coursework10 : HW10 (17.04) - Machine Learning III, Regression analysis

Fortunat Mutunda

April 12, 2016

1. Take the following data and simulate K-NN algorithm to predict the class probabilities of points (3, 5) and (4, 6). Report the probabilities with K=1, K=2 and K=3.

- I will remove the id first because it does not give any useful information, therefore it is not going to help with anything

```
str(dataf)
```

```
## 'data.frame':    10 obs. of  4 variables:
##  $ ID     : int  1 2 3 4 5 6 7 8 9 10
##  $ x_coord: int  9 2 3 4 1 3 5 6 6 3
##  $ y_coord: int  3 4 3 1 6 9 6 4 2 7
##  $ class  : int  1 1 1 1 1 0 0 0 0 0
```

```
dataf = dataf[-1]
```

Now we see that we have coordinates x and y and each point has been classified either as 1 or 0, therefore I am going to change class into factor because that's the column I am going to use to classify all the observations.

```
set.seed(9850)
gp = runif(nrow(dataf))
dataf = dataf[order(gp),]
#dataf
dataf$class = as.factor(dataf$class)
str(dataf)
```

```
## 'data.frame':    10 obs. of  3 variables:
##  $ x_coord: int  5 6 1 4 6 3 3 9 3 2
##  $ y_coord: int  6 2 6 1 4 3 7 3 9 4
##  $ class  : Factor w/ 2 levels "0","1": 1 1 2 2 1 2 1 2 1 2
```

now we can see that there is an even classification of 5 points which belong to 0 and 5 points which belong to 1.

Before I normalize the rest I am going to first randomazie everything because all the data are clissified with 5 ones then 5 zeroes I'll make it to show 0s and 1s randomly.

```
head(dataf,4)
```

```
##   x_coord y_coord class
## 7       5       6     0
## 9       6       2     0
## 5       1       6     1
## 4       4       1     1
```

Now that is done, let's proceed into normalizing the data.

```
normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x))) }
dataf_n <- as.data.frame(lapply(dataf[1:2], normalize))
```

I have just normalized x and y because class is the classifier, after doing so I can proceed into training the data

- create a training dataframe

```
#CrossTable(dataf$x_coord, dataf$y_coord)
# taining and testing on given data to see if my formula works
dataf_train = dataf_n[1:8,]
dataf_test = dataf_n[9:10,]
# then I pick the training taget which will be taken from the original dataset
dataf_train_target = dataf [1:8,3]
dataf_test_target = dataf[9:10, 3]
# done now I will try the knn to see if it works
#model1 is the trained data
model1 = knn(train = dataf_train ,test = dataf_test,cl = dataf_train_target,k = 3 )
model1
```

```
## [1] 0 1
## Levels: 0 1
```

- Now that I am sure that my trained model works I am now going to answer the question for the points (3, 5) and (4, 6) for k = 1,2,3

```
## Warning in read.table(file = file, header = header, sep = sep, quote =
## quote, : incomplete final line found by readTableHeader on 'knn_test.csv'
```

```
## [1] 1 0
## Levels: 0 1
```

```
## [1] 1 0
## Levels: 0 1
```

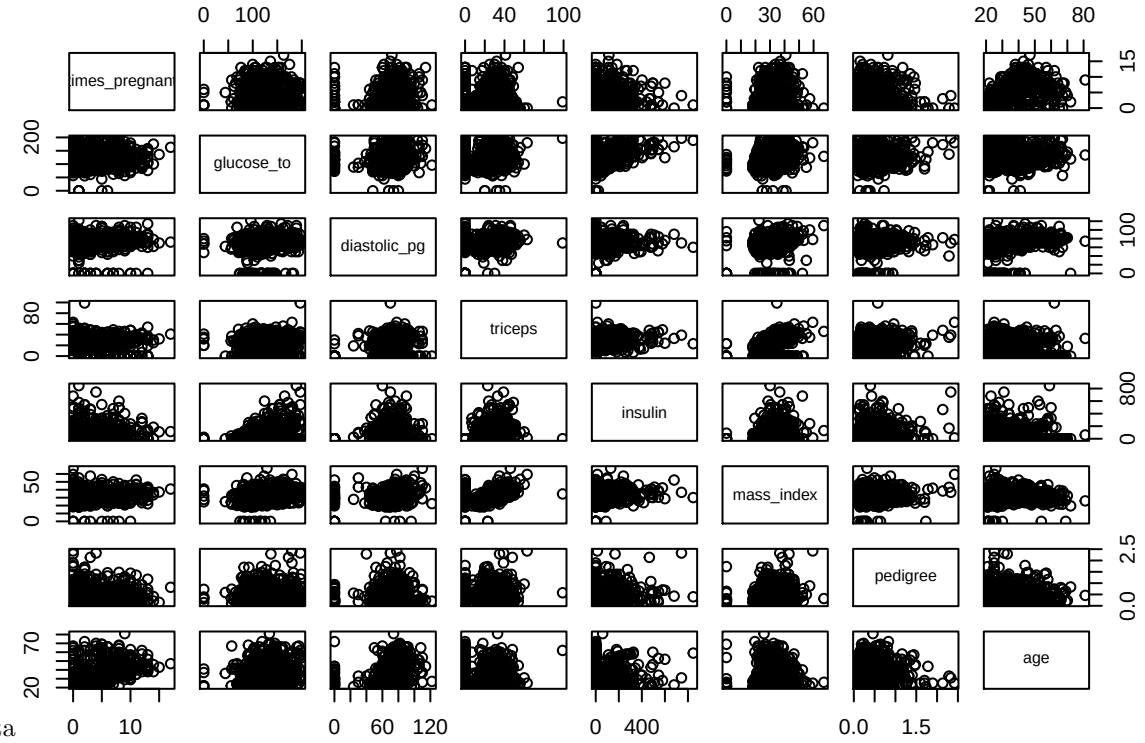
```
## [1] 1 0
## Levels: 0 1
```

Conclusion: for the three measure when k =1 , 2 ,3 the result for the two given points are respectively 1-0 , 1-0 and 1-0.

2. In this task we use diabetes dataset to predict diabetes.

- Split the data randomly on 80% of training and 20% for testing.
- Fit the logistic regression on the training set to predict the class.

- Interpret the model. How the Plasma glucose concentration impacts the odds ratio of having diabetes. What about diabetes pedigree function? Which features do not affect (significantly) the risk of having diabetes?
- Now compute Accuracy, Precision, Recall and F1 score on the test set.



- load the given data
- split the data into two, training and testing data

```
diabetes_training = head(diabetes, 614)
diabetes_testing = tail(diabetes, 154 )
#class_testing = tail(diabetes$class,154)
```

- fit a logistic regression model using training data

```
diabetes_model = glm(class ~.,family = binomial (link='logit') , data = diabetes_training)
anova(diabetes_model, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: class
##
## Terms added sequentially (first to last)
##
##
##                Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                             613     792.69
## times_pregnant  1   26.960       612     765.73 2.077e-07 ***
```

```
## glucose_to      1  132.089       611     633.64 < 2.2e-16 ***
## diastolic_pg    1    0.319       610     633.32 0.5723904
## triceps         1    2.985       609     630.33 0.0840487 .
## insulin         1    0.650       608     629.68 0.4200697
## mass_index      1   42.535       607     587.15 6.942e-11 ***
## pedigree        1   10.944       606     576.20 0.0009391 ***
## age             1    0.834       605     575.37 0.3609844
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- use the fitted model to do prediction for the test data

```
#prediction  = predict(diabetes_model,newdata = diabetes_testing)
#table(prediction)

fitted.results <- predict(diabetes_model,diabetes_testing,type='response')

fitted.results <- ifelse(fitted.results > 0.5,1,0)
misClasificError <- mean(fitted.results != diabetes_testing$class)
print(paste('Accuracy',1-misClasificError))
```

```
## [1] "Accuracy 0.772727272727273"
```

```
table(fitted.results)
```

```
## fitted.results
##   0   1
## 114  40
```

3. Run K-NN on the same data (also using the same setup) to predict diabetes. Try different K's (K=1, K=3). Report the same scores as before (for each K value). Compare the models with F score. Which model has better Accuracy and F score? (logistic or KNN K1 or KNN K3). Optional: plot also roc curves to compare.

```
gp = runif(nrow(diabetes))
diabetes = diabetes[order(gp),]
diabetes$class = as.factor(diabetes$class)
str(diabetes)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ times_pregnant: int  0 5 5 1 3 12 4 1 4 7 ...
##  $ glucose_to    : int  108 166 77 107 78 92 189 153 154 105 ...
##  $ diastolic_pg  : int  68 76 82 68 50 62 110 82 72 0 ...
##  $ triceps       : int  20 0 41 19 32 7 31 42 29 0 ...
##  $ insulin       : int  0 0 42 0 88 258 0 485 126 0 ...
##  $ mass_index    : num  27.3 45.7 35.8 26.5 31 27.6 28.5 40.6 31.3 0 ...
##  $ pedigree      : num  0.787 0.34 0.156 0.165 0.248 0.926 0.68 0.687 0.338 0.305 ...
##  $ age           : int  32 27 35 24 26 44 37 23 37 24 ...
##  $ class         : Factor w/ 2 levels "0","1": 1 2 1 1 2 2 1 1 1 1 ...
```

```
diabetes_n <- as.data.frame(lapply(diabetes[1:8], normalize))
diabetes_train = diabetes_n[1:614,]
diabetes_test = diabetes_n[615:768,]

diabetes_train_target = diabetes [1:614,9]
diabetes_test_target = diabetes[615:768, 9]
modelc = knn(train = diabetes_train ,test = diabetes_test,cl = diabetes_train_target,k = 3 )
table(modelc)
```

```
## modelc
##   0   1
## 103  51
```

```
modelc = knn(train = diabetes_train ,test = diabetes_test,cl = diabetes_train_target,k = 1)
table(modelc)
```

```
## modelc
##   0   1
## 100  54
```

4. In this task we are using diamonds data from the package ggplot2 (data(diamonds)). Build regression models predicting price from the rest of the features, where

A) model 1 has all the features
B) model 2 has all the features + 'carat' and 'depth' of degree 2
C) model 3 has all the features + 3rd degree polynomials of 'carat' and 'depth' (i.e. carat^3, carat^2, carat, depth^3,...)
D) model 4 has all the features + 3rd degree polynomials of 'carat' and 'depth' + 'x','y','z' of degree 2 in R you can use poly(x,d) to evaluate a polynomial of degree d, e.g. lm(price ~ poly(x,3) + ..., data=diamonds) Use the regular 80% train / 20% test split. Measure the RMSE for all the models on the train and test set and plot a graph, where on x-axis models are sorted according to the complexity of the model and on y-axis RMSE for train and test split. What do you observe? Can we diagnose under- or overfitting problems?

5. We have prepared a small Kaggle competition, wherein your task is to apply the regression techniques you have learnt to predict the numerical value of the target variable in the test set. The competition is to be done individually. As you make submissions, you will immediately see how they evaluate (in terms of RMSE) on the public leaderboard. Everyone who makes the first submission before the next week practice session, will be awarded one point. A more detailed description is available on the competition page. Use this link to join the competition.

6. (bonus, until April 30!) For the above competition, students who will finish in the first quaRtile (i.e top 25%) will get 4 extra points, 2nd quartile - 2 extra points, 3rd - 1 point. Important! To avoid overfitting on the public data, kaggle performs the evaluation only on the 50% of the test set. The predictions on the remaining 50% will be used after the competition closes, for the final evaluation.

7. (optional bonus, 1p). Try ridge and lasso regressions for model 4 from task 4 and add the resulting RMSE of training and test set on the plot generated in task 4. Did it help?