP, FP], [FN, TN]]

    acc = (TP+TN)/(TP+FP+FN+TN)

    if TP + FP != 0:
        prec = TP/(TP+FP)
    elif TP + FP == 0:
        prec = 'N/A'
    if TP + FN != 0:
        rec = TP/(TP+FN)
    elif TP + FP ==0:
        rec = 'N/A'
    if TN + FP != 0:
        spec = TN/(TN + FP)
    elif TN + FP == 0:
        spec = 'N/A'
    if TP + FP/2 + FN/2 != 0:
        f1 = TP/(TP + FP/2 + FN/2)
    elif TP + FP/2 + FN/2 == 0:
        f1 = 'N/A'

    res = np.array([auc, acc, prec, rec, f1, spec, [[TP, FP], [FN, TN]], r2])

    # Output : ( 1. AUC, 2, Accuracy, 3. Precision 4. Recall 5. f1-score 6. Specificity, 7. Confusion Matri
x, 8. R^2 in LOO)

    print('■ Leave-One-Out Cross Validation with Random Forest')
    print('')
    print('Sample size of all dataset: ' + str(len(x_train) + len(x_test)) )
    print('Sample size of training data: ' + str(len(x_train)) + '    Explanatory variables: ' + str(x_train.
shape) + '   Target value: ' + str(t_train.shape))
    print('Sample size of test data: ' + str(len(x_test)) + '    Explanatory variables: ' + str(x_test.shap
e) + '   Target value: ' + str(t_test.shape))
```

```
    print('')
    print('AUC: ' + str(my_round(auc, 5)) + '   Accuracy: ' + str(my_round(acc,5)) + '   R2: ' + str(my_
round(r2,5)))
    print('')
    print(pd.DataFrame(np.array(confmat), index=['Predict True', 'Predict False'], columns=['Actual T
rue', 'Actual False']))
    print('')
    print('Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confution matr
ix 8. R^2 in LOO')
    print('')

    from pprint import pprint
    print('◆ The settings of the hyperparameters')
    pprint(rf.get_params())
    # n_estimators, max_features should be optimized first
    # then, max_depth, min_sample_split, min_samples_leaf, bootstrap
    print('')

    return res
```

In [ ]:

```
randomforest(41, True, 20, 'sqrt', 1, 4, 300)
```

■ Leave-One-Out Cross Validation with Random Forest

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.53295 Accuracy: 0.78431 R2: -0.275

	Actual True	Actual False
Predict True	1	1
Predict False	10	39

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

◆ The settings of the hyperparameters
{'bootstrap': True,
'ccp_alpha': 0.0,
'class_weight': None,
'criterion': 'gini',
'max_depth': 20,
'max_features': 'sqrt',
'max_leaf_nodes': None,
'max_samples': None,
'min_impurity_decrease': 0.0,
'min_impurity_split': None,
'min_samples_leaf': 1,
'min_samples_split': 4,
'min_weight_fraction_leaf': 0.0,
'n_estimators': 300,
'n_jobs': None,
'oob_score': False,
'random_state': 41,
'verbose': 0,
'warm_start': False}

Out[ ]:

```
array([0.5329545454545453, 0.7843137254901961, 0.5, 0.090909090909090
91,
       0.15384615384615385, 0.975, list([[1, 1], [10, 39]]),
       -0.27500000000000036], dtype=object)
```

In [ ]:

```
randomforest(625, False, 20, 'sqrt', 1, 4, 300)
```

■ Leave-One-Out Cross Validation with Random Forest

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.53295 Accuracy: 0.78431 R2: -0.275

	Actual True	Actual False
Predict True	1	1
Predict False	10	39

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

◆ The settings of the hyperparameters
{'bootstrap': True,
'ccp_alpha': 0.0,
'class_weight': None,
'criterion': 'gini',
'max_depth': 20,
'max_features': 'sqrt',
'max_leaf_nodes': None,
'max_samples': None,
'min_impurity_decrease': 0.0,
'min_impurity_split': None,
'min_samples_leaf': 1,
'min_samples_split': 4,
'min_weight_fraction_leaf': 0.0,
'n_estimators': 300,
'n_jobs': None,
'oob_score': False,
'random_state': 625,
'verbose': 0,
'warm_start': False}

Out[ ]:

```
array([0.5329545454545453, 0.7843137254901961, 0.5, 0.090909090909090
91,
       0.15384615384615385, 0.975, list([[1, 1], [10, 39]]),
       -0.27500000000000036], dtype=object)
```

Optimization with Optuna

In [ ]:

```
def randomFOptuna(trial):
    import numpy as np
    import pandas as pd
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import LeaveOneOut
    import optuna
    from sklearn.metrics import roc_auc_score

    random_state = trial.suggest_int('random_state', 1, 2000)

    x = []
    t = []

    # Predicted value
    pred = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)

    x = np.array(x)
    t = np.array(t)

    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)
    rf = RandomForestClassifier(random_state = random_state)

    for train_index, test_index in loo.split(x):
        x_train, x_test = x[train_index], x[test_index]
        t_train, t_test = t[train_index], t[test_index]
        rf.fit(x_train, t_train)
        result = rf.predict(x_test)
        pred.append(result[0])

    auc = roc_auc_score(t, pred)

    # minimize 1/AUC
    return 1/auc


def randomFTrial(n_trials):
    import optuna
    study = optuna.create_study()
    study.optimize(randomFOptuna, n_trials)
    # result
    print()
    print('hyperparameter:', study.best_params)
    print('AUC:', 1/study.best_value)
    print()
    return study.best_params['random_state']
```

In [ ]:

```
randomFTrial(100)
```

```
[I 2020-09-19 15:37:06,041] A new study created in memory with name: no-name-0
9782be3-9633-48e3-b32d-8f05601f4e45
[I 2020-09-19 15:37:13,276] Trial 0 finished with value: 1.8763326226012798 and
parameters: {'random_state': 739}. Best is trial 0 with value: 1.876332622601279
8.
[I 2020-09-19 15:37:20,449] Trial 1 finished with value: 1.8763326226012798 and
parameters: {'random_state': 32}. Best is trial 0 with value: 1.8763326226012798.
[I 2020-09-19 15:37:27,634] Trial 2 finished with value: 1.8763326226012798 and
parameters: {'random_state': 1593}. Best is trial 0 with value: 1.876332622601279
8.
[I 2020-09-19 15:37:34,823] Trial 3 finished with value: 1.8763326226012798 and
parameters: {'random_state': 1318}. Best is trial 0 with value: 1.876332622601279
8.
[I 2020-09-19 15:37:42,068] Trial 4 finished with value: 1.8333333333333335 and
parameters: {'random_state': 1883}. Best is trial 4 with value: 1.833333333333333
5.
[I 2020-09-19 15:37:49,290] Trial 5 finished with value: 1.8763326226012798 and
parameters: {'random_state': 1377}. Best is trial 4 with value: 1.833333333333333
5.
[I 2020-09-19 15:37:56,516] Trial 6 finished with value: 1.8333333333333335 and
parameters: {'random_state': 1014}. Best is trial 4 with value: 1.833333333333333
5.
[I 2020-09-19 15:38:03,725] Trial 7 finished with value: 1.8763326226012798 and
parameters: {'random_state': 633}. Best is trial 4 with value: 1.833333333333333
5.
[I 2020-09-19 15:38:10,917] Trial 8 finished with value: 1.8763326226012798 and
parameters: {'random_state': 540}. Best is trial 4 with value: 1.833333333333333
5.
[I 2020-09-19 15:38:18,180] Trial 9 finished with value: 1.8763326226012798 and
parameters: {'random_state': 1303}. Best is trial 4 with value: 1.833333333333333
5.
[I 2020-09-19 15:38:25,402] Trial 10 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1932}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:38:32,573] Trial 11 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1958}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:38:39,803] Trial 12 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 24}. Best is trial 4 with value: 1.833333333333333
5.
[I 2020-09-19 15:38:47,030] Trial 13 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 946}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:38:54,227] Trial 14 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 273}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:39:01,410] Trial 15 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1691}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:39:08,482] Trial 16 finished with value: 2.0 and parameters: {'rand
om_state': 1073}. Best is trial 4 with value: 1.8333333333333335.
[I 2020-09-19 15:39:15,626] Trial 17 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1771}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:39:22,746] Trial 18 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1006}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:39:29,974] Trial 19 finished with value: 2.0512820512820515 a
nd parameters: {'random_state': 349}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:39:37,103] Trial 20 finished with value: 1.8333333333333335 a
```

```
nd parameters: {'random_state': 1490}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:39:44,239] Trial 21 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1985}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:39:51,386] Trial 22 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1881}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:39:58,488] Trial 23 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1521}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:40:05,690] Trial 24 finished with value: 2.0 and parameters: {'rand
om_state': 1183}. Best is trial 4 with value: 1.8333333333333335.
[I 2020-09-19 15:40:12,921] Trial 25 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 845}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:40:20,070] Trial 26 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1514}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:40:27,254] Trial 27 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1745}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:40:34,351] Trial 28 finished with value: 2.0512820512820515 a
nd parameters: {'random_state': 860}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:40:41,560] Trial 29 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 799}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:40:48,628] Trial 30 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 744}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:40:55,705] Trial 31 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1140}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:41:02,878] Trial 32 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 839}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:41:09,989] Trial 33 finished with value: 2.0512820512820515 a
nd parameters: {'random_state': 527}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:41:17,073] Trial 34 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 267}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:41:24,182] Trial 35 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 717}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:41:31,240] Trial 36 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1252}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:41:38,557] Trial 37 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 580}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:41:46,176] Trial 38 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 883}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:41:53,277] Trial 39 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 688}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:42:00,440] Trial 40 finished with value: 1.921397379912664 and
parameters: {'random_state': 1487}. Best is trial 4 with value: 1.833333333333333
5.
```

[I 2020-09-19 15:42:07,598] Trial 41 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1391}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:42:15,341] Trial 42 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1069}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:42:23,064] Trial 43 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 810}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:42:31,028] Trial 44 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 451}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:42:38,229] Trial 45 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 967}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:42:45,405] Trial 46 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1878}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:42:52,444] Trial 47 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1840}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:42:59,525] Trial 48 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 809}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:43:06,679] Trial 49 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1822}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:43:13,773] Trial 50 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1672}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:43:20,887] Trial 51 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1846}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:43:28,095] Trial 52 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1992}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:43:35,187] Trial 53 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1634}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:43:42,238] Trial 54 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1629}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:43:49,269] Trial 55 finished with value: 2.0512820512820515 a
nd parameters: {'random_state': 1421}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:43:56,367] Trial 56 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 629}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:44:03,453] Trial 57 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 913}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:44:10,614] Trial 58 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1057}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:44:17,692] Trial 59 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 784}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:44:24,782] Trial 60 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1724}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:44:31,823] Trial 61 finished with value: 1.8763326226012798 a

```
nd parameters: {'random_state': 1931}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:44:38,869] Trial 62 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1590}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:44:45,969] Trial 63 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1920}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:44:53,025] Trial 64 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1769}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:45:00,114] Trial 65 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1307}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:45:07,238] Trial 66 finished with value: 2.0 and parameters: {'rand
om_state': 1038}. Best is trial 4 with value: 1.8333333333333335.
[I 2020-09-19 15:45:14,320] Trial 67 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1306}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:45:21,516] Trial 68 finished with value: 2.0512820512820515 a
nd parameters: {'random_state': 1802}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:45:28,739] Trial 69 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1889}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:45:35,908] Trial 70 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1981}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:45:43,046] Trial 71 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1968}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:45:50,188] Trial 72 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1148}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:45:57,361] Trial 73 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 671}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:46:04,556] Trial 74 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1234}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:46:11,683] Trial 75 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1465}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:46:18,787] Trial 76 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1431}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:46:26,011] Trial 77 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1543}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:46:33,156] Trial 78 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1559}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:46:40,365] Trial 79 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 939}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:46:47,635] Trial 80 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1370}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:46:55,360] Trial 81 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1110}. Best is trial 4 with value: 1.8333333333333
335.
```

```
[I 2020-09-19 15:47:02,563] Trial 82 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1101}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:47:09,699] Trial 83 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1104}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:47:16,882] Trial 84 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 847}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:47:24,191] Trial 85 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 832}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:47:31,366] Trial 86 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1233}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:47:39,197] Trial 87 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1003}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:47:47,248] Trial 88 finished with value: 1.921397379912664 and
parameters: {'random_state': 985}. Best is trial 4 with value: 1.833333333333333
5.
[I 2020-09-19 15:47:55,474] Trial 89 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1676}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:48:02,831] Trial 90 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1463}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:48:10,111] Trial 91 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1612}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:48:17,360] Trial 92 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1331}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:48:24,432] Trial 93 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 907}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:48:31,621] Trial 94 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1525}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:48:38,824] Trial 95 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 762}. Best is trial 4 with value: 1.83333333333333
35.
[I 2020-09-19 15:48:46,117] Trial 96 finished with value: 1.8763326226012798 a
nd parameters: {'random_state': 1555}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:48:53,276] Trial 97 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1276}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:49:00,544] Trial 98 finished with value: 1.8333333333333335 a
nd parameters: {'random_state': 1278}. Best is trial 4 with value: 1.8333333333333
335.
[I 2020-09-19 15:49:07,684] Trial 99 finished with value: 1.921397379912664 and
parameters: {'random_state': 1272}. Best is trial 4 with value: 1.833333333333333
5.

hyperparameter: {'random_state': 1883}
AUC: 0.5454545454545454
```

Out[ ]:

1883

In [ ]:

```
# This random_state value is one example of hyperparameter optimization. There are other values th
at yield a similar AUC value.
randomF(1883)
```

■ Leave-One-Out Cross Validation with Random Forest

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.54545 Accuracy: 0.80392 R2: -0.15909

	Actual True	Actual False
Predict True	1	0
Predict False	10	40

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

array([0.5454545454545454, 0.803921568627451, 1.0, 0.0909090909090909
1,
0.16666666666666666, 1.0, list([[1, 0], [10, 40]]),
-0.1590909090909094], dtype=object)

7) XGBoost ; AUC 0.668

In [ ]:

```
def xgboost(eta):
    import numpy as np
    import pandas as pd
    from sklearn.model_selection import LeaveOneOut
    import xgboost as xgb

    # Statistical functions
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import r2_score

    # Rounding of fractional point
    import math
    def my_round(val, digit=0):
        p = 10 ** digit
        return (val * p * 2 + 1) // 2 / p

    x = []
    t = []

    # Predicted value
    round = []
    pred = []
    pairs = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)

    x = np.array(x)
    t = np.array(t)


    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)


    # Hypter-parameter settings
    params = {
        'objective' : 'binary:logistic',
        'silent' : 0,
        'eta' : eta,
        'random_state' : 71,
        'max_depth' : 5,
        'eval_metric' : 'logloss'
    }

    for train_index, test_index in loo.split(x):
        x_train, x_test = x[train_index], x[test_index]
        t_train, t_test = t[train_index], t[test_index]
        # transform data structure for XGBoost
        d_train = xgb.DMatrix(x_train, label=t_train)
        d_test = xgb.DMatrix(x_test, label=t_test)
        # training
        model = xgb.train(params, d_train)
        result = model.predict(d_test)
        pred.append(result[0])
```

```
        # Category of prediction
        if (result[0]<0.5):
            round.append(int(0))
        else:
            round.append(int(1))

    for i in range(len(pred)):
        pairs.append([pred[i], t[i]])

    output = pd.DataFrame(pairs)
    # output.to_csv('drive/My Drive/program/data/cstatdata.csv', index=False)

    # Coefficient of Determination
    r2 = r2_score(t, pred)
    # AUC
    auc = roc_auc_score(t, pred)

    # Confusion Matrix
    TP = 0
    FP = 0
    FN = 0
    TN = 0

    for i in range(len(round)):
        if (round[i]==1) & (t[i]==1):
            TP += 1
        elif (round[i]==1) & (t[i]==0):
            FP += 1
        elif (round[i]==0) & (t[i]==1):
            FN += 1
        elif (round[i]==0) & (t[i]==0):
            TN += 1

    confmat = [[TP, FP], [FN, TN]]


    acc = (TP+TN)/(TP+FP+FN+TN)

    if TP + FP != 0:
        prec = TP/(TP+FP)
    elif TP + FP == 0:
        prec = 'N/A'
    if TP + FN != 0:
        rec = TP/(TP+FN)
    elif TP + FP ==0:
        rec = 'N/A'
    if TN + FP != 0:
        spec = TN/(TN + FP)
    elif TN + FP == 0:
        spec = 'N/A'
    if TP + FP/2 + FN/2 != 0:
        f1 = TP/(TP + FP/2 + FN/2)
    elif TP + FP/2 + FN/2 == 0:
        f1 = 'N/A'

    res = np.array([auc, acc, prec, rec, f1, spec, [[TP, FP], [FN, TN]], r2])

    # Output : ( 1. AUC, 2, Accuracy, 3. Precision 4. Recall 5. f1-score 6. Specificity, 7. Confusion Matri
x, 8. R^2 in LOO)
```

```
    print('■ Leave-One-Out Cross Validation with XGBoost')
    print('')
    print('Sample size of all dataset: ' + str(len(x_train) + len(x_test)) )
    print('Sample size of training data: ' + str(len(x_train)) + '    Explanatory variables: ' + str(x_train.
shape) + '   Target value: ' + str(t_train.shape))
    print('Sample size of test data: ' + str(len(x_test)) + '    Explanatory variables: ' + str(x_test.shap
e) + '   Target value: ' + str(t_test.shape))
    print('')
    print('AUC: ' + str(my_round(auc, 5)) + '   Accuracy: ' + str(my_round(acc,5)) + '   R2: ' + str(my_
round(r2,5)))
    print('')
    print(pd.DataFrame(np.array(confmat), index=['Predict True', 'Predict False'], columns=['Actual T
rue', 'Actual False']))
    print('')
    print('Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confution matr
ix 8. R^2 in LOO')
    print('')


    return res
```

In [ ]:

```
xgboost(0.1)
```

```
■ Leave-One-Out Cross Validation with XGBoost

Sample size of all dataset: 51
Sample size of training data: 50    Explanatory variables: (50, 211)   Target value: (5
0,)
Sample size of test data: 1    Explanatory variables: (1, 211)   Target value: (1,)

AUC: 0.64091   Accuracy: 0.72549   R2: -0.03556

              Actual True  Actual False
Predict True            1             4
Predict False          10            36

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO
```

Out[ ]:

```
array([0.640909090909091, 0.7254901960784313, 0.2, 0.0909090909090909
1,
       0.125, 0.9, list([[1, 4], [10, 36]]), -0.03556390450401259],
      dtype=object)
```

In [ ]:

```
xgboost(0.3)
```

■ Leave-One-Out Cross Validation with XGBoost

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.57273 Accuracy: 0.76471 R2: -0.11337

	Actual True	Actual False
Predict True	1	2
Predict False	10	38

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.5727272727272728, 0.7647058823529411, 0.3333333333333333,
       0.09090909090909091, 0.14285714285714285, 0.95,
       list([[1, 2], [10, 38]]), -0.11337225409718865], dtype=object)
```

In [ ]:

```
xgboost(0.2)
```

■ Leave-One-Out Cross Validation with XGBoost

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.675 Accuracy: 0.78431 R2: 2e-05

	Actual True	Actual False
Predict True	1	1
Predict False	10	39

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.675, 0.7843137254901961, 0.5, 0.09090909090909091,
       0.15384615384615385, 0.975, list([[1, 1], [10, 39]]),
       1.922716373370381e-05], dtype=object)
```

In [ ]:

```
xgboost(0.911)
```

■ Leave-One-Out Cross Validation with XGBoost

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.59091 Accuracy: 0.76471 R2: -0.19333

	Actual True	Actual False
Predict True	2	3
Predict False	9	37

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.5909090909090908, 0.7647058823529411, 0.4, 0.1818181818181
82,
       0.25, 0.925, list([[2, 3], [9, 37]]), -0.1933297138660992],
      dtype=object)
```

Optimization with Optuna

In [ ]:

```
def xgboostOptuna(trial):
    import numpy as np
    import pandas as pd
    from sklearn.model_selection import LeaveOneOut
    import xgboost as xgb

    import optuna

    # Statistical functions
    from sklearn.metrics import roc_auc_score

    eta = trial.suggest_uniform('eta', 0, 1)

    x = []
    t = []

    round = []
    pred = []
    pairs = []

    for row in data:
        u = []
        t.append(int(row[-1]))
        for i in range(0, len(row)-1):
            u.append(float(row[i]))
        x.append(u)

    x = np.array(x)
    t = np.array(t)


    loo = LeaveOneOut()
    entire_count = loo.get_n_splits(x)


    # Hypter-parameter settings
    params = {
        'objective' : 'binary:logistic',
        'silent' : 0,
        'eta' : eta,
        'random_state' : 71,
        'max_depth' : 5,
        'eval_metric' : 'logloss'
    }

    for train_index, test_index in loo.split(x):
        x_train, x_test = x[train_index], x[test_index]
        t_train, t_test = t[train_index], t[test_index]
        # transform data structure for XGBoost
        d_train = xgb.DMatrix(x_train, label=t_train)
        d_test = xgb.DMatrix(x_test, label=t_test)
        # training
        model = xgb.train(params, d_train)
        result = model.predict(d_test)
        pred.append(result[0])

    auc = roc_auc_score(t, pred)

    res = auc
```

```
    # minimize 1/AUC
    return 1/res


def xgboostTrial(n_trials):
    import optuna
    study = optuna.create_study()
    study.optimize(xgboostOptuna, n_trials)

    # result
    print()
    print('hyperparameter:', study.best_params)
    print('AUC:', 1/study.best_value)
```

In [ ]:

```
xgboostTrial(100)
```

```
[I 2020-09-19 16:02:42,740] A new study created in memory with name: no-name-3
b31c3e0-d521-4348-acc5-9232931ea4a1
[I 2020-09-19 16:02:43,339] Trial 0 finished with value: 1.71875 and parameters:
{'eta': 0.6435127138634738}. Best is trial 0 with value: 1.71875.
[I 2020-09-19 16:02:43,910] Trial 1 finished with value: 1.7120622568093389 and
parameters: {'eta': 0.7680080428910264}. Best is trial 1 with value: 1.712062256
8093389.
[I 2020-09-19 16:02:44,519] Trial 2 finished with value: 1.7054263565891472 and
parameters: {'eta': 0.470759986317866}. Best is trial 2 with value: 1.7054263565
891472.
[I 2020-09-19 16:02:45,035] Trial 3 finished with value: 1.6923076923076923 and
parameters: {'eta': 0.94708517550152}. Best is trial 3 with value: 1.6923076923
076923.
[I 2020-09-19 16:02:45,599] Trial 4 finished with value: 1.5714285714285714 and
parameters: {'eta': 0.9754406062384664}. Best is trial 4 with value: 1.571428571
4285714.
[I 2020-09-19 16:02:46,162] Trial 5 finished with value: 1.71875 and parameters:
{'eta': 0.48424198743269964}. Best is trial 4 with value: 1.5714285714285714.
[I 2020-09-19 16:02:46,769] Trial 6 finished with value: 1.8644067796610169 and
parameters: {'eta': 0.3457042099424291}. Best is trial 4 with value: 1.571428571
4285714.
[I 2020-09-19 16:02:47,377] Trial 7 finished with value: 1.6923076923076925 and
parameters: {'eta': 0.4041473283159833}. Best is trial 4 with value: 1.571428571
4285714.
[I 2020-09-19 16:02:47,944] Trial 8 finished with value: 1.7670682730923695 and
parameters: {'eta': 0.6128525520570003}. Best is trial 4 with value: 1.571428571
4285714.
[I 2020-09-19 16:02:48,530] Trial 9 finished with value: 1.6988416988416988 and
parameters: {'eta': 0.6079269350460001}. Best is trial 4 with value: 1.571428571
4285714.
[I 2020-09-19 16:02:49,061] Trial 10 finished with value: 1.8106995884773662 a
nd parameters: {'eta': 0.043897044888114356}. Best is trial 4 with value: 1.57142
85714285714.
[I 2020-09-19 16:02:49,668] Trial 11 finished with value: 1.660377358490566 and
parameters: {'eta': 0.9969024178313565}. Best is trial 4 with value: 1.571428571
4285714.
[I 2020-09-19 16:02:50,193] Trial 12 finished with value: 1.6666666666666667 a
nd parameters: {'eta': 0.9240076352074904}. Best is trial 4 with value: 1.5714285
714285714.
[I 2020-09-19 16:02:50,751] Trial 13 finished with value: 1.7254901960784312 a
nd parameters: {'eta': 0.9425076962143007}. Best is trial 4 with value: 1.5714285
714285714.
[I 2020-09-19 16:02:51,298] Trial 14 finished with value: 1.685823754789272 and
parameters: {'eta': 0.8024614110648411}. Best is trial 4 with value: 1.571428571
4285714.
[I 2020-09-19 16:02:51,912] Trial 15 finished with value: 1.6296296296296295 a
nd parameters: {'eta': 0.1762280921481318 4}. Best is trial 4 with value: 1.571428
5714285714.
[I 2020-09-19 16:02:52,543] Trial 16 finished with value: 1.5277777777777777 a
nd parameters: {'eta': 0.1959690871816175 4}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:02:53,126] Trial 17 finished with value: 1.588447653429603 and
parameters: {'eta': 0.2298077021948576}. Best is trial 16 with value: 1.52777777
77777777.
[I 2020-09-19 16:02:53,746] Trial 18 finished with value: 1.7813765182186236 a
nd parameters: {'eta': 0.07663116953912 03}. Best is trial 16 with value: 1.527777
7777777777.
[I 2020-09-19 16:02:54,339] Trial 19 finished with value: 1.6176470588235294 a
nd parameters: {'eta': 0.2683856867311064}. Best is trial 16 with value: 1.527777
7777777777.
[I 2020-09-19 16:02:54,939] Trial 20 finished with value: 1.611721611721612 and
```

parameters: {'eta': 0.13566679907715207}. Best is trial 16 with value: 1.5277777
777777777.
[I 2020-09-19 16:02:55,555] Trial 21 finished with value: 1.5942028985507246 a
nd parameters: {'eta': 0.2845145601955423 3}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:02:56,150] Trial 22 finished with value: 1.7054263565891474 a
nd parameters: {'eta': 0.2174442610115866}. Best is trial 16 with value: 1.527777
7777777777.
[I 2020-09-19 16:02:56,754] Trial 23 finished with value: 2.7160493827160495 a
nd parameters: {'eta': 0.004700297186714747}. Best is trial 16 with value: 1.5277
777777777777.
[I 2020-09-19 16:02:57,348] Trial 24 finished with value: 1.795918367346939 and
parameters: {'eta': 0.3367103213261914 4}. Best is trial 16 with value: 1.5277777
777777777.
[I 2020-09-19 16:02:57,973] Trial 25 finished with value: 1.6417910447761193 a
nd parameters: {'eta': 0.1710416335819682 8}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:02:58,597] Trial 26 finished with value: 1.6793893129770991 a
nd parameters: {'eta': 0.09271237323564291}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:02:59,174] Trial 27 finished with value: 1.6296296296296295 a
nd parameters: {'eta': 0.3759249528502209 3}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:02:59,789] Trial 28 finished with value: 1.6479400749063668 a
nd parameters: {'eta': 0.2656910039515204 5}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:03:00,384] Trial 29 finished with value: 1.5658362989323846 a
nd parameters: {'eta': 0.5602150874689004}. Best is trial 16 with value: 1.527777
7777777777.
[I 2020-09-19 16:03:00,947] Trial 30 finished with value: 2.135922330097087 and
parameters: {'eta': 0.707235298587656 2}. Best is trial 16 with value: 1.52777777
77777777.
[I 2020-09-19 16:03:01,523] Trial 31 finished with value: 1.7120622568093384 a
nd parameters: {'eta': 0.5491368764406567}. Best is trial 16 with value: 1.527777
7777777777.
[I 2020-09-19 16:03:02,060] Trial 32 finished with value: 1.746031746031746 and
parameters: {'eta': 0.827616686856238}. Best is trial 16 with value: 1.527777777
7777777.
[I 2020-09-19 16:03:02,650] Trial 33 finished with value: 2.0183486238532113 a
nd parameters: {'eta': 0.6897062207460923}. Best is trial 16 with value: 1.527777
7777777777.
[I 2020-09-19 16:03:03,235] Trial 34 finished with value: 1.660377358490566 and
parameters: {'eta': 0.4232882531655147 7}. Best is trial 16 with value: 1.5277777
777777777.
[I 2020-09-19 16:03:03,833] Trial 35 finished with value: 1.6176470588235294 a
nd parameters: {'eta': 0.5136483356897427}. Best is trial 16 with value: 1.527777
7777777777.
[I 2020-09-19 16:03:04,449] Trial 36 finished with value: 1.6176470588235294 a
nd parameters: {'eta': 0.2210820413025001 7}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:03:05,020] Trial 37 finished with value: 2.6506024096385543 a
nd parameters: {'eta': 0.0010761236465836044}. Best is trial 16 with value: 1.527
77777777777777.
[I 2020-09-19 16:03:05,567] Trial 38 finished with value: 1.8106995884773662 a
nd parameters: {'eta': 0.8724480936348354}. Best is trial 16 with value: 1.527777
7777777777.
[I 2020-09-19 16:03:06,178] Trial 39 finished with value: 1.6417910447761197 a
nd parameters: {'eta': 0.3157612926682 9}. Best is trial 16 with value: 1.52777777
77777777.
[I 2020-09-19 16:03:06,756] Trial 40 finished with value: 1.9047619047619047 a
nd parameters: {'eta': 0.7369197608341385}. Best is trial 16 with value: 1.527777

```
7777777777.
[I 2020-09-19 16:03:07,340] Trial 41 finished with value: 1.660377358490566 and
parameters: {'eta': 0.2889277863469429}. Best is trial 16 with value: 1.52777777
77777777.
[I 2020-09-19 16:03:07,925] Trial 42 finished with value: 1.6417910447761197 a
nd parameters: {'eta': 0.43461237784869705}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:03:08,551] Trial 43 finished with value: 1.7054263565891474 a
nd parameters: {'eta': 0.2159636569318746}. Best is trial 16 with value: 1.527777
7777777777.
[I 2020-09-19 16:03:09,160] Trial 44 finished with value: 1.6236162361623614 a
nd parameters: {'eta': 0.1299263810994002}. Best is trial 16 with value: 1.527777
7777777777.
[I 2020-09-19 16:03:09,805] Trial 45 finished with value: 1.6296296296296295 a
nd parameters: {'eta': 0.17195603079502844}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:03:10,428] Trial 46 finished with value: 1.7054263565891472 a
nd parameters: {'eta': 0.5712622217080006}. Best is trial 16 with value: 1.527777
7777777777.
[I 2020-09-19 16:03:11,026] Trial 47 finished with value: 1.7670682730923697 a
nd parameters: {'eta': 0.38034098249027837}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:03:11,624] Trial 48 finished with value: 1.7120622568093384 a
nd parameters: {'eta': 0.46955100027677277}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:03:12,225] Trial 49 finished with value: 1.685823754789272 and
parameters: {'eta': 0.24803144298245067}. Best is trial 16 with value: 1.5277777
777777777.
[I 2020-09-19 16:03:12,776] Trial 50 finished with value: 1.6541353383458648 a
nd parameters: {'eta': 0.9885058054950668}. Best is trial 16 with value: 1.527777
7777777777.
[I 2020-09-19 16:03:13,370] Trial 51 finished with value: 1.673003802281369 and
parameters: {'eta': 0.11746035302069932}. Best is trial 16 with value: 1.5277777
777777777.
[I 2020-09-19 16:03:13,958] Trial 52 finished with value: 1.8181818181818181 a
nd parameters: {'eta': 0.0656726054689672}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:03:14,566] Trial 53 finished with value: 1.6858237547892718 a
nd parameters: {'eta': 0.12512970356336844}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:03:15,165] Trial 54 finished with value: 1.746031746031746 and
parameters: {'eta': 0.3092636769484737}. Best is trial 16 with value: 1.52777777
77777777.
[I 2020-09-19 16:03:15,720] Trial 55 finished with value: 2.0276497695852536 a
nd parameters: {'eta': 0.0251338381865035}. Best is trial 16 with value: 1.527777
7777777777.
[I 2020-09-19 16:03:16,354] Trial 56 finished with value: 1.7391304347826089 a
nd parameters: {'eta': 0.15342516890309912}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:03:16,995] Trial 57 finished with value: 1.5827338129496402 a
nd parameters: {'eta': 0.2072506811734776}. Best is trial 16 with value: 1.52777
77777777777.
[I 2020-09-19 16:03:17,593] Trial 58 finished with value: 1.588447653429603 and
parameters: {'eta': 0.19456891776525698}. Best is trial 16 with value: 1.5277777
777777777.
[I 2020-09-19 16:03:18,184] Trial 59 finished with value: 1.522491349480969 and
parameters: {'eta': 0.19876448027921312}. Best is trial 59 with value: 1.5224913
49480969.
[I 2020-09-19 16:03:18,808] Trial 60 finished with value: 1.71875 and parameters:
{'eta': 0.21584168485267735}. Best is trial 59 with value: 1.522491349480969.
[I 2020-09-19 16:03:19,417] Trial 61 finished with value: 1.5017064846416384 a
```

nd parameters: {'eta': 0.19882553849367446}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:20,019] Trial 62 finished with value: 1.6923076923076925 a
nd parameters: {'eta': 0.24867867703339938}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:20,682] Trial 63 finished with value: 1.7054263565891474 a
nd parameters: {'eta': 0.18590623819553004}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:21,314] Trial 64 finished with value: 1.752988047808765 and
parameters: {'eta': 0.09081318076244645}. Best is trial 61 with value: 1.5017064
846416384.
[I 2020-09-19 16:03:21,926] Trial 65 finished with value: 1.6923076923076925 a
nd parameters: {'eta': 0.18849041048317017}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:22,526] Trial 66 finished with value: 1.6988416988416988 a
nd parameters: {'eta': 0.364433758825625}. Best is trial 61 with value: 1.5017064
846416384.
[I 2020-09-19 16:03:23,080] Trial 67 finished with value: 1.8410041841004183 a
nd parameters: {'eta': 0.6554321193512281}. Best is trial 61 with value: 1.501706
4846416384.
[I 2020-09-19 16:03:23,660] Trial 68 finished with value: 1.71875 and parameters:
{'eta': 0.3350255027725738}. Best is trial 61 with value: 1.5017064846416384.
[I 2020-09-19 16:03:24,261] Trial 69 finished with value: 1.6858237547892718 a
nd parameters: {'eta': 0.24849195802485483}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:24,864] Trial 70 finished with value: 1.774193548387097 and
parameters: {'eta': 0.05477315212261136}. Best is trial 61 with value: 1.5017064
846416384.
[I 2020-09-19 16:03:25,489] Trial 71 finished with value: 1.5770609318996418 a
nd parameters: {'eta': 0.20577492349761892}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:26,108] Trial 72 finished with value: 1.7322834645669294 a
nd parameters: {'eta': 0.14900454011220698}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:26,728] Trial 73 finished with value: 1.660377358490566 and
parameters: {'eta': 0.2890306903234272}. Best is trial 61 with value: 1.50170648
46416384.
[I 2020-09-19 16:03:27,329] Trial 74 finished with value: 1.752988047808765 and
parameters: {'eta': 0.10931339277655552}. Best is trial 61 with value: 1.5017064
846416384.
[I 2020-09-19 16:03:27,926] Trial 75 finished with value: 1.5017064846416384 a
nd parameters: {'eta': 0.1989257328822957}. Best is trial 61 with value: 1.501706
4846416384.
[I 2020-09-19 16:03:28,560] Trial 76 finished with value: 1.71875 and parameters:
{'eta': 0.15320893357652007}. Best is trial 61 with value: 1.5017064846416384.
[I 2020-09-19 16:03:29,167] Trial 77 finished with value: 1.5120274914089347 a
nd parameters: {'eta': 0.19632357029568903}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:29,770] Trial 78 finished with value: 1.7054263565891472 a
nd parameters: {'eta': 0.2602784049427478 4}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:30,308] Trial 79 finished with value: 1.7959183673469385 a
nd parameters: {'eta': 0.892290337319714 7}. Best is trial 61 with value: 1.501706
4846416384.
[I 2020-09-19 16:03:30,868] Trial 80 finished with value: 1.7670682730923695 a
nd parameters: {'eta': 0.034745634348070986}. Best is trial 61 with value: 1.5017
064846416384.
[I 2020-09-19 16:03:31,512] Trial 81 finished with value: 1.605839416058394 and
parameters: {'eta': 0.2097998061167708}. Best is trial 61 with value: 1.50170648
46416384.
[I 2020-09-19 16:03:32,109] Trial 82 finished with value: 1.6356877323420076 a

```
nd parameters: {'eta': 0.17492299388009847}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:32,721] Trial 83 finished with value: 1.6666666666666667 a
nd parameters: {'eta': 0.2799808690210507}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:33,322] Trial 84 finished with value: 1.6923076923076925 a
nd parameters: {'eta': 0.09620875676736138}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:33,914] Trial 85 finished with value: 1.7054263565891472 a
nd parameters: {'eta': 0.2416812187670549}. Best is trial 61 with value: 1.501706
4846416384.
[I 2020-09-19 16:03:34,523] Trial 86 finished with value: 1.7599999999999998 a
nd parameters: {'eta': 0.3081232019762159}. Best is trial 61 with value: 1.501706
4846416384.
[I 2020-09-19 16:03:35,113] Trial 87 finished with value: 1.533101045296167 and
parameters: {'eta': 0.2012762837254132}. Best is trial 61 with value: 1.50170648
46416384.
[I 2020-09-19 16:03:35,725] Trial 88 finished with value: 1.6923076923076925 a
nd parameters: {'eta': 0.15819345837402388}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:36,317] Trial 89 finished with value: 1.6 and parameters: {'eta':
0.23016211851086163}. Best is trial 61 with value: 1.5017064846416384.
[I 2020-09-19 16:03:36,879] Trial 90 finished with value: 1.6541353383458648 a
nd parameters: {'eta': 0.7745205564938219}. Best is trial 61 with value: 1.501706
4846416384.
[I 2020-09-19 16:03:37,494] Trial 91 finished with value: 1.5827338129496404 a
nd parameters: {'eta': 0.1944985403844865}. Best is trial 61 with value: 1.501706
4846416384.
[I 2020-09-19 16:03:38,099] Trial 92 finished with value: 1.6793893129770991 a
nd parameters: {'eta': 0.13630814951821735}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:38,746] Trial 93 finished with value: 1.7529880478087654 a
nd parameters: {'eta': 0.2124868304252281}. Best is trial 61 with value: 1.501706
4846416384.
[I 2020-09-19 16:03:39,355] Trial 94 finished with value: 1.6541353383458648 a
nd parameters: {'eta': 0.2736918320856467 4}. Best is trial 61 with value: 1.50170
64846416384.
[I 2020-09-19 16:03:39,965] Trial 95 finished with value: 1.4965986394557822 a
nd parameters: {'eta': 0.1990049866560162}. Best is trial 95 with value: 1.496598
6394557822.
[I 2020-09-19 16:03:40,570] Trial 96 finished with value: 1.588447653429603 and
parameters: {'eta': 0.23373469134394087}. Best is trial 95 with value: 1.4965986
394557822.
[I 2020-09-19 16:03:41,155] Trial 97 finished with value: 1.6479400749063673 a
nd parameters: {'eta': 0.16786274366854875}. Best is trial 95 with value: 1.49659
86394557822.
[I 2020-09-19 16:03:41,821] Trial 98 finished with value: 1.6988416988416988 a
nd parameters: {'eta': 0.11419051962570578}. Best is trial 95 with value: 1.49659
86394557822.
[I 2020-09-19 16:03:42,441] Trial 99 finished with value: 1.611721611721612 and
parameters: {'eta': 0.135635053437839}. Best is trial 95 with value: 1.496598639
4557822.

hyperparameter: {'eta': 0.1990049866560162}
AUC: 0.6681818181818182
```

In [ ]:

```
# This eta value is one example of hyperparameter optimization. There are other values that yield a si
milar AUC value.
xgboost(0.1990049866560162)
```

■ Leave-One-Out Cross Validation with XGBoost

Sample size of all dataset: 51
Sample size of training data: 50 Explanatory variables: (50, 211) Target value: (5
0,)
Sample size of test data: 1 Explanatory variables: (1, 211) Target value: (1,)

AUC: 0.66818 Accuracy: 0.78431 R2: 8e-05

	Actual True	Actual False
Predict True	1	1
Predict False	10	39

Output: 1. AUC 2. Accuracy 3. Precision 4. Recall 5. f1-score 6. Specificity 7. Confutio
n matrix 8. R^2 in LOO

Out[ ]:

```
array([0.6681818181818182, 0.7843137254901961, 0.5, 0.090909090909090
91,
       0.15384615384615385, 0.975, list([[1, 1], [10, 39]]),
       8.417951884320818e-05], dtype=object)
```

8) Symbolic Regression ; AUC 0.784

See Leave-One-Out CV with renaldata by SR via GP.pdf file for the calculation processes of LOO with Symbolic Regression

- Results of Leave-One-Out CV with SR via GP
- AUC 0.784
- Accuracy 82.35%
- Precision 60.0%
- Recall 54.55%
- Sepcificity 90.0%
- F1-measure 57.14
- C-statistics 0.784

In [ ]:

	Actual True	Actual False
Predicted True	6	4
Predicted False	5	36