|  |
| --- |
| > # Logistic Regression Analysis  > # ex) --> able to predict products in marketing  > getwd()  [1] "C:/Users/junhe/Documents"  > #importing the dataset  > dataset = read.csv("Social\_Network\_Ads.csv")  > head(dataset) #head,str to become friendly with the data  User.ID Gender Age EstimatedSalary Purchased  1 15624510 Male 19 19000 0  2 15810944 Male 35 20000 0  3 15668575 Female 26 43000 0  4 15603246 Female 27 57000 0  5 15804002 Male 19 76000 0  6 15728773 Male 27 58000 0  > str(dataset) # 'data.frame': 400 obs. of 5 variables  'data.frame': 400 obs. of 5 variables:  $ User.ID : int 15624510 15810944 15668575 15603246 15804002 15728773 15598044 15694829 15600575 15727311 ...  $ Gender : chr "Male" "Male" "Female" "Female" ...  $ Age : int 19 35 26 27 19 27 27 32 25 35 ...  $ EstimatedSalary: int 19000 20000 43000 57000 76000 58000 84000 150000 33000 65000 ...  $ Purchased : int 0 0 0 0 0 0 0 1 0 0 ...  > # Purchased = int--> factor  > View(dataset) # able to to see the whole dataset in a table format  > dataset = dataset[3:5] # choose Age, EstimatedSalary, Purchased[column no3,4,5]  > # can estimate something people wil buy based on Age and Salary  > head(dataset)  Age EstimatedSalary Purchased  1 19 19000 0  2 35 20000 0  3 26 43000 0  4 27 57000 0  5 19 76000 0  6 27 58000 0  > str(dataset)  'data.frame': 400 obs. of 3 variables:  $ Age : int 19 35 26 27 19 27 27 32 25 35 ...  $ EstimatedSalary: int 19000 20000 43000 57000 76000 58000 84000 150000 33000 65000 ...  $ Purchased : int 0 0 0 0 0 0 0 1 0 0 ...  > #Encoding the target feature as factor ( Change data type of elements for further analysis)  > dataset$Purchased = factor(dataset$Purchased,levels = c(0,1)) #Access to the variable purchased and change it from int to factor  > library(caTools)  > set.seed(123)  > split = sample.split(dataset$Purchased,SplitRatio=0.75) #75% of the whole data goes to the training set  > training\_set = subset(dataset,split==TRUE) # 75% training set  > test\_set = subset(dataset,split==FALSE) #25% test set  > head(training\_set)  Age EstimatedSalary Purchased  1 19 19000 0  3 26 43000 0  6 27 58000 0  7 27 84000 0  8 32 150000 1  10 35 65000 0  > head(test\_set)  Age EstimatedSalary Purchased  2 35 20000 0  4 27 57000 0  5 19 76000 0  9 25 33000 0  12 26 52000 0  18 45 26000 1  > #Feature scaling --> scale and normalize the dataset to make the appropriate model  > training\_set[,1:2] = scale(training\_set[,1:2])  > test\_set[,1:2] = scale(test\_set[,1:2])  > training\_set[,1:2]  Age EstimatedSalary  1 -1.76554750 -1.47334137  3 -1.09629664 -0.78837605  6 -1.00068938 -0.36027273  7 -1.00068938 0.38177303  8 -0.52265305 2.26542765  10 -0.23583125 -0.16049118  11 -1.09629664 0.26761214  13 -1.66994024 0.43885347  14 -0.52265305 -1.50188159  15 -1.86115477 0.32469259  16 -0.80947485 0.26761214  17 0.91145593 -1.30210004  21 0.72024140 -1.38772071  23 1.00706320 -0.84545650  24 0.72024140 -1.38772071  25 0.81584866 -1.35918049  26 0.91145593 -1.44480115  27 1.10267046 -1.21647938  28 0.91145593 -1.15939893  30 -0.61826032 -1.50188159  31 -0.61826032 0.09637081  33 -1.57433297 -1.55896204  36 -0.23583125 -1.24501960  37 -0.42704579 -1.21647938  39 -1.09629664 0.03929037  40 -1.00068938 -1.13085871  41 -1.00068938 -1.53042182  42 -0.42704579 -0.56005428  43 -0.23583125 1.06673835  44 -0.71386758 -1.58750226  47 -1.19190391 0.23907192  49 -0.71386758 1.83732433  50 -0.61826032 0.52447414  51 -1.28751117 -1.10231849  53 -0.80947485 0.35323281  54 -0.23583125 -1.35918049  55 -1.00068938 -0.36027273  56 -1.28751117 -0.44589340  57 -1.38311844 -0.64567495  58 -0.90508211 0.23907192  59 -1.47872570 -1.50188159  60 -0.52265305 1.32360034  61 -1.00068938 -1.44480115  62 -1.19190391 0.46739370  63 -1.38311844 -0.13195096  64 -0.52265305 1.40922101  65 2.05874311 0.35323281  67 -1.28751117 -1.47334137  68 -1.38311844 0.32469259  70 -0.61826032 -0.07487051  71 -1.19190391 0.26761214  72 -1.28751117 -1.24501960  73 -1.66994024 -1.35918049  76 -0.33143852 1.18089923  77 -1.86115477 -0.53151406  78 -1.47872570 -1.24501960  79 -0.90508211 0.46739370  80 -1.09629664 -1.53042182  81 -0.71386758 0.26761214  83 -1.66994024 -0.61713472  88 -0.90508211 0.41031325  90 -0.23583125 -0.58859450  91 -1.47872570 0.29615237  92 -0.71386758 1.29506012  93 -1.09629664 -1.58750226  94 -0.80947485 -1.21647938  95 -0.80947485 0.35323281  96 -0.23583125 -0.75983583  97 -0.23583125 -1.30210004  98 -0.90508211 1.49484167  99 -0.23583125 0.06783059  100 -0.90508211 -0.95961738  101 -1.00068938 0.49593392  102 -0.90508211 -0.33173251  105 -1.76554750 -1.41626093  106 -1.57433297 0.03929037  110 0.05099054 0.26761214  111 0.14659781 0.01075015  112 -0.04461672 0.01075015  113 0.05099054 -0.27465207  114 -0.04461672 -0.44589340  115 0.43341960 0.26761214  116 0.24220507 -0.38881295  118 -0.14022399 -0.53151406  119 0.24220507 -0.33173251  120 0.33781234 -0.33173251  121 -0.14022399 0.12491104  122 -0.04461672 0.03929037  123 0.24220507 0.12491104  125 0.33781234 -0.56005428  128 -1.09629664 -1.10231849  129 -0.71386758 -1.53042182  130 -1.09629664 0.38177303  132 -0.42704579 -1.13085871  133 -0.71386758 0.46739370  135 -0.90508211 -0.44589340  136 -1.38311844 -0.21757162  137 -1.66994024 0.32469259  138 -0.71386758 1.03819813  140 -1.76554750 -1.30210004  141 -1.76554750 0.41031325  142 -1.86115477 -0.07487051  143 -0.23583125 -0.33173251  144 -0.71386758 0.52447414  145 -0.33143852 -1.30210004  146 -1.28751117 0.52447414  147 -1.00068938 0.72425569  149 -0.80947485 -0.27465207  150 -1.66994024 0.09637081  151 -1.09629664 -1.58750226  152 0.33781234 -0.73129561  153 -0.61826032 0.15345126  155 0.24220507 -0.67421517  157 0.81584866 -0.33173251  158 -0.80947485 0.12491104  160 -0.52265305 1.83732433  161 -0.52265305 0.83841658  164 -0.23583125 -0.93107716  165 -0.42704579 -0.04633029  166 -1.86115477 0.43885347  167 -1.47872570 -0.44589340  168 -0.23583125 0.01075015  169 -0.80947485 2.20834721  171 -1.57433297 0.49593392  172 -0.33143852 1.26651990  173 -1.09629664 1.35214056  174 -0.33143852 -0.78837605  177 -0.23583125 -0.67421517  178 -1.19190391 -1.38772071  179 -1.28751117 -1.35918049  180 -0.61826032 -1.04523805  181 -1.09629664 -1.55896204  182 -0.61826032 0.01075015  183 -0.52265305 1.32360034  184 -0.42704579 -0.78837605  185 -0.42704579 -0.30319229  186 -0.61826032 -0.13195096  187 -1.66994024 0.32469259  188 -0.42704579 -0.84545650  189 -0.23583125 0.03929037  190 -0.90508211 -1.10231849  191 -1.28751117 0.38177303  192 -1.76554750 -1.27355982  194 -1.76554750 -0.01779007  195 -0.90508211 0.52447414  196 -0.33143852 -0.78837605  197 -0.71386758 0.23907192  198 -1.66994024 -0.98815761  201 -0.23583125 -0.90253694  202 1.10267046 0.09637081  203 0.14659781 1.80878411  204 0.33781234 0.01075015  205 1.96313585 0.86695680  206 0.91145593 -0.67421517  207 1.67631405 1.69462322  209 0.24220507 2.03710588  210 0.81584866 -1.38772071  211 1.00706320 0.72425569  212 1.38949226 2.26542765  214 -0.23583125 -0.36027273  215 0.91145593 -0.78837605  216 2.15435038 1.06673835  217 1.10267046 -0.16049118  218 0.24220507 0.21053170  219 0.81584866 0.72425569  220 2.05874311 2.06564610  221 0.33781234 0.26761214  222 -0.23583125 0.58155458  223 -0.04461672 2.09418633  225 -0.23583125 -0.30319229  227 -0.14022399 1.58046234  231 -0.23583125 2.17980699  232 0.14659781 -0.81691628  233 0.24220507 1.03819813  235 0.05099054 1.18089923  238 -0.04461672 0.26761214  240 1.48509952 2.06564610  242 0.05099054 -0.33173251  243 1.19827773 0.49593392  244 1.77192132 0.95257746  245 0.33781234 0.03929037  246 1.29388499 2.15126677  247 -0.23583125 -0.58859450  248 1.86752858 1.46630145  249 0.33781234 -0.53151406  250 -0.23583125 0.75279591  251 0.62463413 -0.90253694  252 -0.04461672 -0.53151406  253 1.00706320 1.80878411  254 -0.04461672 2.15126677  256 1.38949226 0.55301436  257 0.33781234 0.03929037  258 0.24220507 -0.38881295  259 1.96313585 0.69571547  260 0.72024140 1.72316344  261 -0.23583125 0.18199148  262 -0.14022399 2.09418633  263 1.67631405 1.55192212  267 0.24220507 0.12491104  268 -0.04461672 0.09637081  269 0.91145593 2.09418633  270 0.24220507 -0.27465207  271 0.52902687 1.78024389  272 2.05874311 0.15345126  275 1.86752858 -1.27355982  276 1.86752858 0.09637081  277 0.05099054 0.01075015  278 1.10267046 0.49593392  279 1.38949226 -0.93107716  280 1.19827773 -0.98815761  282 -0.23583125 -0.27465207  283 -0.04461672 -0.01779007  284 1.38949226 -1.41626093  285 1.00706320 2.00856566  287 -0.04461672 -0.24611184  288 1.00706320 1.92294500  289 0.33781234 0.23907192  290 -0.04461672 0.21053170  291 0.14659781 1.80878411  293 1.67631405 -0.90253694  294 -0.04461672 0.18199148  295 -0.23583125 -0.38881295  296 -0.14022399 -0.21757162  297 0.43341960 0.06783059  298 0.52902687 1.18089923  300 0.81584866 1.32360034  301 1.96313585 -0.93107716  303 -0.04461672 1.89440477  304 -0.04461672 0.23907192  306 0.43341960 -0.47443362  308 0.91145593 1.20943946  309 -0.14022399 1.55192212  311 0.43341960 -0.01779007  312 0.14659781 0.72425569  313 0.05099054 -0.58859450  314 1.10267046 2.00856566  315 0.14659781 0.23907192  317 1.58070679 0.95257746  318 -0.23583125 -0.44589340  319 0.72024140 -1.10231849  320 -0.14022399 -0.30319229  321 1.38949226 1.92294500  322 1.48509952 0.32469259  323 0.33781234 -0.53151406  325 1.00706320 1.72316344  327 0.33781234 0.03929037  328 0.43341960 0.12491104  329 -0.14022399 1.35214056  330 0.91145593 1.03819813  331 0.05099054 -0.56005428  333 0.43341960 -0.16049118  334 0.24220507 -0.16049118  335 1.86752858 -0.30319229  336 -0.14022399 -0.47443362  337 1.96313585 2.09418633  338 -0.23583125 0.23907192  340 0.14659781 1.46630145  342 -0.23583125 0.12491104  344 0.91145593 -0.56005428  345 0.91145593 0.98111768  346 0.33781234 -0.21757162  348 1.58070679 1.06673835  349 0.14659781 0.18199148  350 0.05099054 -0.27465207  351 0.05099054 1.20943946  352 -0.04461672 0.12491104  354 -0.04461672 -0.38881295  355 -0.14022399 0.80987635  356 2.15435038 -1.04523805  357 1.58070679 -0.01779007  358 0.33781234 0.03929037  359 0.24220507 0.01075015  360 0.43341960 -0.47443362  361 0.52902687 1.66608300  362 1.48509952 -1.04523805  365 0.43341960 0.95257746  366 2.05874311 -1.18793916  370 1.58070679 -1.27355982  371 2.15435038 -0.70275539  374 2.05874311 1.69462322  375 -0.04461672 0.26761214  376 0.81584866 -1.10231849  377 0.81584866 0.09637081  378 0.43341960 -0.50297384  379 0.33781234 0.46739370  381 0.43341960 -0.18903140  382 1.00706320 -1.07377827  384 1.10267046 -1.21647938  385 1.86752858 -1.07377827  386 1.77192132 -0.30319229  387 1.10267046 -0.90253694  388 0.14659781 0.01075015  390 1.00706320 -1.01669783  391 1.00706320 -1.07377827  393 0.72024140 -0.73129561  394 2.15435038 -0.81691628  396 0.81584866 -0.84545650  397 1.29388499 -1.35918049  398 1.19827773 -1.44480115  399 -0.14022399 -1.07377827  > # Fitting the logistic Regression to the Training set  > classifier = glm(formula = Purchased ~.,family=binomial,data=training\_set)  > #Predicting the test set results  > prob\_pred = predict(classifier,type='response',newdata = test\_set[-3])  > #test\_set[-3] = test\_set[,1:2]. use test set to assess the model  > View(prob\_pred)  > y\_pred = ifelse(prob\_pred >0.5,1,0) # Target value Y is between 0~0.5~1  > y\_pred  2 4 5 9 12 18 19 20 22 29 32 34 35 38 45 46 48 52 66 69 74 75 82 84  0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  85 86 87 89 103 104 107 108 109 117 124 126 127 131 134 139 148 154 156 159 162 163 170 175  0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  176 193 199 200 208 213 224 226 228 229 230 234 236 237 239 241 255 264 265 266 273 274 281 286  0 0 0 0 1 1 1 0 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 0  292 299 302 305 307 310 316 324 326 332 339 341 343 347 353 363 364 367 368 369 372 373 380 383  1 1 1 0 1 0 0 0 0 1 0 1 0 1 1 1 1 1 1 0 1 0 1 1  389 392 395 400  0 0 0 1  > #Making the Confusion Matrix -> assessment of the built model  > cm = table(test\_set[,3],y\_pred) #compare the real one and prediction  > cm # 83% accuracy(57+26), 17% error(10+7)  y\_pred  0 1  0 57 7  1 10 26  > (cm1 = table(test\_set[,3],y\_pred>0.5)) # boolian form    FALSE TRUE  0 57 7  1 10 26  > View(test\_set)  > #Visualizing the Training set results  > library(ElemStatLearn)  > set = training\_set  > X1 = seq(min(set[,1]) -1,max(set[,1]) +1,by=0.01) #age. use 1st column in all rows. by is interval. +1 : not to sketch(?) in the graph(Cannot understand pronunciation in video 10.3 12:55) . // whole length by interval. This depends on dataset in analysis  > X2 = seq(min(set[,2]) -1,max(set[,2]) +1,by=0.01) #estimatedSalary  > grid\_set = expand.grid(X1,X2) # drawing basic graph  > colnames(grid\_set) = c('Age','EstimatedSalary')  > prob\_set = predict(classifier, type = 'response', newdata = grid\_set)  > y\_grid = ifelse(prob\_set > 0.5,1,0) # seperate by two to predict target group  > plot(set[,-3],main = 'Logistic Regression(Training set)',xlab = 'Age', ylab = 'Estimated Salary',xlim = range(X1),ylim = range(X2))  > contour(X1,X2,matrix(as.numeric(y\_grid),length(X1),length(X2)),add=TRUE)# make x,y scale of the graph  > points(grid\_set,pch='.',col=ifelse(y\_grid==1,'springgreen3','tomato'))  > points(set,pch=21,bg=ifelse(set[,3]==1,'green4','red3')) # distinguish by color  > # Visualizing the Test set results  > library(ElemStatLearn)  > set = test\_set  > X1 = seq(min(set[,1]) -1,max(set[,1]) +1,by=0.01)  > X2 = seq(min(set[,2]) -1,max(set[,2]) +1,by=0.01)  > grid\_set = expand.grid(X1,X2)  > colnames(grid\_set) = c('Age','EstimatedSalary')  > prob\_set = predict(classifier, type = 'response', newdata = grid\_set)  > y\_grid = ifelse(prob\_set > 0.5,1,0)  > plot(set[,-3],main = 'Logistic Regression(Test set)',xlab = 'Age', ylab = 'Estimated Salary',xlim = range(X1),ylim = range(X2))  > contour(X1,X2,matrix(as.numeric(y\_grid),length(X1),length(X2)),add=TRUE)  > points(grid\_set,pch='.',col=ifelse(y\_grid==1,'springgreen3','tomato'))  > points(set,pch=21,bg=ifelse(set[,3]==1,'green4','red3')) |
|  |
| |  | | --- | |  | |