Praktikum Pemodelan Statistika terapan

Dosen Pengampu Ronny Sutsetyoko

Wahyu Ikbal Maulana

3323600056

D4 SDT B

Politeknik Elektronika Negeri Surabaya

Percobaan ke-1: Studi Kasus 1

In [ ]:
```
library(readr)
tissue <- read_csv("breast_tissue.csv") # Checking the structure of adult data
str(tissue)
```

```
Rows: 106 Columns: 11
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (1): Class
dbl (10): patient_id, I0, PA500, HFS, DA, Area, A/DA, Max IP, DR, P

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
spc_tbl_ [106 × 11] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ patient_id: num [1:106] 1 2 3 4 5 6 7 8 9 10 ...
 $ Class     : chr [1:106] "car" "car" "car" "car" ...
 $ I0        : num [1:106] 525 330 552 380 363 ...
 $ PA500     : num [1:106] 0.187 0.227 0.232 0.241 0.201 ...
 $ HFS       : num [1:106] 0.0321 0.2653 0.0635 0.2862 0.2443 ...
 $ DA        : num [1:106] 229 121 265 138 125 ...
 $ Area      : num [1:106] 6844 3163 11888 5402 3290 ...
 $ A/DA      : num [1:106] 29.9 26.1 44.9 39.2 26.3 ...
 $ Max IP    : num [1:106] 60.2 69.7 77.8 88.8 69.4 ...
 $ DR        : num [1:106] 220.7 99.1 253.8 105.2 103.9 ...
 $ P         : num [1:106] 557 400 657 494 425 ...
 - attr(*, "spec")=
  .. cols(
  ..   patient_id = col_double(),
  ..   Class = col_character(),
  ..   I0 = col_double(),
  ..   PA500 = col_double(),
  ..   HFS = col_double(),
  ..   DA = col_double(),
  ..   Area = col_double(),
  ..   `A/DA` = col_double(),
  ..   `Max IP` = col_double(),
  ..   DR = col_double(),
  ..   P = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

In [ ]:
```
tissue <- tissue[, -1]
tissue$Class <- as.factor(tissue$Class)
levels(tissue$Class)[levels(tissue$Class)  %in%  c("fad",  "gla", "mas")] <-
levels(tissue$Class)
```

'adi' · 'car' · 'con' · 'other'

Analisis : Dari dataset BreastTissue, kemudian direduksi kelas dimana fibro-adenoma, mastopati, dan glandular dijadikan satu kelas diberi nama 'others'.

In [ ]:
```
#Splitting the data using a function from dplyr package
library(caret)
library(ggplot2)
```

```
index <- createDataPartition(tissue$Class, p = .70, list = FALSE)
train <- tissue[index,]
test <- tissue[-index,]
```

In [ ]:
```
# Setting the reference
train$Class <- relevel(train$Class, ref = "adi")
```

In [ ]:
```
library(nnet)
# Training the multinomial model
multinom_model <- multinom(Class ~ ., data = tissue)
# Checking the model
summary(multinom_model)
```

```
# weights:  44 (30 variable)
initial  value 146.947202
iter  10 value 107.785951
iter  20 value 71.796827
iter  30 value 17.709626
iter  40 value 11.374947
iter  50 value 6.775080
iter  60 value 5.795296
iter  70 value 5.581882
iter  80 value 4.879180
iter  90 value 4.199680
iter 100 value 4.160692
final  value 4.160692
stopped after 100 iterations
Call:
multinom(formula = Class ~ ., data = tissue)

Coefficients:
      (Intercept)          I0      PA500        HFS         DA         Area
car      86.45689 -0.92420610  35.426343 -28.995689 -2.7614791 -0.009627356
con      65.29309 -0.02131995   3.507425   5.180119  0.3799671 -0.006557466
other    94.35136 -0.44817721 -10.573482  45.314817 -0.2506004 -0.011107358
          `A/DA`   `Max IP`         DR           P
car   -0.6522402 2.3039134  3.0645844  0.67990594
con    1.3401948 0.2354590 -0.1683327 -0.07552706
other  1.5460282 0.1120201  0.6247094  0.29486854

Std. Errors:
      (Intercept)        I0        PA500         HFS        DA         Area
car    0.03476847 0.1066177 0.0177918233 0.037347078 0.6130240 0.029524661
con    0.00246189 0.5774913 0.0003169089 0.002276567 0.3423545 0.005154037
other  0.03434195 0.1013918 0.0176815177 0.037334873 0.6274376 0.029559302
         `A/DA`  `Max IP`        DR         P
car   0.5424341 0.4562191 0.5804292 0.1709975
con   0.2034479 0.6789553 0.2860865 0.4041700
other 0.5335004 0.4787424 0.5696903 0.1633894

Residual Deviance: 8.321383
AIC: 68.32138
```

🖉 **Analisis :** Dalam regresi logistik multinomial memakai library nnet, menghasilkan model training yang mengiterasi optimisasi train model. Optimisasi tersebut bertujual untuk meminimalkan fungsi objektivitas untuk menemukan model parameter yang optimal. Optimisasi terhenti setelah mencapai kriteria convergence, saat dimana improvisasi objek sudah kecil dan maksimum iterasi tercapai. Nilai AIC yang cenderung rendah menunjukkan bahwa model memiliki kualitas yang baik.

In [ ]:
```
exp(coef(multinom_model))
```

A matrix: 3 × 10 of type dbl

	(Intercept)	I0	PA500	HFS	DA	Area	`A
car	3.529816e+37	0.3968464	2.429210e+15	2.554656e-13	0.06319822	0.9904188	0.520
con	2.272107e+28	0.9789057	3.336224e+01	1.777040e+02	1.46223641	0.9934640	3.819
other	9.468380e+40	0.6387915	2.558557e-05	4.786024e+19	0.77833332	0.9889541	4.692

In [ ]:
```
head(round(fitted(multinom_model), 2),20)
```

A matrix: 20 × 4 of type dbl

	adi	car	con	other
1	0	0.97	0	0.03
2	0	1.00	0	0.00
3	0	1.00	0	0.00
4	0	1.00	0	0.00
5	0	1.00	0	0.00
6	0	0.96	0	0.04
7	0	0.77	0	0.23
8	0	0.53	0	0.47
9	0	1.00	0	0.00
10	0	1.00	0	0.00
11	0	1.00	0	0.00
12	0	1.00	0	0.00
13	0	0.91	0	0.09
14	0	1.00	0	0.00
15	0	1.00	0	0.00
16	0	1.00	0	0.00
17	0	1.00	0	0.00
18	0	0.55	0	0.45
19	0	1.00	0	0.00
20	0	0.96	0	0.04

In [ ]:
```
tail(round(fitted(multinom_model), 2),20)
```

A matrix: 20 × 4 of type dbl

	adi	car	con	other
87	1	0	0	0
88	1	0	0	0
89	1	0	0	0
90	1	0	0	0
91	1	0	0	0
92	1	0	0	0
93	1	0	0	0
94	1	0	0	0
95	1	0	0	0
96	1	0	0	0
97	1	0	0	0
98	1	0	0	0
99	1	0	0	0
100	1	0	0	0
101	1	0	0	0
102	1	0	0	0
103	1	0	0	0
104	1	0	0	0
105	1	0	0	0
106	1	0	0	0

🖉 **Analisis :** Regresi multinomial memprediksi probabilitas pengamatan tertentu untuk menjadi bagian dari kelas tertentu. Kolom mewakili tingkat klasifikasi dan baris mewakili pengamatan. 20 baris pertama terklasifikasi sebagai carsinoma dan 20 baris terakhir tergolong sebagai adipose

In [ ]:
```
# Predicting the values for train dataset
train$ClassPredicted <- predict(multinom_model, newdata = train, "class")
# Building classification table
tab <- table(train$Class, train$ClassPredicted)
# Calculating accuracy - sum of diagonal elements divided by total obs
round((sum(diag(tab))/sum(tab))*100,2)
```

98.68

In [ ]:
```
# Predicting the class for test dataset
test$ClassPredicted <- predict(multinom_model, newdata = test, "class")
# Building classification table
tab <- table(test$Class, test$ClassPredicted)
tab
```

```
       adi car con other
adi      6   0   0     0
car      0   6   0     0
con      0   0   4     0
other    0   0   0    14
```

🖉 **Analisis :** Didapat untuk akurasi skor dari tiap tiap konfigurasi dan semua plot yang. Didapat bahwa akurasinya sangat tinggi yaitu 98.68%. Disimpulkan bahwa model bagus dan stabil.

Percobaan ke-3: Studi Kasus 3

In [ ]:
```
# test classification dataset
from collections import Counter
from sklearn.datasets import make_classification
# define dataset
X, y = make_classification(n_samples=1000, n_features=10, n_informative=5,
                           n_redundant=5, n_classes=3, random_state=1)

# summarize the dataset
print(X.shape, y.shape)
print(Counter(y))
```

```
(1000, 10) (1000,)
Counter({1: 334, 2: 334, 0: 332})
```

🖉 **Analisis :** Kode di atas digunakan untuk membuat dataset klasifikasi sintetis dengan 1000 sampel, 10 fitur, dan 3 kelas. Dan distribusi kelas terbagi menjadi 334 untuk tiap kelas. Disini kita dapat memahami struktur dataset klasifikasi dan distribusi kelasnya.

In [ ]:
```
# define the multinomial logistic regression model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(multi_class='multinomial', solver='lbfgs')
```

🖉 **Analisis :** Memanggil fungsi multinomial logistic regression kemudian di fit menggunakan cross-entropy loss dan akan dipredict nilainya untuk tiap label kelasnya.

In [ ]:
```
# evaluate multinomial logistic regression model
from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.linear_model import LogisticRegression
# define dataset
X, y = make_classification(n_samples=1000, n_features=10, n_informative=5,
                           n_redundant=5, n_classes=3, random_state=1)
# define the multinomial logistic regression model
model = LogisticRegression(multi_class='multinomial', solver='lbfgs')
# define the model evaluation procedure
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
# evaluate the model and collect the scores
n_scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
# report the model performance
print('Mean Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

```
Mean Accuracy: 0.681 (0.042)
```

🖉 **Analisis :** Dalam kasus ini kita bisa melihat multinomial logistic regression model dengan penalty default dan didapat dengan mean classification akurasi sekitar 68.1% dari dataset klasifikasi sintetis. Dengan 3 kali perulangan dengan 10 folds, dimana nilai ini adalah nilai default dari repeated stratified k-fold cross-validation, kita bisa mengevaluasi model menggunakan klasifikasi.

In [ ]:
```
# make a prediction with a multinomial logistic regression model
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
# define dataset
X, y = make_classification(n_samples=1000, n_features=10, n_informative=5, n_red
# define the multinomial logistic regression model
model = LogisticRegression(multi_class='multinomial', solver='lbfgs')
# fit the model on the whole dataset
model.fit(X, y)
# define a single row of input data
row = [1.89149379, -0.39847585, 1.63856893, 0.01647165, 1.51892395, -3.52651223,
# predict the class label
yhat = model.predict([row])
# summarize the predicted class
print('Predicted Class: %d' % yhat[0])
```

```
Predicted Class: 1
```

In [ ]:
```
# predict probabilities with a multinomial logistic regression model
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
# define dataset
X, y = make_classification(n_samples=1000, n_features=10, n_informative=5, n_red
# define the multinomial logistic regression model
model = LogisticRegression(multi_class='multinomial', solver='lbfgs')
# fit the model on the whole dataset
model.fit(X, y)
# define a single row of input data
row = [1.89149379, -0.39847585, 1.63856893, 0.01647165, 1.51892395, -3.52651223,
# predict a multinomial probability distribution
yhat = model.predict_proba([row])
# summarize the predicted probabilities
print('Predicted Probabilities: %s' % yhat[0])
```

```
Predicted Probabilities: [0.16470456 0.50297138 0.33232406]
```

🖉 **Analisis :** Disini kemudian memasukkan sebuah satu baris data dummy dengan variable row yang kemudian dilakukan prediksi, didapat bahwa dalam baris tersebut merupakan kelas pertama. Beserta probrabilitas untuk tiap tiap kelasnya.

In [ ]:
```
# define the multinomial logistic regression model with a default penalty
LogisticRegression(multi_class='multinomial', solver='lbfgs', penalty='l2', C=1.
```

Out[ ]:
```
▾ LogisticRegression
LogisticRegression(multi_class='multinomial')
```

🖉 **Analisis :** Selanjutnya dari dataset kemudian dilakukan prediksi dengan solver Limited-memory Broyden-Fletcher-Goldfarb-Shanno untuk optimisasinya.

🖉 **Analisis :** Mari kita lihat dari L2 penalti dengan berat value dari range 0.0001 hingga 1.0. Dengan perhitungan komplex dari multinomial logistic regression di bawah ini.

In [ ]:
```
# tune regularization for multinomial logistic regression
from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.linear_model import LogisticRegression
from matplotlib import pyplot

# get the dataset
def get_dataset():
        X, y = make_classification(n_samples=1000, n_features=20, n_informative=
                                    n_redundant=5, random_state=1, n_classes=3)
        return X, y
```

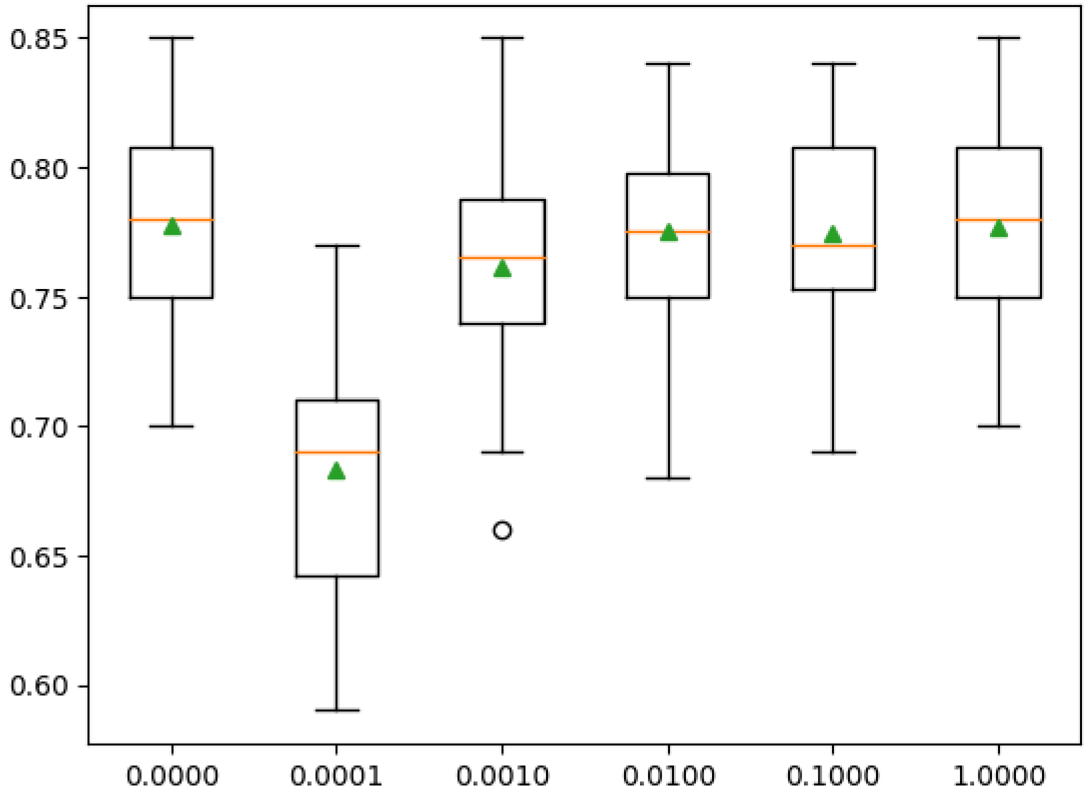
In [ ]:
```
# get a list of models to evaluate
def get_models():
        models = dict()
        for p in [0.0, 0.0001, 0.001, 0.01, 0.1, 1.0]:
                # create name for model
                key = '%.4f' % p
                # turn off penalty in some cases
                if p == 0.0:
                        # no penalty in this case
                        models[key] = LogisticRegression(multi_class='multinomia
                else:
                        models[key] = LogisticRegression(multi_class='multinomia
        return models

# evaluate a give model using cross-validation
def evaluate_model(model, X, y):
        # define the evaluation procedure
        cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
        # evaluate the model
        scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=
        return scores
```

In [ ]:
```
# define dataset
X, y = get_dataset()
# get the models to evaluate
models = get_models()
# evaluate the models and store results
results, names = list(), list()
for name, model in models.items():
        # evaluate the model and collect the scores
        scores = evaluate_model(model, X, y)
```

```
        # store the results
        results.append(scores)
        names.append(name)
        # summarize progress along the way
        print('>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))
# plot model performance for comparison
pyplot.boxplot(results, labels=names, showmeans=True)
pyplot.show()
```

```
>0.0000 0.777 (0.037)
>0.0001 0.683 (0.049)
>0.0010 0.762 (0.044)
>0.0100 0.775 (0.040)
>0.1000 0.774 (0.038)
>1.0000 0.777 (0.037)
```



Box and Whisker Plots of L2 Penalty Configuration vs. Accuracy for Multinomial Logistic Regression

✏️ **Analisis :** Kode di atas memungkinkan ntuk membandingkan kinerja model regresi logistik dengan berbagai tingkat regularisasi. Dari gambar box plot di atas untuk akurasi skor dari tiap tiap konfigurasi dan kita dapat memahami bagaimana performa model berubah dengan perubahan parameter regularisasi. Dalam kasus ini C Value dari 0.1 memiliki skor terbaik dengan sekitar 77.7%. Yang skornya sama dengan yang tanpa penalti.