pca.R

Magilan

Mon Oct 08 15:05:45 2018

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.0.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(boot)
library(forecast)
library(tseries)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:boot':
##
##     logit
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
library(devtools)
library(ggbiplot)
```

```
## Loading required package: plyr
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:purrr':
##
##     compact
```

```
## Loading required package: scales
```

```
##
## Attaching package: 'scales'
```

```
## The following objects are masked from 'package:psych':
##
##     alpha, rescale
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
## Loading required package: grid
```

```r
library(sp)
library(class)


data <- read.csv("C:/Users/Magilan/Desktop/ML_project/austin_weather.csv",header = TRUE)
data1=na.omit(data,invert=FALSE)
attach(data1)

# Principal Component analysis

pc = prcomp(data1[,-c(1,20,21,22)],
            center=TRUE,
            scale. = TRUE)
pc$center
```

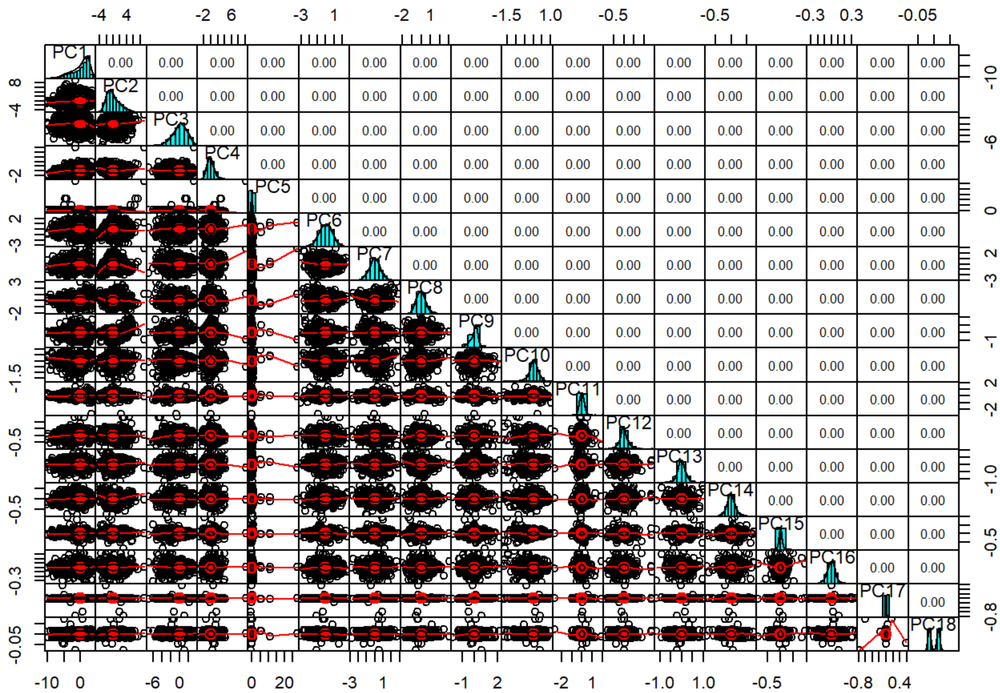
```
##                   TempHighF                   TempAvgF
##                   80.792337                  70.557854
##                    TempLowF              DewPointHighF
##                   59.819923                  61.516475
##                DewPointAvgF               DewPointLowF
##                   56.636782                  50.944061
##         HumidityHighPercent         HumidityAvgPercent
##                   87.833716                  66.662835
##          HumidityLowPercent SeaLevelPressureHighInches
##                   44.983908                  30.112337
##   SeaLevelPressureAvgInches  SeaLevelPressureLowInches
##                   30.022835                  29.931609
##         VisibilityHighMiles          VisibilityAvgMiles
##                    9.991571                   9.162452
##          VisibilityLowMiles                WindHighMPH
##                    6.842912                  13.245211
##                  WindAvgMPH                WindGustMPH
##                    5.009195                  21.383908
```

```r
summary(pc)
```

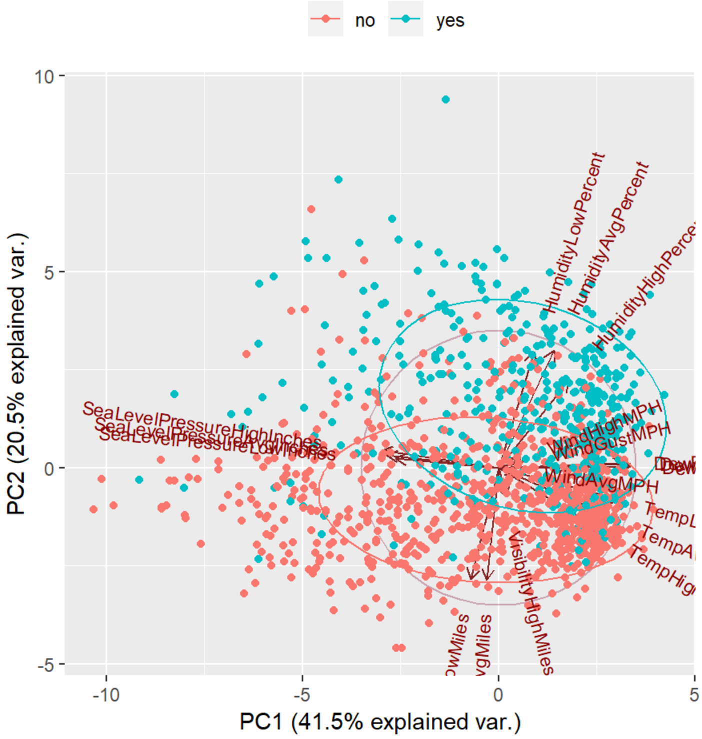
```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     2.7336 1.9211 1.6574 1.09700 0.98168 0.81429
## Proportion of Variance 0.4152 0.2050 0.1526 0.06686 0.05354 0.03684
## Cumulative Proportion  0.4152 0.6202 0.7728 0.83967 0.89321 0.93005
##                            PC7    PC8     PC9    PC10    PC11    PC12
## Standard deviation     0.69570 0.5597 0.40987 0.31203 0.25950 0.21910
## Proportion of Variance 0.02689 0.0174 0.00933 0.00541 0.00374 0.00267
## Cumulative Proportion  0.95693 0.9743 0.98367 0.98908 0.99282 0.99549
##                           PC13    PC14    PC15    PC16    PC17    PC18
## Standard deviation     0.20639 0.14985 0.09830 0.07064 0.03619 0.01485
## Proportion of Variance 0.00237 0.00125 0.00054 0.00028 0.00007 0.00001
## Cumulative Proportion  0.99785 0.99910 0.99964 0.99992 0.99999 1.00000
```

```r
# Orthogonality of PC

pairs.panels(pc$x,gap=0,pch=21)
```



```
g <- ggbiplot(pc,
              obs.scale = 1,
              var.scale = 1,
              groups = data1$Rain,
              ellipse = TRUE,
              circle = TRUE,
              ellipse.prob = 0.68)
g <- g + scale_color_discrete(name = '')
g <- g + theme(legend.direction = 'horizontal',
               legend.position = 'top')
print(g)
```



```r
pc.df=data.frame(pc$x)

index <- createDataPartition(Rain, p = 0.7, list = FALSE)
# Training set
train.df <- pc.df[index,]
train.Y = data1[index,22]
train.Y1 = data1[index,21]
train = cbind(train.df,train.Y)

# Testing dataset
test.df <- pc.df[-index,]
test.Y = data1[-index,22]
test.Y1 =data1[-index,21]
test = cbind(test.df,test.Y)

# Logistic Regression With PCA

model <- glm(train$train.Y ~. , data = train)
summary(model)
```

```
##
## Call:
## glm(formula = train$train.Y ~ ., data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0002  -0.1913  -0.0299   0.1747   0.9579
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.341247   0.011161  30.575  < 2e-16 ***
## PC1          0.025835   0.004132   6.253 6.23e-10 ***
## PC2          0.148393   0.005669  26.178  < 2e-16 ***
## PC3         -0.039171   0.006708  -5.839 7.33e-09 ***
## PC4         -0.020073   0.010145  -1.979  0.04817 *
## PC5         -0.011699   0.010288  -1.137  0.25578
## PC6         -0.088247   0.014063  -6.275 5.43e-10 ***
## PC7         -0.032025   0.016442  -1.948  0.05176 .
## PC8          0.147705   0.020477   7.213 1.16e-12 ***
## PC9         -0.142736   0.027550  -5.181 2.73e-07 ***
## PC10        -0.078385   0.035364  -2.217  0.02691 *
## PC11        -0.011828   0.042141  -0.281  0.77903
## PC12        -0.195926   0.050709  -3.864  0.00012 ***
## PC13         0.229570   0.053763   4.270 2.16e-05 ***
## PC14        -0.036846   0.072598  -0.508  0.61191
## PC15        -0.033466   0.118913  -0.281  0.77844
## PC16        -0.040601   0.157613  -0.258  0.79678
## PC17        -0.118198   0.266033  -0.444  0.65693
## PC18        -0.578076   0.745721  -0.775  0.43843
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.113517)
##
##     Null deviance: 205.93  on 914  degrees of freedom
## Residual deviance: 101.71  on 896  degrees of freedom
## AIC: 626.6
##
## Number of Fisher Scoring iterations: 2
```

```r
predicted_values <- predict(model, test.df, type = "response")
head(predicted_values)
```

```
##            1            3           22           24           28
##  0.849981638 -0.071333565  0.276149681  0.331586724  0.140525906
##           32
## -0.008151737
```

```
#Vlaidation
table(Rain)
```

```
## Rain
##  no yes
## 859 446
```

```
nrows_prediction<-nrow(test.df)
prediction <- data.frame(c(1:nrows_prediction))
colnames(prediction) <- c("Rain")
str(prediction)
```

```
## 'data.frame':    390 obs. of  1 variable:
##  $ Rain: int  1 2 3 4 5 6 7 8 9 10 ...
```

```
prediction$Rain <- as.character(prediction$Rain)
prediction$Rain <- "yes"
prediction$Rain[ predicted_values < 0.5] <- "no"
prediction$Rain <- as.factor(prediction$Rain)

#Confusion Matrix

table(prediction$Rain, test.Y1)
```

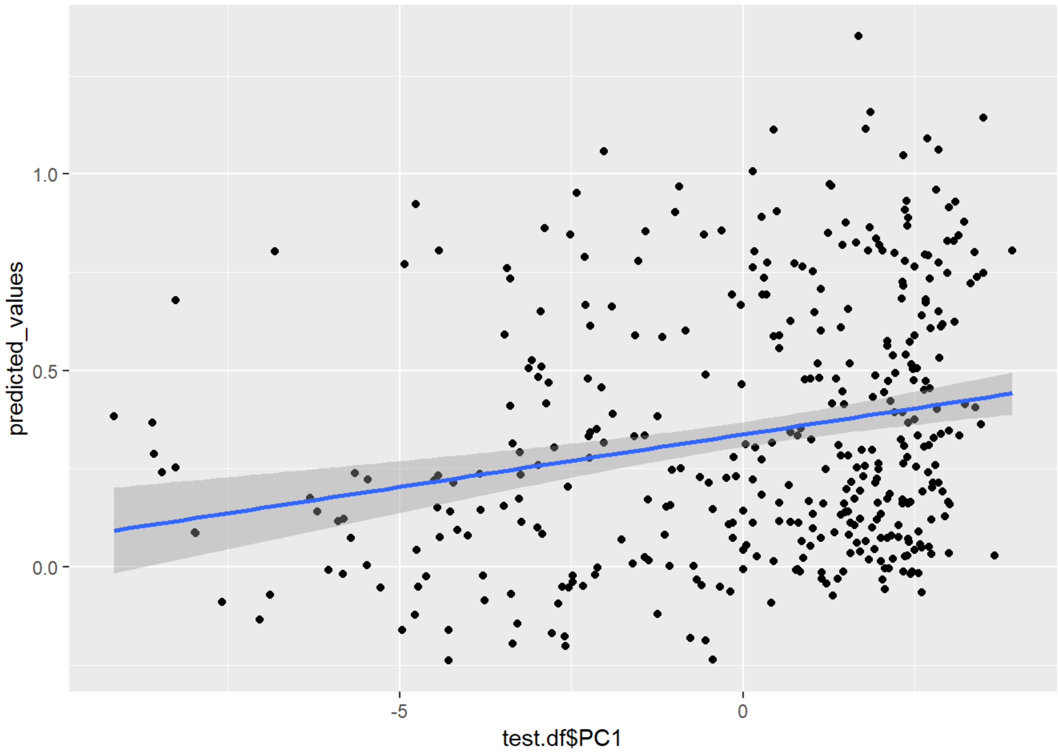
```
##      test.Y1
##        no yes
##   no  232  40
##   yes  25  93
```

```
confusionMatrix(prediction$Rain,test.Y1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  232  40
##        yes  25  93
##
##                Accuracy : 0.8333
##                  95% CI : (0.7926, 0.869)
##     No Information Rate : 0.659
##     P-Value [Acc > NIR] : 1.045e-14
##
##                   Kappa : 0.6188
##  Mcnemar's Test P-Value : 0.08248
##
##             Sensitivity : 0.9027
##             Specificity : 0.6992
##          Pos Pred Value : 0.8529
##          Neg Pred Value : 0.7881
##              Prevalence : 0.6590
##          Detection Rate : 0.5949
##    Detection Prevalence : 0.6974
##       Balanced Accuracy : 0.8010
##
##        'Positive' Class : no
##
```

```
#Plotting

ggplot(test, aes(x = test.df$PC1, y = predicted_values))+
  geom_point() + # add points
  geom_smooth(method = "lm", # plot a regression...
              method.args = list())
```




```
# KNN After PCA


model.knn = knn(train.df,test.df,train.Y1,k=1)
head(data.frame(model.knn,test.Y1))
```

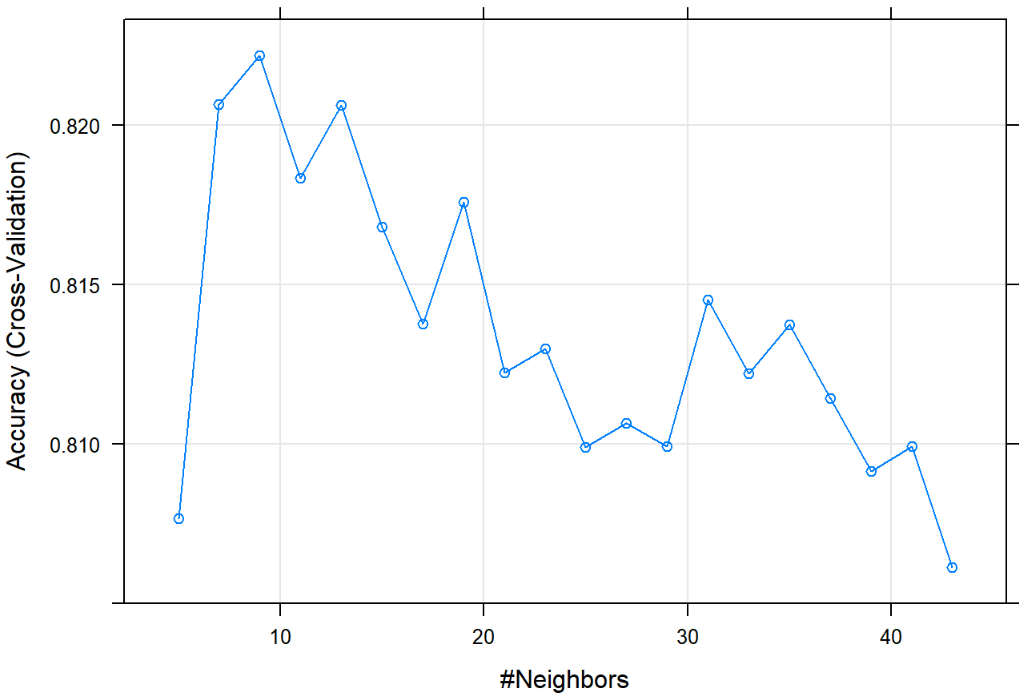
```
##   model.knn test.Y1
## 1       yes     yes
## 2        no      no
## 3        no      no
## 4        no      no
## 5        no      no
## 6        no      no
```

```
confusionMatrix(model.knn,test.Y1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  219  50
##        yes  38  83
##
##                Accuracy : 0.7744
##                  95% CI : (0.7296, 0.8149)
##     No Information Rate : 0.659
##     P-Value [Acc > NIR] : 4.51e-07
##
##                   Kappa : 0.4868
##  Mcnemar's Test P-Value : 0.241
##
##             Sensitivity : 0.8521
##             Specificity : 0.6241
##          Pos Pred Value : 0.8141
##          Neg Pred Value : 0.6860
##              Prevalence : 0.6590
##          Detection Rate : 0.5615
##    Detection Prevalence : 0.6897
##       Balanced Accuracy : 0.7381
##
##        'Positive' Class : no
##
```

```r
tr=cbind(pc.df,Rain)

model.cv <- train(
  Rain ~., data = tr, method = "knn",
  trControl = trainControl("cv", number = 10),
  preProcess = c("center","scale"),
  tuneLength = 20
)
plot(model.cv)
```




```r
K=model.cv$bestTune
K
```

```
##   k
## 3 9
```

```r
model.knn = knn(train.df,test.df,train.Y1,k=K)
head(data.frame(model.knn,test.Y1))
```

```
##   model.knn test.Y1
## 1       yes     yes
## 2        no      no
## 3        no      no
## 4        no      no
## 5        no      no
## 6        no      no
```

```r
confusionMatrix(model.knn,test.Y1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  235  53
##        yes  22  80
##
##                Accuracy : 0.8077
##                  95% CI : (0.765, 0.8456)
##     No Information Rate : 0.659
##     P-Value [Acc > NIR] : 6.184e-11
##
##                   Kappa : 0.5466
##  Mcnemar's Test P-Value : 0.000532
##
##             Sensitivity : 0.9144
##             Specificity : 0.6015
##          Pos Pred Value : 0.8160
##          Neg Pred Value : 0.7843
##              Prevalence : 0.6590
##          Detection Rate : 0.6026
##    Detection Prevalence : 0.7385
##       Balanced Accuracy : 0.7580
##
##        'Positive' Class : no
##
```