Gabi Rivera

20Nov2022

ADS502-01

Introduction to Data Mining: Exercise 4.14

20. Consider the XOR problem where there are four training points: (1, 1, −), (1, 0, +), (0, 1, +), (0, 0, −).

Transform the data into the following feature space: φ= (1,√2x1,√2x2,√2x1x2, x21, x22). Find the maximum margin linear decision boundary in the transformed space.

Answer: Maximum margin linear decision boundary is x1x2.

XOR Truth Table

Instance	Class	x1	x2	Output
1	-	1	1	0
2	+	1	0	1
3	+	0	1	1
4	-	0	0	0

*Output: 0 if x1 & x2 are same value, 1 if they are different

Feature Space

Instance	1	sqrt(2)*x1	sqrt(2)*x2	sqrt(2)*x1x2	x1^2	x2^2	x1x2
1	1	sqrt(2)	sqrt(2)	sqrt(2)	1	1	1
2	1	sqrt(2)	0	0	1	0	0
3	1	0	sqrt(2)	0	0	1	0
4	1	0	0	0	0	0	0

XOR Features with Original x1 and x2

Instances	x1	x2	1	sqrt(2)*x1	sqrt(2)*x2	sqrt(2)*x1x2	x1^2	x2^2	x1x2
1	1	1	0	1	1	1	0	0	0
2	1	0	1	1	1	1	1	1	1
3	0	1	1	1	1	1	1	1	1
4	0	0	1	0	0	0	0	0	0

*Feature space 1: 0 if 1 (FS table), x1 ,& x2 are same value, 1 if they are different.

**Essentially compare Feature Space table to XOR table and determine outputs.

Module 4 Assignment

Gabi Rivera

2022-11-20

Data Science Using Python and R: Chapter 13

For the following exercises, work with the clothing_sales_training and clothing_sales_test data sets. Use either Python or R to solve each problem.

13. Create a logistic regression model to predict whether or not a customer has a store credit card, based on whether they have a web account and the days between purchases. Obtain the summary of the model.

```
library(lattice)
library(tidyverse)
```

Import test and training datasets:

```
cs_train = read.csv("clothing_sales_training.csv", sep = ",")
cs_test = read.csv("clothing_sales_test.csv", sep = ",")
```

Create logistic regression:

```
logreg01 = glm(formula = CC ~ Days + Web, data = cs_train, family = binomial)
summary(logreg01)
```

```
##
## Call:
## glm(formula = CC ~ Days + Web, family = binomial, data = cs_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9035  -1.1458  -0.6078   1.0895   2.1044
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.4961706  0.0886529   5.597 2.18e-08 ***
## Days        -0.0037016  0.0004381  -8.449  < 2e-16 ***
## Web          1.2536955  0.3306672   3.791  0.00015 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2009.9  on 1450  degrees of freedom
## Residual deviance: 1903.6  on 1448  degrees of freedom
## AIC: 1909.6
##
## Number of Fisher Scoring iterations: 4
```

14. Are there any variables that should be removed from the model? If so, remove them and rerun the model. Both variables will be included because there's no sign of multicollinearity amongst the variables. The z-values have reasonable scores with significantly small p-values less than 0.05 level of significance.

15. Write the descriptive form of the logistic regression model using the coefficients obtained from Question 1.

Model for credit score based on web account and days between purchases: ln(CC) = 0.4961706 -0.0037016Days + 1.2536955Web

16. Validate the model using the test data set.

```
logreg01_test = glm(formula = CC ~ Days + Web, data = cs_test, family = binomial)
summary(logreg01_test)
```

```
##
## Call:
## glm(formula = CC ~ Days + Web, family = binomial, data = cs_test)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8458  -1.1588  -0.5775   1.1022   2.0513
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.4634478  0.0872706   5.310 1.09e-07 ***
## Days        -0.0034721  0.0004221  -8.226  < 2e-16 ***
## Web          1.0972994  0.2829570   3.878 0.000105 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1932.8  on 1394  degrees of freedom
## Residual deviance: 1832.7  on 1392  degrees of freedom
## AIC: 1838.7
##
## Number of Fisher Scoring iterations: 4
```

17. Obtain the predicted values of the response variable for each record in the data set. Predicted values of test dataset:

```
cs_test$pred_prob <- predict(object = logreg01, newdata = cs_test, type='response')
head(cs_test$pred_prob)
```

```
## [1] 0.4630895 0.5428533 0.5780543 0.5567058 0.3820027 0.5708153
```

Convert predicted values to binary:

```
cs_test$pred <- (cs_test$pred_prob > 0.5)*1
head(cs_test$pred)
```

```
## [1] 0 1 1 1 0 1
```

Data Science Using Python and R: Chapter 9

For the following exercises, work with the bank_marketing_training and the bank_marketing_test data set. Use either Python or R to solve each problem.

24. Prepare the data set for neural network modeling, including standardizing the variables. Import train and test dataset:

```
bm_train = read.csv("bank_marketing_training", sep = ",")
bm_test = read.csv("bank_marketing_test", sep = ",")
```

Libraries:

```
library(nnet)
library(NeuralNetTools)
```

Convert variables to factors:

```
bm_train$response = as.factor(bm_train$response)
bm_train$response = as.factor(bm_train$response)
bm_train$job = as.factor(bm_train$job)
bm_train$marital = as.factor(bm_train$marital)

bm_test$response = as.factor(bm_test$response)
bm_test$response = as.factor(bm_test$response)
bm_test$job = as.factor(bm_test$job)
bm_test$marital = as.factor(bm_test$marital)
```

Total number of response in training dataset:

```
table(bm_train$response)
```

```
##
##    no   yes
## 23886  2988
```

Total number of response in test dataset:

```
table(bm_test$response)
```

```
##
##    no   yes
## 23886  2988
```

25. Using the training data set, create a neural network model to predict a customer's Response using whichever predictors you think appropriate. Obtain the predicted responses. Neural network model predicting response based on job:

```
nnet01 = nnet(response ~ job + marital,
              data = bm_train,
              maxit = 100, # number of iterations
              size = 25) # number of nodes in the hidden layer
```

```
## # weights:  401
## initial  value 10948.929568
## iter  10 value 9081.227399
## iter  20 value 9072.451443
## iter  30 value 9066.532533
## iter  40 value 9060.796838
## iter  50 value 9059.071585
## iter  60 value 9058.020484
## iter  70 value 9057.324128
## iter  80 value 9056.594230
## iter  90 value 9055.962198
## iter 100 value 9055.750037
## final  value 9055.750037
## stopped after 100 iterations
```

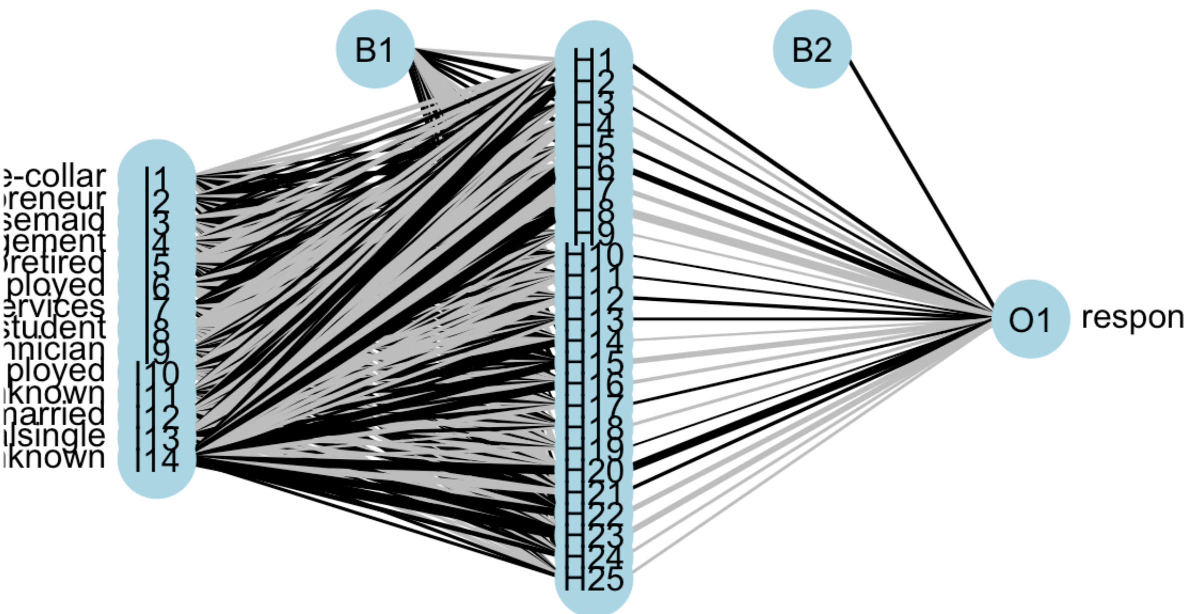
Predicted train responses:

```
bm_train$pred_prob = predict(object = nnet01, newdata = bm_train)
bm_train$pred = (bm_train$pred_prob > 0.5)*1
head(bm_train$pred)
```

```
##   [,1]
## 1    0
## 2    0
## 3    0
## 4    0
## 5    0
## 6    0
```

26. Plot the neural network.

```
plotnet(nnet01)
```



27. Evaluate the neural network model using the test data set. Construct a contingency table to compare the actual and predicted values of Response.

Neutral network model against test data: Prediction

```
bm_test$pred_prob = predict(object = nnet01, newdata = bm_test)
bm_test$pred = (bm_test$pred_prob > 0.5)*1
head(bm_test$pred)
```

```
##   [,1]
## 1    0
## 2    0
## 3    0
## 4    0
## 5    0
## 6    0
```

Contingency table comparing actual against predicted response:

```
t1 = table(bm_test$response, bm_test$pred)
row.names(t1) = c("Actual: 0", "Actual: 1")

colnames(t1) = c("Predicted: 0", "Predicted: 1")
t1 = addmargins(A = t1, FUN = list(Total = sum), quiet =TRUE);
t1
```

```
##
##              Predicted: 0 Predicted: 1 Total
##   Actual: 0         23885            1 23886
##   Actual: 1          2985            3  2988
##   Total             26870            4 26874
```

28. Which baseline model do we compare your neural network model against? Did it outperform the baseline according to accuracy? Baseline model: Accuracy (all negative model) = TAN/GT

```
round((23885/26874)*100, 2)
```

```
## [1] 88.88
```

Accuracy of neural network model: Accuracy = (TN+TP) / GT

```
NNet = round(((23885+3)/26874), 4)*100
NNet
```

```
## [1] 88.89
```

Answer: The neural network model performs similarly as the baseline at 88.89% accuracy.

29. Using the same predictors you used for your neural network model, build models to predict Response using the following algorithms:

   a. CART

```
library(rpart)
library(rpart.plot)
```

CART training model:

```
cart01 = rpart(formula = response ~ job + marital,
               data = bm_train, method = "class")
```

CART prediction:

```
X = data.frame(job = bm_test$job, marital = bm_test$marital)
predCART = predict(object = cart01, newdata = X, type = "class")
```

Contingency table:

```
t2 = table(bm_test$response, predCART)
row.names(t2) = c("Actual: 0", "Actual: 1")

colnames(t2) = c("Predicted: 0", "Predicted: 1")
t2 = addmargins(A = t2, FUN = list(Total = sum), quiet =TRUE);
t2
```

```
##            predCART
##             Predicted: 0 Predicted: 1 Total
##   Actual: 0        23886            0 23886
##   Actual: 1         2988            0  2988
##   Total            26874            0 26874
```

b. C5.0

```
library(C50)
```

C5.0 Training model:

```
C5 = C5.0(formula = response ~ job + marital,
           data = bm_train, control = C5.0Control(minCases=75))
```

C5.0 prediction:

```
predC5.0 = predict(object = C5, newdata = X)
```

Contingency table:

```
t3 = table(bm_test$response, predC5.0)
row.names(t3) = c("Actual: 0", "Actual: 1")

colnames(t3) = c("Predicted: 0", "Predicted: 1")
t3 = addmargins(A = t3, FUN = list(Total = sum), quiet =TRUE);
t3
```

```
##            predC5.0
##             Predicted: 0 Predicted: 1 Total
##   Actual: 0        23886            0 23886
##   Actual: 1         2988            0  2988
##   Total            26874            0 26874
```

c. Naïve Bayes

```
library(e1071)
```

Naive bayes training model:

```
nb01 = naiveBayes(formula = response ~ job + marital, data = bm_train)
nb01
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        no       yes
## 0.8888145 0.1111855
##
## Conditional probabilities:
##      job
## Y           admin. blue-collar entrepreneur   housemaid  management     retired
##   no  0.247132211 0.235744788  0.035250775 0.026752072 0.070334087 0.035669430
##   yes 0.285809906 0.140562249  0.024096386 0.023427041 0.069946452 0.097389558
##      job
## Y     self-employed    services     student  technician  unemployed     unknown
##   no    0.034539061 0.099639956 0.016913673 0.166289877 0.023988948 0.007745123
##   yes   0.031124498 0.067269076 0.064926372 0.155622490 0.031459170 0.008366801
##
##      marital
## Y         divorced     married      single     unknown
##   no  0.114837143 0.610357532 0.272712049 0.002093276
##   yes 0.104417671 0.538152610 0.355087015 0.002342704
```

Prediction:

```
ypred = predict(object = nb01, newdata = bm_test)
```

Contingency table:

```
t4 = table(bm_test$response, ypred)
row.names(t4) = c("Actual: 0", "Actual: 1")

colnames(t4) = c("Predicted: 0", "Predicted: 1")
t4 = addmargins(A = t4, FUN = list(Total = sum), quiet =TRUE);
t4
```

```
##              ypred
##               Predicted: 0 Predicted: 1 Total
##    Actual: 0         23886            0 23886
##    Actual: 1          2988            0  2988
##    Total             26874            0 26874
```

30. Compare the results of your neural network model with the three models from the previous exercise, according to the following criteria. Discuss in detail which model performed best and worst according to each criterion.

  a. Accuracy: Accuracy = (TN+TP) / GT

```
CART_A = round((23886+0) / 26874, 4)*100
C5.0_A = round((23886+0) / 26874, 4)*100
Nbayes_A = round((23886+0) / 26874, 4)*100
```

  b. Sensitivity: Recall = TP/TAP

```
Nnet_R = round((3) / 2988, 4)*100
CART_R = round((0) / 2988, 4)*100
C5.0_R = round((0) / 2988, 4)*100
NBayes_R = round((0) / 2988, 4)*100
```

  c. Specificity: Specificity = TN/TAN

```
Nnet_S = round((23885) / 23886, 4)*100
CART_S = round((23886) / 23886, 4)*100
C5.0_S = round((23886) / 23886, 4)*100
NBayes_S = round((23886) / 23886, 4)*100
```

Metric comparison table:

```
Table_c = matrix(c(NNet, Nnet_R, Nnet_S, CART_A, CART_R, CART_S, C5.0_A, C5.0_R,
                 C5.0_S, Nbayes_A, NBayes_R, NBayes_S), ncol=4, byrow=FALSE)

colnames(Table_c) = c("NNet", "CART", "C5.0", "NBayes")
rownames(Table_c) = c('Accuracy', 'Sensitivity', 'Specificity')

Table_c
```

```
##                NNet   CART   C5.0 NBayes
## Accuracy      88.89  88.88  88.88  88.88
## Sensitivity    0.10   0.00   0.00   0.00
## Specificity 100.00 100.00 100.00 100.00
```

Answer: All the models are performing relatively similar in terms of accuracy compared to the baseline. There's not much difference between specificity scores at 100% across the models. This suggest that all models are able to capture or predict the entire true negative values. Sensitivity is non-existent across the models with only the neutral network model able to detect 0.10% of true positive values. This suggests that none of the models are able to capture or predict the same proportion of positive values in the dataset. The training dataset will need to be re-balanced to represent each class in the response variable proportionately similar to real world scenario.

Data Science Using Python and R: Chapter 6

For Exercises 14–20, work with the adult_ch6_training and adult_ch6_test data sets. Use either Python or R to solve each problem.

19. Use random forests on the training data set to predict income using marital status and capital gains and losses.

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Prepare and import train and test dataset:

```
a_train = read.csv("adult_ch6_training", sep = ",")
colnames(a_train)[1] = "maritalStatus"
a_train$Income = factor(a_train$Income)
a_train$maritalStatus = factor(a_train$maritalStatus)

a_test = read.csv("adult_ch6_test", sep = ",")
colnames(a_test)[1] = "maritalStatus"
a_test$Income = factor(a_test$Income)
a_test$maritalStatus = factor(a_test$maritalStatus)
```

Predict income based on marital status and capital gains of train dataset:

```
rf01 = randomForest(formula = Income ~ maritalStatus + Cap_Gains_Losses,data = a_trai
n, ntree = 100, type ="classification")
a_train$pred = rf01$predicted
head(a_train)
```

```
##   maritalStatus Income Cap_Gains_Losses  pred
## 1 Never-married  <=50K          0.02174 <=50K
## 2      Divorced  <=50K          0.00000 <=50K
## 3       Married  <=50K          0.00000 <=50K
## 4       Married  <=50K          0.00000 <=50K
## 5       Married  <=50K          0.00000 <=50K
## 6       Married   >50K          0.00000 <=50K
```

20. Use random forests using the test data set that utilizes the same target and predictor variables.Does the test data result match the training data result?

Predict income based on marital status and capital gains of test dataset:

```
rf02 = randomForest(formula = Income ~ maritalStatus + Cap_Gains_Losses,data = a_test
, ntree = 100, type ="classification")
a_test$pred = rf02$predicted
head(a_test)
```

```
##   maritalStatus Income Cap_Gains_Losses  pred
## 1       Married  <=50K         0.000000 <=50K
## 2       Married   >50K         0.051781  >50K
## 3 Never-married  <=50K         0.000000 <=50K
## 4      Divorced   >50K         0.000000 <=50K
## 5       Married   >50K         0.000000 <=50K
## 6       Married  <=50K         0.000000 <=50K
```

Total number of predicted results in the test dataset:

```
table(a_test$pred)
```

```
##
## <=50K  >50K
##  5717   438
```

```
(426/(5729+426)*100)
```

```
## [1] 6.921202
```

Total number of predicted results in the train dataset:

```
table(a_train$pred)
```

```
##
## <=50K  >50K
## 17400  1361
```

```
(1350/(17411+1350)*100)
```

```
## [1] 7.195778
```

Answer: There's relatively the same at 7% percent of >50K predicted using both the training and test dataset. This suggest that the test data result match the train data result as the proportion of >50K and <=50K are the same based on job and marital variables.

Module 4 Assignment

Gabi Rivera || 20Nov2022 || ADS502-01

In [1]:
```
import os
os.getcwd()
```

Out[1]:
```
'/Users/gabirivera/Desktop/MSADS2/ADS502-01/Module4/Assignment'
```

In [25]:
```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
```

Data Science Using Python and R: Chapter 13

1. Create a logistic regression model to predict whether or not a customer has a store credit card, based on whether they have a web account and the days between purchases. Obtain the summary of the model.

Import train and test datasets:

In [7]:
```
cs_train = pd.read_csv("clothing_sales_training.csv", sep = ",")
cs_train.head()
```

Out[7]:

	CC	Days	Web	Sales per Visit
0	0	333.0	0	184.230000
1	0	171.5	0	38.500000
2	0	213.0	0	150.326667
3	1	71.4	1	104.240000
4	1	145.0	0	782.080000

In [6]:
```
cs_test = pd.read_csv("clothing_sales_test.csv", sep = ",")
cs_test.head()
```

Out[6]:

	CC	Days	Web	Sales per Visit
0	1	174.00	0	64.5000
1	1	87.62	0	105.7575
2	0	49.00	0	87.4400
3	0	72.50	0	60.0000
4	0	264.00	0	318.5000

Prepare X and y variables:

In [9]:
```
X = pd.DataFrame(cs_train[['Days', 'Web']])
X = sm.add_constant(X)
```

```
y = pd.DataFrame(cs_train[['CC']])
```

Create logistic regression:

In [12]:
```
logreg01 = sm.Logit(y, X).fit()
logreg01.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.655955
         Iterations 5
```

Out[12]:

Model:	Logit	Pseudo R-squared:	0.053
Dependent Variable:	CC	AIC:	1909.5825
Date:	2022-11-20 10:00	BIC:	1925.4226
No. Observations:	1451	Log-Likelihood:	-951.79
Df Model:	2	LL-Null:	-1004.9
Df Residuals:	1448	LLR p-value:	8.3668e-24
Converged:	1.0000	Scale:	1.0000
No. Iterations:	5.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	0.4962	0.0887	5.5968	0.0000	0.3224	0.6699
Days	-0.0037	0.0004	-8.4491	0.0000	-0.0046	-0.0028
Web	1.2537	0.3307	3.7914	0.0001	0.6056	1.9018

1. Are there any variables that should be removed from the model? If so, remove them and rerun the model.

Both variables will be included because there's no sign of multicollinearity amongst the variables. The z-values have reasonable scores with significantly small p-values less than 0.05 level of significance.

1. Write the descriptive form of the logistic regression model using the coefficients obtained from Question 1.

ln(CC) = 0.4962 -0.0037Days + 1.2537Web

1. Validate the model using the test data set.

In [13]:
```
X_test = pd.DataFrame(cs_test[['Days', 'Web']])
X_test = sm.add_constant(X_test)
y_test = pd.DataFrame(cs_test[['CC']])

logreg01_test = sm.Logit(y_test, X_test).fit()
logreg01_test.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.656885
         Iterations 5
```

Out[13]:

Model:	Logit	Pseudo R-squared:	0.052
Dependent Variable:	CC	AIC:	1838.7104

Date:	2022-11-20 10:02	BIC:	1854.4324
No. Observations:	1395	Log-Likelihood:	-916.36
Df Model:	2	LL-Null:	-966.40
Df Residuals:	1392	LLR p-value:	1.8534e-22
Converged:	1.0000	Scale:	1.0000
No. Iterations:	5.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	0.4634	0.0873	5.3105	0.0000	0.2924	0.6345
Days	-0.0035	0.0004	-8.2261	0.0000	-0.0043	-0.0026
Web	1.0973	0.2830	3.8780	0.0001	0.5427	1.6519

1. Obtain the predicted values of the response variable for each record in the data set.

In [ ]:
```
Predicted values fo test datast:
```

In [17]:
```
predictions_prob = logreg01.predict(X_test)
predictions_prob.head()
```

Out[17]:
```
0    0.463090
1    0.542853
2    0.578054
3    0.556706
4    0.382003
dtype: float64
```

In [ ]:
```
Convert predictions to binary results:
```

In [18]:
```
predictions = (logreg01.predict(X_test) > 0.5).astype(int)
predictions.head()
```

Out[18]:
```
0    0
1    1
2    1
3    1
4    0
dtype: int64
```

Data Science Using Python and R: Chapter 6

1. Use random forests on the training data set to predict income using marital status and capital gains and losses.

In [23]:
```
a_train = pd.read_csv("adult_ch6_training", sep = ",")
a_train.head()
```

Out[23]:

	Marital status	Income	Cap_Gains_Losses
0	Never-married	<=50K	0.02174
1	Divorced	<=50K	0.00000
2	Married	<=50K	0.00000
3	Married	<=50K	0.00000

4	Married	<=50K	0.00000

In [24]:
```
a_test = pd.read_csv("adult_ch6_test", sep = ",")
a_test.head()
```

Out[24]:

	Marital status	Income	Cap_Gains_Losses
0	Married	<=50K	0.000000
1	Married	>50K	0.051781
2	Never-married	<=50K	0.000000
3	Divorced	>50K	0.000000
4	Married	>50K	0.000000

Transform dependent variable to one-dimention:

In [53]:
```
y = a_train[['Income']]
rfy = np.ravel(y)

mar_np = np.array(a_train['Marital status'])
(mar_cat, mar_cat_dict) = stattools.categorical(mar_np, drop=True, dictnames = True)

mar_cat_pd = pd.DataFrame(mar_cat)
X = pd.concat((a_train[['Cap_Gains_Losses']], mar_cat_pd), axis = 1)
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tools/tools.py:1
52: FutureWarning: categorical is deprecated. Use pandas Categorical to represent catego
rical data and can get_dummies to construct dummy arrays. It will be removed after relea
se 0.13.
  warnings.warn(
```

Create random forrets train model:

In [54]:
```
rf01 = RandomForestClassifier(n_estimators = 100, criterion="gini").fit(X,rfy)
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py:
1858: FutureWarning: Feature names only support names that are all strings. Got feature
names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

In [56]:
```
rf01_d = rf01.predict(X)
a_train['pred'] = rf01_d.tolist()
a_train.head()
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py:
1858: FutureWarning: Feature names only support names that are all strings. Got feature
names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

Out[56]:

	Marital status	Income	Cap_Gains_Losses	pred
0	Never-married	<=50K	0.02174	<=50K
1	Divorced	<=50K	0.00000	<=50K
2	Married	<=50K	0.00000	<=50K
3	Married	<=50K	0.00000	<=50K
4	Married	<=50K	0.00000	<=50K

In [60]:
```
a_train['pred'].value_counts()
```

Out[60]:
```
<=50K    17375
>50K      1386
Name: pred, dtype: int64
```

In [68]:
```
(1386/(17375+1386)*100)
```

Out[68]:
```
7.387665902670433
```

1. Use random forests using the test data set that utilizes the same target and predictor variables.Does the test data result match the training data result?

Transform test dataset's dependent variable to one-dimention:

In [32]:
```
y2 = a_test[['Income']]
rfy2 = np.ravel(y2)

mar_np2 = np.array(a_test['Marital status'])
(mar_cat2, mar_cat_dict2) = stattools.categorical(mar_np2, drop=True, dictnames = True)

mar_cat_pd2 = pd.DataFrame(mar_cat2)
X2 = pd.concat((a_test[['Cap_Gains_Losses']], mar_cat_pd2), axis = 1)
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tools/tools.py:1
52: FutureWarning: categorical is deprecated. Use pandas Categorical to represent catego
rical data and can get_dummies to construct dummy arrays. It will be removed after relea
se 0.13.
  warnings.warn(
```

In [33]:
```
rf02 = RandomForestClassifier(n_estimators = 100, criterion="gini").fit(X2,rfy2)
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py:
1858: FutureWarning: Feature names only support names that are all strings. Got feature
names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

In [48]:
```
rf02_d = rf02.predict(X)
a_test['pred'] = rf02_d.tolist()
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py:
1858: FutureWarning: Feature names only support names that are all strings. Got feature
names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

In [49]:
```
a_test.head()
```

Out[49]:

	Marital status	Income	Cap_Gains_Losses	pred
0	Married	<=50K	0.000000	<=50K
1	Married	>50K	0.051781	>50K
2	Never-married	<=50K	0.000000	<=50K
3	Divorced	>50K	0.000000	<=50K
4	Married	>50K	0.000000	<=50K

In [59]:
```
a_test['pred'].value_counts()
```

Out[59]:
```
<=50K    5702
>50K      453
Name: pred, dtype: int64
```

In [67]:
```
(453/(5702+453)*100)
```
Out[67]:
```
7.359870024370431
```