MO433 – Aprendizado de Máquina Não Supervisionado

Trabalho 3

Thiago Bruschi Martins RA 120212

Obs: Este relatório foi desenvolvido utilizando a linguagem R no RStudio.

```
library(dbscan)
library(isotree)
```

```
## Warning: package 'isotree' was built under R version 4.1.2
```

```
library(devtools)
```

```
## Carregando pacotes exigidos: usethis
```

```
devtools::install_github("pridiltal/stray")
```

```
## Skipping install of 'stray' from a github remote, the SHA1 (192ec52b) has not changed since last ins
##   Use 'force = TRUE' to force installation
```

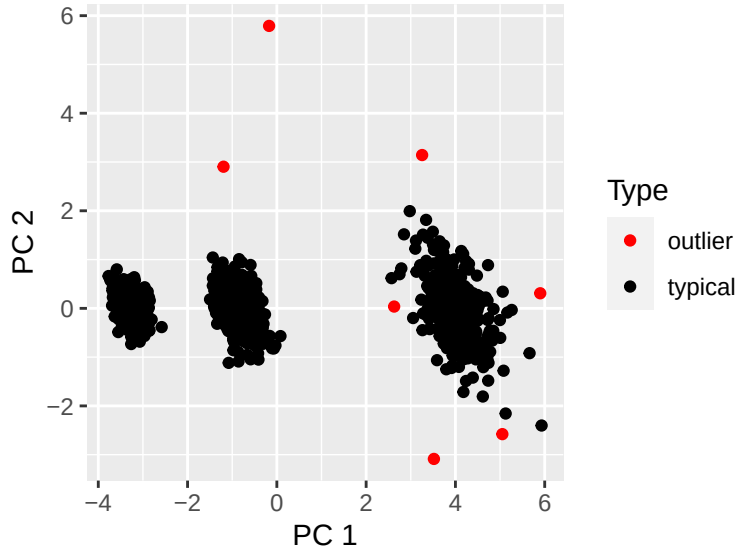
```
# Setando o workspace
setwd("C:\\Users\\thiag\\OneDrive\\Documents\\Unicamp\\Master\\Unsupervised-Learning\\Trabalho 3")

# Leitura dos dados
data <- read.csv('dados3.csv', sep = ",", header = TRUE)
head(data)
```

```
##      V1     V2     V3     V4     V5     V6     V7     V8    V9     V10
## 1 -2.97  1.020 -2.340  3.460  1.630  0.157 -2.660  0.559 -5.27  1.960
## 2  4.30 -0.817  1.410 -2.160  0.673  0.870 -1.220  1.620  3.43 -0.771
## 3 -2.62  0.378 -1.010  1.430 -0.278 -0.384  0.613 -0.880 -2.14  0.465
## 4  2.38 -0.356  0.731 -1.250  0.391  0.362 -0.817  1.000  1.85 -0.260
## 5  1.87 -0.568  0.440 -0.856  0.401  0.576 -0.568  0.793  1.55 -0.412
## 6 -2.17  0.255 -1.010  1.200 -0.128 -0.291  0.332 -0.552 -1.77  0.499
```

1º Algoritmo: K-vizinhos mais próximos

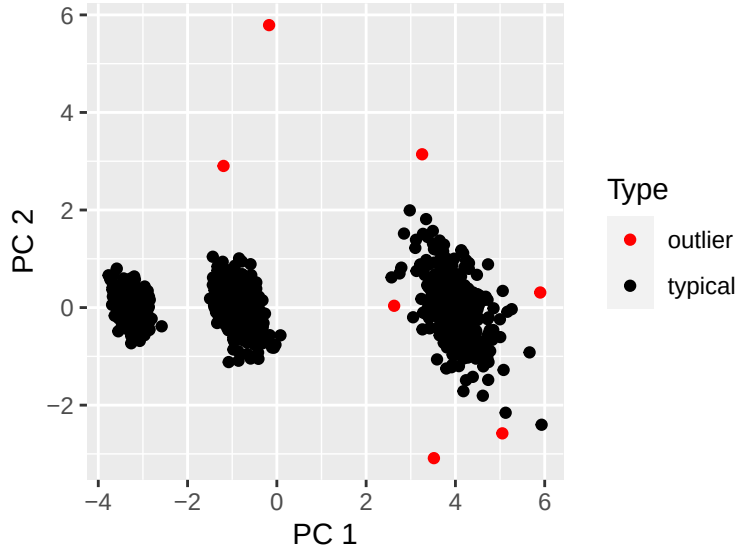
Esta função do pacote stray, apresentada em aula, utiliza a ténica de verificar a distância entre os k-vizinhos mais próximos de cada ponto e ordená-los de acordo com estas distâncias. Após esta ordenação, é selecionado um threshold para classificar os dados como anomalias ou não utilizando a Teoria do Valor extremo. Utilizando esta função obtivemos o seguinte resultado:

```
outliers_k <- stray::find_HDoutliers(data)
stray::display_HDoutliers(data, outliers_k)
```



Observando a figura acima, vemos que antes de aplicar a técnica de k-vizinhos mais próximos, que é baseada em distância, a função aplicou uma redução de dimensionalidade com PCA. Isto significa que na verdade houve uma combinação de técnicas aqui. O total de outliers encontrando pela função foi 7. De acordo com os dados do enunciado, de que existem 7 outliers nos dados, aparentemente o resultado foi ótimo. Pela imagem observamos que pelo menos 3 desses 7 outliers são bem evidentes após a técnica do PCA. Mas como não sabemos exatamente quais são os pontos que são outliers, não podemos garantir que o resultado está correto. Obs: O valor de K default para esta função é 10, mas testamos com K variando de 1 a 10 e o resultado se manteve durante todo este intervalo. Segue a imagem para K = 1, mostrando que o resultado se manteve.

```
outliers_k1 <- stray::find_HDoutliers(data, k = 1)
stray::display_HDoutliers(data, outliers_k1)
```



Agora vamos verificar quais foram os pontos identificados como outliers

```
print(outliers_k[1])
```

```
## $outliers
## [1]   1 224 359 555 665 754 833
```

2º Algoritmo: Local Outlier Factor (LOF)

Este algoritmo utiliza a distância entre os k-vizinhos mais próximos para realizar uma comparação entre a densidade dos dados. Lembrando que quanto maior o LOF, maior a diferença de densidade entre os dados observados, e portanto, maior chance de detercarmos uma anomalia. Aqui listamos os LOF mais altos (em ordem decrescente) e depois listamos os dados com LOF > 3, que resultou nos 7 dados com mais chances de serem anomalias. Coincidentemente, foi o mesmo resultado do algoritmo anterior, o que sugere que seja um bom resultado, dado que os dois algoritmos convergiram.

Obs: O valor default de k para esta função é 5.

```
message('Lista decrescente dos LOF mais altos:\n')
```

```
## Lista decrescente dos LOF mais altos:
```

```
lof <- lof(data)
head(sort(lof, decreasing = TRUE), 20)
```

```
##  [1] 11.546486  7.074966  6.484763  5.688474  5.677321  5.666452  3.838198
##  [8]  2.440832  2.422457  2.406871  2.363613  2.351568  2.321905  2.170431
## [15]  2.015497  1.952570  1.909699  1.844144  1.840022  1.763041
```

```
message('\nIndice dos dados com LOF > 3 (anomalias):\n')
```

```
##
## Indice dos dados com LOF > 3 (anomalias):
```

```
which(lof > 3)
```

```
## [1]   1 224 359 555 665 754 833
```

3º Algoritmo: Isolation Forest

Este algoritmo cria n árvores que dividem hierarquicamente os dados de forma aleatória. Este modelo assume que os outliers, por estarem mais distantes de dados "normais", serão isolados pelas árvores mais rapidamente. Desta forma, terão uma altura média menor do que a dos outros dados. Utilizamos aqui os valores default do modelo e obtivemos um resultado coerente com as outras duas técnicas utilizadas. A predição deste modelo é um score para cada observação, quanto mais alto esse score, maior a chance da observação ser uma anomalia. Sendo assim, ordenamos o resultado de forma decrescente para observamos os 10 maiores scores obtidos. Vemos que os 7 maiores scores obtidos foram acima de 0.72. Enquanto o 8º já se encontra em torno de 0.63.

```r
iso <- isolation.forest(data)
pred <- predict(iso, data)
head(sort(pred, decreasing = TRUE), 10)
```

```
##       665         1       359       555       224       833       754       713
## 0.7987895 0.7947479 0.7762913 0.7714252 0.7601147 0.7290603 0.7246536 0.6307393
##        41       500
## 0.5748651 0.5608050
```

```r
#cat("Point with highest outlier score: ",  which.max(pred), "\n")
```

Conclusão

Os três algoritmos utilizados apontaram os mesmos 7 pontos como outliers, são eles: 1, 224, 359, 555, 665, 754, 833. Dado que os 3 algoritmos utilizam técnicas diferentes e os 3 convergiram para o mesmo resultado, provavelmente o resultado é coerente.