Tarefa 3 de Aprendizado não Supervisionado (MO 433)

Anomalias
30/11/2021

Autores

- Fábio Kenji Jojima (RA 232024)
- Lucas Alves Racoci (RA 156331)

Objetivo

Encontrar outliers (anomalias) nos dados [1] fornecidos pelo professor utilizando três técnicas diferentes.

Metodologia

Para realizar esse objetivo, usamos a linguagem R e Python, seguindo os seguintes passos:

- Leitura dos dados (em R)
- Detecção de anomalia usando Isolation forests (em R)
- Detecção de anomalias por densidade (em R)
- Leitura dos dados (em Python)
- Detecção de anomalia por mean shift (em Python)

Estes passos serão detalhados a seguir. Para mais detalhes sobre os scripts completos executados, veja os apêndices [2] e [3].

Leitura dos Dados

Para ler os dados foi utilizado a função `read` nativa do R:

```
url <- "https://www.ic.unicamp.br/~wainer/cursos/2s2021/433/dados3.csv"
df <- read.csv(url)
```

Isolation forests

Para encontrar os valores das árvores isoladas, foi utilizada a função `isolation.forest` da biblioteca `"isotree"`. Para instalá-la, usamos o seguinte comando:

```
install.packages("isotree")
library("isotree")
```

Assim, para obter os valores desejados:

```
iforest <- isolation.forest(df)
```

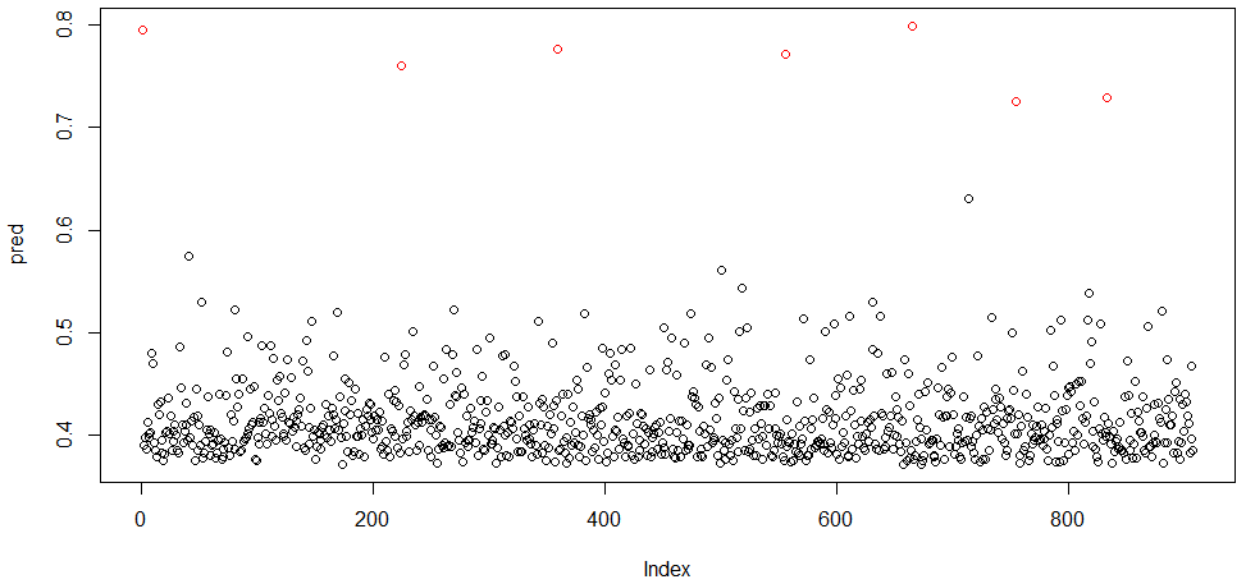
Obs: Os valores *default* foram utilizados neste script, também foram feitos testes para diferentes número de árvores (*ntrees*) porém o resultado não foi significativamente diferente.

Com os valores das arvores isoladas, é então calculado a pontuação de anomalias (outlier) utilizando a função **predict** também da biblioteca "isotree" :

```
pred <- predict(iforest, df)
```

Para melhor visualização é feito o gráfico dos resultados:

```
plot(pred, col = ifelse(pred < 0.7,'black','red'))
```



Os índices dos valores encontrados podem ser obtidos utilizando a linha de comando abaixo:

```
> pred[pred>0.7]
```

```
        1       224       359       555       665       754       833
0.7947479 0.7601147 0.7762913 0.7714252 0.7987895 0.7246536 0.7290603
```

Portanto, utilizando o método de isolation forest, foram encontrados 7 anomalias (outliers) representados no gráfico acima com destaque aos valores maiores a 0.7.

Local Outlier Factor Score (LOF)

Para encontrar os valores da densidade LDR (*local readability density*), foi utilizado a função **`lof`** da biblioteca **`"dbscan"`**. Para instalá-la, usamos o seguinte comando:

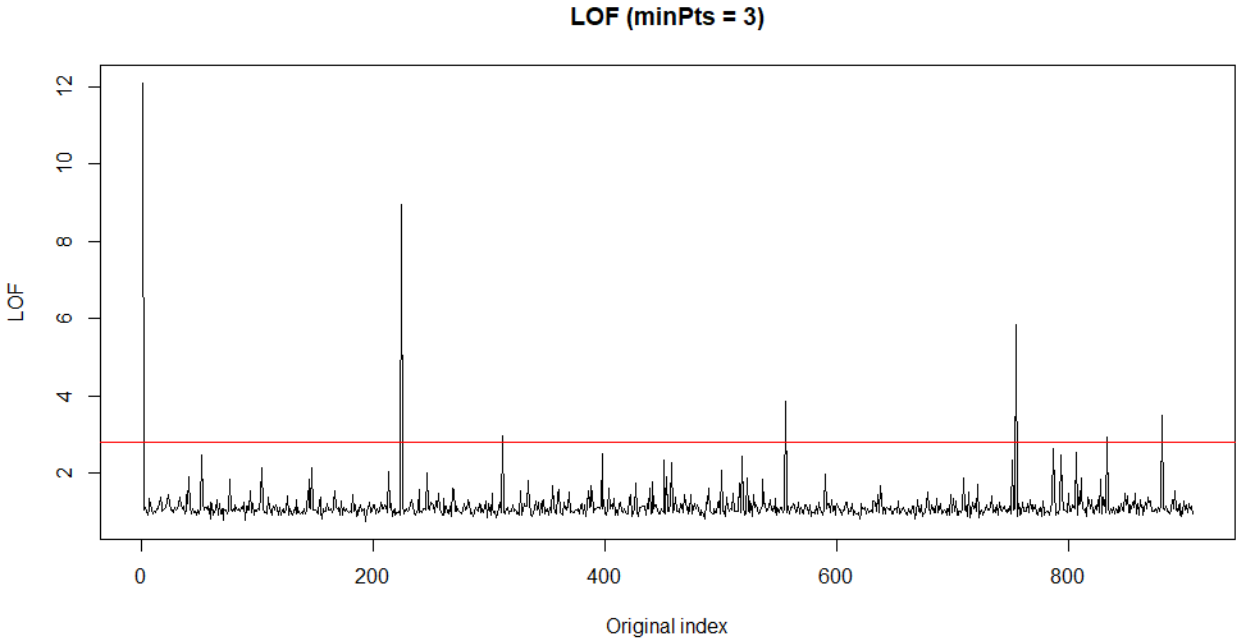
```
install.packages("dbscan")
library("dbscan")
```

Então são obtidos os valores de densidade para os três vizinhos mais próximos com a função **`lof`**:

```
lof_data <- lof(df, minPts = 3)
```

Os valores de LOF que são significativamente maiores que 1 indicam que possuem um LDR pequeno em relação aos vizinhos, indicando uma anomalia. Após uma análise dos dados, para se obter os 7 maiores valores, foram escolhidos os valores acima de 2.8. Para uma melhor visualização foi gerado o gráfico abaixo.

```
plot(lof_data, type = "l",  main = "LOF (minPts = 3)", xlab = "Original
index", ylab = "LOF")
abline(h=2.8, col="red")
```



Para obter explicitamente os índices dos sete maiores valores, foram nomeados os valores pelos índices do dataset e então filtrados os valores superiores a 2.8:

```
> lof_names <- setNames(lof_data, 1:length(lof_data))

> lof_names[lof_names > 2.8]

        1       224       311       555       754       833       880

12.088215  8.943467  2.965006  3.860715  5.824949  2.938469  3.479505
```

Utilizando a detecção de anomalias por densidade são então encontrados os 7 maiores valores que são significantemente maiores que 1, indicando que são valores de anomalias.

Mean Shift

Para detectar as anomalias pelo método Mean Shift usa-se a biblioteca **`MeanShift`** do pacote `cluster` do **`sklearn`** em Python. O script completo usado pode ser encontrado em [3]. Também usamos numpy e **`pandas`** na manipulação de dados e o módulo **`express`** do **`plotly`** para gerar um gráfico de dispersão

```
from sklearn.cluster import MeanShift
import numpy as np
import pandas as pd
import plotly.express as px
```

Os dados foram lidos usando o método **`df = pd.read_csv(url)`** como pode ser visto a seguir:

```
url = "https://www.ic.unicamp.br/~wainer/cursos/2s2021/433/dados3.csv"
df = pd.read_csv(url)
df
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	-2.97	1.020	-2.340	3.460	1.630	0.157	-2.660	0.559	-5.27	1.960
1	4.30	-0.817	1.410	-2.160	0.673	0.870	-1.220	1.620	3.43	-0.771
2	-2.62	0.378	-1.010	1.430	-0.278	-0.384	0.613	-0.880	-2.14	0.465
3	2.38	-0.356	0.731	-1.250	0.391	0.362	-0.817	1.000	1.85	-0.260
4	1.87	-0.568	0.440	-0.856	0.401	0.576	-0.568	0.793	1.55	-0.412
...	...	...	...	...	...	...	...	...	...	...
902	-2.19	0.632	-0.457	0.880	-0.589	-0.723	0.860	-1.010	-1.63	0.413

903	4.63	-0.851	1.540	-2.330	0.716	0.921	-1.310	1.730	3.67	-0.828
904	2.16	-0.497	0.508	-0.949	0.525	0.568	-0.867	0.997	1.60	-0.310
905	-2.15	0.923	-0.301	0.690	-0.706	-1.040	0.799	-0.996	-1.65	0.627
906	2.07	-0.401	0.709	-0.975	0.342	0.475	-0.577	0.733	1.59	-0.428

907 rows × 10 columns

O método **fit** do **MeanShift** retorna um objeto **clusters** com um arranjo com as etiquetas (**labels**) do grupo a que pertence cada entrada do conjunto de dados.

Como o parâmetro **bandwidth** não foi explicitado, ele será estimado pelo método **estimate_bandwidth** [4, 5]

```
clusters = MeanShift().fit(df)
labels = clusters.labels_
labels
```

```
array([7, 0, 2,..., 1, 2, 1])
```

O arranjo tem uma entrada para cada elemento do conjunto de dados, onde o valor da entrada é a etiqueta do grupo a que pertence o elemento.

Para contar quantas entradas estão em cada grupo, foi usada a função **np.unique** com o parâmetro **return_counts=True**, o que faz com que a função retorne um par de arranjos. Ambos os arranjos terão o mesmo tamanho, o número de etiquetas únicas. O primeiro arranjo (**unique_labels**) contém o valor de cada uma das etiquetas e o segundo (**count**) tem a contagem do número de entradas com a etiqueta correspondente.

```
unique_labels, count = np.unique(labels, return_counts=True)
np.stack((unique_label, count)).T
```

```
array([[  0, 300],
       [  1, 300],
       [  2, 300],
       [  3,   1],
       [  4,   1],
       [  5,   1],
       [  6,   1],
       [  7,   1],
       [  8,   1],
       [  9,   1]])
```

Assim, do resultado anterior, foi observado que há trezentas entradas em cada um dos três primeiros grupos (0 a 2) e apenas uma entrada nos próximos sete (3 a 9).

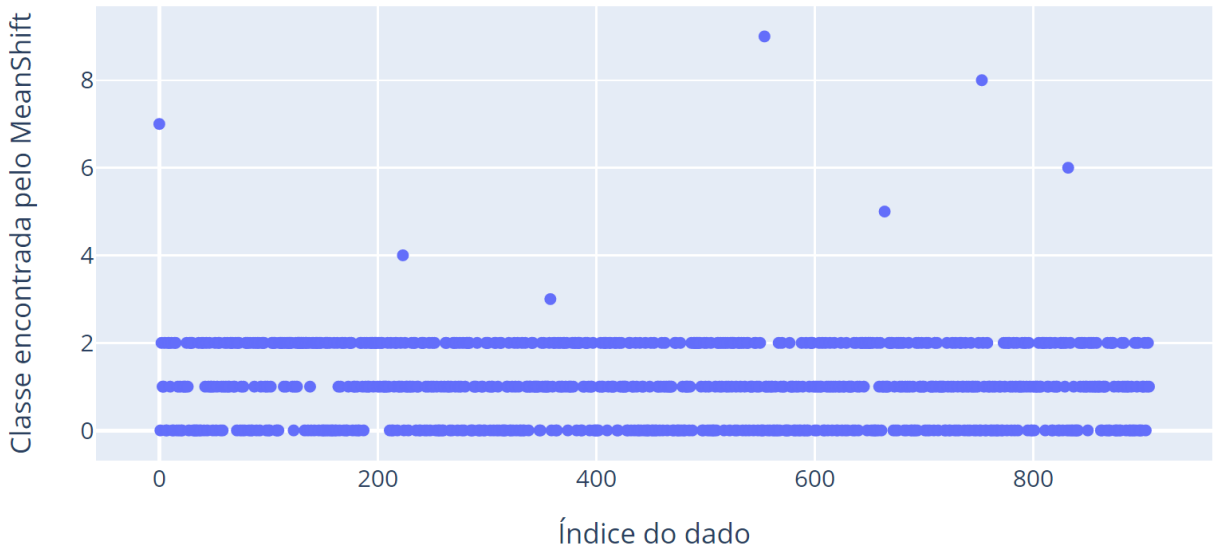
Para melhor visualizar as etiquetas de cada entrada, construiu-se o arranjo **labeled_indexes**, com a primeira coluna (0) contendo as etiquetas (**labels**) e a segunda os índices (**indexes**). Isso foi feito da seguinte maneira:

```
indexes = np.arange(etiquetas.size).reshape(etiquetas.shape)
indexed_labels = np.stack((labels, indexes)).T
indexed_labels
```

```
array([[  7,   0],
       [  0,   1],
       [  2,   2],
       ...,
       [  1, 904],
       [  2, 905],
       [  1, 906]])
```

Também foi gerado um gráfico de dispersão mostrando a classe no eixo vertical a que pertence cada índice (eixo horizontal) da entrada do conjunto de dados original.

```
fig = px.scatter(
    pd.DataFrame(indexed_labels),
    x=1, y=0,
    labels = {
        '1': "Índice do dado",
        '0': "Classe encontrada pelo MeanShift"
    }
)
fig.write_html("meanshift-scatterplot.html")
fig.show()
```



Foi assumido que os sete grupos com apenas um elemento, os com etiqueta maior que dois, são as anomalias. Assim, para explicitar os índices dessas entradas podemos filtrar o arranjo **`indexed_labels`** e manter só as entradas cujas etiquetas (coluna 0) são maiores que dois, da seguinte forma:

```
indexed_labels[indexed_labels[:,0]>2]
```

```
array([[  7,   0],
       [  4, 223],
       [  3, 358],
       [  9, 554],
       [  5, 664],
       [  8, 753],
       [  6, 832]])
```

Assim, os índices das anomalias encontradas por esse método são: 0, 223, 358, 554, 664, 753 e 832.

Conclusão

Juntando os valores obtidos em cada um dos métodos foi construída a seguinte tabela. Os valores em vermelho indicam os que não foram encontrados pelos outros métodos.

	Índices do dataset original						
Isolation Forrest	1	224	311	555	665	754	833
Local Outlier Factor	1	224	311	555	754	833	880
Mean Shift *	1	224	359	555	665	754	833

*Os índices para o Mean Shift foram incrementados, porque em Python a indexação começa em zero.

Apesar de os métodos não terem concordado perfeitamente, foi observado que são apenas duas anomalias encontradas que divergem dos outros métodos.

Apêndice

[1] Dados disponíveis em : https://www.ic.unicamp.br/~wainer/cursos/2s2021/433/dados3.csv (último acesso 16:03 do dia 13/11/2021).

[2] O script R completo utilizado:

```
1.  install.packages("isotree")
2.  install.packages("dbscan")
3.
4.  library("isotree")
5.  library("dbscan")
6.
7.  url <- "https://www.ic.unicamp.br/~wainer/cursos/2s2021/433/dados3.csv"
8.  df <- read.csv(url)
9.
10. # Isolation Forest
11. iforest <- isolation.forest(df)
12. pred <- predict(iforest, df)
13. plot(pred, col = ifelse(pred < 0.7,'black','red'))
14. pred[pred > 0.7]
15.
16. # Local Outlier Factor Score (LOF)
17. lof_data <- lof(df, minPts = 3)
18. plot(lof_data, type = "l",  main = "LOF (minPts = 3)", xlab = "Original
    index", ylab = "LOF")
19. abline(h=2.8, col="red")
20. lof_names <- setNames(lof_data, 1:length(lof_data))
21. lof_names[lof_names > 2.8]
```

[3] Script em python usado para encontrar as anomalias através da clusterização por mean shift. Também pode ser encontrado na seguinte url do colab

https://colab.research.google.com/drive/1jQy7v3VeF1S3pRpBhPTmemLAmmMO8arH?usp=sharing

```
from sklearn.cluster import MeanShift
import numpy as np
import pandas as pd
import plotly.express as px
url = "https://www.ic.unicamp.br/~wainer/cursos/2s2021/433/dados3.csv"
df = pd.read_csv(url)
print(df)

clusters = MeanShift().fit(df)
labels = clusters.labels_
print(labels)

unique_labels, count = np.unique(labels, return_counts=True)
print(np.stack((unique_labels, count)).T)

indexes = np.arange(labels.size).reshape(labels.shape)
indexed_labels = np.stack((labels, indexes)).T
print(indexed_labels)

fig = px.scatter(
   pd.DataFrame(indexed_labels),
   x=1, y=0,
   labels = {
       '1': "Índice do dado",
       '0': "Classe encontrada pelo MeanShift"
   }
)
fig.write_html("meanshift-scatterplot.html")
fig.show()
print(indexed_labels[indexed_labels[:,0]>2])
```

[4] Documentação do método MeanShift (último acesso 15:17 do dia 29/11/2021): https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html

[5] Documentação do método estimate_bandwidth: (último acesso 15:18 do dia 29/11/2021): https://scikit-learn.org/stable/modules/generated/sklearn.cluster.estimate_bandwidth.html