
UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
JOÃO MONLEVADE


João Marcos Torres Gardingo - 17.1.8102

# Lista VI

CSI693 – Avaliação de Desempenho de Sistemas

Professor: Alexandre Magno de Sousa

JOÃO MONLEVADE/MG

9 de Fevereiro de 2023

a) As informações abaixo foram obtidas através de uma amostra "sample"dos dados originais, devido ao grande tamanho do arquivo original, 15500 linhas do arquivo .csv foram utilizadas para amostragem, representando 5% dos dados originais

| | **DURATION** | **SEND** | **RECEIVE** |
|---|---|---|---|
| Contagem | 15500.0000 | 15500.0000 | 15500.0000 |
| Média | 13.5380 | 18.1071 | 47.8745 |
| Desvio Padrão | 29.7168 | 127.7313 | 196.7760 |
| Mediana | 14.0016 | 14.4153 | 44.0664 |
| Variância | 49006676.0398 | 134376525.0735 | 180096570.1206 |
| Coeficiente de Variação | 2.1951 | 7.0542 | 4.1103 |
| 1° quartil | 3.7164 | 0.8879 | 4.5694 |
| 2° quartil | 7.6796 | 3.0625 | 13.5001 |
| 3° quartil | 14.4651 | 10.7235 | 40.2584 |
| min | 0.0689 | 0.0007 | 0.0250 |
| max | 2558.6003 | 11337.2172 | 14382.4323 |
| Range | 2558.5315 | 11337.2165 | 14382.4073 |
| Soma | 20702.5132 | 38356.4163 | 44616.0197 |

Tabela 1: Informações das *features* - 15500 amostras (5%)

b) • **Histogramas**

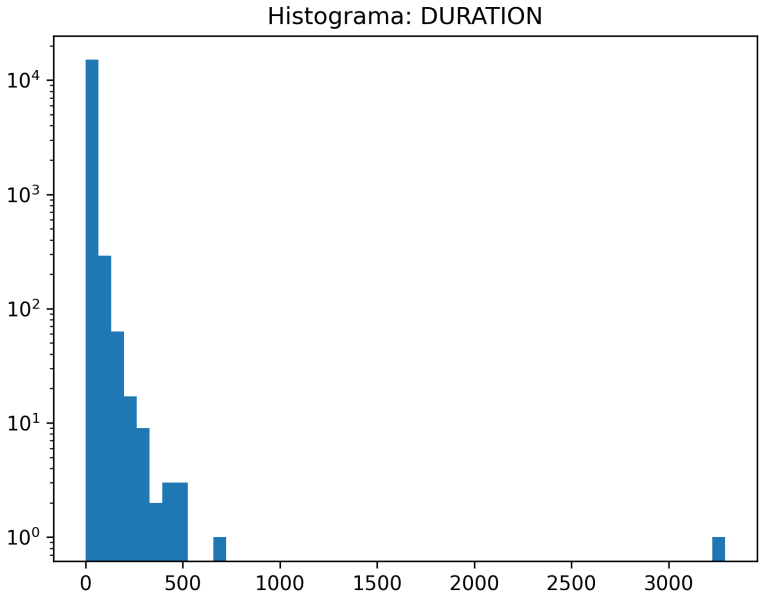


Figura 1: Duration - Histograma

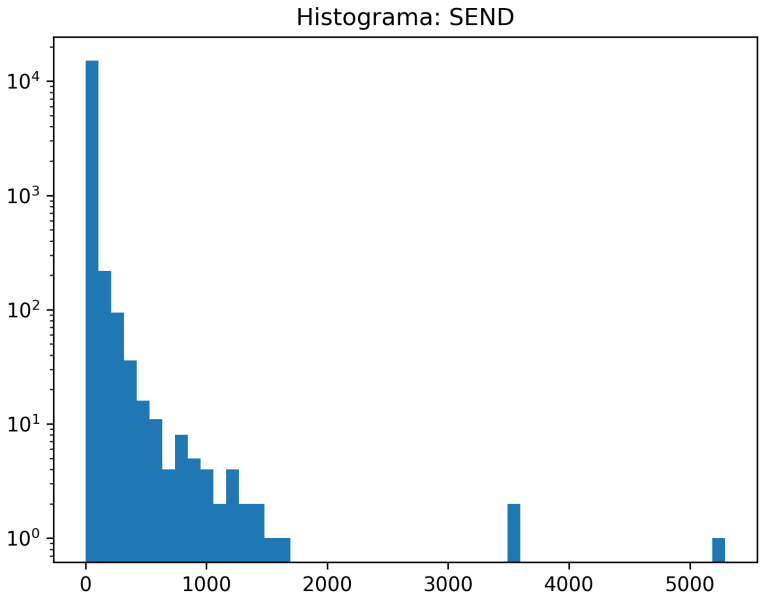


Figura 2: Send - Histograma

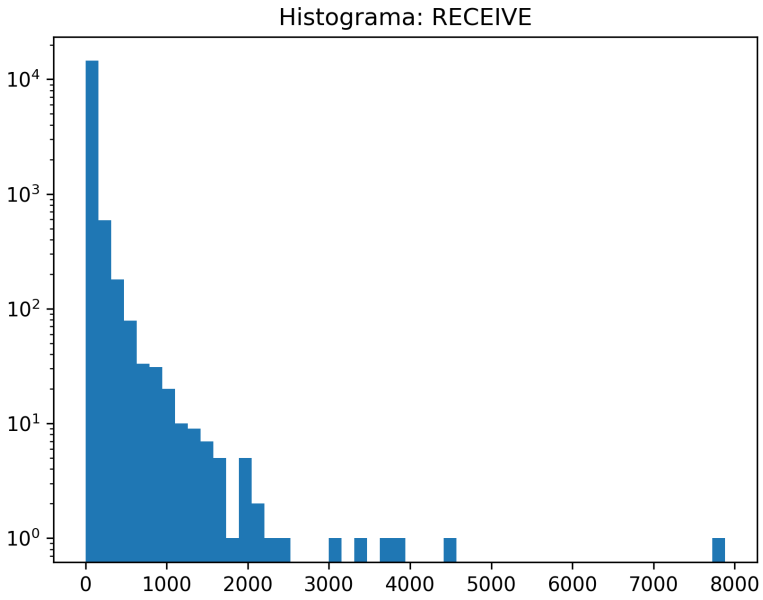


Figura 3: Receive - Histograma

- **Boxplot**

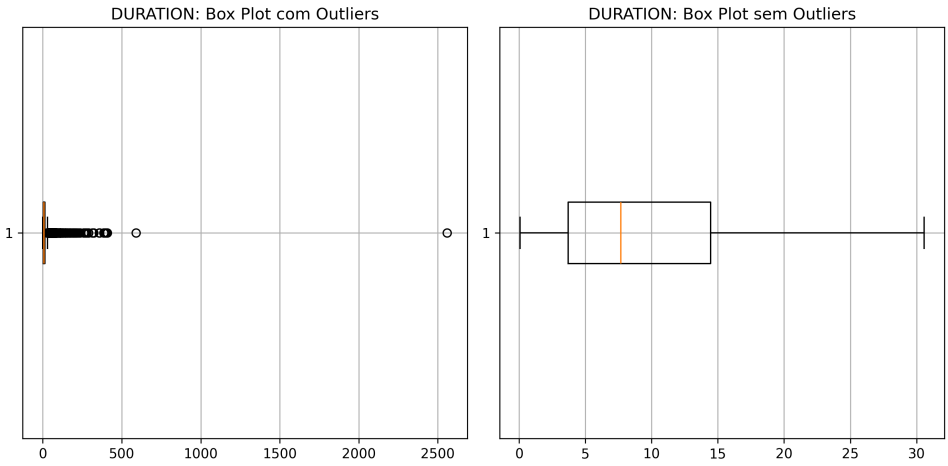


Figura 4: Duration - Boxplot

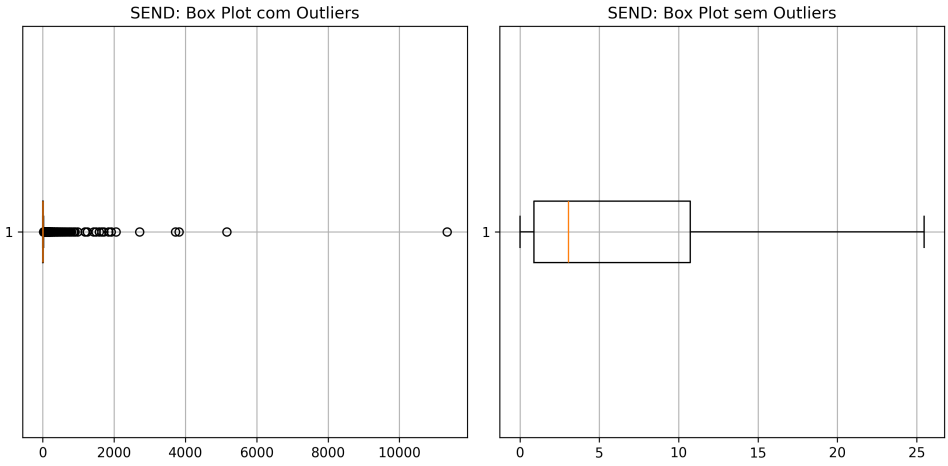


Figura 5: Send - Boxplot

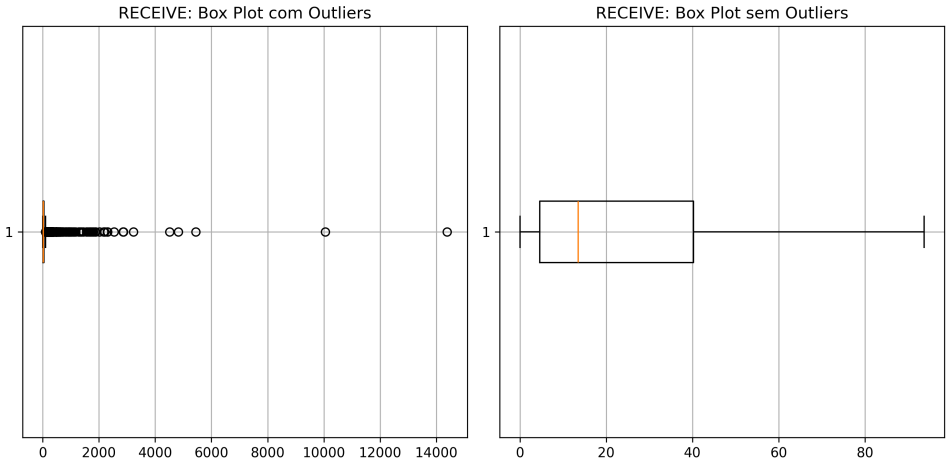


Figura 6: Receive - Boxplot

- **Cumulative Distribution Function (CDF)**

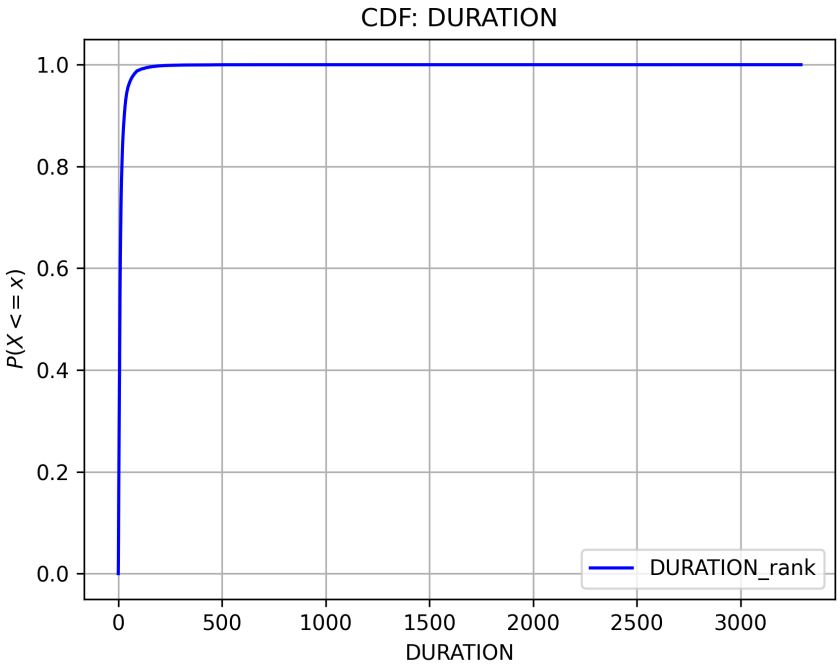


Figura 7: Duration - CDF

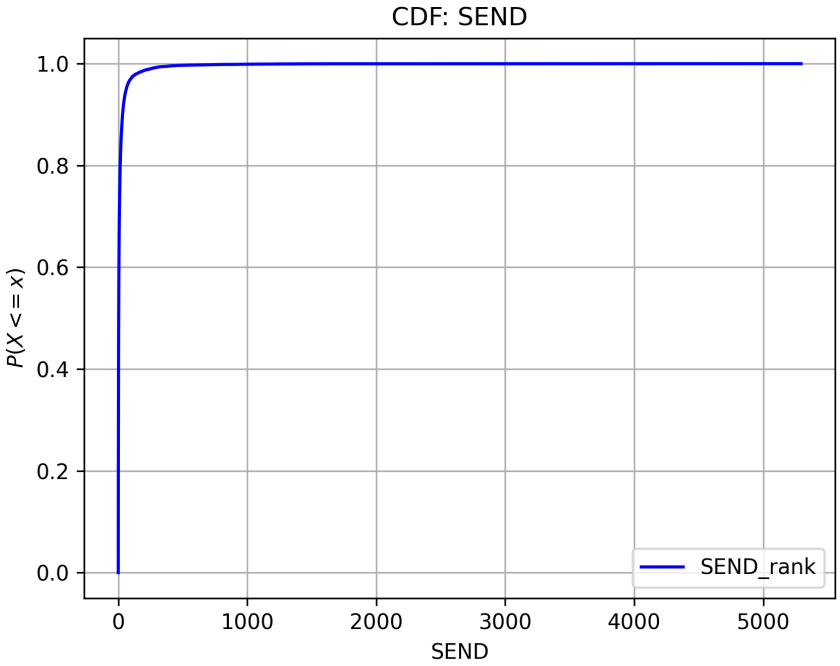


Figura 8: Send - CDF

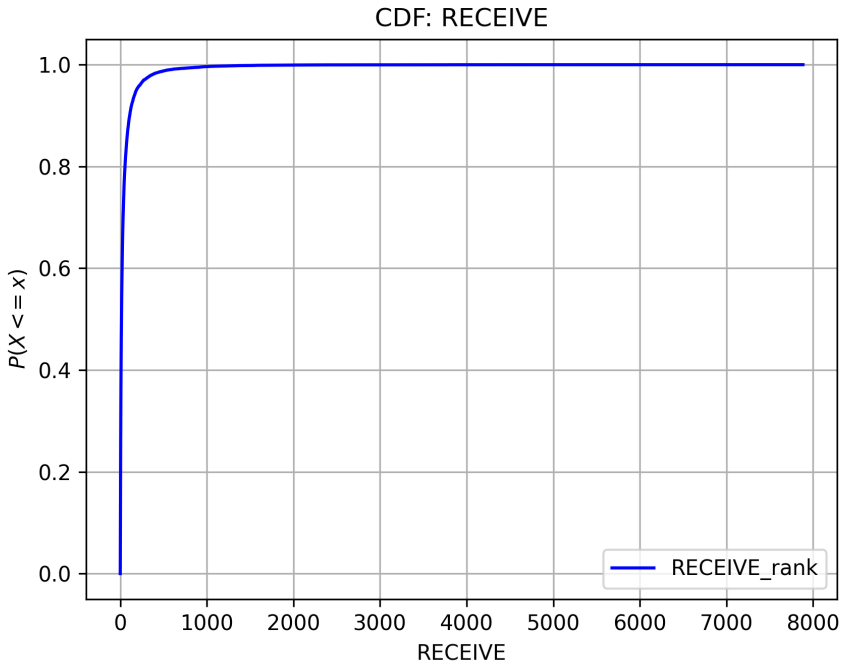


Figura 9: Receive - CDF

- **Dispersão**

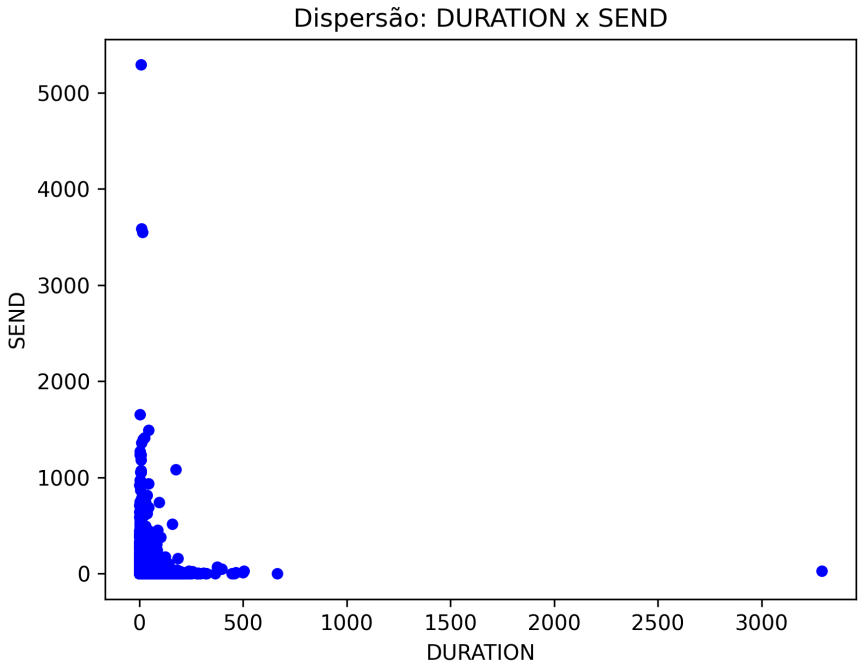


Figura 10: Duration x Send - Dispersão

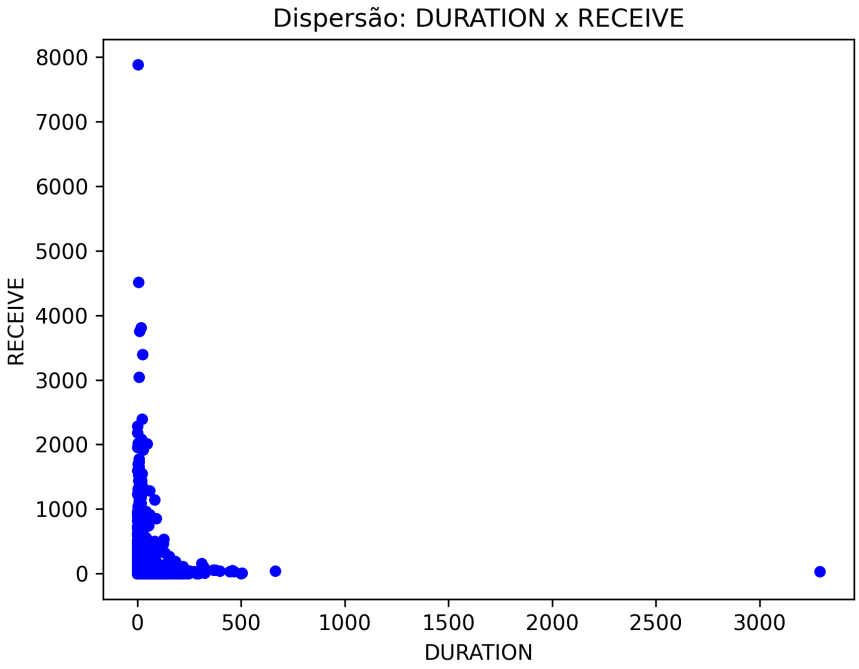


Figura 11: Duration x Receive - Dispersão

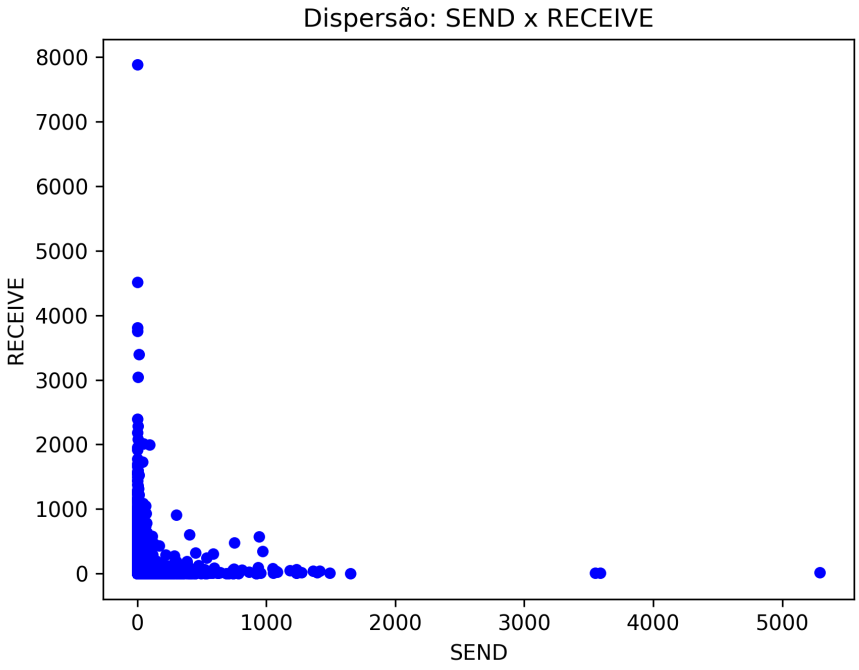


Figura 12: Send x Receive - Dispersão

c) Foi realizada uma análise nos dados com objetivo de determinar a necessidade ou não de transformação nos mesmos. Nesse caso, seguindo os passos da seção 15.4 do livro The Art of Computer Systems Performance Analysis, de Raj Rain, encontrei uma fórmula que se encaixava no nosso problema, onde temos uma grande base de dados. Para determinar a necessidade de transformação dos dados, foi encontrada a razão entre o maior e o menor valor de cada *feature*, então, como sugerido, foi realizada uma transformação logarítimica, pois a grandeza dessa razão foi muito alta, justificando essa ação, como pode ser visto abaixo:

| | **DURATION** | **SEND** | **RECEIVE** |
|---|---|---|---|
| $y_{max}$ | 2558.6003 | 11337.2172 | 14382.4323 |
| $y_{min}$ | 0.0689 | 0.0007 | 0.0250 |
| Razão | 37145.4115 | 15763783.4019 | 575880.0157 |

Tabela 2: Razão entre os valores $y_{min}$ e $y_{max}$ das *features*

Após realizada a transformação logarítmica (logaritmo natural, restringido em valores $> 0$), obtivemos novos gráficos:

- **Histogramas**

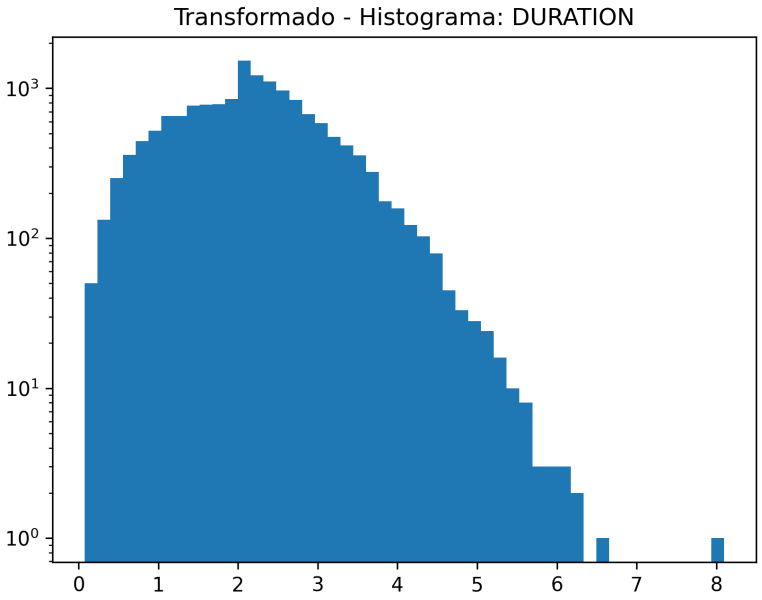


Figura 13: Duration - Histograma

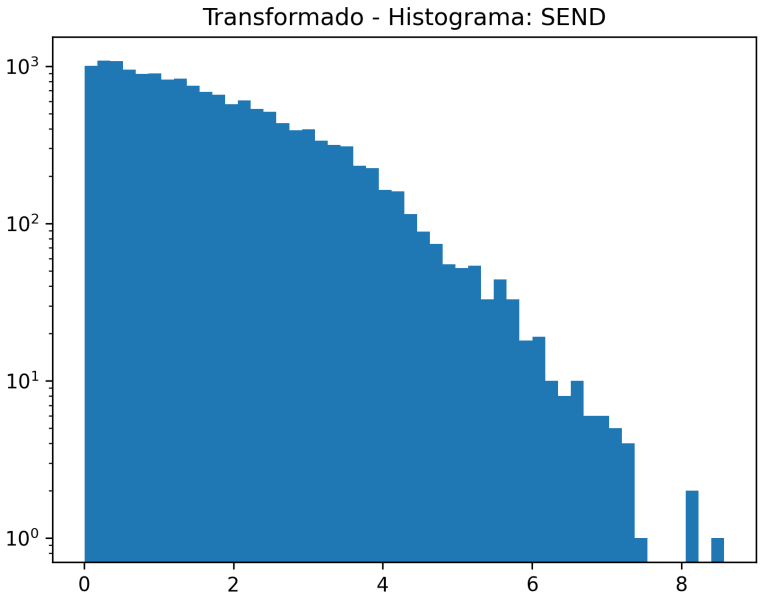


Figura 14: Send - Histograma

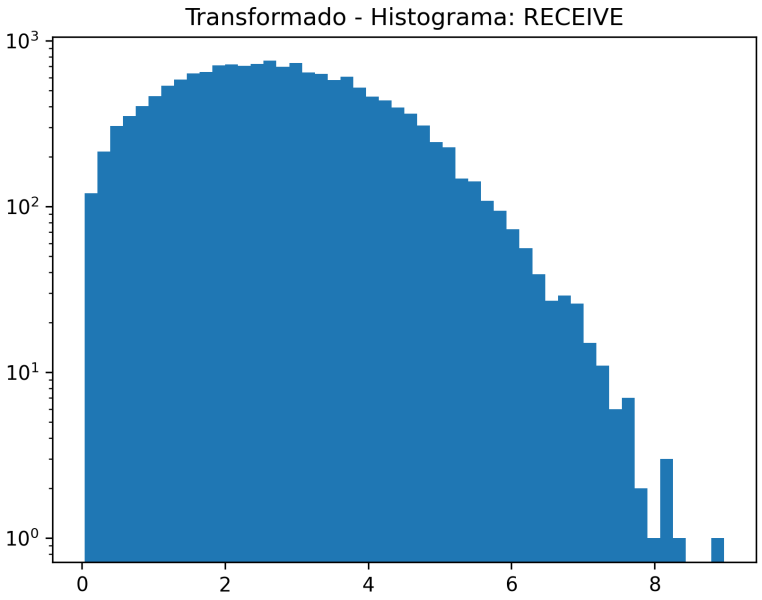


Figura 15: Receive - Histograma

- **Boxplot**

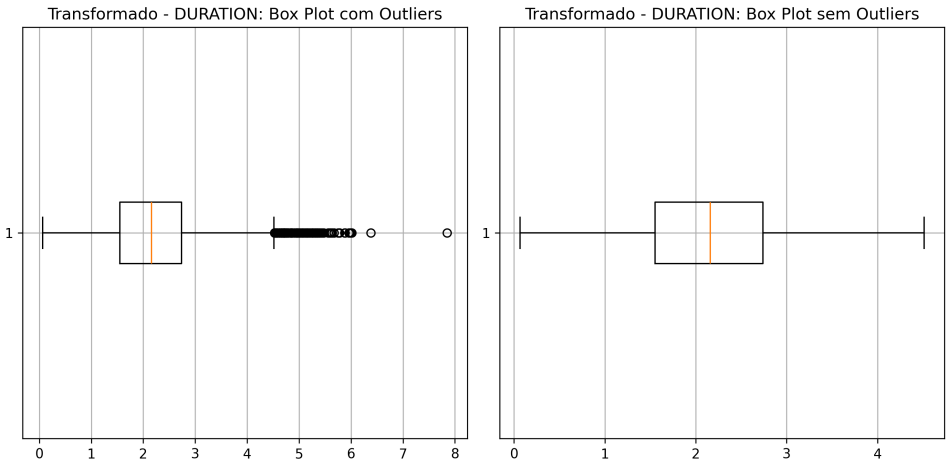


Figura 16: Duration - Boxplot

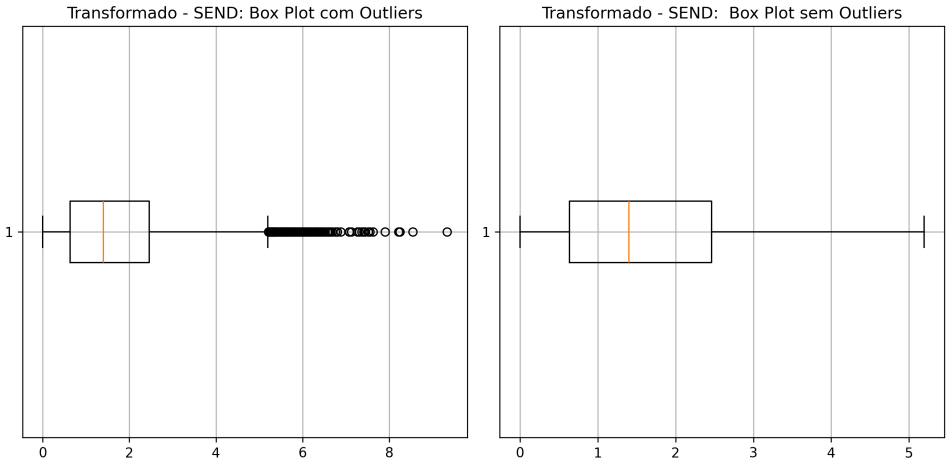


Figura 17: Send - Boxplot

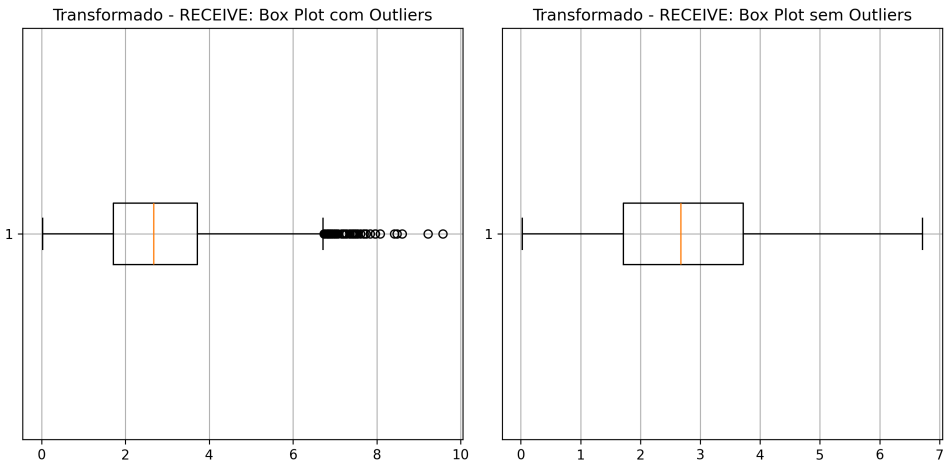


Figura 18: Receive - Boxplot

- **Cumulative Distribution Function (CDF)**

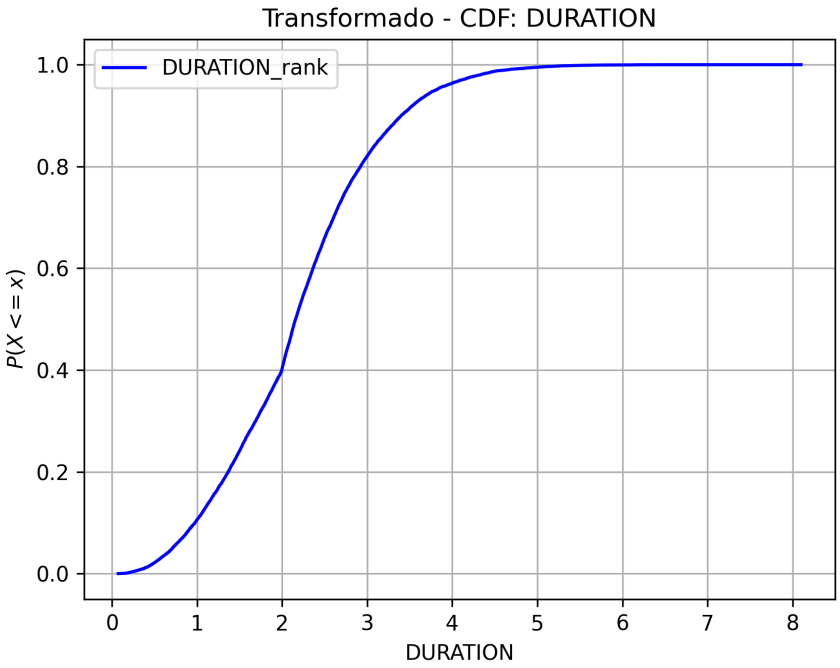


Figura 19: Duration - CDF

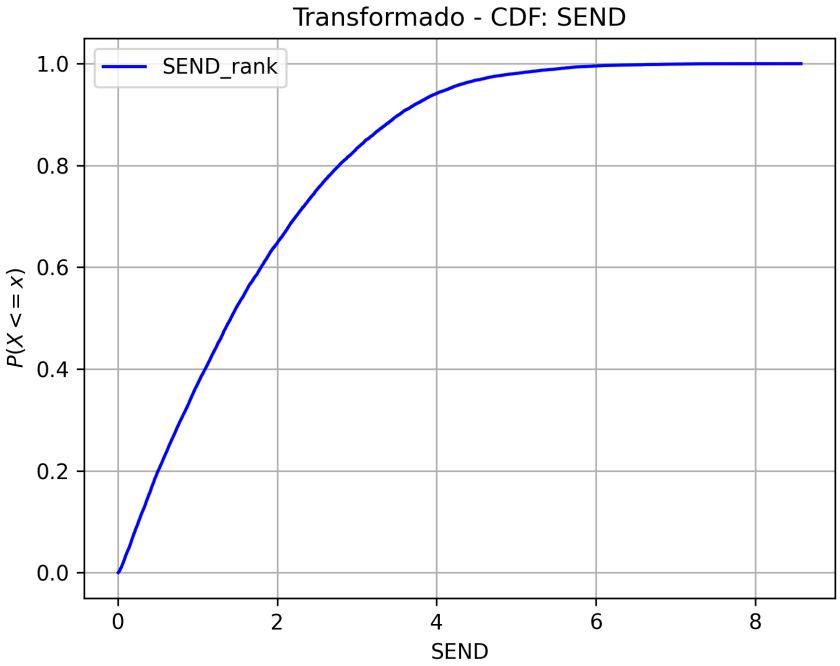


Figura 20: Send - CDF

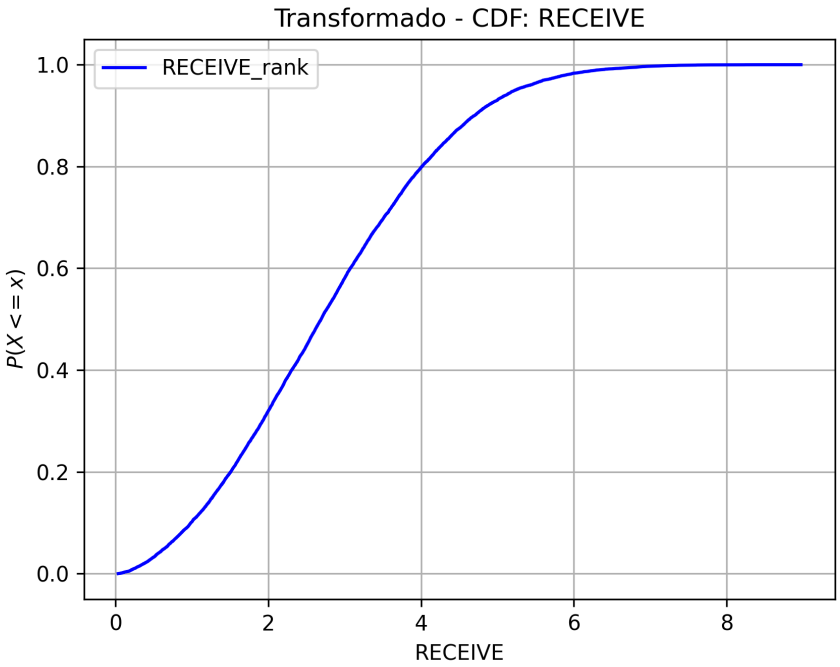


Figura 21: Receive - CDF

- **Dispersão**

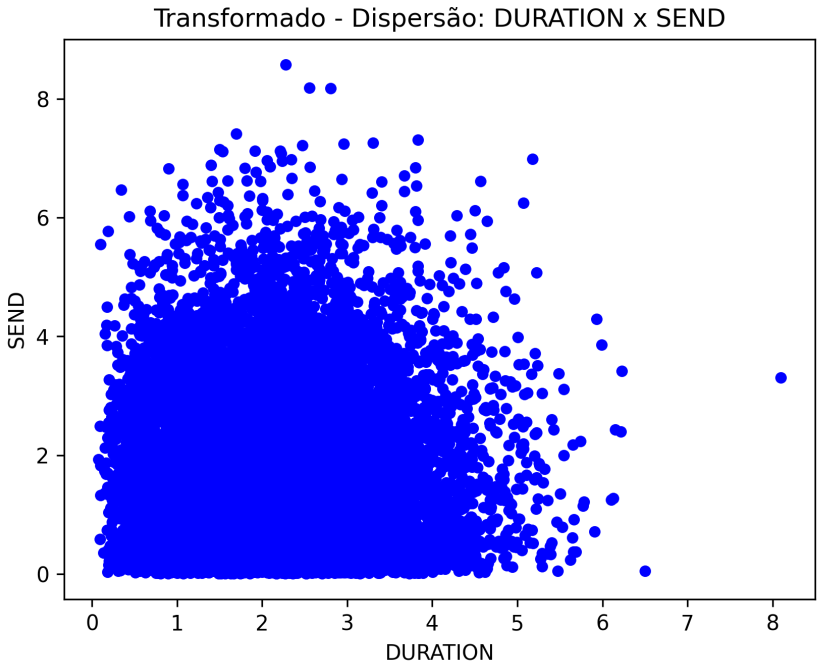


Figura 22: Duration x Send - Dispersão

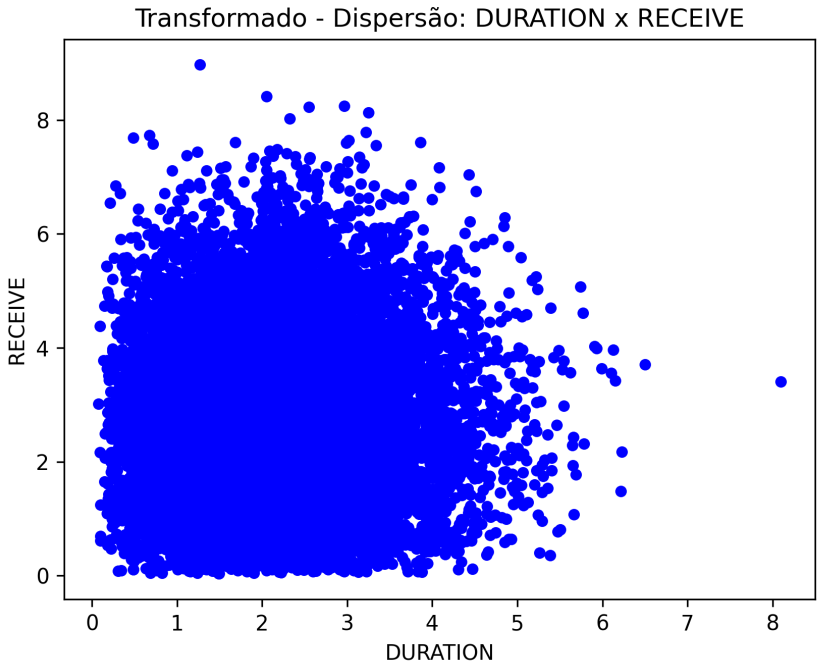


Figura 23: Duration x Receive - Dispersão

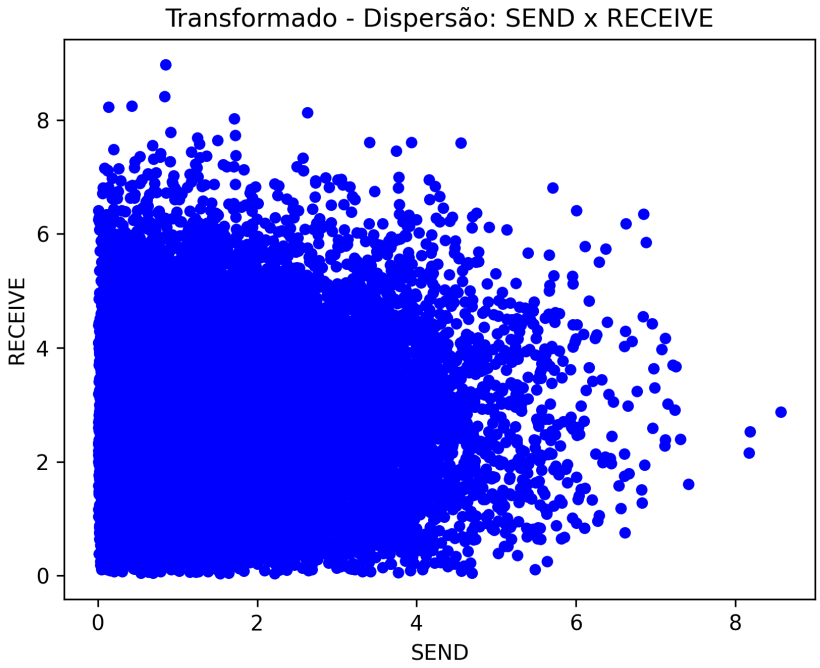


Figura 24: Send x Receive - Dispersão

Como pode ser observado, a introdução da transformação facilitou o entendimento de alguns gráficos, como o caso do bloxpot, onde podemos observar melhor a informação sem a necessidade de omitir os *outliers*.

d) Confesso que tive bastante dúvidas com relação à remoção dos *outliers* ou parte deles, contudo, como estamos lidando com dados coletados por uma fonte confiável, isso nos sugere que esses outliers não são erros de medição (o que poderia ser removido) mas sim informações reais e sobre o tráfego dos usuários, como não tive certeza nessa etapa, optei por não remover os *outliers* por não conseguir mensurar de forma concreta o impacto dessa remoção no processo de caracterização.

e) Conforme estudado em sala de aula, o cálculo do PCA foi feito utilizando funções da biblioteca *sklearn*, inicialmente, foi definido o número de componentes que o PCA teria, correspondendo ao número das *features* presentes (DURATION, SEND, RECEIVE).

Inicialmente, o seguinte trecho de código foi executado nos dados transformados:

```
df = df_transformed
pca = PCA(n_components=3)
X=df[['DURATION','SEND','RECEIVE']].values
pca.fit(X)
X_pca = pca.transform(X)
// pca.explained_variance_ratio == [0.44, 0.37, 0.19]
```

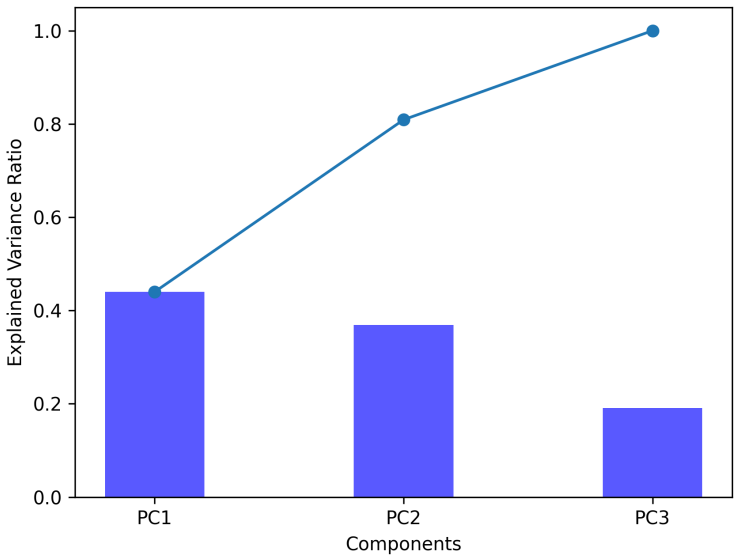


Figura 25: Componentes PCA x Taxa de Variância Explicada

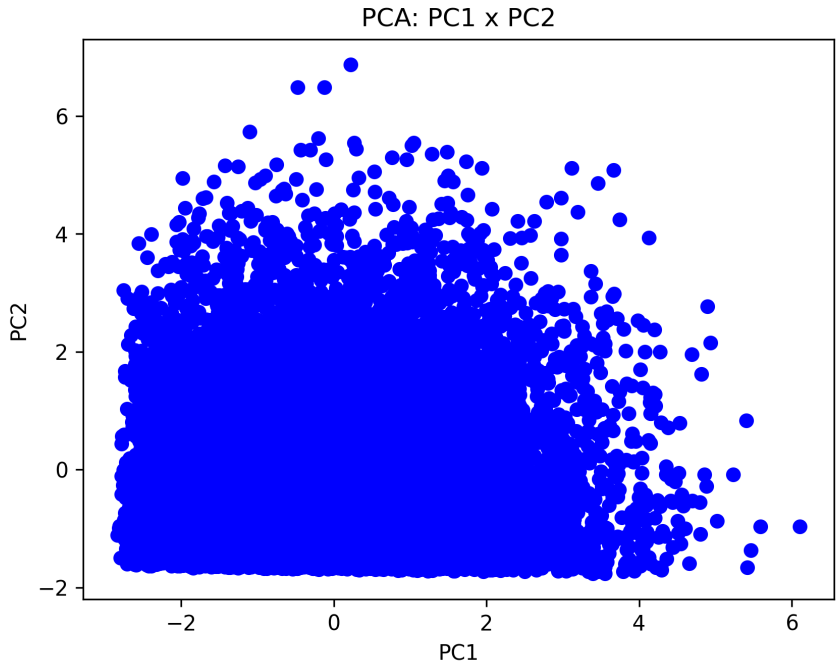


Figura 26: Correlação entre os componentes PC1 e PC2

f) Com base no gráfico dos componentes do PCA com relação a taxa de variância explicada é possível observar que a soma cumulativa dos 3 componentes é necessária para superar 85%, além disso é possível ver que para os dados transformados, a correlação entre os componentes PC1 e PC2 (os que possuem significancia maior no contexto da taxa de variância explicada) não nos diz tanta coisa, pois a faixa de grandeza dos dois componentes é praticamente a mesma.

g) Em seguida, foi executado o algoritmo *KMeans*, onde primeiramente descobrimos o valor de clusters ideais e, em seguida, utilizamos esse valor para finalizar a clusterização, como pode ser visto abaixo:

```
1 lsilhouette = []
2 lknumber = range(2, 19)
3
4 for k in lknumber:
5     np.random.seed(int(time.time()))
6     rand_number = np.random.randint(2**11)
7     kmeans = KMeans(n_clusters=k, random_state=
          rand_number)
```

```
8       kmeans.fit(X_pca)  # X_pca - a variavel que contem os
            componentes principais
9       cluster_labels = kmeans.labels_
10      silhouette = silhouette_score(X_pca, cluster_labels,
            metric='euclidean')
11      lsilhouette.append(silhouette)
12      print("Rand number = %d, K = %d: %.2f" % (rand_number
            , k, silhouette))
```

Resultando em:

```
Rand number = 277, K = 2: 0.28
Rand number = 618, K = 3: 0.30
Rand number = 1859, K = 4: 0.25
Rand number = 1040, K = 5: 0.25
Rand number = 1097, K = 6: 0.24
Rand number = 124, K = 7: 0.24
Rand number = 1176, K = 8: 0.24
Rand number = 928, K = 9: 0.23
Rand number = 640, K = 10: 0.23
Rand number = 1988, K = 11: 0.23
Rand number = 825, K = 12: 0.23
Rand number = 1010, K = 13: 0.23
Rand number = 1148, K = 14: 0.22
Rand number = 822, K = 15: 0.23
Rand number = 2010, K = 16: 0.23
Rand number = 1187, K = 17: 0.23
Rand number = 1890, K = 18: 0.23
```

Assim, podemos observar que o valor K = 3 é onde o índice de Silhouette se estabiliza.

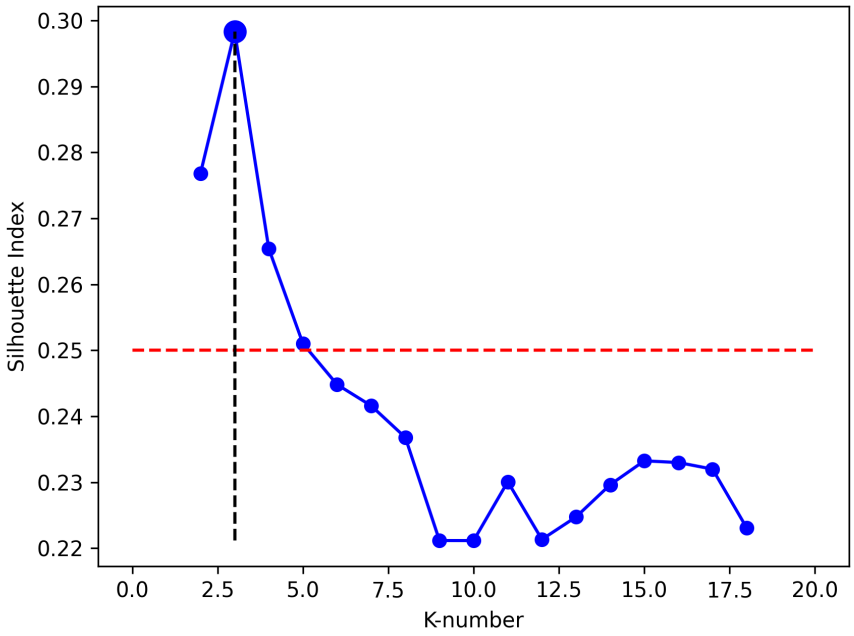


Figura 27: Índice de Silhouette - *KMeans*

h) **Dados dos Clusters:**

- **Cluster 0**

| | **DURATION** | **SEND** | **RECEIVE** |
|---|---|---|---|
| Contagem | 6640.0000 | 6640.0000 | 6640.0000 |
| Média | 2.1306 | 1.0262 | 1.7874 |
| Desvio Padrão | 0.9419 | 0.6726 | 0.7792 |
| Mediana | 2.1206 | 0.9838 | 1.8162 |
| Variância | 6699171.4368 | 6689738.7082 | 6693695.5634 |
| Coeficiente de Variação | 0.4421 | 0.6554 | 0.4359 |
| 1° quartil | 1.4321 | 0.4369 | 1.1915 |
| 2° quartil | 2.1105 | 0.9413 | 1.8449 |
| 3° quartil | 2.7239 | 1.5504 | 2.4423 |
| min | 0.0733 | 0.0050 | 0.0352 |
| max | 5.7827 | 2.7597 | 3.1962 |
| Range | 5.7094 | 2.7546 | 3.1610 |
| Soma | 6663.4671 | 6651.7859 | 6656.6898 |

Tabela 3: Cluster 0

- **Cluster 1**

| | DURATION | SEND | RECEIVE |
|---|---|---|---|
| Contagem | 5100.0000 | 5100.0000 | 5100.0000 |
| Média | 2.2604 | 1.2622 | 4.2801 |
| Desvio Padrão | 0.9134 | 0.8598 | 0.9057 |
| Mediana | 2.2399 | 1.2056 | 4.1973 |
| Variância | 3956717.1263 | 3951414.2617 | 3965773.2743 |
| Coeficiente de Variação | 0.4041 | 0.6812 | 0.2116 |
| 1° quartil | 1.6641 | 0.5525 | 3.5756 |
| 2° quartil | 2.2194 | 1.1490 | 4.1145 |
| 3° quartil | 2.7849 | 1.8456 | 4.8025 |
| min | 0.0945 | 0.0057 | 2.9342 |
| max | 6.5028 | 4.7008 | 8.9727 |
| Range | 6.4083 | 4.6952 | 6.0385 |
| Soma | 5125.4919 | 5116.9576 | 5140.0327 |

Tabela 4: Cluster 1

- **Cluster 2**

| | DURATION | SEND | RECEIVE |
|---|---|---|---|
| Contagem | 3760.0000 | 3760.0000 | 3760.0000 |
| Média | 2.1926 | 3.4760 | 2.5402 |
| Desvio Padrão | 0.9365 | 0.9445 | 1.0895 |
| Mediana | 2.1728 | 3.3844 | 2.5422 |
| Variância | 2155039.5141 | 2158217.5154 | 2154788.7753 |
| Coeficiente de Variação | 0.4271 | 0.2717 | 0.4289 |
| 1° quartil | 1.5335 | 2.7802 | 1.7718 |
| 2° quartil | 2.1530 | 3.2928 | 2.5442 |
| 3° quartil | 2.7259 | 3.9518 | 3.2550 |
| min | 0.0915 | 2.0257 | 0.0469 |
| max | 8.0993 | 8.5742 | 6.8139 |
| Range | 8.0078 | 6.5484 | 6.7670 |
| Soma | 3788.3400 | 3795.2497 | 3787.7995 |

Tabela 5: Cluster 2

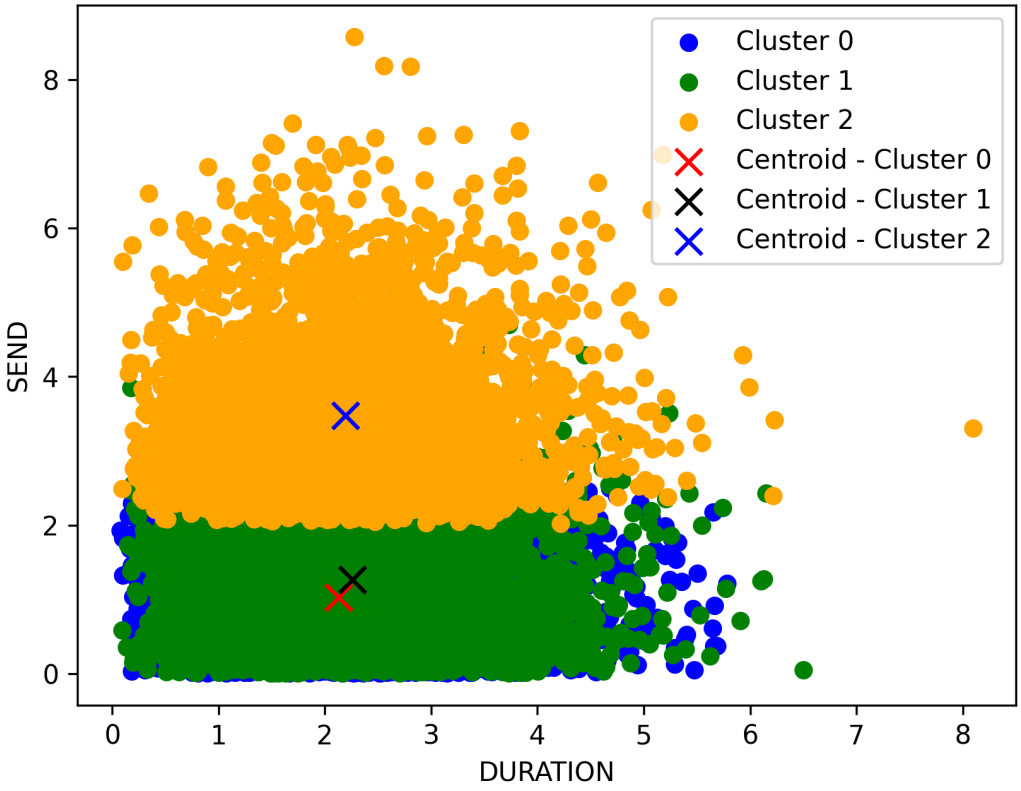


Figura 28: Clusters e Centróides

i) Eu tive dificuldades de rotular os clusters, o que pude observar é que as sessões dos usuários de todos os Clusters costumam ter aproximadamente o mesmo tempo, porém os usuários no Cluster 1 enviam mais que o dobro de dados em comparação aos outros clusters. Podemos dizer que os usuários do Cluster 1 tem mais demanda de upload, uma espécie de *Heavy User*, já os outros são usuários comuns.

j) Já acabei respondendo no item anterior.

# 1 Link para os códigos:

https://github.com/joaomarcostg/ADS/blob/main/listaVI.ipynb