MAS-12 lecture code

Chapter 12 Classification

In [960…

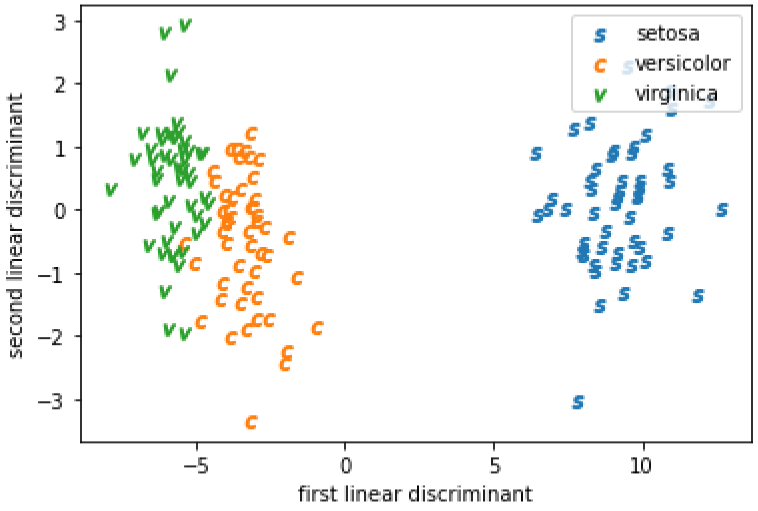
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.datasets import load_iris

#print(iris)
#print(iris.DESCR,'\n')
#print(iris.data,'\n')
#print(iris.feature_names,'\n')
#print(iris.target,'\n')
#print(iris.target_names,'\n')

#iris dataset.
iris = load_iris()
ir = pd.DataFrame(data=iris.data, columns=iris.feature_names)
ir_species = pd.Series(iris.target)
ir_species = ir_species.map({0:"s", 1:"c", 2:"v"})
#print(ir.head(),'\n\n',ir_species.head())

## Crabs dataset.
crabs = pd.read_csv("./input/crabs.csv")
lcrabs = np.log(crabs.iloc[:,3:])
lcrabs_grp = list(np.repeat(["B","b","O","o"], [50]*4, axis = 0))
lcrabs_target = np.repeat([0,1,2,3], [50]*4, axis = 0)
lcrabs_target_name = np.array(["B","b","O","o"])
#print(crabs.head(),"\n\n", lcrabs_target,"\n\n",lcrabs_grp)

## fgl dataset.
fgl=pd.read_csv("./input/fgl.csv")
fgl_target=fgl.type
fgl_targetnames=['WinNF', 'WinF' ,'Head' ,'Veh' ,'Con', 'Tabl']
#print(fgl_target.value_counts())

##Cushings dataset.
Cushings=pd.read_csv("./input/Cushings.csv")
Cushings_target=Cushings.Type

#print(Cushings_target)
```

12.1 Discriminant Analysis

In [961…

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.preprocessing import StandardScaler

#iris dataset을 LDA 모델에 적합.
std_ir = StandardScaler().fit_transform(np.log(ir))
ir_lda=LDA()
ir_ld=ir_lda.fit_transform(std_ir,ir_species)
ir_ld[:,0]=-ir_ld[:,0]
```

```
#print(ir_ld)

#그래프를 그리기 위해 LDA 결과값을 dataframe으로 변환.
lda_columns=['LD1','LD2']
ir_ld = pd.DataFrame(ir_ld, columns=lda_columns)
#ir_ld['target']=iris.target
#print(ir_ld.iloc[0,:])

#각 범주의 알파벳을 그래프에 나타내기 위해 marker 지정.
markers=['$s$', '$c$', '$v$']

#그래프 그리기.
for i, marker in enumerate(markers):
    x_axis_data = ir_ld.loc[iris.target==i,:]['LD1']
    y_axis_data = ir_ld.loc[iris.target==i,:]['LD2']

    plt.scatter(x_axis_data, y_axis_data, marker=marker,label=iris.target_na

plt.legend(loc='upper right')
plt.xlabel('first linear discriminant')
plt.ylabel('second linear discriminant')
plt.show()
```



In [962…

```
#print(lcrabs)
std_cr = StandardScaler().fit_transform(lcrabs.drop(["BD"],axis=1))
dcrabs_lda = LDA().fit(std_cr,crabs.sex)
a=pd.DataFrame({'sex':crabs.sex,'predict_sex':dcrabs_lda.predict(std_cr)})
print(pd.crosstab(index=a.sex,columns=a.predict_sex))

#print(lcrabs.drop(["BD"],axis=1))
dcrabs_lda4 = LDA(n_components=3).fit(std_cr,lcrabs_grp)
dcrabs_pr4=pd.DataFrame(dcrabs_lda4.predict_proba(std_cr),columns=["B","O","
#print(pd.Series(dcrabs_lda4.predict(std_cr)).value_counts())

#R에서 predict 함수의 dimen=2를 구현 못해서 다른 값이 나옴. 나중에 확인.
dcrabs_pr2=dcrabs_pr4[["B","O"]] @ [1,1]
#print(dcrabs_pr4)
print(pd.crosstab(index=a.sex,columns=[dcrabs_pr2>0.5]))
```

```
predict_sex   F   M
sex
F            97   3
M             3  97
col_0  False  True
sex
F         96     4
M          4    96
```

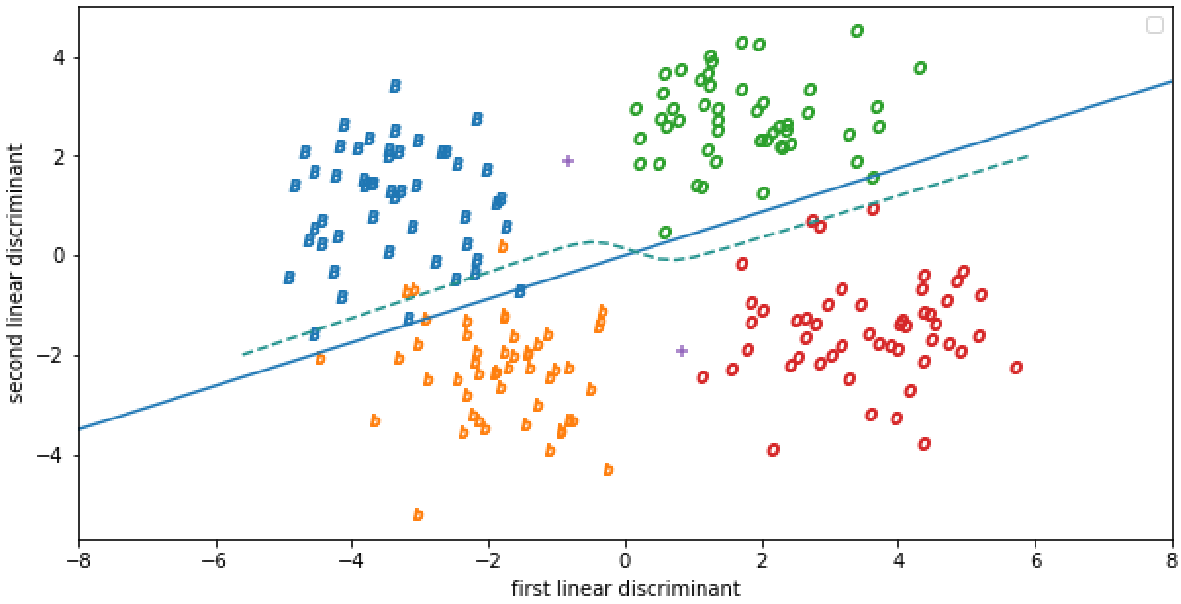
In [963…

```
cr_t = dcrabs_lda4.transform(std_cr)[:,[0,1]]
#cr_t[:,1]=-cr_t[:,1]
#print(ir_ld)

#그래프를 그리기 위해 LDA 결과값을 dataframe으로 변환.
lda_columns=['LD1','LD2']
cr_t = pd.DataFrame(cr_t, columns=lda_columns)

#각 범주의 알파벳을 그래프에 나타내기 위해 marker 지정.
markers=['$B$','$b$', '$O$', '$o$']

#그래프 그리기.
fig=plt.figure(figsize=(10,5))
p1=fig.add_subplot(111)

for i, marker in enumerate(markers):
    x_axis_data = cr_t.loc[lcrabs_target==i,:]['LD1']
    y_axis_data = cr_t.loc[lcrabs_target==i,:]['LD2']

    p1.scatter(x_axis_data, y_axis_data, marker=marker)

p1.legend(loc='upper right')
p1.set_xlim(-8,8)
p1.set_xlabel('first linear discriminant')
p1.set_ylabel('second linear discriminant')

def perp(x,y):
    u=np.array([-8,8])
    m = (x+y)/2
    s = - (x[0] - y[0])/(x[1] - y[1])
    p1.plot(u,s*u+m)

cr_m = LDA().fit(cr_t,crabs.sex).means_
#print(dcrabs_lda)
p1.scatter(cr_m[:,0],cr_m[:,1],marker="+")
perp(cr_m[0,:],cr_m[1,:])

cr_lda=LDA().fit(cr_t,lcrabs_grp)
x=np.arange(-6, 6.25, 0.25)
y=np.arange(-2, 2.25, 0.25)

x1= np.tile(x,len(y))
x2=np.repeat(y,len(x))

#print(x2)

Xcon = pd.DataFrame({'x1':x1,'x2':x2})
#print(Xcon)

#cr_pr= cr_lda.predict_proba(Xcon)
cr_pr=pd.DataFrame(cr_lda.predict_proba(Xcon),columns=["B","O",'b',"o"])[["B
cr_pr=cr_pr.values.reshape((len(y),len(x)))
#cr_pr=cr_pr.values.reshape((len(y),len(x))).transpose()
#print(cr_pr)

p1.contour(x,y,cr_pr,levels=1,linestyles='dashed' )
```

```
No artists with labels found to put in legend.  Note that artists whose labe
l start with an underscore are ignored when legend() is called with no argum
ent.
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/base.py:49
3: FutureWarning: The feature names should match those that were passed duri
ng fit. Starting version 1.2, an error will be raised.
Feature names unseen at fit time:
- x1
- x2
Feature names seen at fit time, yet now missing:
- LD1
- LD2

  warnings.warn(message, FutureWarning)
```

Out[963]:

```
<matplotlib.contour.QuadContourSet at 0x2913f32e0>
```



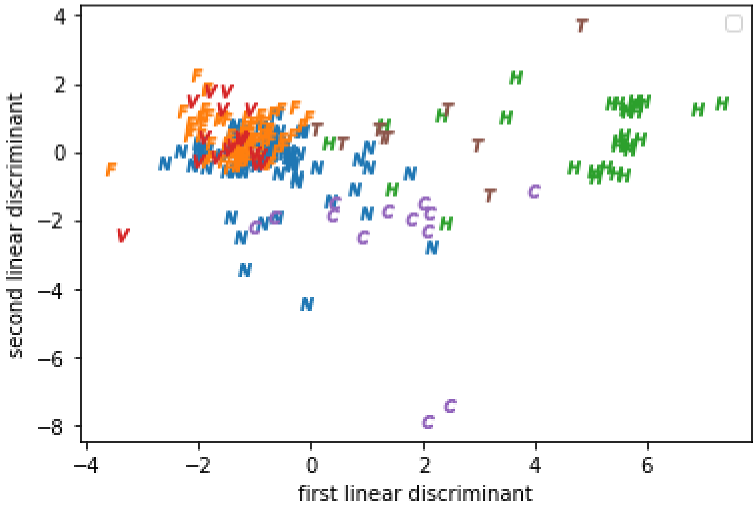
In [964…

```
for i in range(0,4):
    c=lcrabs.loc[lcrabs_target==i,:]
    name=["B","b","O","o"][i]
    print(f"Covariance matrix of the column {name}","\n",pd.DataFrame.cov(c)
Covariance matrix of the column B
          FL        RW        CL        CW        BD
FL  0.052964  0.043251  0.056633  0.056006  0.058748
RW  0.043251  0.037247  0.046800  0.046336  0.048304
CL  0.056633  0.046800  0.061045  0.060328  0.063118
CW  0.056006  0.046336  0.060328  0.059738  0.062495
BD  0.058748  0.048304  0.063118  0.062495  0.066021


Covariance matrix of the column b
          FL        RW        CL        CW        BD
FL  0.043578  0.043610  0.046080  0.045675  0.050083
RW  0.043610  0.045040  0.046544  0.046172  0.050729
CL  0.046080  0.046544  0.049147  0.048631  0.053537
CW  0.045675  0.046172  0.048631  0.048261  0.053015
BD  0.050083  0.050729  0.053537  0.053015  0.059463


Covariance matrix of the column O
          FL        RW        CL        CW        BD
FL  0.048712  0.040674  0.052366  0.052165  0.053486
RW  0.040674  0.035038  0.044230  0.044110  0.045120
CL  0.052366  0.044230  0.056893  0.056651  0.057999
CW  0.052165  0.044110  0.056651  0.056510  0.057791
BD  0.053486  0.045120  0.057999  0.057791  0.059476


Covariance matrix of the column o
          FL        RW        CL        CW        BD
FL  0.032173  0.029149  0.031774  0.031767  0.032387
RW  0.029149  0.028188  0.029426  0.029453  0.029828
CL  0.031774  0.029426  0.031953  0.031859  0.032594
CW  0.031767  0.029453  0.031859  0.031921  0.032481
BD  0.032387  0.029828  0.032594  0.032481  0.033844
```

In [965…

```python
std_fgl = StandardScaler().fit_transform(fgl.drop(["type"],axis=1))

# LDA 함수의 method="t"를 구현해서 한 번 더 그림. 나중에 확인.
fgl_ld=LDA().fit(std_fgl,fgl_target).transform(std_fgl)[:,[0,1]]
fgl_ld[:,0]=-fgl_ld[:,0]
fgl_ld=pd.DataFrame(data=fgl_ld,columns=lda_columns)
#print(fgl_ld[:10])

#각 범주의 알파벳을 그래프에 나타내기 위해 marker 지정.
markers=[ "$N$","$F$",  "$H$","$V$", "$C$", "$T$"]

#그래프 그리기.
for i, marker in enumerate(markers):
    name=fgl_targetnames[i]
    x_axis_data = fgl_ld.loc[fgl_target==name,:]['LD1']
    y_axis_data = fgl_ld.loc[fgl_target==name,:]['LD2']

    plt.scatter(x_axis_data, y_axis_data, marker=marker)

plt.legend(loc='upper right')
plt.xlabel('first linear discriminant')
plt.ylabel('second linear discriminant')
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose labe
l start with an underscore are ignored when legend() is called with no argum
ent.
```



12.2 Classification theory

In [966…

```
##

def predplot(object,fig,t):

    len=100
    p1=fig.add_subplot(220+t)


#    p1.scatter(Cushings.iloc[:,0], Cushings.iloc[:,1])

    #각 범주의 알파벳을 그래프에 나타내기 위해 marker 지정.
    markers=[(f"${al}$")  for al in Cushings.Type.unique()]

    #그래프 그리기.
    for i, marker in enumerate(markers):
        s = Cushings.Type==Cushings.Type.unique()[i]
        name=fgl_targetnames[i]
        x_axis_data = Cushings.loc[s, :].iloc[:,0]
        y_axis_data = Cushings.loc[s, :].iloc[:,1]

        plt.scatter(x_axis_data, y_axis_data, marker=marker)
        plt.title("LDA" if t ==1 else "QDA")


    xp = np.linspace(0.6, 4.0, len)
    yp = np.linspace(-3.25, 2.45, len)
    cushT =pd.DataFrame([(x, y)  for y in yp for x in xp],
                    columns=['Tetrahydrocortisone','Pregnanetriol'])
#    p1.set_ylim(min(yp),max(yp))
#    p1.set_xlim(min(xp),max(xp))
#    p1.set_ylim(-0.5,3.0)
#    p1.set_xlim(0,50)

    Z1 = object.predict(cushT)
    Z2 = object.predict_proba(cushT)
#    print(Z2[:,2])
    zp = Z2[:,2] - np.maximum(Z2[:,1], Z2[:,0])
    zp=zp.reshape((len,-1))
    p1.contour(np.exp(xp),
               np.exp(yp), zp,
               levels = 0)

    zp=Z2[:,0] - np.maximum(Z2[:,1], Z2[:,2])
    zp=zp.reshape((len,-1))
    p1.contour(np.exp(xp),
               np.exp(yp), zp,
               levels = 0)
    p1.set_xlabel("Tetrahydrocortisone")
    p1.set_ylabel("Pregnanetriol")
    p1.semilogy(base=10)
    p1.semilogx(base=10)
```

In [967…

```
def cushplot(xp, yp, Z,name):


    fig=plt.figure(figsize=(15,10))
    p1=fig.add_subplot(111)

    #각 범주의 알파벳을 그래프에 나타내기 위해 marker 지정.
    markers=[(f"${al}$")  for al in Cushings.Type.unique()]

    #그래프 그리기.
    for i, marker in enumerate(markers):
        s = Cushings.Type==Cushings.Type.unique()[i]
        name=fgl_targetnames[i]
        x_axis_data = Cushings.loc[s, :].iloc[:,0]
        y_axis_data = Cushings.loc[s, :].iloc[:,1]

        plt.scatter(x_axis_data, y_axis_data, marker=marker)

    zp = Z[:,2] - np.maximum(Z[:,1], Z[:,0])
    zp=zp.reshape((n_p,-1))
    p1.contour(np.exp(xp), np.exp(yp), zp, levels = 0)
#     print(Z[:,2])

    zp = Z[:,0] - np.maximum(Z[:,1], Z[:,2])
    zp=zp.reshape((n_p,-1))
    p1.contour(np.exp(xp), np.exp(yp), zp, levels = 0)

    p1.set_xlabel("Tetrahydrocortisone")
    p1.set_ylabel("Pregnanetriol")
    p1.semilogy(base=10)
    p1.semilogx(base=10)
    p1.set_title(name)
```

In [968…

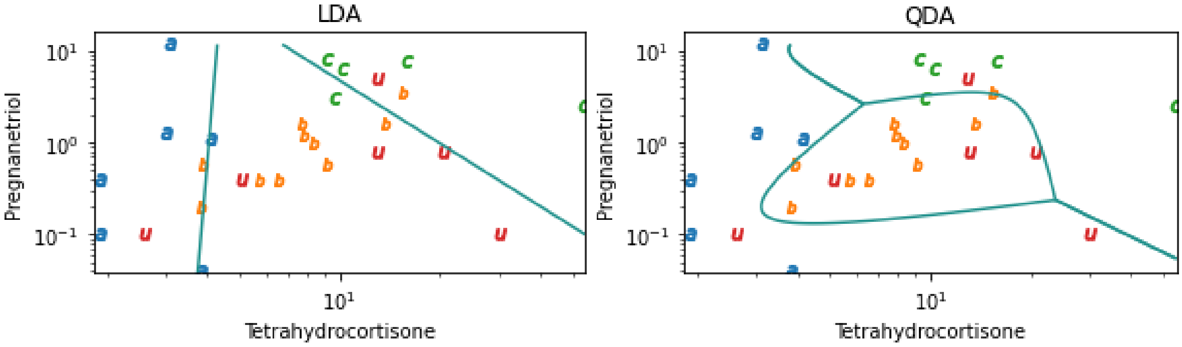
```
##

cush=np.log(Cushings.drop(Cushings.columns[[2]],axis=1))
tp = Cushings.Type[0:21]
#print(tp)
```

In [969…

```
##
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as Q

cush_lda = LDA().fit(cush[0:21],tp)
cush_qda = QDA().fit(cush[0:21],tp)
```

In [970…

```
fig=plt.figure(figsize=(10,5))

predplot(cush_lda,fig, 1)
predplot(cush_qda,fig, 2)

#R의 predict 함수의 method 부분을 구현 못 함. 이 부분 다시 확인.
#predplot(cush_qda,3, "QDA (predictive)", method = "predictive")
#predplot(cush_qda,4, "QDA (debiased)", method = "debiased")
```



In [971…

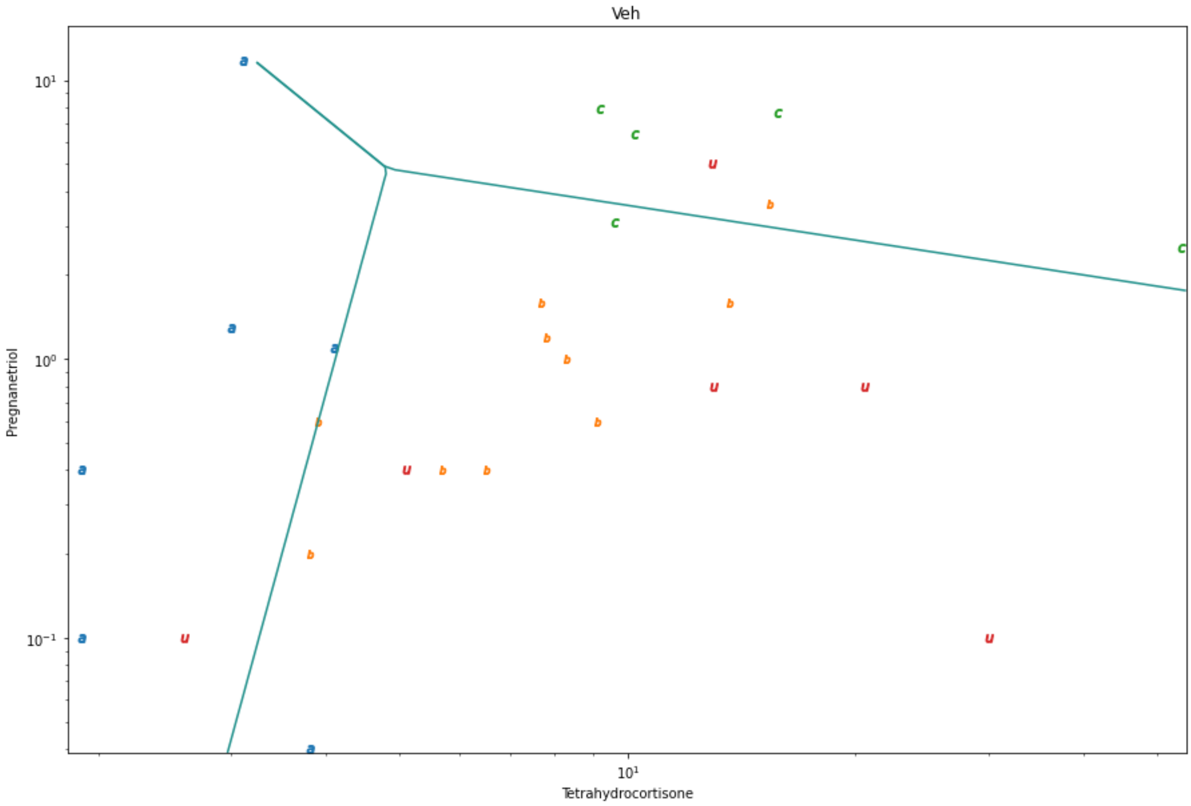
```
#
from sklearn.linear_model import LogisticRegression

#par(mfrow = c(1,2))

Cf = pd.DataFrame({'tp' : tp,
                   'Tetrahydrocortisone' : np.log(Cushings[0:21].iloc[:,0]),
                   'Pregnanetriol' : np.log(Cushings[0:21].iloc[:,1])} )
#print(Cf.iloc[:,1:3][0:10])

#cush_multinom=LogisticRegression(multi_class='multinomial', solver='lbfgs')

cush_multinom=LogisticRegression(multi_class='multinomial', solver='lbfgs').
#cush_multinom.predict_proba(cushT)
```

In [972…

```
##

xp = np.linspace(0.6, 4.0, 100)
n_p = len(xp)
yp = np.linspace(-3.25, 2.45, 100)

cushT =pd.DataFrame([(x, y)  for y in yp for x in xp],
                    columns=['Tetrahydrocortisone','Pregnanetriol'])
```

In [973…

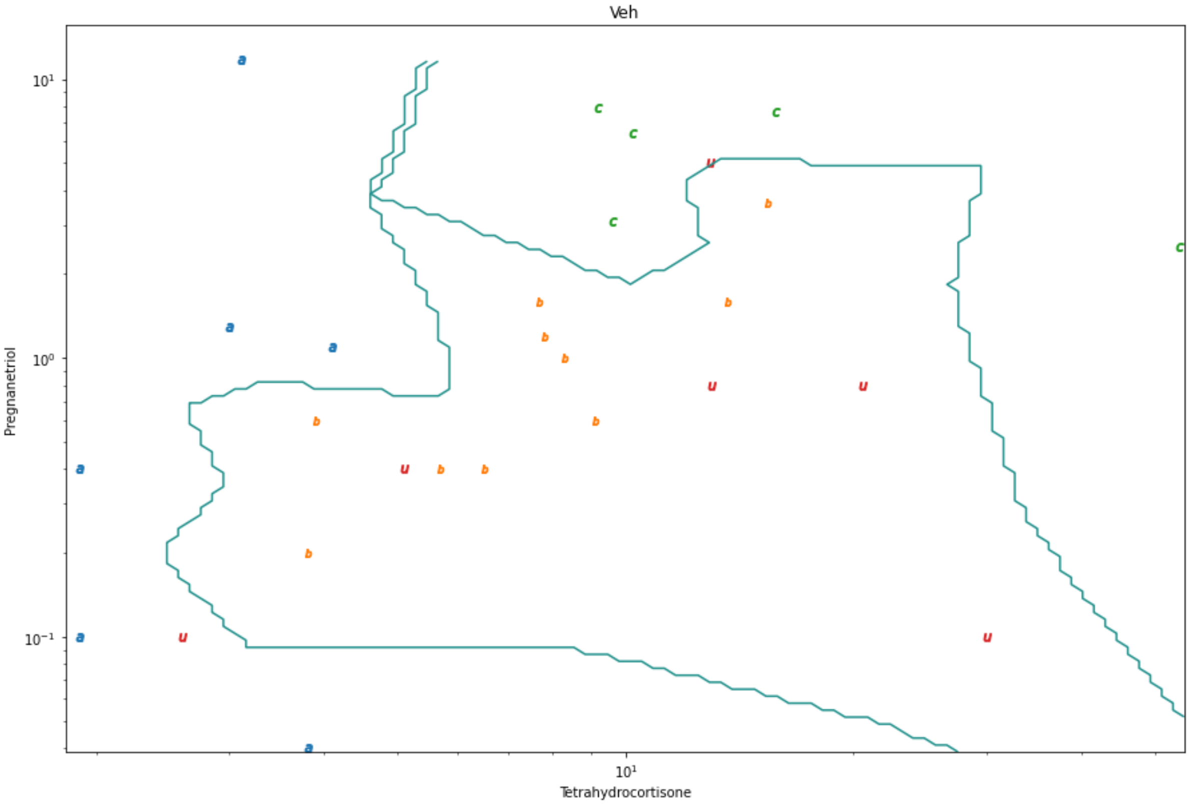
```
#

Z = cush_multinom.predict_proba(cushT)
#print(type(Z))
cushplot(xp, yp, Z, "Logistic Regression")
```



12.3 Non-parametric rules

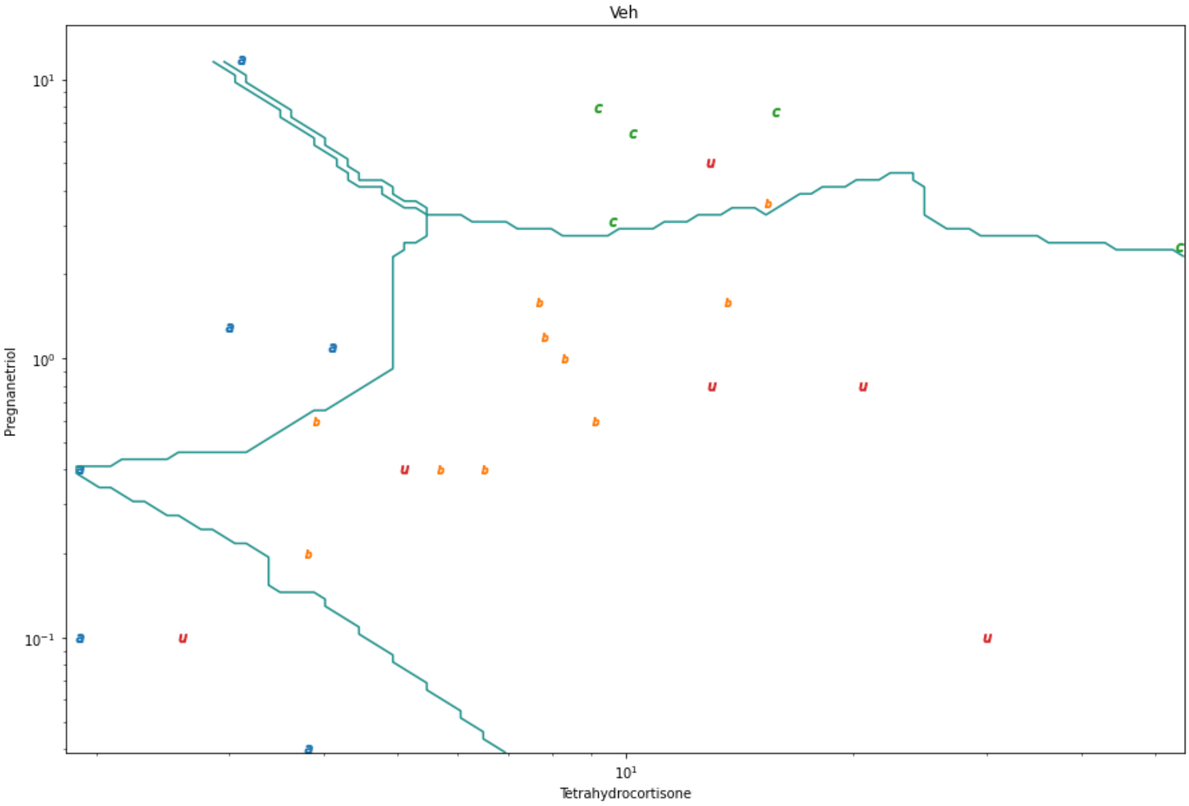
In [974…

```
#

from sklearn.neighbors import KNeighborsClassifier
from sklearn import preprocessing

#center=False, scale=True

def not_center_scale(x):
    t =x.iloc[:,0]/3.4
    p = x.iloc[:,1]/5.7
    std_cu=pd.concat([t,p],axis=1)
    return std_cu
#print(cush)

cu1=not_center_scale(cush[0:21])
cu2=not_center_scale(cushT)

knn=KNeighborsClassifier(n_neighbors=1)
knn.fit(cu1,tp)
Z=knn.predict(cu2)

#print(np.unique(Z,return_counts=True))
class_ind_Z=np.array(pd.get_dummies(Z))
cushplot(xp, yp, class_ind_Z, "1-NN")
#print(pd.get_dummies(Z))
```



In [975…

```
#

knn=KNeighborsClassifier(n_neighbors = 3)
knn.fit(cu1,tp)
Z=knn.predict(cu2)
#print(np.unique(Z,return_counts=True))
class_ind_Z=np.array(pd.get_dummies(Z))
cushplot(xp, yp, class_ind_Z,"3-NN")
```



12.4 Neural networks

In [976…

```
def pltnn(main,t, *args):

    #fig=plt.figure(figsize=(15,10))
    #p1=fig.add_subplot(220+t)

    markers=[(f"${al}$")  for al in Cushings.Type.unique()]

    #그래프 그리기.
    for i, marker in enumerate(markers):
        s = Cushings.Type==Cushings.Type.unique()[i]
        name=fgl_targetnames[i]
        x_axis_data = Cushings.loc[s, :].iloc[:,0]
        y_axis_data = Cushings.loc[s, :].iloc[:,1]

        plt.scatter(x_axis_data, y_axis_data, marker=marker)

    p1.set_xlabel("Tetrahydrocortisone")
    p1.set_ylabel("Pregnanetriol")
    p1.semilogy(base=10)
    p1.semilogx(base=10)
    p1.set_title(main)
```

In [977…

```
from sklearn.neural_network import MLPClassifier

def plt_bndry(col,learning_rate):

    cush_nn = MLPClassifier(hidden_layer_sizes=(27),
                            activation='logistic',
                            solver='sgd',
                            max_iter=1000,learning_rate_init=learning_rate)
    cush_nn.fit(cush, tp)
#     print(cush_nn.predict_proba(cushT))
    b1(cush_nn.predict_proba(cushT),col)
```

In [978…

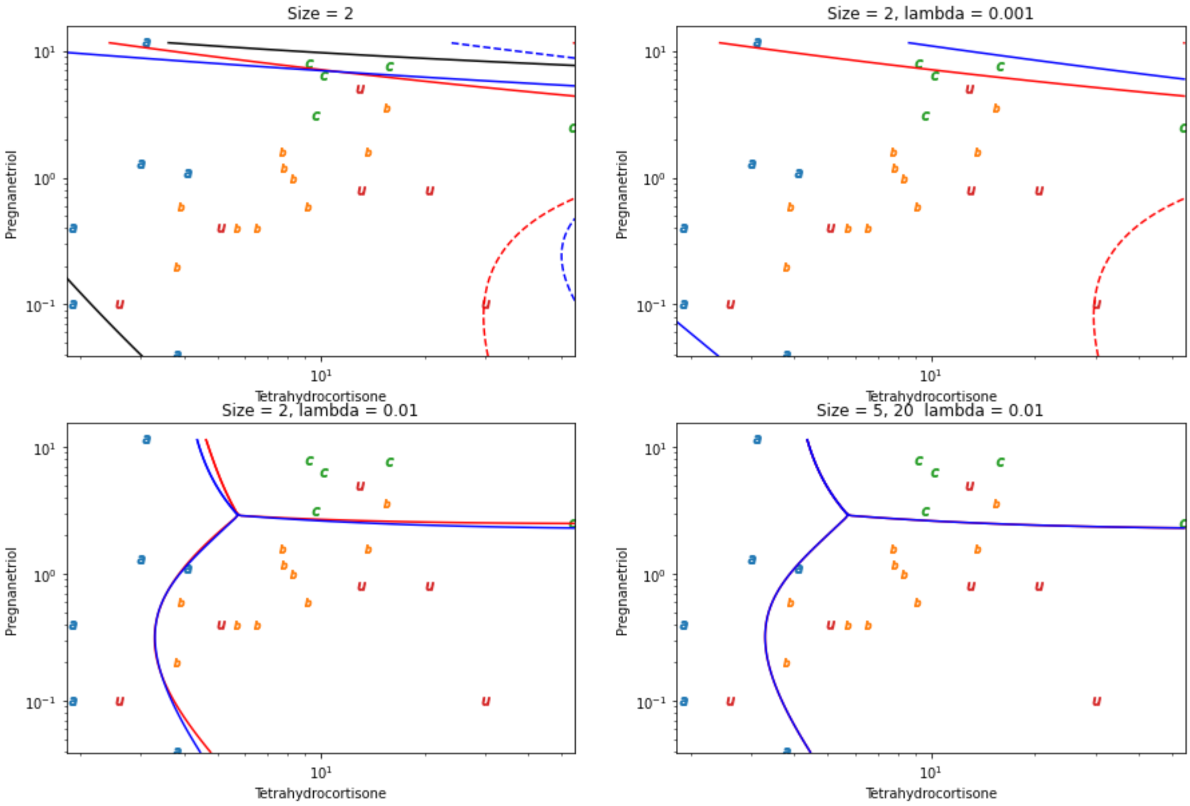
```
def b1(Z,col):

    zp = Z[:,2] - np.maximum(Z[:,1], Z[:,0])
    zp=zp.reshape((n_p,-1))
    p1.contour(np.exp(xp), np.exp(yp), zp,levels=0,colors=col)

    zp = Z[:,0] - np.maximum(Z[:,1], Z[:,2])
    zp=zp.reshape((n_p,-1))
    p1.contour(np.exp(xp), np.exp(yp), zp,levels=0,colors=col)
```

In [979…

```
# 여기 그림이 다름 재확인 필요!!!!!

cush = cush[0:21]
#print(cush)

tpi = pd.get_dummies(tp)
#print(tpi)


# functions pltnn and plt_bndry given in the scripts
fig=plt.figure(figsize=(15,10))
p1=fig.add_subplot(221)

pltnn("Size = 2",1)
np.random.seed(1); plt_bndry('red',0.001)
np.random.seed(3); plt_bndry('b',0.001)
```

```
plt_bndry('k',0.001)

p1=fig.add_subplot(222)
pltnn("Size = 2, lambda = 0.001",2)
np.random.seed(1); plt_bndry( col='red', learning_rate = 0.001)
np.random.seed(2); plt_bndry( col='b', learning_rate = 0.001)

p1=fig.add_subplot(223)
pltnn("Size = 2, lambda = 0.01",3)
np.random.seed(1); plt_bndry( col='red', learning_rate = 0.01)
np.random.seed(2); plt_bndry( col='b', learning_rate = 0.01)

p1=fig.add_subplot(224)
pltnn("Size = 5, 20  lambda = 0.01",4)
np.random.seed(2); plt_bndry(col='red',learning_rate = 0.01)
np.random.seed(2); plt_bndry(col='b', learning_rate = 0.01)
```

```
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
```

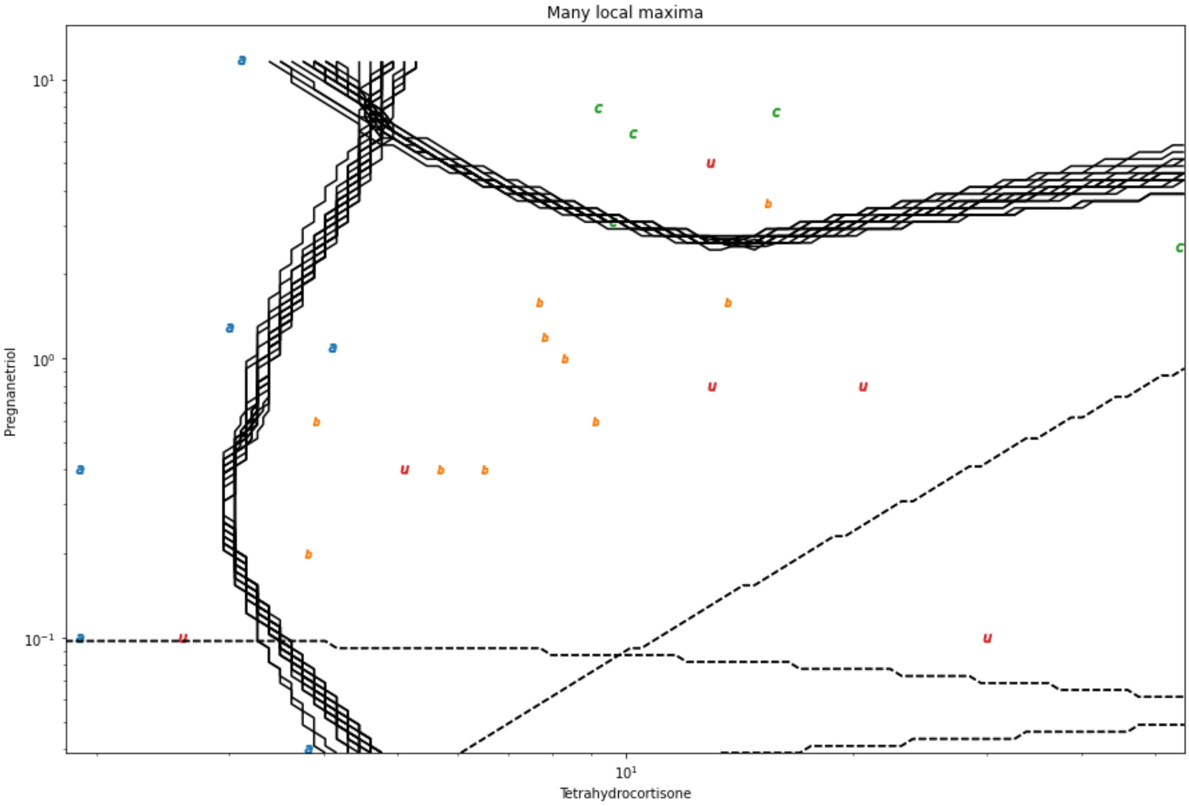


In [980…

```
fig=plt.figure(figsize=(15,10))
p1=fig.add_subplot(111)
# functions pltnn and b1 are in the scripts
pltnn("Many local maxima",t=1)


Z = np.zeros( (cushT.shape[0], tpi.shape[1]))


for iter in range(0,20) :
    np.random.seed(iter)
#     cush_nn = nnet(cush, tpi, skip = TRUE, softmax = TRUE, size = 3,
#        decay = 0.01, maxit = 1000, trace = FALSE)
    cush_nn= MLPClassifier(hidden_layer_sizes=(27),
                             activation='logistic',
                             solver='sgd',
                             max_iter=1000,learning_rate_init=0.01)
    Z = Z + cush_nn.fit(cush,tpi).predict_proba( cushT)
#     print("final value", f"{cush_nn.coefs_}", "\n")
    b1(cush_nn.predict(cushT), col = 'k')
```

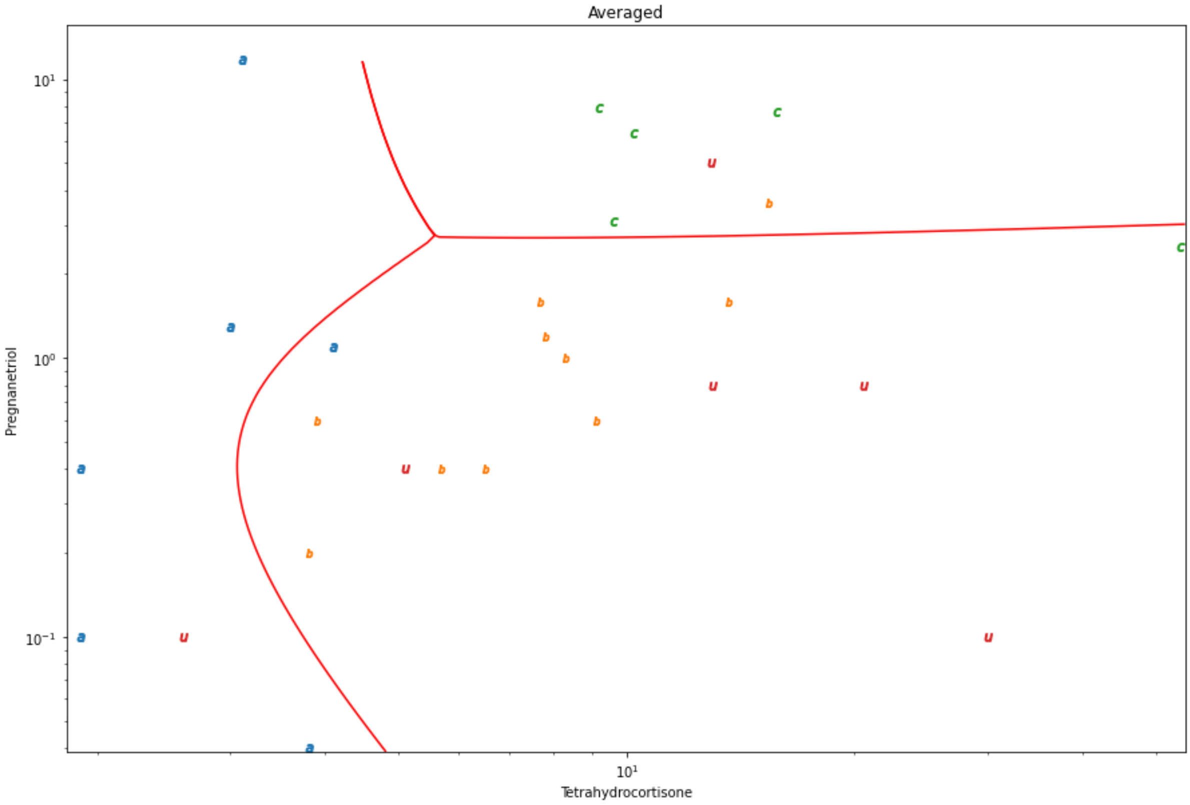
```
/var/folders/sc/618cfvvs4dq7jx43m7fskwjw0000gn/T/ipykernel_81137/3109009077.
py:5: UserWarning: No contour levels were found within the data range.
  p1.contour(np.exp(xp), np.exp(yp), zp,levels=0,colors=col)
/var/folders/sc/618cfvvs4dq7jx43m7fskwjw0000gn/T/ipykernel_81137/3109009077.
py:9: UserWarning: No contour levels were found within the data range.
  p1.contour(np.exp(xp), np.exp(yp), zp,levels=0,colors=col)
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/var/folders/sc/618cfvvs4dq7jx43m7fskwjw0000gn/T/ipykernel_81137/3109009077.
py:5: UserWarning: No contour levels were found within the data range.
  p1.contour(np.exp(xp), np.exp(yp), zp,levels=0,colors=col)
/var/folders/sc/618cfvvs4dq7jx43m7fskwjw0000gn/T/ipykernel_81137/3109009077.
py:9: UserWarning: No contour levels were found within the data range.
  p1.contour(np.exp(xp), np.exp(yp), zp,levels=0,colors=col)
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/var/folders/sc/618cfvvs4dq7jx43m7fskwjw0000gn/T/ipykernel_81137/3109009077.
py:5: UserWarning: No contour levels were found within the data range.
  p1.contour(np.exp(xp), np.exp(yp), zp,levels=0,colors=col)
/var/folders/sc/618cfvvs4dq7jx43m7fskwjw0000gn/T/ipykernel_81137/3109009077.
py:9: UserWarning: No contour levels were found within the data range.
  p1.contour(np.exp(xp), np.exp(yp), zp,levels=0,colors=col)
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
```

```
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (1000) reached and the optimization hasn't converged y
et.
  warnings.warn(
```



In [981…

```
fig=plt.figure(figsize=(15,10))
p1=fig.add_subplot(111)
pltnn("Averaged",1)
b1(Z, col='red')
```



12.5 Support vector machines

In [982…

```
from sklearn import svm
crabs_svm = svm.SVC(kernel='linear', C=100)
crabs_svm.fit(lcrabs,crabs.sp)

print(pd.crosstab(index=crabs.sp,
                  columns=crabs_svm.predict(lcrabs)))
```

```
col_0     B    O
sp
B       100    0
O         0  100
```

12.6 Forensic glass example

In [983…

```
##

np.random.seed(123)
# dump random partition from S-PLUS
rand = [9, 6, 7, 10, 8, 8, 2, 2, 10, 1, 5, 2, 3, 8, 6, 8, 2, 6, 4,
4, 6, 1, 3, 2, 5, 5, 5, 3, 1, 9, 10, 2, 8, 2, 1, 6, 2, 7, 7, 8, 4, 1,
9, 5, 5, 1, 4, 6, 8, 6, 5, 7, 9, 2, 1, 1, 10, 9, 7, 6, 4, 7, 4, 8, 9,
9, 1, 8, 9, 5, 3, 3, 4, 8, 8, 6, 6, 9, 3, 10, 3, 10, 6, 6, 5, 10, 10,
2, 10, 6, 1, 4, 7, 8, 9, 10, 7, 10, 8, 4, 6, 8, 9, 10, 1, 9, 10, 6, 8,
4, 10, 8, 2, 10, 2, 3, 10, 1, 5, 9, 4, 4, 8, 2, 7, 6, 4, 8, 10, 4, 8,
10, 6, 10, 4, 9, 4, 1, 6, 5, 3, 2, 4, 1, 3, 4, 8, 4, 3, 7, 2, 5, 4, 5,
10, 7, 4, 2, 6, 3, 2, 2, 8, 4, 10, 8, 10, 2, 10, 6, 5, 2, 3, 2, 6, 2,
7, 7, 8, 9, 7, 10, 8, 6, 7, 9, 7, 10, 3, 2, 7, 5, 6, 1, 3, 9, 7, 7, 1,
8, 7, 8, 8, 8, 10, 4, 5, 9, 4, 6, 9, 6, 10, 2]
rand=np.array(rand)
```

In [984…

```
##

def con(*args):
    tab = pd.crosstab(*args)
    print(tab)
    tab = np.matrix(tab)

    r, c = tab.shape
    print(r,c)
    for i in range(0,r): tab[i,i]=0
    print(tab)
    print("error rate = ",
        round(100*(np.sum(tab))/len([*args[0]]), 2), "\n\n")
    return tab
```

In [985…

```
#
##
def CVtest(fitfn, predfn, *args):
    res = np.array(["aaaaaaaa"]*214)

    for i in sorted(np.unique(rand)) :
        print("fold  ", i, "\n")
#         print(rand!=i)
        learn = fitfn(rand != i, *args)
        res[rand == i] = predfn(learn, rand == i)
        print(sum(fgl.type!=res))
    return res

def func1(x,*args):
    print()
    return LogisticRegression(multi_class='multinomial',
                              max_iter=1000).fit(fgl.iloc[x,0:9], fgl.type.i

def func2(obj,x):
    return obj.predict(fgl[x].iloc[:,0:9])

from sklearn import linear_model

res_multinom= CVtest(func1,func2)

print(res_multinom)
```

```
fold   1


204
fold   2


191
fold   3
```

```
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/linear_mod
el/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/linear_mod
el/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/linear_mod
el/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
```

```
184
fold   4


172
fold   5


162
fold   6
```

```
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/linear_mod
el/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/linear_mod
el/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/linear_mod
el/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
```

```
144
fold   7


136
fold   8


115
fold   9
```

```
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/linear_mod
el/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/linear_mod
el/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/linear_mod
el/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
```

```
103
fold   10
```

```
86
['WinF' 'WinNF' 'WinNF' 'WinF' 'WinF' 'WinNF' 'WinF' 'WinF' 'WinF' 'WinNF'
 'WinNF' 'WinF' 'WinNF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinNF'
 'WinNF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinNF' 'WinF' 'WinNF' 'WinF'
 'WinF' 'WinF' 'WinF' 'WinNF' 'WinF' 'WinF' 'WinNF' 'WinF' 'WinF' 'WinF'
 'WinF' 'WinF' 'WinNF' 'WinF' 'WinNF' 'WinNF' 'WinF' 'WinF' 'WinF' 'WinNF'
 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'Veh' 'WinF' 'WinF' 'WinF'
 'WinF' 'WinNF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF'
 'Tabl' 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF'
 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF'
 'WinNF' 'WinNF' 'WinF' 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF'
 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinF' 'WinNF' 'Con' 'Con'
 'WinNF' 'Tabl' 'Tabl' 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinF' 'WinF' 'WinF'
 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinNF'
 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinNF' 'WinF'
 'WinF' 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinNF' 'WinNF'
 'WinNF' 'WinF' 'WinF' 'WinF' 'WinNF' 'WinF' 'WinNF' 'WinF' 'WinF' 'WinNF'
 'WinF' 'WinF' 'WinF' 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinF' 'Head'
 'WinNF' 'WinNF' 'WinNF' 'Con' 'Con' 'Con' 'WinNF' 'Con' 'Con' 'WinNF'
 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'Tabl' 'WinNF' 'Head' 'Tabl' 'Head'
 'Tabl' 'Head' 'WinNF' 'Head' 'WinF' 'WinNF' 'Head' 'WinNF' 'Head' 'Head'
 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Con' 'Head'
 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Head'
 'Head']
```

```
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/sklearn/linear_mod
el/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=
1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
```

In [986…

```
#!!!!!!!!!!!!!

tab=con(fgl.type, res_multinom)

#print(tab)
```

```
col_0  Con  Head  Tabl  Veh  WinF  WinNF
type
Con      5     1     0    0     0      7
Head     1    24     0    0     1      3
Tabl     0     3     3    0     0      3
Veh      0     0     0    0    11      6
WinF     0     0     0    1    47     22
WinNF    2     0     3    0    22     49
6 6
[[ 0  1  0  0  0  7]
 [ 1  0  0  0  1  3]
 [ 0  3  0  0  0  3]
 [ 0  0  0  0 11  6]
 [ 0  0  0  1  0 22]
 [ 2  0  3  0 22  0]]
error rate =  40.19
```

In [987…

```
'''
res_lda = CVtest(
  function(x, ...) lda(type ~ ., fgl[x, ], ...),
  function(obj, x) predict(obj, fgl[x, ]) )
'''

def func9(x,*args):
    print()
    return LDA().fit(fgl[x].iloc[:,0:9], fgl.type[x])

def func2(obj,x):
    return obj.predict(fgl[x].iloc[:,0:9])

res_lda=CVtest(func9,func2)
print(res_lda)
```

```
fold  1


204
fold  2


189
fold  3


182
fold  4


167
fold  5


156
fold  6


140
fold  7


128
fold  8


109
fold  9


96
fold  10


80
['WinF' 'WinF' 'WinNF' 'WinNF' 'WinF' 'WinNF' 'WinF' 'WinF' 'WinF' 'WinF'
 'WinNF' 'WinF' 'WinNF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF'
 'WinNF' 'Veh' 'WinF' 'WinF' 'WinF' 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinF'
 'WinF' 'WinF' 'WinNF' 'WinF' 'WinF' 'Veh' 'WinF' 'WinF' 'WinF' 'WinF'
 'WinF' 'WinNF' 'WinF' 'WinF' 'WinNF' 'WinF' 'WinF' 'WinNF' 'WinF' 'WinNF'
 'WinF' 'WinF' 'WinF' 'WinNF' 'WinNF' 'WinNF' 'Veh' 'WinNF' 'WinF' 'WinF'
 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF'
 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinF' 'WinNF' 'WinNF' 'WinF'
 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinNF'
 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF'
 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinF' 'WinF' 'Con' 'Head'
 'WinNF' 'Tabl' 'Tabl' 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinF' 'WinF'
 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF'
 'WinNF' 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'Con' 'WinF' 'WinNF'
 'WinF' 'WinF' 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinF'
 'WinNF' 'WinF' 'WinNF' 'WinF' 'WinF' 'WinF' 'WinNF' 'WinNF' 'WinF' 'WinF'
 'WinNF' 'WinNF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinNF' 'WinNF' 'WinF'
 'WinNF' 'Head' 'WinNF' 'WinNF' 'WinNF' 'Con' 'Con' 'Con' 'Con' 'Con'
 'Con' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF' 'Tabl' 'WinNF' 'Tabl' 'Tabl'
 'Head' 'Tabl' 'Head' 'WinF' 'Head' 'WinF' 'Con' 'Head' 'Head' 'Head'
 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Con'
 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Head' 'Head'
 'Head' 'Head']
```

In [988…

```
#

con(fgl.type, res_lda)
```

```
col_0  Con  Head  Tabl  Veh  WinF  WinNF
type
Con      6     1     0    0     0      6
Head     2    25     0    0     2      0
Tabl     0     2     4    0     1      2
Veh      0     0     0    0    10      7
WinF     0     0     0    3    49     18
WinNF    2     1     2    0    21     50
6 6
[[ 0  1  0  0  0  6]
 [ 2  0  0  0  2  0]
 [ 0  2  0  0  1  2]
 [ 0  0  0  0 10  7]
 [ 0  0  0  3  0 18]
 [ 2  1  2  0 21  0]]
error rate =  37.38
```

Out[988]:

```
matrix([[ 0,  1,  0,  0,  0,  6],
        [ 2,  0,  0,  0,  2,  0],
        [ 0,  2,  0,  0,  1,  2],
        [ 0,  0,  0,  0, 10,  7],
        [ 0,  0,  0,  3,  0, 18],
        [ 2,  1,  2,  0, 21,  0]])
```

In [989…

```
#

fgl0 = fgl.drop(fgl.columns[[9]],axis=1) # drop type
print(fgl0)

def res_knn1():
    res = np.array(["aaaaaaaa"]*214)
    knn=KNeighborsClassifier(n_neighbors = 1)



    for i in sorted(np.unique(rand)):
        print("fold ", i ,"\n")
        knn.fit(fgl0[rand != i],fgl.type[rand != i])
        res[rand == i]=knn.predict( fgl0[rand == i])

 #       res[rand == i] = knn(fgl0[rand != i], fgl0[rand == i], fgl.type[ran
 #                            k = 1)

    return res

print(res_knn1())
```

```
       RI     Na    Mg    Al     Si     K    Ca    Ba   Fe
0    3.01  13.64  4.49  1.10  71.78  0.06  8.75  0.00  0.0
1   -0.39  13.89  3.60  1.36  72.73  0.48  7.83  0.00  0.0
2   -1.82  13.53  3.55  1.54  72.99  0.39  7.78  0.00  0.0
3   -0.34  13.21  3.69  1.29  72.61  0.57  8.22  0.00  0.0
4   -0.58  13.27  3.62  1.24  73.08  0.55  8.07  0.00  0.0
..    ...    ...   ...   ...    ...   ...   ...   ...  ...
209 -1.77  14.14  0.00  2.88  72.61  0.08  9.18  1.06  0.0
210 -1.15  14.92  0.00  1.99  73.06  0.00  8.40  1.59  0.0
211  2.65  14.36  0.00  2.02  73.42  0.00  8.44  1.64  0.0
212 -1.49  14.38  0.00  1.94  73.61  0.00  8.48  1.57  0.0
213 -0.89  14.23  0.00  2.08  73.36  0.00  8.62  1.67  0.0

[214 rows x 9 columns]
fold  1

fold  2

fold  3

fold  4

fold  5

fold  6

fold  7

fold  8

fold  9

fold  10

['Veh' 'WinF' 'WinNF' 'WinF' 'WinF' 'WinNF' 'WinF' 'WinF' 'WinF' 'WinF'
 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'Veh' 'WinF' 'WinF'
 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF'
 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'Veh' 'WinF' 'WinF' 'WinF' 'WinF'
 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinNF' 'WinNF' 'WinF' 'WinF'
 'WinF' 'WinF' 'WinF' 'WinF' 'WinF' 'WinNF' 'WinNF' 'WinF' 'WinF' 'WinF'
 'Veh' 'WinF' 'WinF' 'WinF' 'WinF' 'Veh' 'WinF' 'WinF' 'WinF' 'WinF'
 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF'
 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'Veh' 'WinNF' 'WinNF'
 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF'
 'WinF' 'WinNF' 'WinF' 'WinNF' 'WinF' 'WinF' 'WinF' 'WinF' 'Con' 'WinNF'
 'WinNF' 'Con' 'Tabl' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF'
 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF'
 'WinNF' 'Veh' 'WinNF' 'WinNF' 'WinNF' 'Con' 'WinNF' 'WinNF' 'WinNF'
 'WinNF' 'WinNF' 'WinF' 'WinNF' 'WinNF' 'WinNF' 'WinNF' 'WinF' 'WinNF'
 'WinF' 'Veh' 'WinNF' 'Veh' 'Veh' 'WinNF' 'WinNF' 'Veh' 'Veh' 'Veh' 'Veh'
 'WinNF' 'WinNF' 'Veh' 'Veh' 'Veh' 'Veh' 'Veh' 'WinF' 'WinF' 'Head' 'Tabl'
 'Con' 'Con' 'Con' 'Head' 'Con' 'WinNF' 'Con' 'Con' 'Con' 'WinNF' 'Con'
 'Tabl' 'Tabl' 'Tabl' 'Tabl' 'WinF' 'Tabl' 'Con' 'Tabl' 'Head' 'WinNF'
 'WinNF' 'WinNF' 'Veh' 'Head' 'WinNF' 'Head' 'Head' 'Head' 'Head' 'Head'
 'Head' 'Head' 'Head' 'Head' 'Head' 'Con' 'Head' 'Head' 'Head' 'Head'
 'Head' 'Head' 'Head' 'Head' 'Head' 'Tabl' 'Head' 'Head']
```

In [990…

```
#

con(fgl.type, res_knn1())
```

```
fold  1

fold  2

fold  3

fold  4

fold  5

fold  6

fold  7

fold  8

fold  9

fold  10

col_0   Con  Head  Tabl  Veh  WinF  WinNF
type
Con       8     2     1    0     0      2
Head      1    22     1    1     0      4
Tabl      1     1     6    0     1      0
Veh       0     0     0   11     2      4
WinF      0     0     0    5    59      6
WinNF     3     0     1    3    12     57
6 6
[[ 0  2  1  0  0  2]
 [ 1  0  1  1  0  4]
 [ 1  1  0  0  1  0]
 [ 0  0  0  0  2  4]
 [ 0  0  0  5  0  6]
 [ 3  0  1  3 12  0]]
error rate =  23.83
```

Out[990]:

```
matrix([[ 0,  2,  1,  0,  0,  2],
        [ 1,  0,  1,  1,  0,  4],
        [ 1,  1,  0,  0,  1,  0],
        [ 0,  0,  0,  0,  2,  4],
        [ 0,  0,  0,  5,  0,  6],
        [ 3,  0,  1,  3, 12,  0]])
```

In [991…

```
##

knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(fgl0,fgl.type)
res_lb=knn.predict_proba(fgl0)
table_res=pd.DataFrame(np.unique(np.max(res_lb,axis=1),return_counts=True))
print(table_res)
```

```
          0          1      2
0   0.333333   0.666667    1.0
1  10.000000  64.000000  140.0
```

In [992…

```
#ㅎㅎㅎㅎㅎㅎㅎㅎㅎㅎ

from sklearn.tree import DecisionTreeClassifier

def func3(x, *args):
    tr=DecisionTreeClassifier(criterion='entropy',
                              max_depth=3,
                              random_state=0).fit(fgl.iloc[x,0:9],
                                                  fgl.type.iloc[x])
    cp = tr.cptable
    r = cp[:, 4] + cp[:, 5]
    rmin = np.min(seq(along = r)[cp.iloc[:, 4] < min(r)])
    cp0 = cp[rmin, 1]

    print("size chosen was", cp.iloc[rmin, 2] + 1, "\n")
    prune(tr, cp = 1.01*cp0)
```

In [993…

```
res_rpart= CVtest(func3,func2)
```

```
fold   1
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [993], in <cell line: 1>()
----> 1 res_rpart= CVtest(func3,func2)

Input In [985], in CVtest(fitfn, predfn, *args)
      7         print("fold  ", i, "\n")
      8 #          print(rand!=i)
----> 9         learn = fitfn(rand != i, *args)
     10         res[rand == i] = predfn(learn, rand == i)
     11         print(sum(fgl.type!=res))

Input In [992], in func3(x, *args)
      5 def func3(x, *args):
      7     tr=DecisionTreeClassifier(criterion='entropy',
      8                               max_depth=3,
      9                               random_state=0).fit(fgl.iloc[x,0:9],
     10                                                   fgl.type.iloc[x])
---> 11     cp = tr.cptable
     12     r = cp[:, 4] + cp[:, 5]
     13     rmin = np.min(seq(along = r)[cp.iloc[:, 4] < min(r)])

AttributeError: 'DecisionTreeClassifier' object has no attribute 'cptable'
```

In [ ]:

```
con( fgl.type, res_rpart)
```

In [ ]:

```
fgl1 = np.matrix(fgl.iloc[:,0:9])

max_f=np.max(fgl1[0:9])
min_f=np.min(fgl1[0:9])

#print(min_f,max_f)

fgl1[0:9]=(fgl1[0:9]-min_f)/(max_f-min_f)

#print(np.max(fgl1[0:9]))

#min 과 max가 다른 값이 나옴! 이상
```

In [ ]:

```
#gggggggggggggggg

CVnn2 = function(formula, data,
                 size = rep(6,2), lambda = c(0.001, 0.01),
                 nreps = 1, nifold = 5, verbose = 99, ...)
{
    CVnn1 = function(formula, data, nreps=1, ri, verbose,  ...)
    {
        truth = data[,deparse(formula[[2]])]
        res =  matrix(0, nrow(data), length(levels(truth)))
        if(verbose > 20) cat("  inner fold")
        for (i in sort(unique(ri))) {
            if(verbose > 20) cat(" ", i,  sep="")
            for(rep in 0:nreps) {
                learn = nnet(formula, data[ri !=i,], trace = FALSE, ...)
                res[ri == i,] = res[ri == i,] +
                    predict(learn, data[ri == i,])
            }
        }
        if(verbose > 20) cat("\n")
        sum(as.numeric(truth) != max.col(res/nreps))
    }
    truth = data[,deparse(formula[[2]])]
    res =  matrix(0, nrow(data), length(levels(truth)))
    choice = numeric(length(lambda))
    for (i in sort(unique(rand))) {
        if(verbose > 0) cat("fold ", i,"\n", sep="")
        ri = sample(nifold, sum(rand!=i), replace=TRUE)
        for(j in seq(along=lambda)) {
            if(verbose > 10)
                cat("  size =", size[j], "decay =", lambda[j], "\n")
            choice[j] = CVnn1(formula, data[rand != i,], nreps=nreps,
                               ri=ri, size=size[j], decay=lambda[j],
                               verbose=verbose, ...)
        }
        decay = lambda[which.is.max(-choice)]
        csize = size[which.is.max(-choice)]
        if(verbose > 5) cat("  #errors:", choice, "  ") #
        if(verbose > 1) cat("chosen size = ", csize,
                            " decay = ", decay, "\n", sep="")
        for(rep in 0:nreps) {
            learn = nnet(formula, data[rand != i,], trace=FALSE,
                          size=csize, decay=decay, ...)
            res[rand == i,] = res[rand == i,] +
                predict(learn, data[rand == i,])
        }
    }
    factor(levels(truth)[max.col(res/nreps)], levels = levels(truth))
}

if(FALSE) { # only run this if you have time to wait
res.nn2 = CVnn2(type ~ ., fgl1, skip = TRUE, maxit = 500, nreps = 10)
con(true = fgl$type, predicted = res.nn2)
}
```

In [ ]:

```
def func4(x,*args):
    fgl_svm=svm.SVC(kernel='linear', C=100).fit(fgl.iloc[x,0:9],fgl.type.ilc
    return fgl_svm


'''
def func2(obj,x):
    return obj.predict(fgl[x].iloc[:,0:9])
'''
res_svm=CVtest(func4,func2)
```

In [ ]:

```
##

con( fgl.type, res_svm)
```

In [ ]:

```
##

#svm(type ~ ., data = fgl, cost = 100, gamma = 1, cross = 10)
from sklearn import svm

fgl_svm2 = svm.SVC(kernel='linear', C=100).fit(fgl.iloc[x,0:9],fgl.type.iloc
fgl_svm2
```

In [ ]:

```
#!pip install sklvq
from sklvq.models import LVQBaseClass
```

In [ ]:

```
'''
cd0 = lvqinit(fgl0, fgl.type, prior = rep(1, 6)/6, k = 3)
cd1 = olvq1(fgl0, fgl.type, cd0)
'''

cd0= LVQBaseClass().fit(fgl0, fgl.type)


con(fgl.type, lvqtest(cd0, fgl0))
```

In [ ]:

```
#

def CV_lvq():

    res = np.array(["aaaaaaaa"]*214)

    for i in sorted(np.unique(rand)):
        print("doing fold", i, "\n")

        cd0 = lvqinit(fgl0[rand != i,], fgl.type[rand != i],
                      prior = rep(1, 6)/6, k = 3)
        cd1 = olvq1(fgl0[rand != i,], fgl.type[rand != i], cd0)
        cd1 = lvq3(fgl0[rand != i,], fgl.type[rand != i],
                   cd1, niter = 10000)

        res[rand == i] = lvqtest(cd1, fgl0[rand == i, ])

    return res
```

In [ ]:

```
con(fgl.type,CV_lvq())
```

In [ ]:

12.7 Calibration plots

In [ ]:

```
##

def CVprobs(fitfn, predfn, *args):

    res = np.zeros((214,6))

    for i in sorted(np.unique(rand)):
        print("fold ", i, "\n")
        learn = fitfn(rand != i, *args)
        res[rand == i,:] = predfn(learn, rand == i)

    print(res)
    return res
```

In [ ]:

```
##

a=np.array(pd.get_dummies(fgl.type)).flatten()
print(a[0:20])
```

In [ ]:

```
'''
probs_multinom = CVprobs(
  function(x, ...) multinom(type ~ ., fgl[x, ], ...),
  function(obj, x) predict(obj, fgl[x, ], type = "probs"),
  maxit = 1000, trace = FALSE)
  '''


'''
def func1(x,*args):
    print()
    return LogisticRegression(multi_class='multinomial',
                              max_iter=1000).fit(fgl.iloc[x,0:9], fgl.type.i
'''
def func7(obj,x):
    return obj.predict_proba(fgl[x].iloc[:,0:9])

probs_multinom = CVprobs(func1,func7)
```

In [994…

```
##

probs_yes = np.array(pd.get_dummies(fgl.type)).flatten()
probs = np.array(probs_multinom).flatten()
```

In [995…

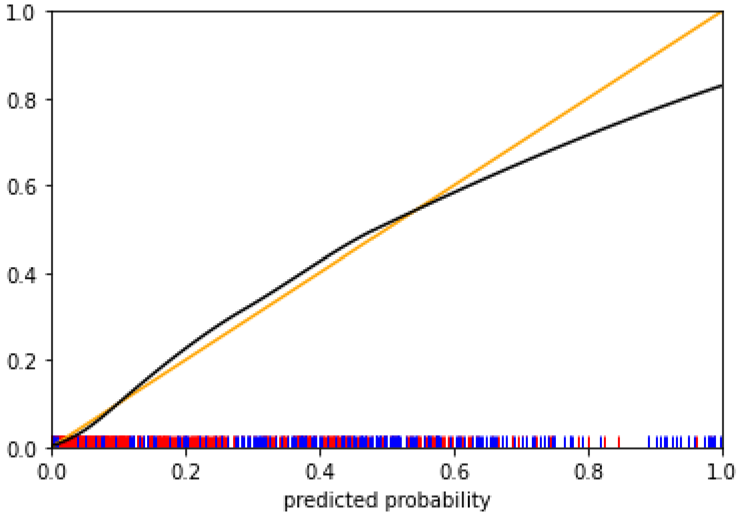
```
#!pip install scikit-misc
from skmisc.loess import loess
```

In [996…

```
'''
par(pty = "s")
plot(c(0, 1), c(0, 1), type = "n", xlab = "predicted probability",
     ylab = "", xaxs = "i", yaxs = "i", las = 1)

rug(probs[probs_yes == 0], 0.02, side = 1, lwd = 0.5)
rug(probs[probs_yes == 1], 0.02, side = 3, lwd = 0.5)

abline(0, 1)
newp = np.linspace(0, 1, 100)
lines(newp, predict(loess(probs.yes ~ probs, span = 1), newp))
'''

import seaborn as sns

g=sns.rugplot(probs[probs_yes == 0],color='r')
sns.rugplot(probs[probs_yes == 1],color='b')
sns.lineplot([0,1],[0,1],color='orange')

newp=np.linspace(0,1,100)
y=loess(probs,probs_yes)
y=y.predict(newp).values
#print(y)

sns.lineplot(newp,y,color='black',x='predicted probability')

g.set(xlim=(0,1),ylim=(0,1),xlabel='predicted probability')
```

```
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/seaborn/_decorator
s.py:36: FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`, and pa
ssing other arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
/Users/kyungseonlee/anaconda3/lib/python3.9/site-packages/seaborn/_decorator
s.py:36: FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`, and pa
ssing other arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

```
Out[996]: [(0.0, 1.0), (0.0, 1.0), Text(0.5, 0, 'predicted probability')]
```



```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```