Análise de dados de provedor de Internet para dispositivos Smart-TV e Chromecast

Nome: Daniel Rodrigues Ferreira

- Introdução

Este trabalho visa analisar o conjunto de dados dos dois dispositivos, com o objetivo de compreender os dados, bem como extrair resultados e hipóteses que possivelmente viriam a ajudar o provedor de serviço de Internet em questão.

O trabalho foi feito com o uso da linguagem Python e redigido no Jupyter, na forma de um notebook para melhor entendimento.

O presente relatório contém trechos de código para demonstração de como os resultados foram extraídos, imagens de gráficos, explicações e conclusões sobre os resultados.

- Bibliotecas

As principais bibliotecas utilizadas são o pandas, que é uma biblioteca para análise e manipulação de dados muito utilizada na área de ciência de dados, numpy, para processamento de matrizes, matplotlib, que se trata de uma biblioteca para visualização de dados e criação de gráficos, e scipy, uma biblioteca usada para computação científica, usada para tarefas de estatística, inteligência artificial e engenharia.

Imports

```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
import statsmodels.api as sm
from scipy.stats import gamma, norm, probplot, pearsonr, chi2_contingency
```

- Leitura e formatação dos dados

Após a leitura dos arquivos csv com pandas, há o consequente reescalonamento dos dados para logaritmo de base 10, uma vez que os valores de taxa upload e download, que serão analisados variam diversas ordens de grandeza.

Para evitar valores negativos, bem como a ocorrência de valores -infinito para valores inicialmente zerados, primeiramente se adiciona 1 à todos os valores.

Seção 1: Dataset

```
chrome = pd.read_csv('dataset_chromecast.csv')
smart = pd.read_csv('dataset_smart-tv.csv')
```

```
chrome.bytes_up = 1 + chrome.bytes_up
chrome.bytes_down = 1 + chrome.bytes_down
smart.bytes_up = 1 + smart.bytes_up
smart.bytes_down = 1 + smart.bytes_down


chrome.bytes_up = np.log10(chrome.bytes_up).to_numpy()
chrome.bytes_down = np.log10(chrome.bytes_down).to_numpy()
smart.bytes_up = np.log10(smart.bytes_up).to_numpy()
smart.bytes_down = np.log10(smart.bytes_down).to_numpy()
```

Em seguida, são removidas linhas com valores nulos e os valores qualitativos são convertidos em valores numéricos com a função fit_transform que usa estatísticas de uma dada coluna para converte-la de forma a manter uma escala. Nominalmente, as estatísticas utilizadas por este método são o desvio padrão e a média.

- Estatísticas gerais

Nesta parte é feita uma análise inicial das taxas de upload e download separadamente para cada tipo de dispositivo, Chromecast e Smart-TV.

Inicialmente são extraídas estatísticas básicas dos dados, com o objetivo de atingir uma compreensão básica dos mesmos.

1. Média, variância e desvio padrão

```
cupload = chrome[['bytes_up']]
cdownload = chrome[['bytes_down']]
supload = smart[['bytes_up']]
sdownload = smart[['bytes_down']]
print(f'''upload chrome: \n média: {np.mean(a=cupload).item()} \n variância: {np.var(a=cupload).item()}  \n desvio pa
\n download chrome: \n média: {np.mean(a=cdownload).item()} \n variância: {np.var(a=cdownload).item()} \n desvio padr
\n upload smart: \n média: {np.mean(a=supload).item()} \n variância: {np.var(a=supload).item()} \n desvio padrão: {np
\n download smart: \n média: {np.mean(a=sdownload).item()} \n variância: {np.var(a=sdownload).item()} \n desvio padrã
''')
```

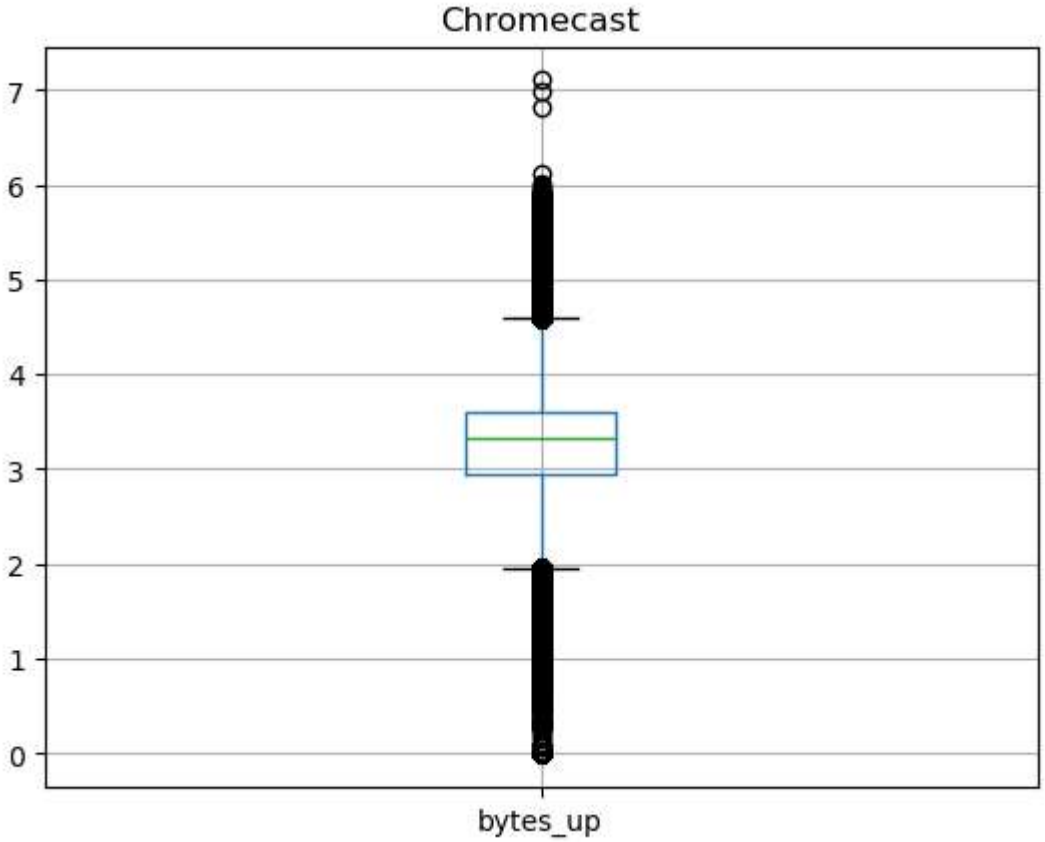
```
upload chrome:
 média: 3.3502996618095193
 variância: 0.4599683624609476
 desvio padrão: 0.6782096744082523

 download chrome:
 média: 3.8000457060383566
 variância: 1.663894572605127
 desvio padrão: 1.289920374521283

 upload smart:
 média: 2.1582882065066804
 variância: 4.110138414288679
 desvio padrão: 2.02734763035072

 download smart:
 média: 2.351678620482843
 variância: 6.721322375969502
 desvio padrão: 2.592551325619129
```

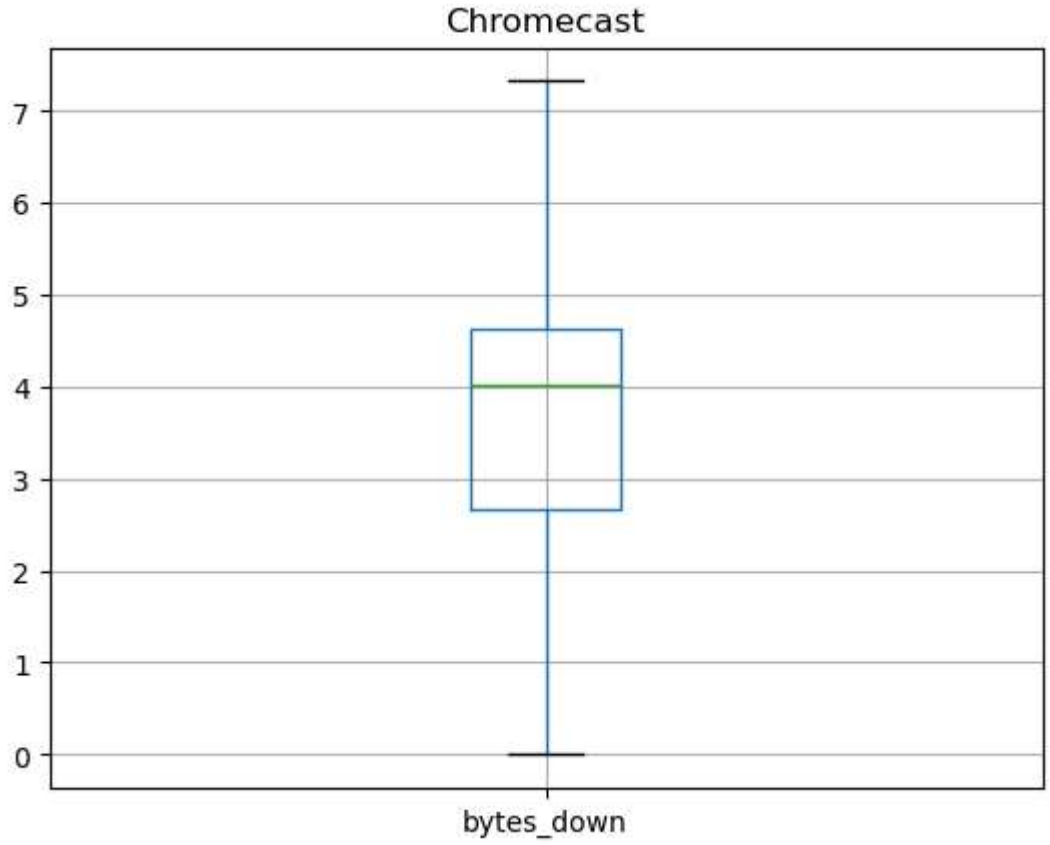
Com o uso do numpy, são calculadas a média, a variância e o desvio padrão das taxas de download e upload para cada tipo de dispositivo. Podemos observar por estes dados que, em média, os dispositivos do tipo Chromecast apresentam taxas de dados recebidos e enviados consideravelmente maiores, cerca de 50%, indicando uma maior demanda do serviço para o provedor de Internet.

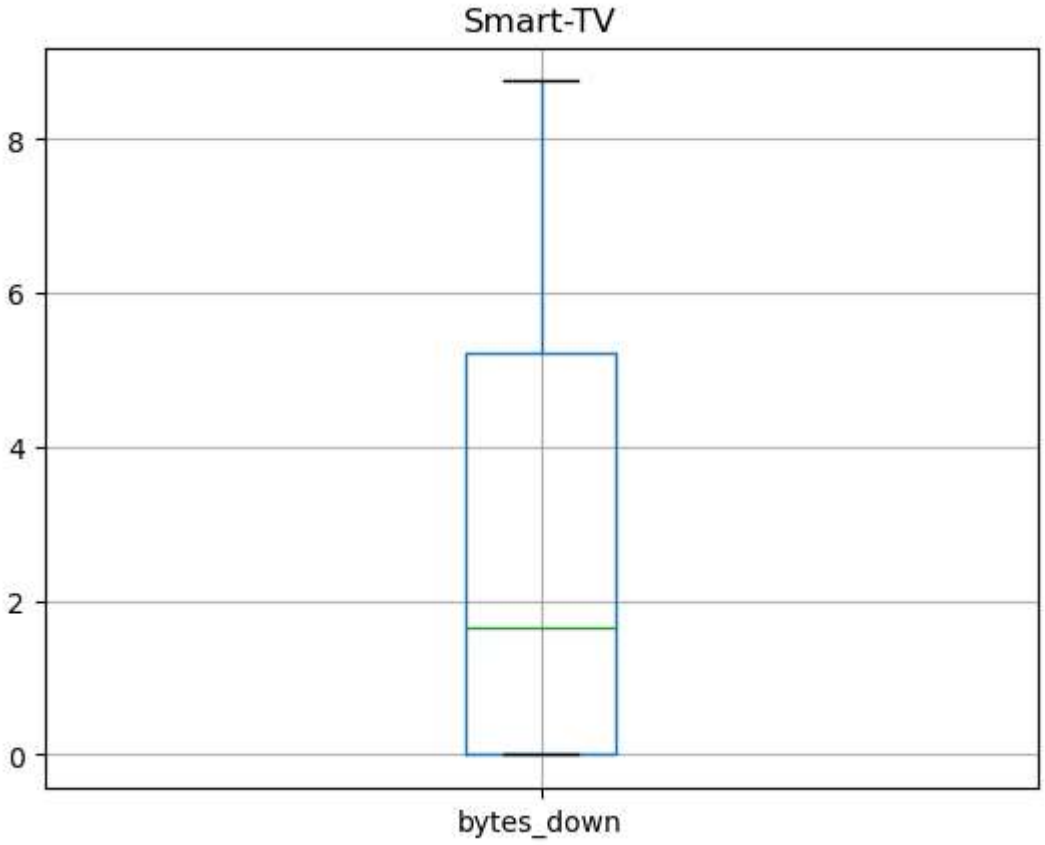
A baixa variância e desvio padrão apresentada pela taxa de upload no Chromecast indica que a taxa de upload varia pouco para diferentes dispositivos ao longo do dia, possivelmente indicando que há um padrão para a mesma.

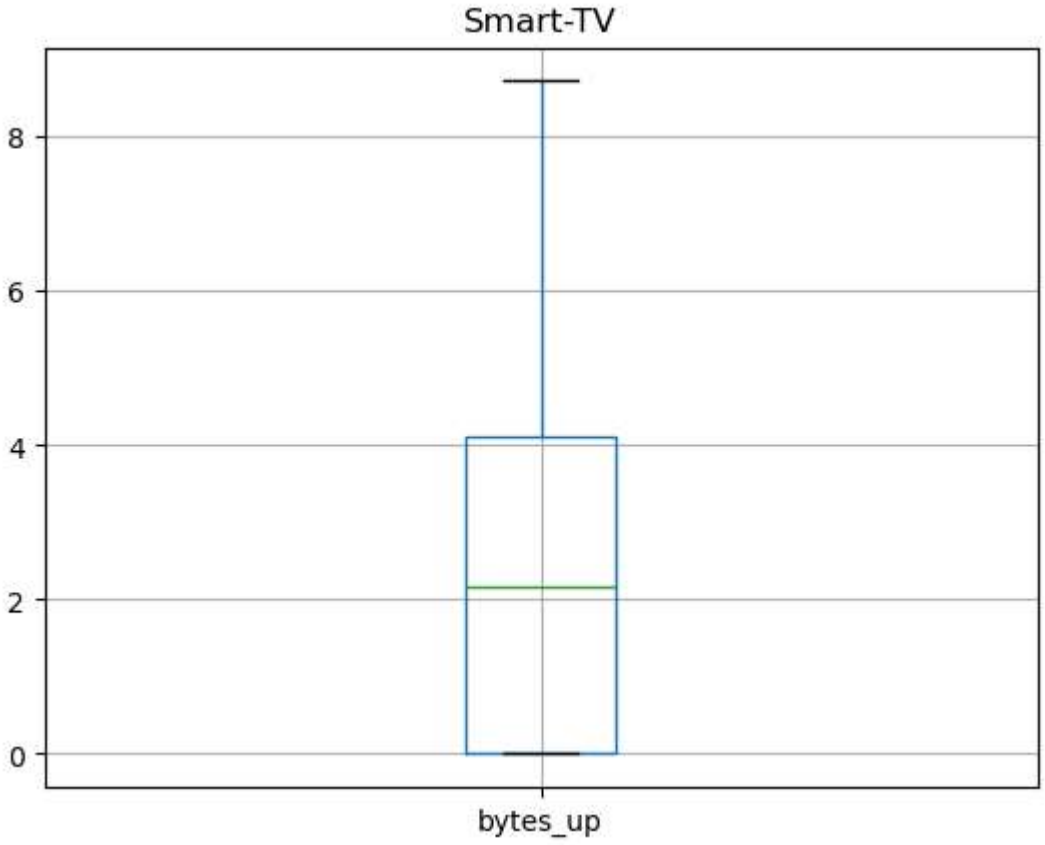
Por outro lado, a alta variância e desvio padrão nos dispositivos Smart-TV indica que o uso da Internet varia muito durante o dia para diferentes dispositivos.

2. Boxplots




Chromecast
7
6
5
4
3
2
1
0
bytes_down


Smart-TV
8
6
4
2
0
bytes_down


Smart-TV
8
6
4
2
0
bytes_up

Os parâmetros para os gráficos foram calculados com o uso do método boxplot do pandas e plotados com o matplotlib.

Com os gráficos de boxplot, é possível obter informações sobre a localidade dos dados, sua concentração e a presença de outliers ou discrepantes. A barra verde representa a mediana, e os dados contidos na “caixa” estão entre o 75-percentil e o 25-percentil.

É possível notar que o gráfico de upload de dispositivos Chromecast apresenta dados mais concentrados, com o IQR variando menos de 1 unidade, significando que, independente da hora ou dispositivo em questão, metade dos dispositivos Chromecast apresentaram uma taxa de upload contida entre 1000 e 10000 bytes. No entanto, nota-se a presença considerável de outliers, possivelmente significando que o comportamento fora dessa região ainda pode variar bastante.

O IQR do gráfico de download de dispositivos Chromecast também varia relativamente pouco, porém sem a presença de outliers.

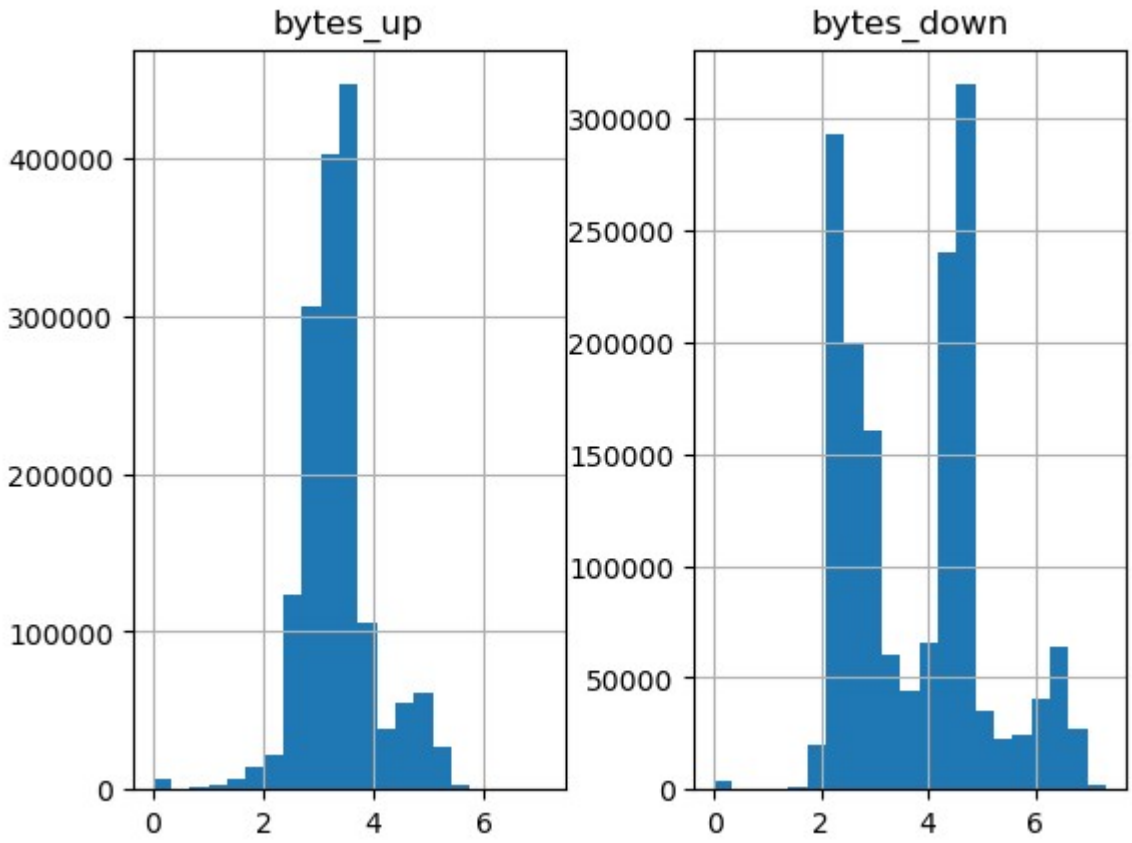
Para dispositivos do tipo Smart-TV podemos notar gráficos de boxplot com comportamento similar, ambos com o valor mínimo do IQR em 0, significando que o 25-percentil está em 0, demonstrando que para uma quantidade grande de horários para diferentes dispositivos a taxa de download e upload está em 0. Nenhum dos dois apresenta outliers, no entanto seus máximos chegam a ser mais altos do que para dispositivos Chromecast.

3. Histogramas

Os histogramas foram plotados com o uso do método hist da biblioteca pandas, com os valores de y representando a frequência e o número de bins foi calculado utilizando o método de Sturges.

Chromecast

```
chrome.hist(column = ['bytes_up','bytes_down'], bins = int(1 + 3.3*math.log(1620529, 10)))
plt.show()
```

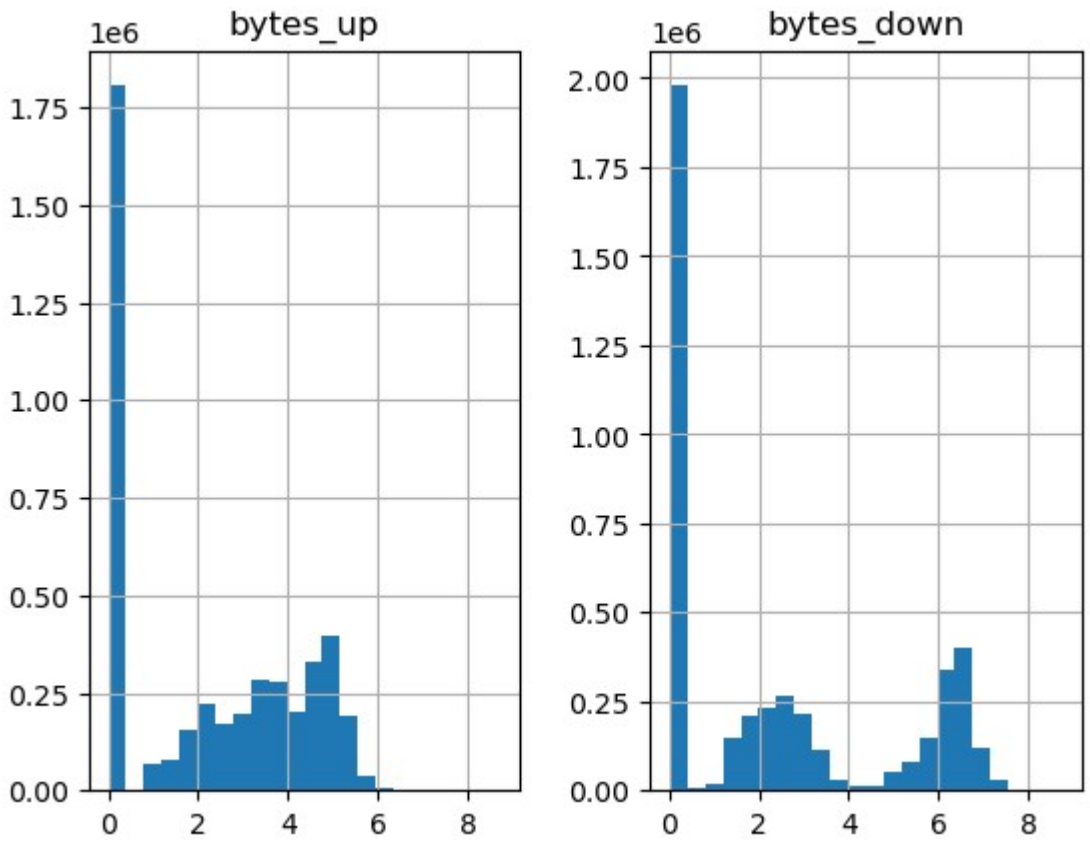


Para os histogramas dos dispositivos Chromecast em relação as taxas de upload, pode-se notar uma concentração altíssima de valores no centro, enquanto as taxas de download demonstração concentração em dois valores distintos.

Nota-se para ambos que há poucos valores próximos de 0, possivelmente significando que dispositivos deste tipo raramente se encontram em repouso.

Smart-TV

```
smart.hist(column = ['bytes_up','bytes_down'], bins = int(1 + 3.3*math.log(4417903, 10)))
plt.show()
```



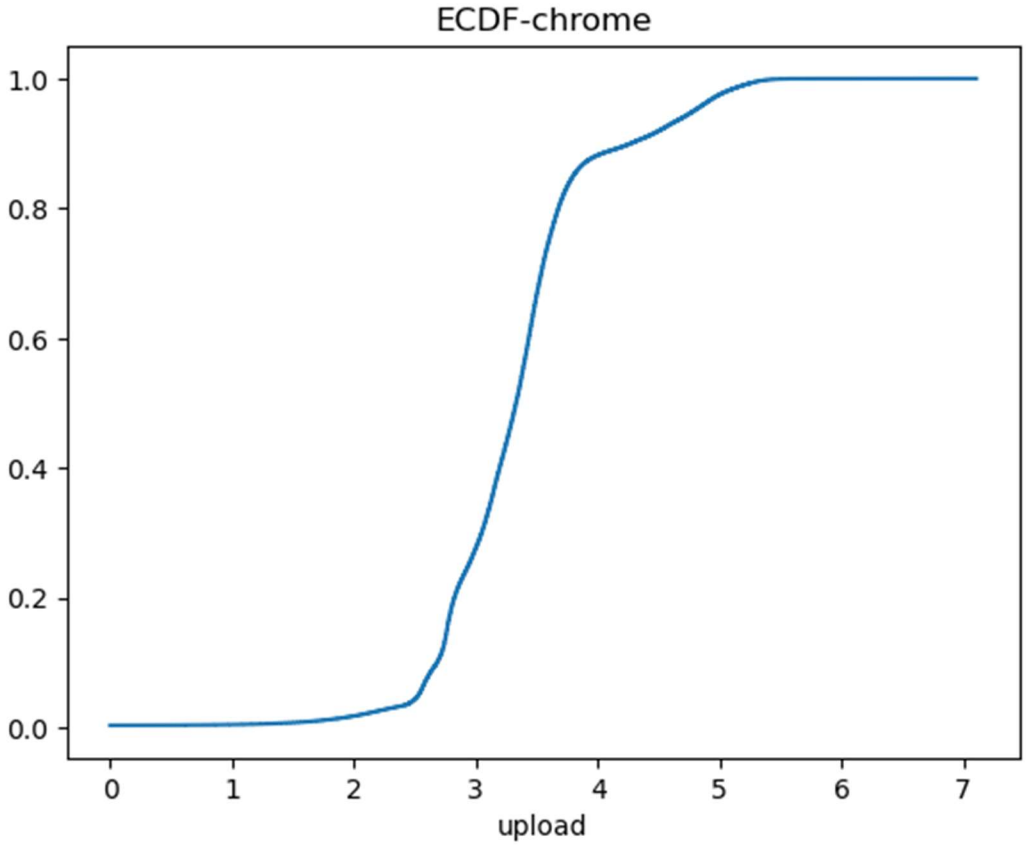
No caso dos dispositivos Smart-TV podemos observar uma altíssima concentração de valores em 0, indicando que possivelmente há um nível altíssimo de ociosidade nestes dispositivos, sendo utilizados esporadicamente.

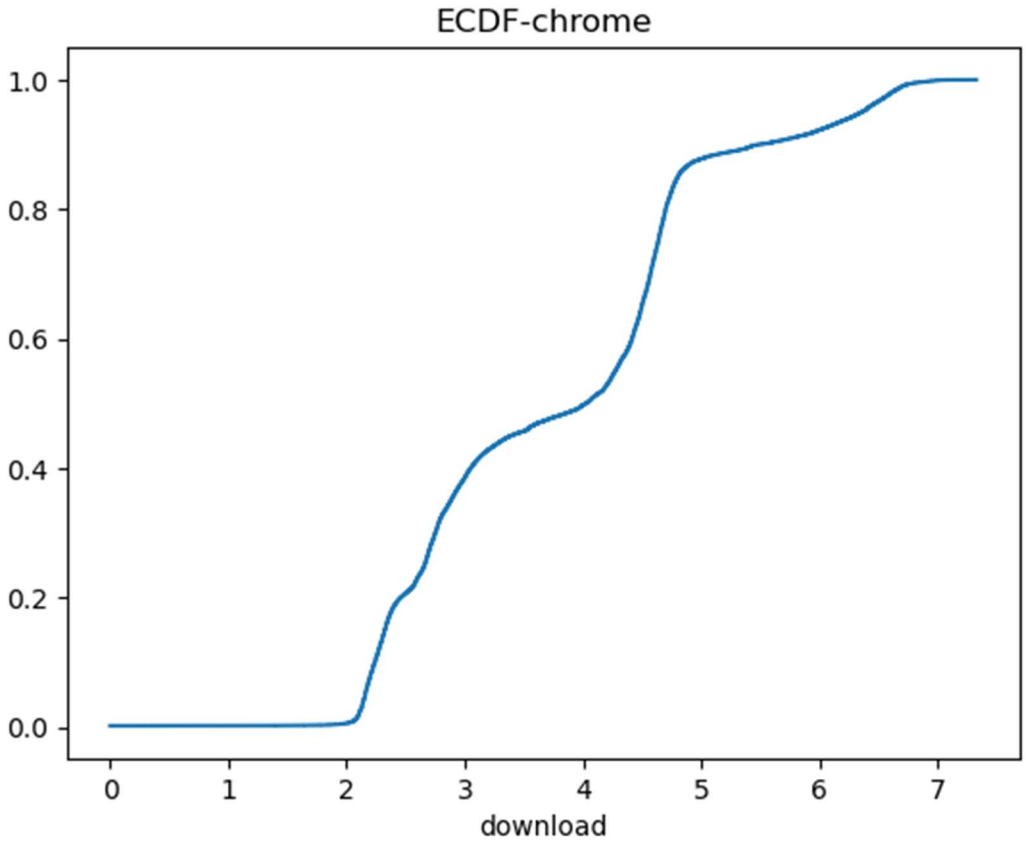
Para as taxas de upload, pode-se notar que para valores diferentes de 0 há uma distribuição sem uma concentração aparente em um determinado valor. Já para as taxas de download há uma concentração em dois intervalos distintos.

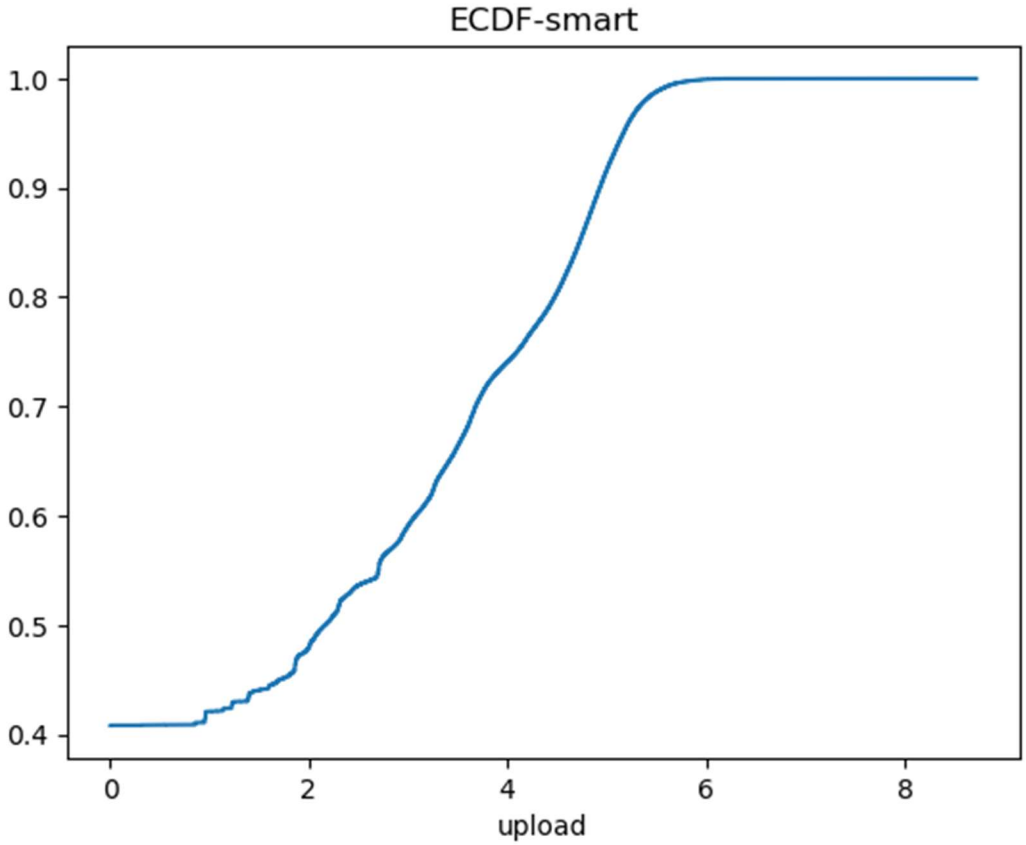
4. Função distribuição empírica

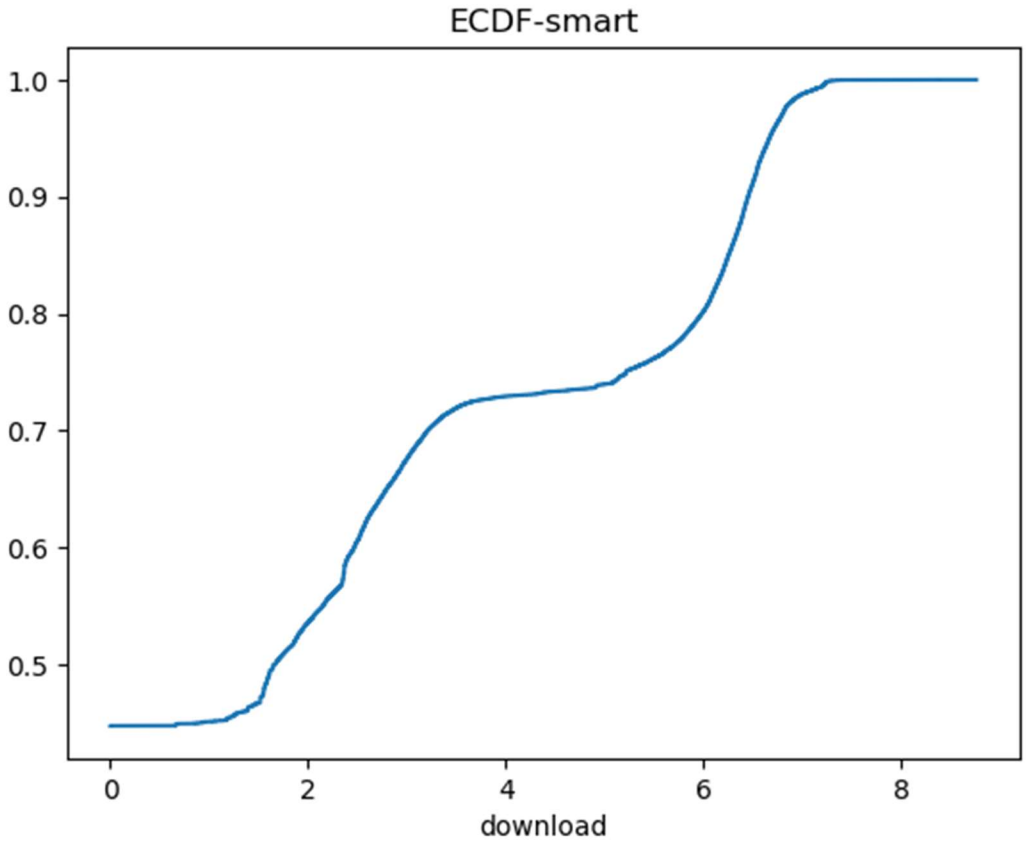
Os parâmetros da distribuição foram calculados com o uso da biblioteca statsmodels, que se trata de uma biblioteca útil para ferramentas relacionadas à modelos estatísticos. Neste caso utiliza-se o método ECDF, como demonstrado abaixo.

```
ecdf = sm.distributions.ECDF(chrome['bytes_up'])

x = np.linspace(min(chrome['bytes_up']), max(chrome['bytes_up']), num = 1000)
y = ecdf(x)
```


ECDF-chrome
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
7
upload


ECDF-chrome
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
7
download


ECDF-smart
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
2
4
6
8
upload


ECDF-smart
1.0
0.9
0.8
0.7
0.6
0.5
0
2
4
6
8
download

Com o uso dos gráficos das ECDFs, podemos notar que cerca de 90% dos dados para taxa de upload em dispositivos Chromecast em diferentes horários estão abaixo de 4 e cerca de 30% estão abaixo de 3.

Além disso, podemos perceber para a taxa de download que cerca de 0% dos mesmos dispositivos em horários variados esta taxa está abaixo de 2, enquanto cerca de 50% está abaixo de 3 e cerca de 90% está abaixo de 5.

Nos dispositivos Smart-TV, nota-se que para ambas a taxa de upload e download, cerca de 40% dos dados estão zerados, indicando que estes dispositivos em diferentes horários se apresentam ociosos em cerca de 40% dos dados medidos. Para a taxa de upload a ECDF parece crescer de maneira relativamente constante, enquanto para o download há uma aparente "barriga", de forma que cerca de 70% dos dados estão abaixo de 3,5 e 100% abaixo de 6.

- Estatísticas por horário

Estas estatísticas têm como objetivo a análise dos dados para cada tipo de dispositivo de acordo com o horário, em que foram gerados, na esperança de identificar padrões e informações interessantes sobre as taxas de upload e download em relação ao horário do dia.

Primeiramente é feita uma manipulação dos dados, de forma a se obter a se gerar uma coluna que identifica a hora do dia em que cada entrada foi gerada baseado na coluna "date-hour".

Seção 3: Estatísticas por horário

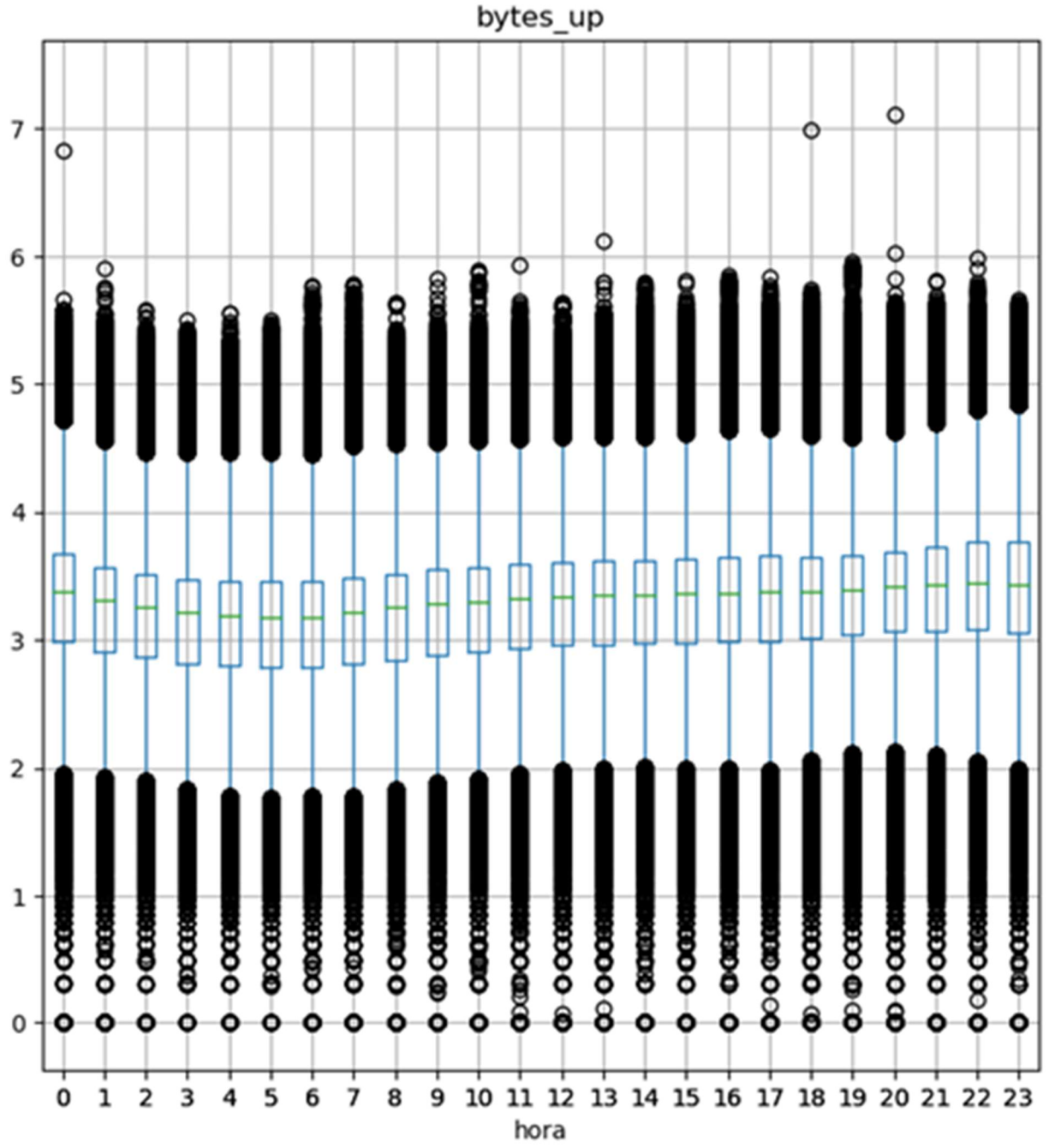
```
chrome['hora'] = chrome.date_hour.str[11:13]
chrome['hora'] = pd.to_numeric(chrome['hora'])
chrome
```

1. Boxplots por horário

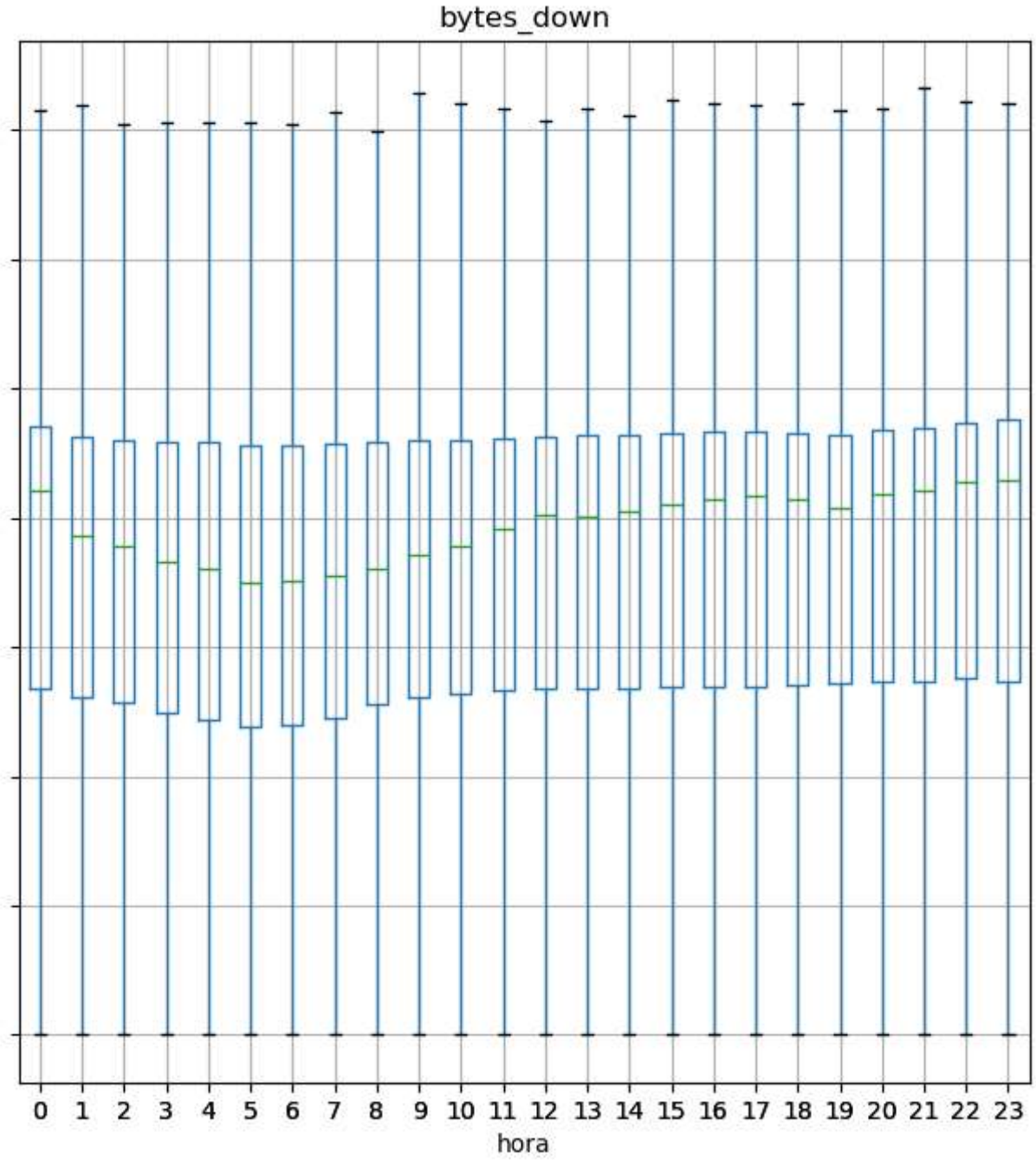
Os boxplots foram gerados com uso do método boxplot do pandas novamente, desta vez separados por hora.

```
chrome.boxplot(column = ['bytes_up', 'bytes_down'], by = 'hora', figsize = (16, 8) )
plt.title('Chromecast')
plt.show()
```

Chromecast

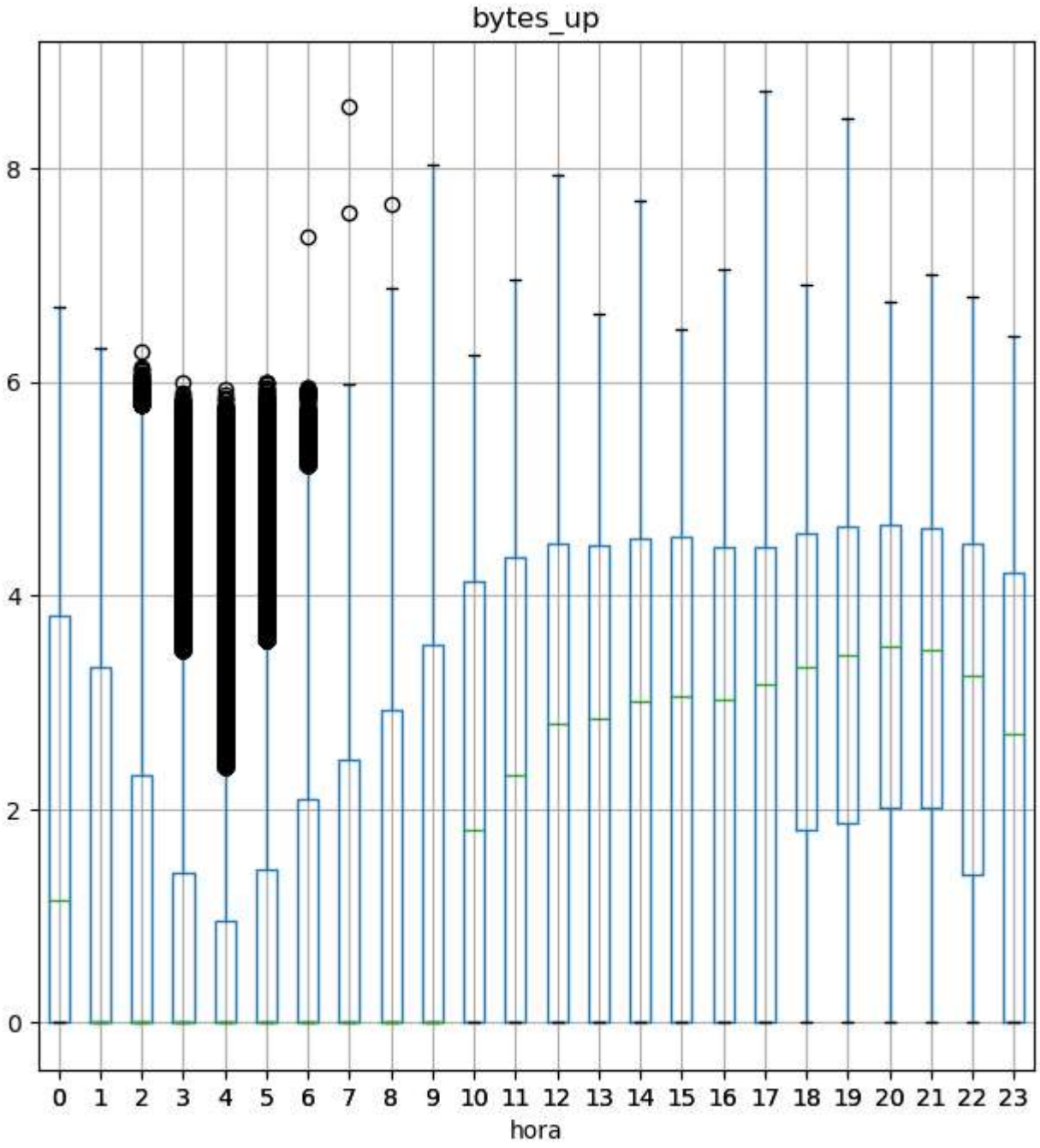


Podemos notar pela divisão dos boxplots dos dispositivos Chromecast por hora que os horários parecem manter o mesmo nível de concentração e o comportamento aparenta similar. Contudo, nota-se que há uma queda geral nas taxas na madrugada, das 3 às 8, com um crescimento no período da noite, a partir das 20.



No caso das taxas de download para os dispositivos Chromecast, nota-se um mesmo comportamento de diminuição pela madrugada e aumento no período da noite, no entanto é possível notar que esse deslocamento do IQR em relação ao horário é menor em proporção, mas com um deslocamento razoável da mediana.

Smart-TV



Com dispositivos Smart-TV, as taxas de upload e download apresentam características bem semelhantes, na sua variação de acordo com a hora. No período da madrugada, a queda é ainda mais aparente, com a presença de outliers nos máximos da vizinhança e a mediana situada em zero.

Em particular, para a taxa de download, até mesmo o 75-percentil está em zero. Além disso, podemos observar que no período da noite, apesar da mediana e os máximos se manterem relativamente constantes, há um deslocamento de quase duas unidades do 25-

percentil, sugerindo que este seja o momento de menor ociosidade de uso de Internet destes dispositivos, com uma altíssima ociosidade no período da madrugada.

2. Média, variância, desvio padrão e mediana

Nesta etapa são calculadas estatísticas básicas para as taxas de upload e download de cada dispositivo para cada horário. Em seguida são traçados gráficos dessas estatísticas em função do horário.

Primeiramente se utilizam métodos da biblioteca numpy para calcular as estatísticas de cada horário, com a criação de um array para cada estatística, de forma que o primeiro valor de cada array representa a estatística na hora 0 e o último representa a estatística na hora 23. Isto é feito como na demonstração abaixo para cada caso de tipo de dispositivo e envio/recibo de dados.

```
horas = []
ucmedias = []
ucvariancias = []
ucdesvios = []
ucmedianas = []
for i in range(24):
    horas.append(i)
    chromei = chrome.loc[chrome['hora'] == i, ['bytes_up']]
    ucmedias.append(np.mean(a=chromei,axis=0).item())
    ucvariancias.append(np.var(a=chromei,axis=0).item())
    ucdesvios.append(np.std(a=chromei,axis=0).item())
    ucmedianas.append(np.median(a=chromei, axis=0).item())
```

Chromecast upload

```
Médias:
[3.432 3.322 3.243 3.202 3.178 3.159 3.157 3.201 3.242 3.286 3.298 3.321
 3.348 3.355 3.363 3.381 3.399 3.408 3.401 3.418 3.468 3.494 3.522 3.508]

Variâncias:
[0.631 0.482 0.341 0.307 0.315 0.291 0.304 0.337 0.39  0.397 0.407 0.408
 0.405 0.429 0.425 0.435 0.478 0.496 0.475 0.484 0.494 0.543 0.596 0.693]

Desvios:
[0.795 0.694 0.584 0.554 0.561 0.539 0.551 0.581 0.624 0.63  0.638 0.638
 0.636 0.655 0.652 0.66  0.692 0.704 0.689 0.696 0.703 0.737 0.772 0.833]

Medianas:
[3.379 3.317 3.257 3.211 3.184 3.172 3.173 3.212 3.251 3.28  3.301 3.328
 3.341 3.349 3.348 3.363 3.368 3.373 3.379 3.388 3.415 3.425 3.444 3.428]
```

Chromecast download

```
Médias:
[3.953 3.776 3.686 3.637 3.618 3.571 3.566 3.616 3.653 3.697 3.708 3.742
 3.779 3.785 3.798 3.833 3.866 3.88  3.858 3.853 3.922 3.968 4.036 4.053]

Variâncias:
[2.065 1.741 1.457 1.407 1.414 1.377 1.371 1.427 1.487 1.509 1.519 1.515
 1.538 1.589 1.582 1.624 1.709 1.73  1.661 1.66  1.751 1.861 1.969 2.159]

Desvios:
[1.437 1.32  1.207 1.186 1.189 1.173 1.171 1.194 1.219 1.229 1.232 1.231
 1.24  1.26  1.258 1.274 1.307 1.315 1.289 1.288 1.323 1.364 1.403 1.469]

Medianas:
[4.209 3.864 3.784 3.655 3.606 3.494 3.516 3.557 3.602 3.713 3.786 3.913
 4.02  4.007 4.052 4.108 4.146 4.169 4.143 4.08  4.178 4.215 4.273 4.286]
```

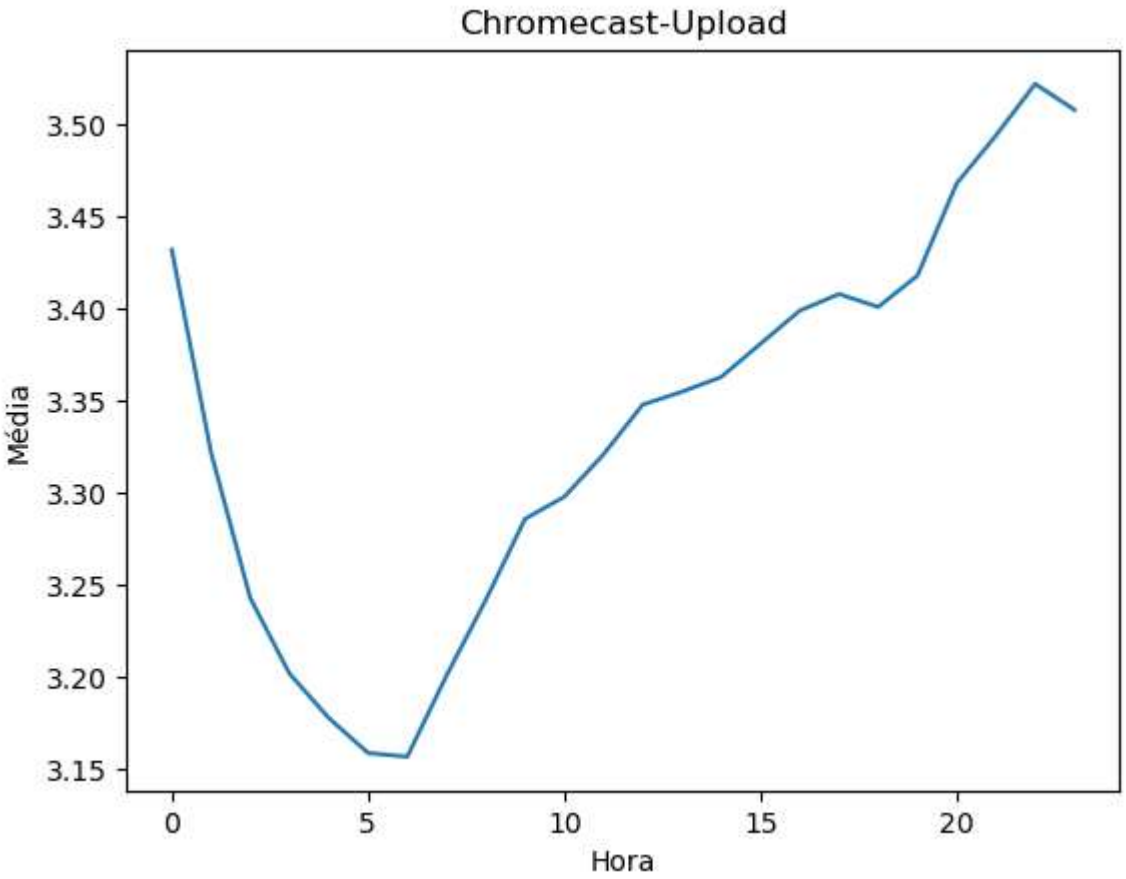
Smart-TV upload

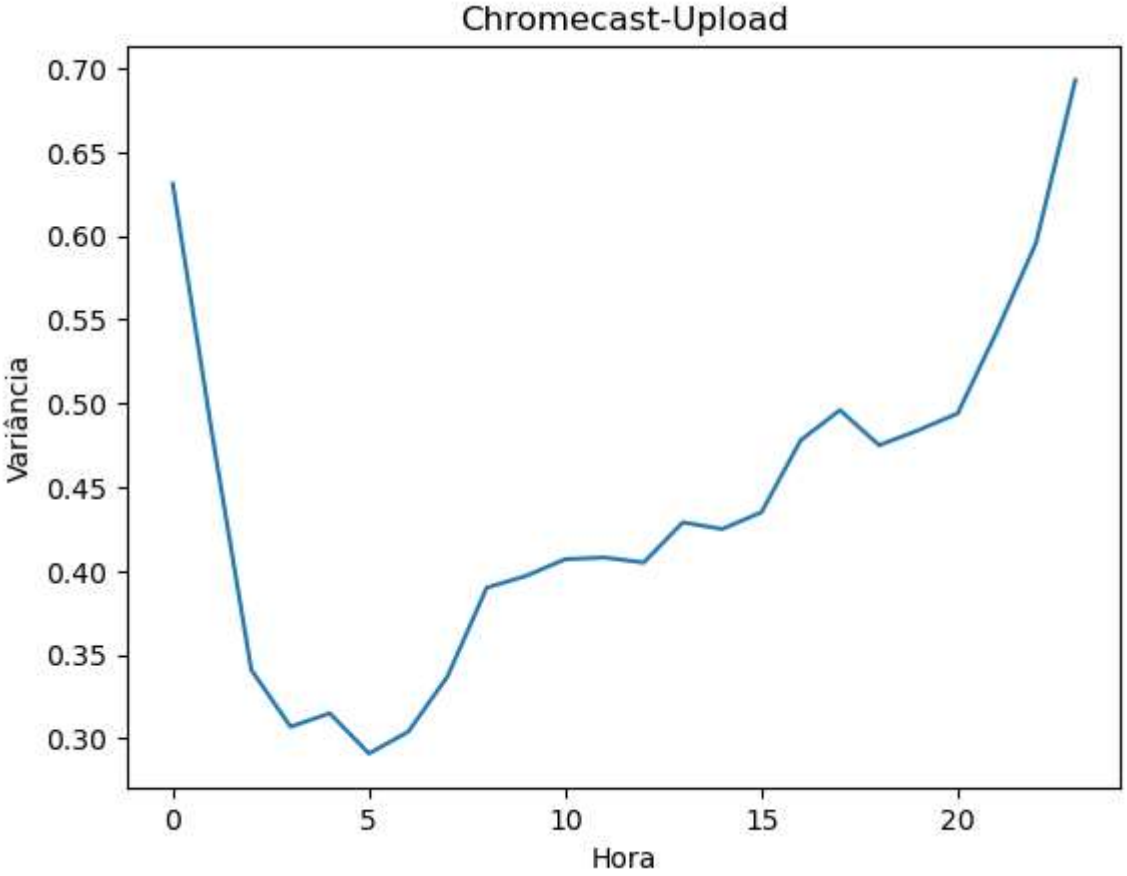
```
Médias:
[1.894 1.467 1.154 0.893 0.769 0.875 1.025 1.197 1.392 1.717 2.017 2.266
 2.474 2.489 2.557 2.606 2.62  2.744 2.951 3.053 3.124 3.103 2.837 2.375]

Variâncias:
[4.157 3.759 3.152 2.465 2.056 2.345 2.64  3.001 3.529 3.971 4.236 4.268
 4.158 4.138 4.218 4.116 3.867 3.591 3.35  3.276 3.169 3.132 3.459 3.939]

Desvios:
[2.039 1.939 1.775 1.57  1.434 1.531 1.625 1.732 1.878 1.993 2.058 2.066
 2.039 2.034 2.054 2.029 1.966 1.895 1.83  1.81  1.78  1.77  1.86  1.985]

Medianas:
[1.142 0.    0.    0.    0.    0.    0.    0.    0.    1.813 2.318
 2.808 2.852 3.004 3.052 3.027 3.169 3.333 3.438 3.531 3.484 3.258 2.71 ]
```

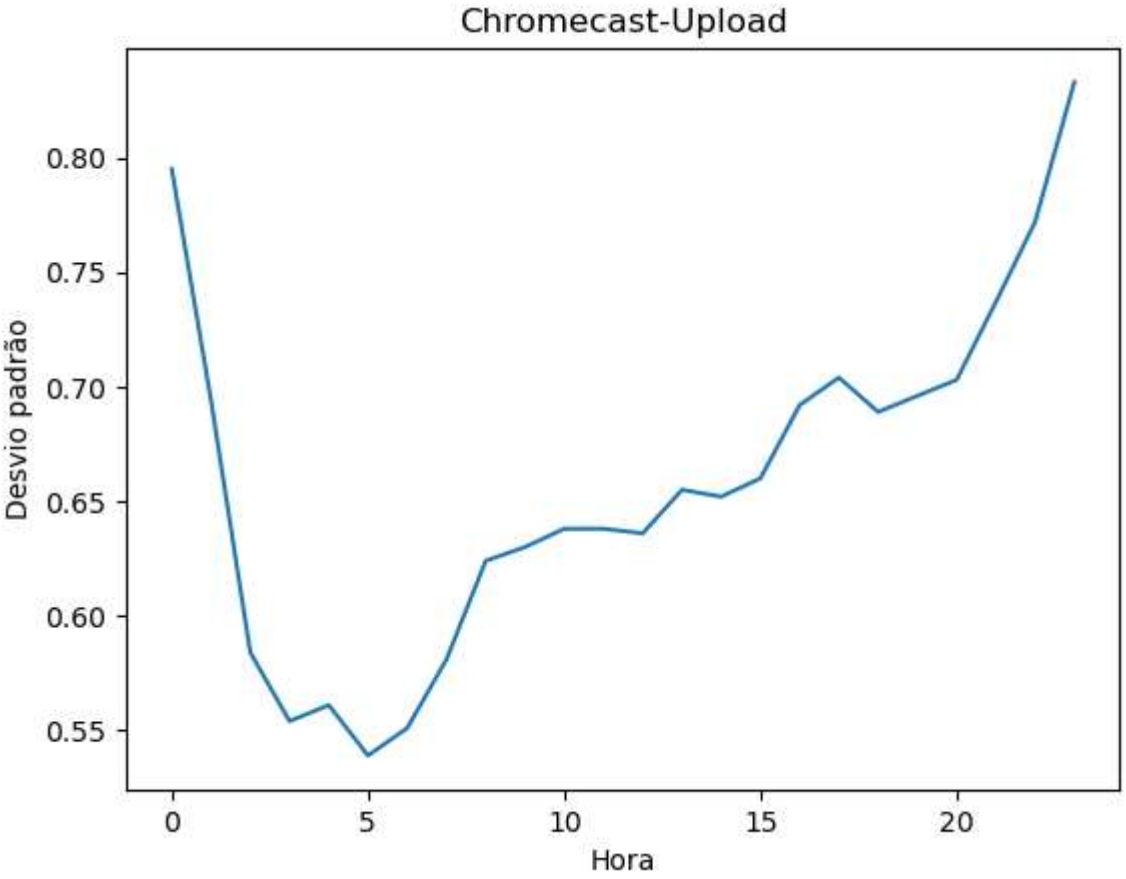
Smart-TV download

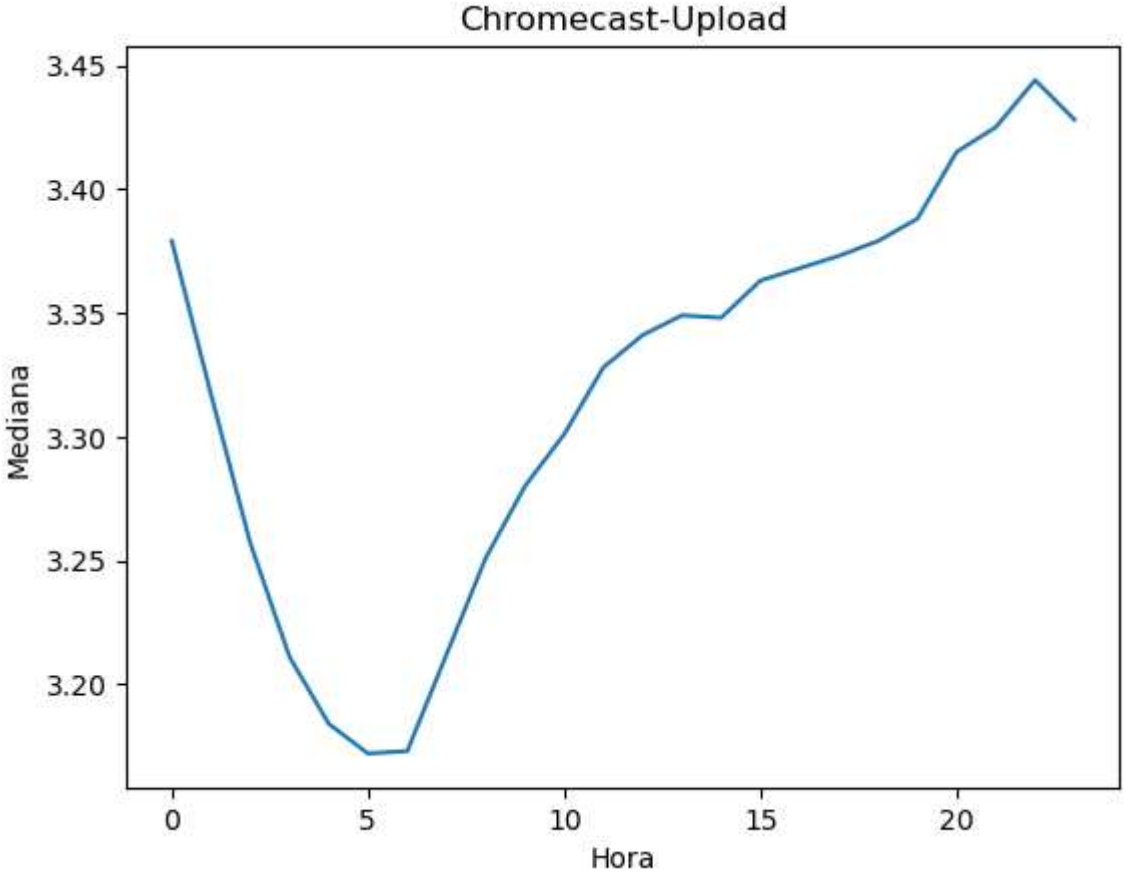
```
Médias:
[2.105 1.602 1.228 0.897 0.736 0.891 1.073 1.245 1.478 1.868 2.23  2.526
 2.775 2.779 2.876 2.92  2.876 2.958 3.191 3.322 3.396 3.366 3.061 2.586]

Variâncias:
[6.888 6.052 4.957 3.628 2.887 3.522 4.    4.401 5.324 6.252 6.884 7.055
 7.044 7.002 7.238 7.163 6.759 6.416 6.242 6.293 6.201 6.125 6.289 6.648]

Desvios:
[2.624 2.46  2.226 1.905 1.699 1.877 2.    2.098 2.307 2.5   2.624 2.656
 2.654 2.646 2.69  2.676 2.6   2.533 2.498 2.509 2.49  2.475 2.508 2.578]

Medianas:
[0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    1.308 1.876
 2.352 2.358 2.388 2.441 2.409 2.512 2.705 2.792 2.89  2.851 2.588 2.185]
```

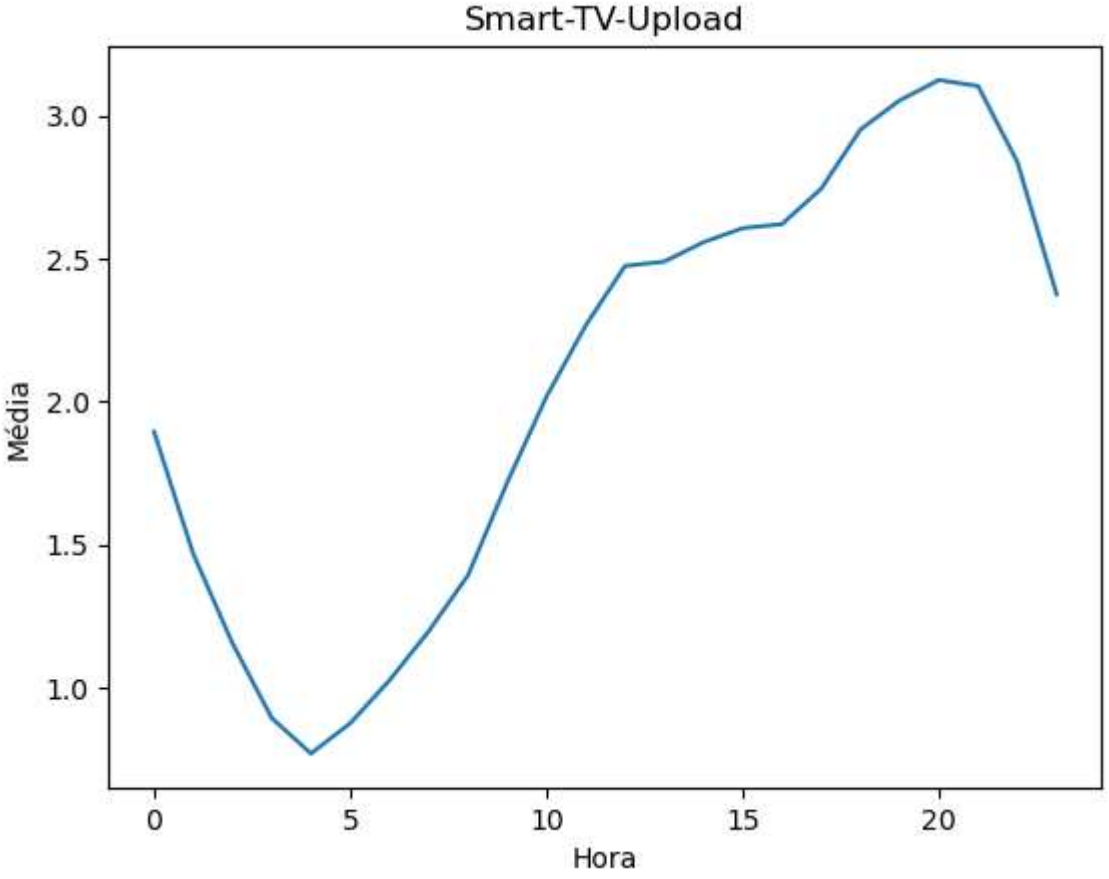
Os gráficos são plotados com uso dos arrays, utilizando a biblioteca matplotlib.

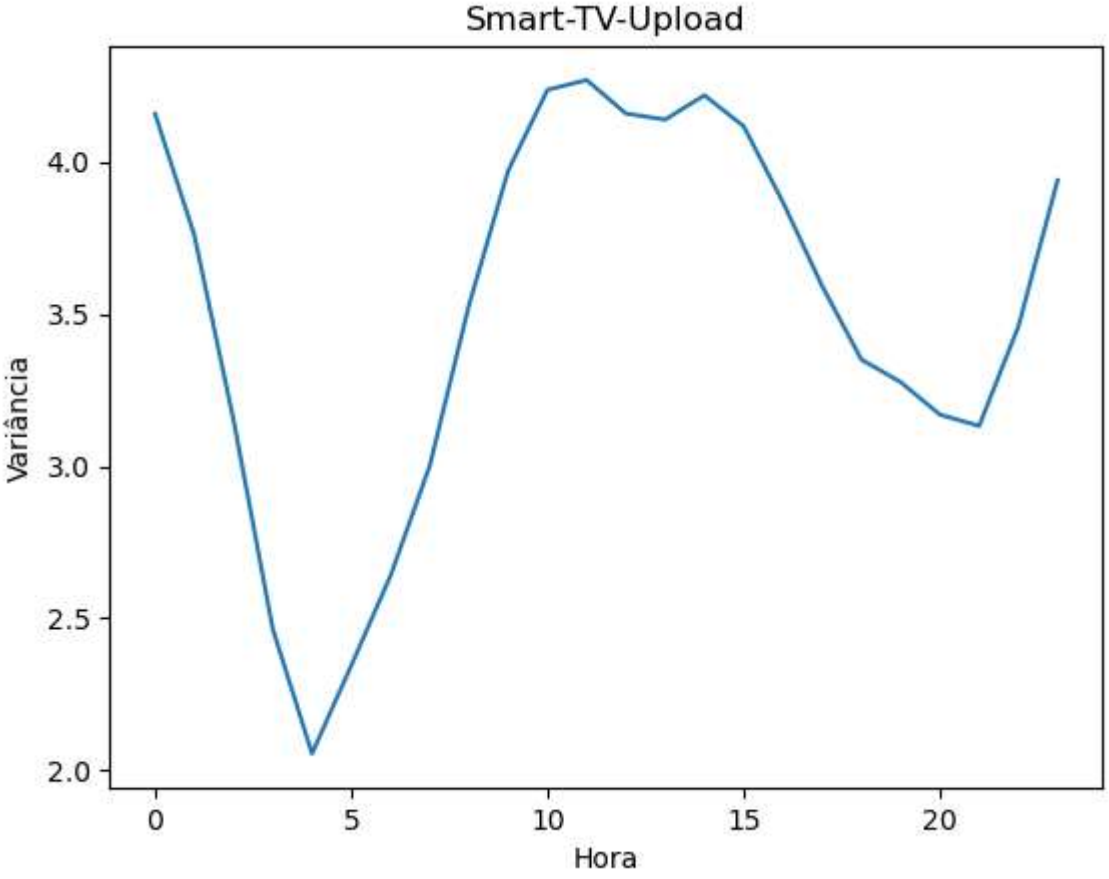


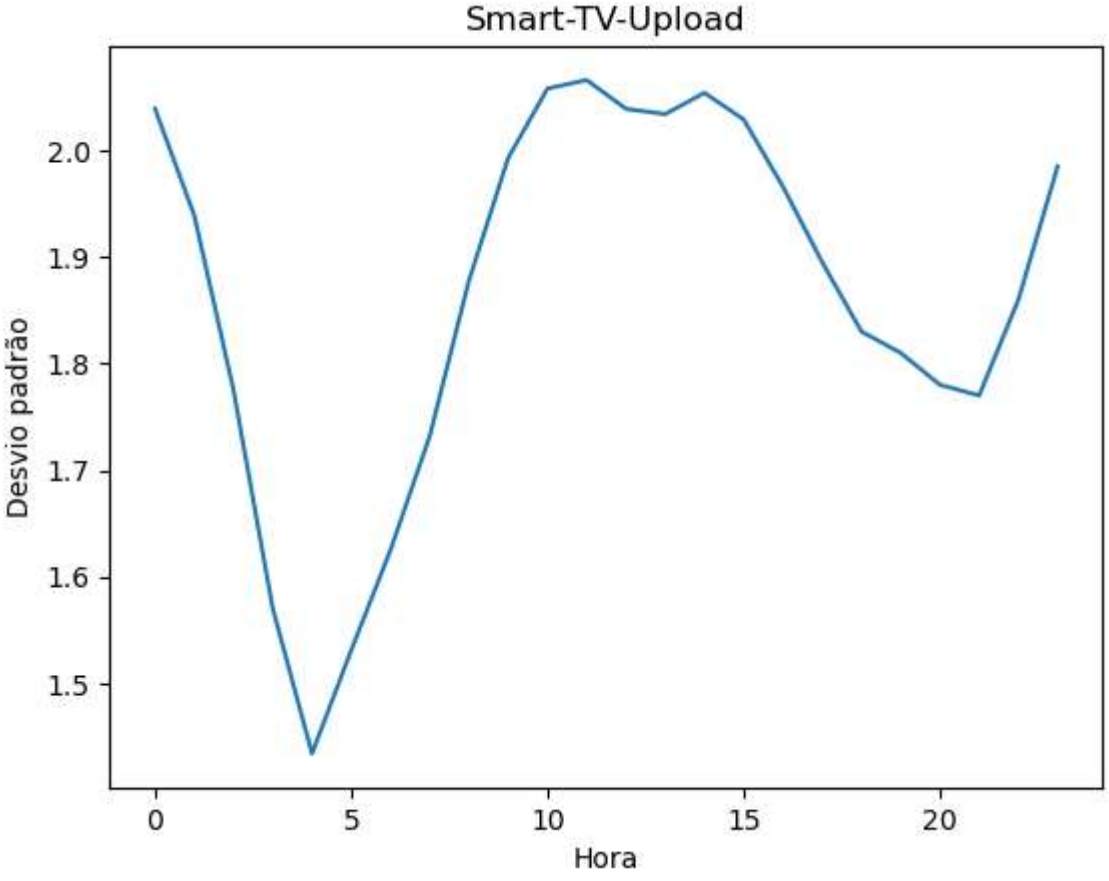

Chromecast-Upload
Variância
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
Hora

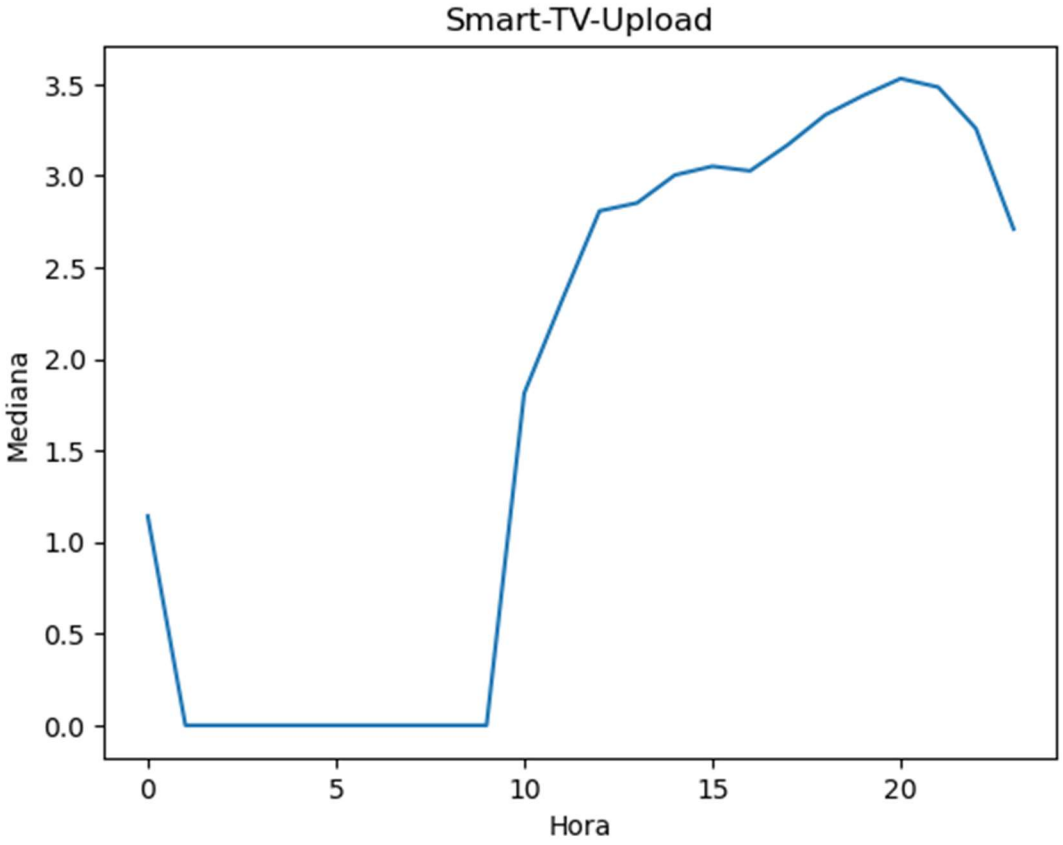

Chromecast-Upload
Desvio padrão
0.80
0.75
0.70
0.65
0.60
0.55
0
5
10
15
20
Hora

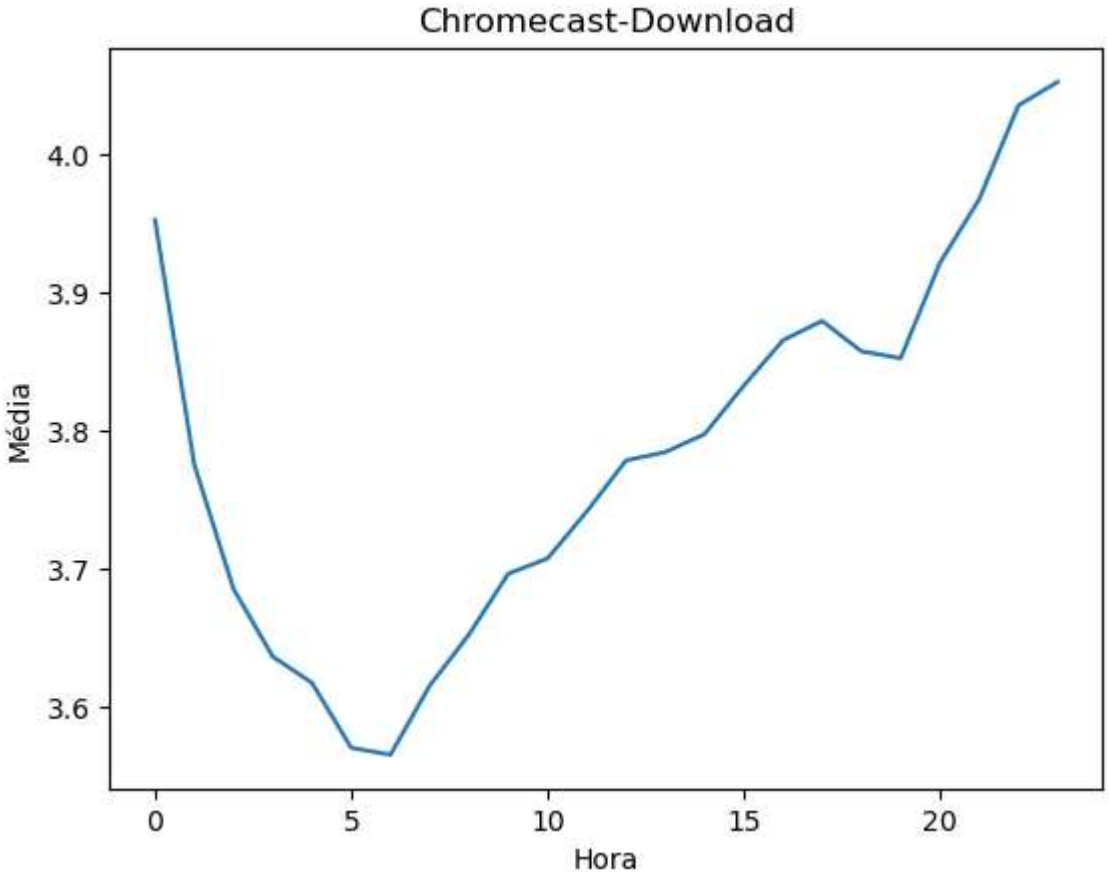

Chromecast-Upload
Mediana
3.45
3.40
3.35
3.30
3.25
3.20
0
5
10
15
20
Hora

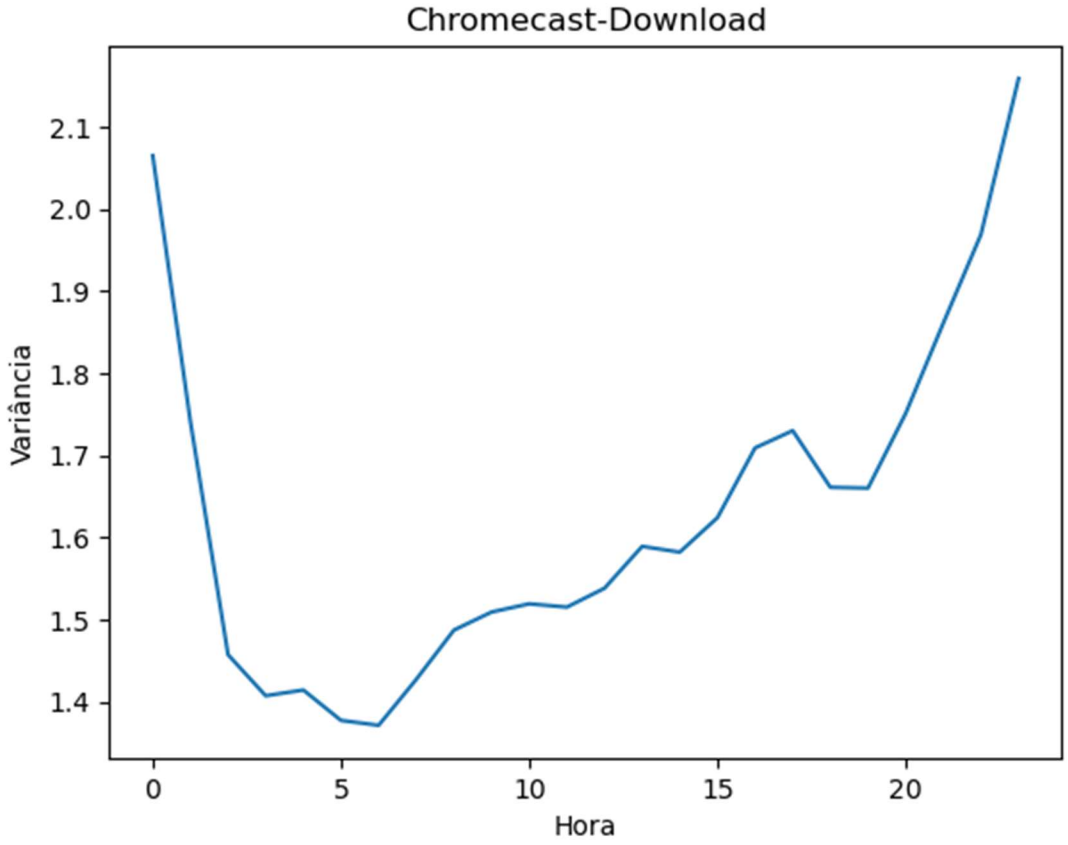

Smart-TV-Upload
Média
3.0
2.5
2.0
1.5
1.0
0
5
10
15
20
Hora

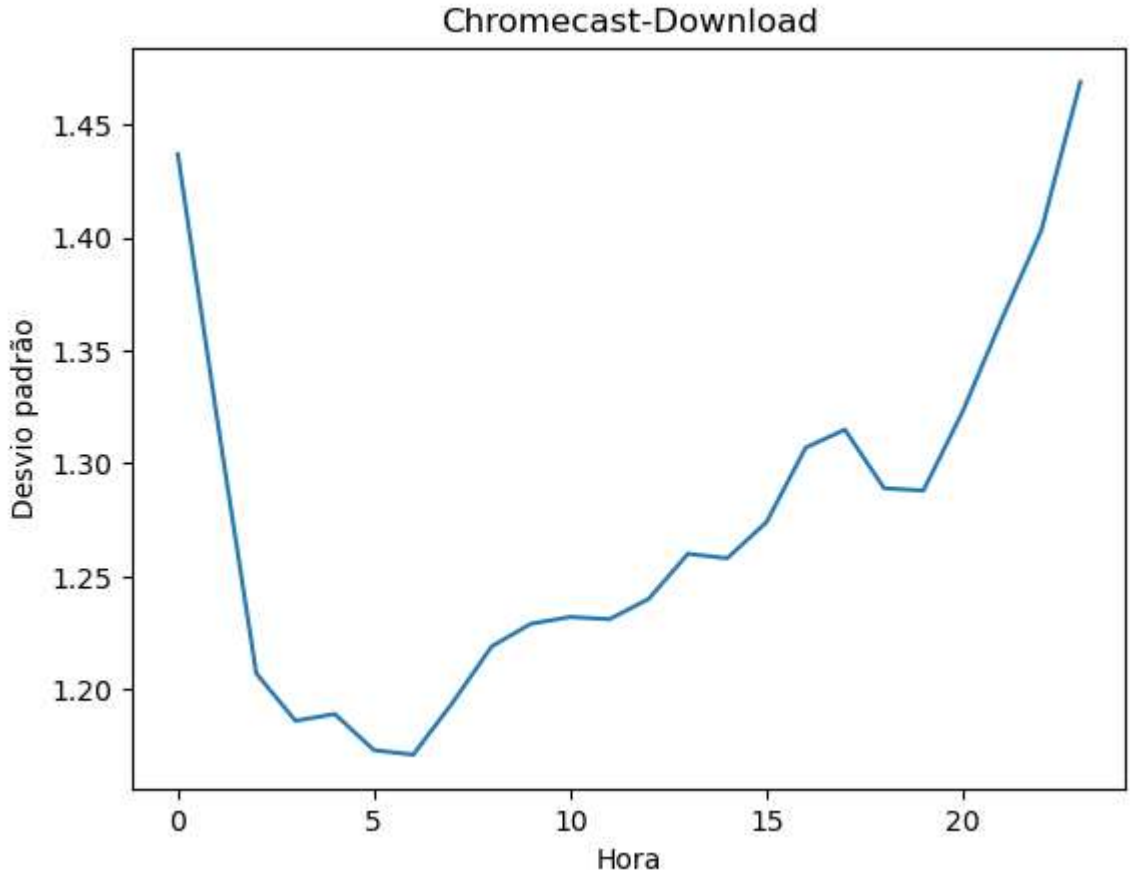

Smart-TV-Upload
Variância
4.0
3.5
3.0
2.5
2.0
0
5
10
15
20
Hora

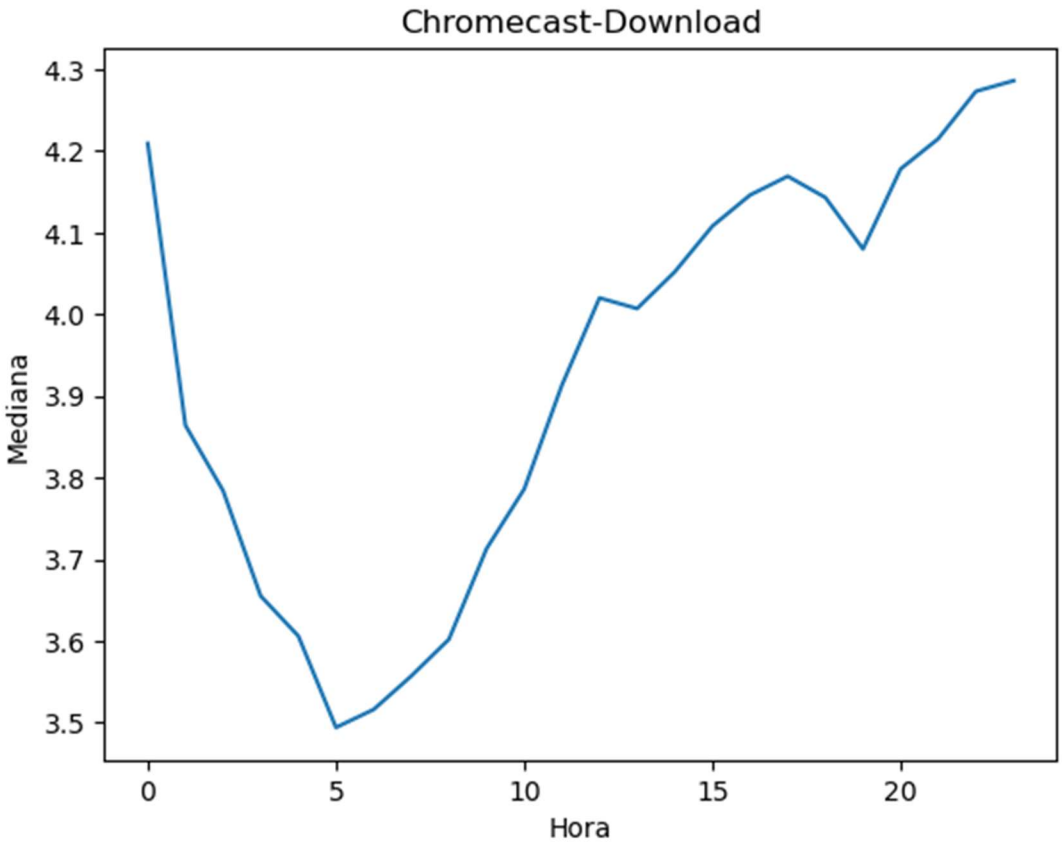

Smart-TV-Upload
Desvio padrão
2.0
1.9
1.8
1.7
1.6
1.5
0
5
10
15
20
Hora

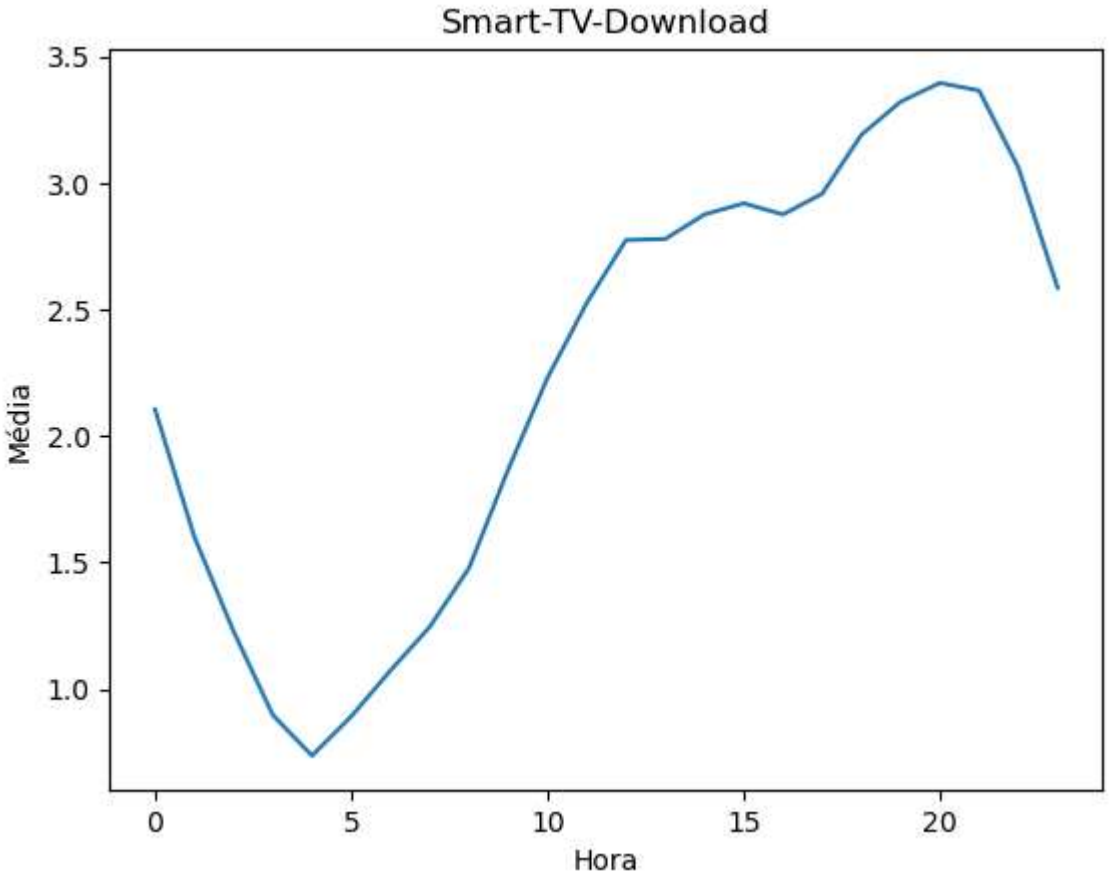

Smart-TV-Upload
Mediana
Hora
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
5
10
15
20

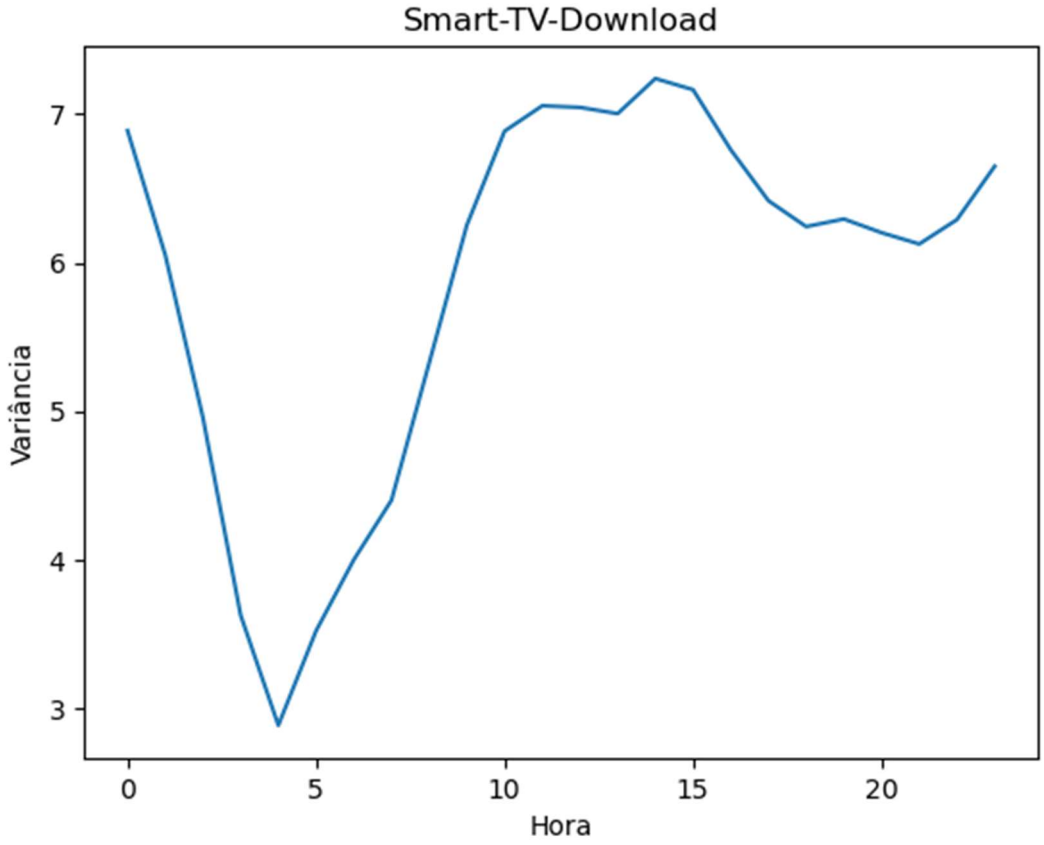

Chromecast-Download
Média
Hora
3.6
3.7
3.8
3.9
4.0
0
5
10
15
20

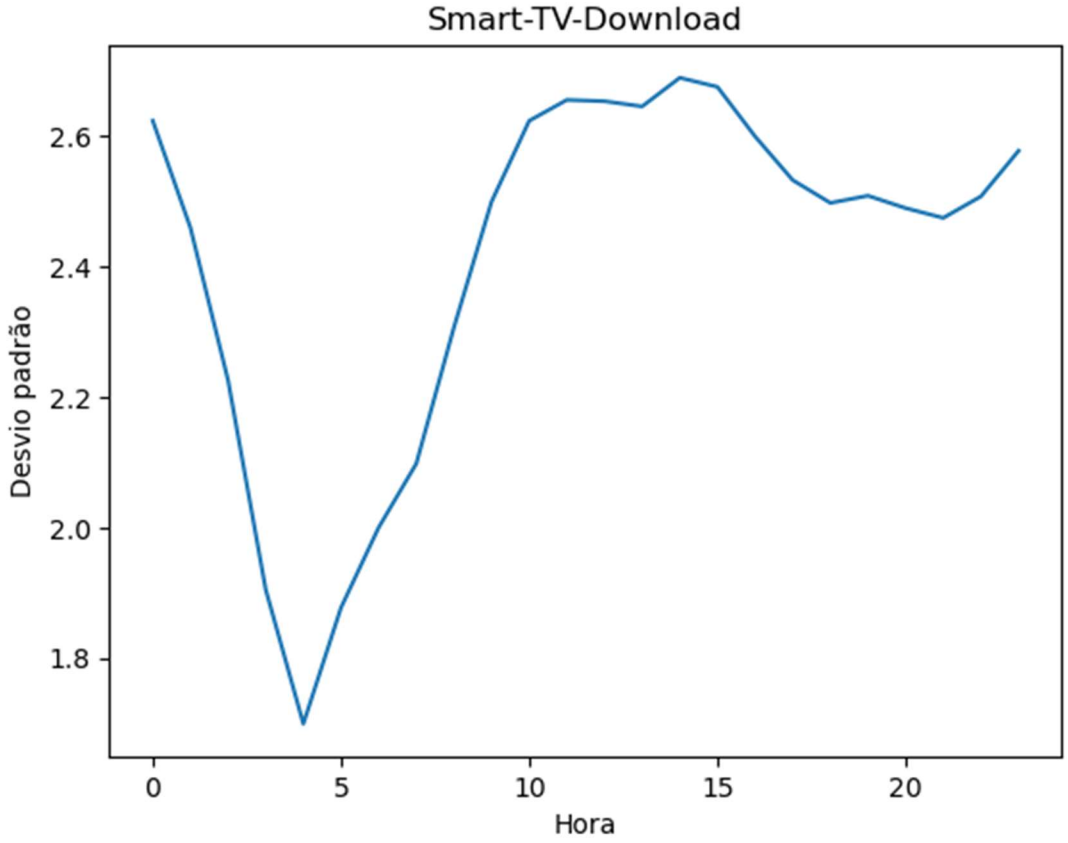

Chromecast-Download
Variância
2.1
2.0
1.9
1.8
1.7
1.6
1.5
1.4
0
5
10
15
20
Hora

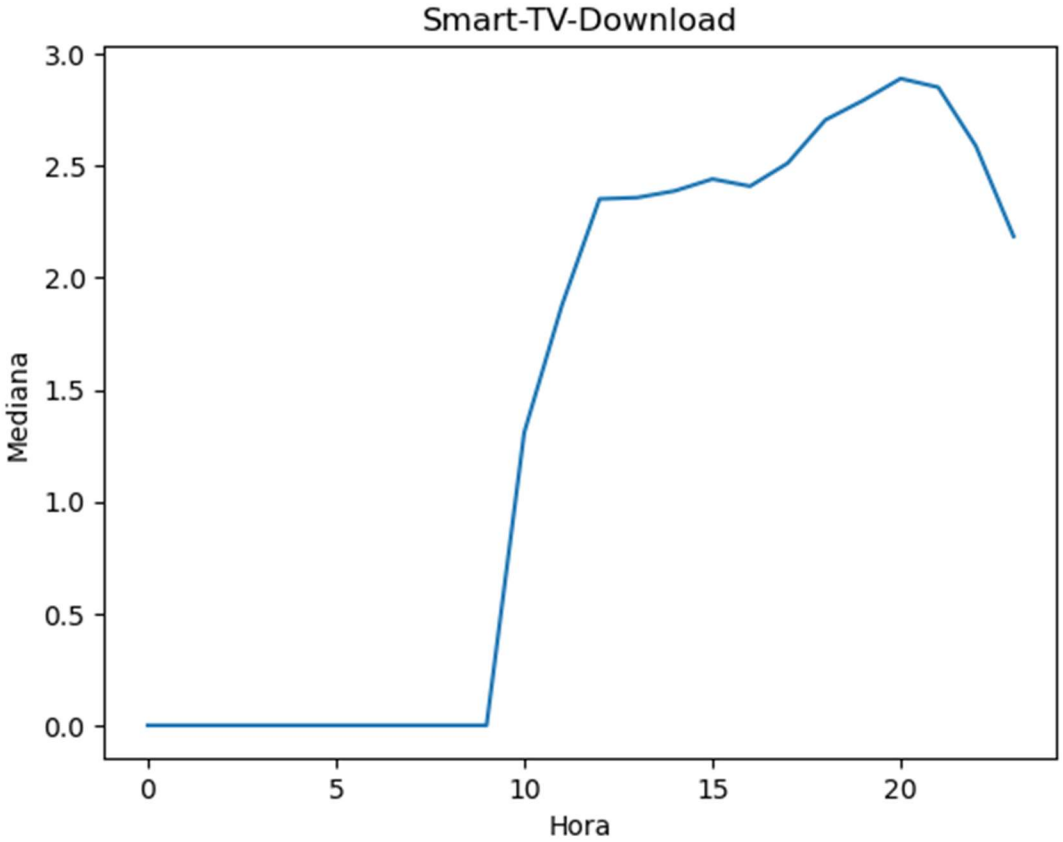

Chromecast-Download
Desvio padrão
1.45
1.40
1.35
1.30
1.25
1.20
0
5
10
15
20
Hora


Chromecast-Download
Mediana
Hora
4.3
4.2
4.1
4.0
3.9
3.8
3.7
3.6
3.5
0
5
10
15
20


Smart-TV-Download
Média
Hora
3.5
3.0
2.5
2.0
1.5
1.0
0
5
10
15
20


Smart-TV-Download
Variância
Hora
7
6
5
4
3
0
5
10
15
20


Smart-TV-Download
Desvio padrão
Hora
2.6
2.4
2.2
2.0
1.8
0
5
10
15
20



No geral, é possível observar grande similaridade entre o comportamento das médias com suas respectivas medianas. Isto indica um baixo nível de skewness, ou viés e poucos outliers, como havíamos observado antes.

Além disso, é notório o comportamento que havia sido observado anteriormente de queda nas taxas parte da madrugada com pico na parte da noite. A variância e o desvio padrão são menores na parte da noite e maiores na parte da noite, quando há respectivo menor e maior uso.

- Horários com maior valor de tráfego

Primeiramente são determinados os horários com maior média e mediana para cada tipo de dispositivo e cada tipo de taxa, com o uso do método argmax do numpy. A partir daí, são determinados oito datasets.

Os quatro primeiros são representados pelos horários com maior média e mediana de taxa de upload, bem como os horários com maior média e mediana de taxa de download de dispositivos do tipo Smart-TV. Incidentalmente, os quatro primeiros coincidem no mesmo horário, 20 horas, como demonstrado a seguir.

```
usmedianas = np.array(usmedianas)
usmedias = np.array(usmedias)
dsmedianas = np.array(dsmedianas)
dsmedias = np.array(dsmedias)

hora1 = np.argmax(usmedianas)
hora2 = np.argmax(usmedias)
hora3 = np.argmax(dsmedianas)
hora4 = np.argmax(dsmedias)

print(f'Hora1: {hora1}  Hora2: {hora2}  Hora3: {hora3}  Hora4: {hora4}')
```

```
Hora1: 20  Hora2: 20  Hora3: 20  Hora4: 20
```

Sendo assim, é gerado um novo dataframe do pandas com os dados deste horário, que representará os datasets 1, 2, 3 e 4.

```
dataset1 = smart.loc[smart['hora'] == 20]

print(f'Dataset 1/2/3/4:\n\n{dataset1}')
```

Os datasets 5 e 6 também coincidem, sendo eles os de 22 horas, horário de maior média e mediana de taxa de upload em dispositivos do tipo Chromecast, bem como os datasets 7 e 8 que representam 23 horas, o horário de maior média e mediana de taxa de download em dispositivos do tipo Chromecast.

```
ucmedianas = np.array(ucmedianas)
ucmedias = np.array(ucmedias)
dcmedianas = np.array(dcmedianas)
dcmedias = np.array(dcmedias)

hora5 = np.argmax(ucmedianas)
hora6 = np.argmax(ucmedias)
hora7 = np.argmax(dcmedianas)
hora8 = np.argmax(dcmedias)

print(f'Hora5: {hora5}  Hora6: {hora6}  Hora7: {hora7}  Hora8: {hora8}')
```

```
Hora5: 22  Hora6: 22  Hora7: 23  Hora8: 23
```

Sendo assim, são gerados dois novos dataframes do pandas com os dados destes horários.

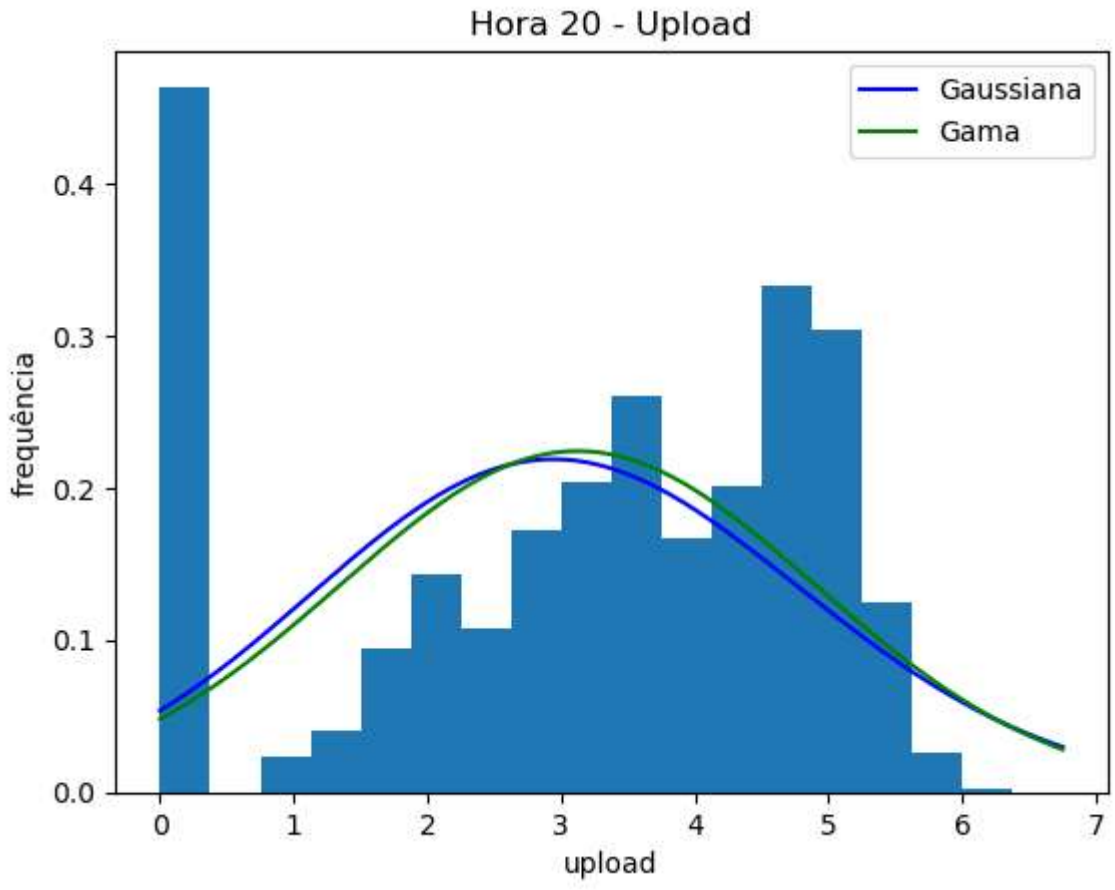
```
dataset5 = chrome.loc[chrome['hora'] == 22]
dataset7 = chrome.loc[chrome['hora'] == 23]

print(f'Dataset 5/6:\n\n{dataset5}\n\nDataset 7/8:\n\n{dataset7}')
```

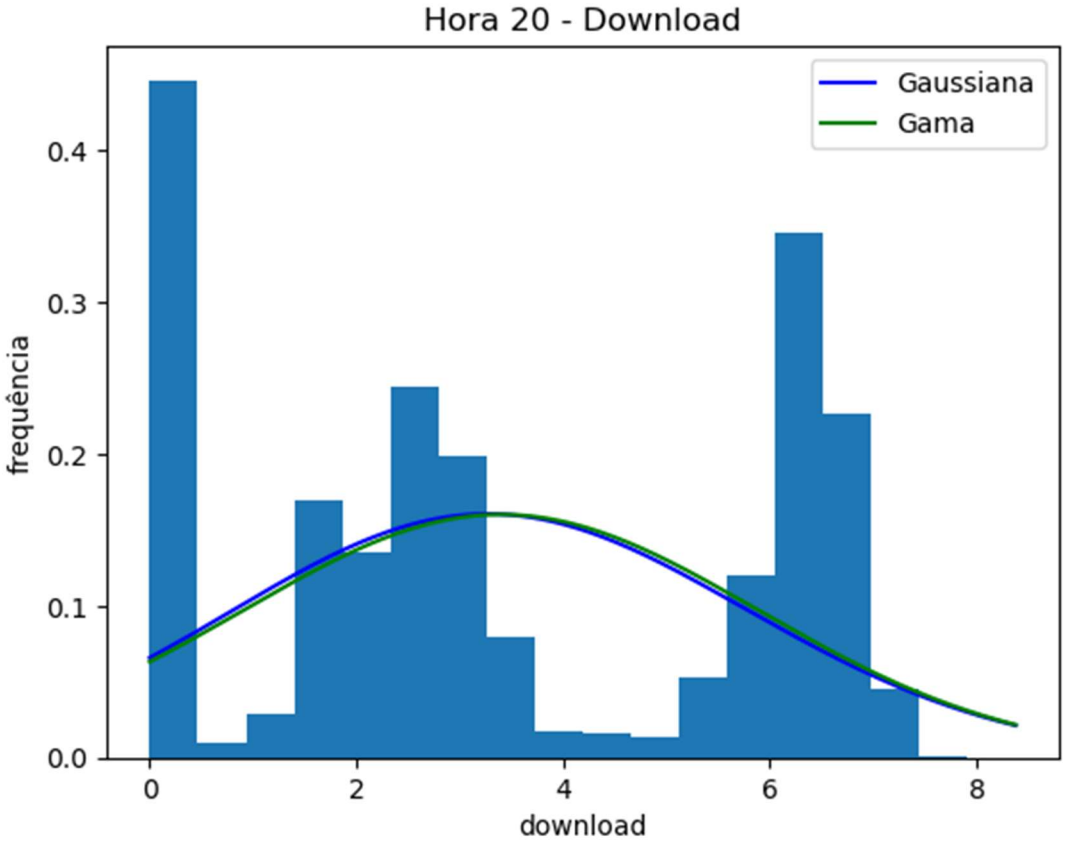
Em seguida são estimados parâmetros para a distribuição Gamma com o uso do método gamma.fit do scipy e é gerado um gráfico com um histograma utilizando o método de Sturges, a função densidade Gaussiana, utilizando os valores de média e desvio padrão encontrados anteriormente, e a função densidade Gamma, para cada um dos datasets, como demonstrado abaixo.

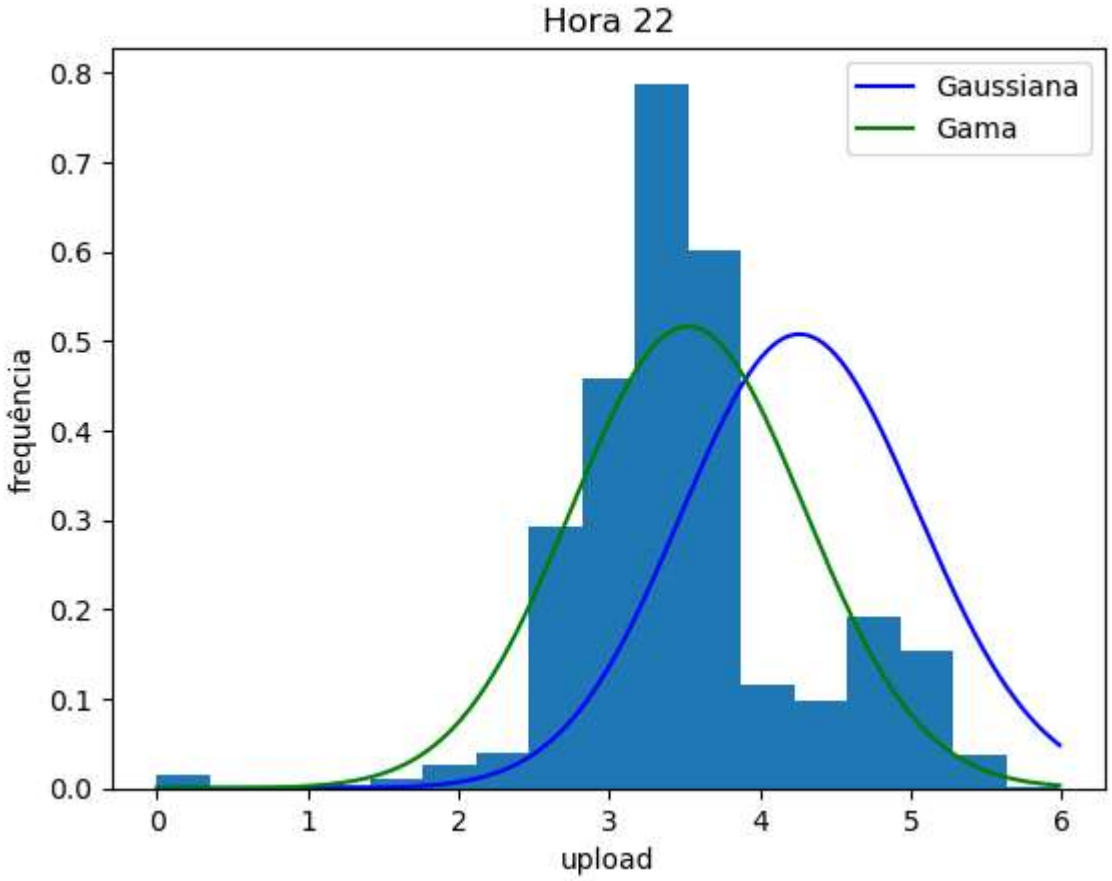
```
alfa1up, posicao1up, beta1up = gamma.fit(dataset1['bytes_up'])
alfa1down, posicao1down, beta1down = gamma.fit(dataset1['bytes_down'])
```

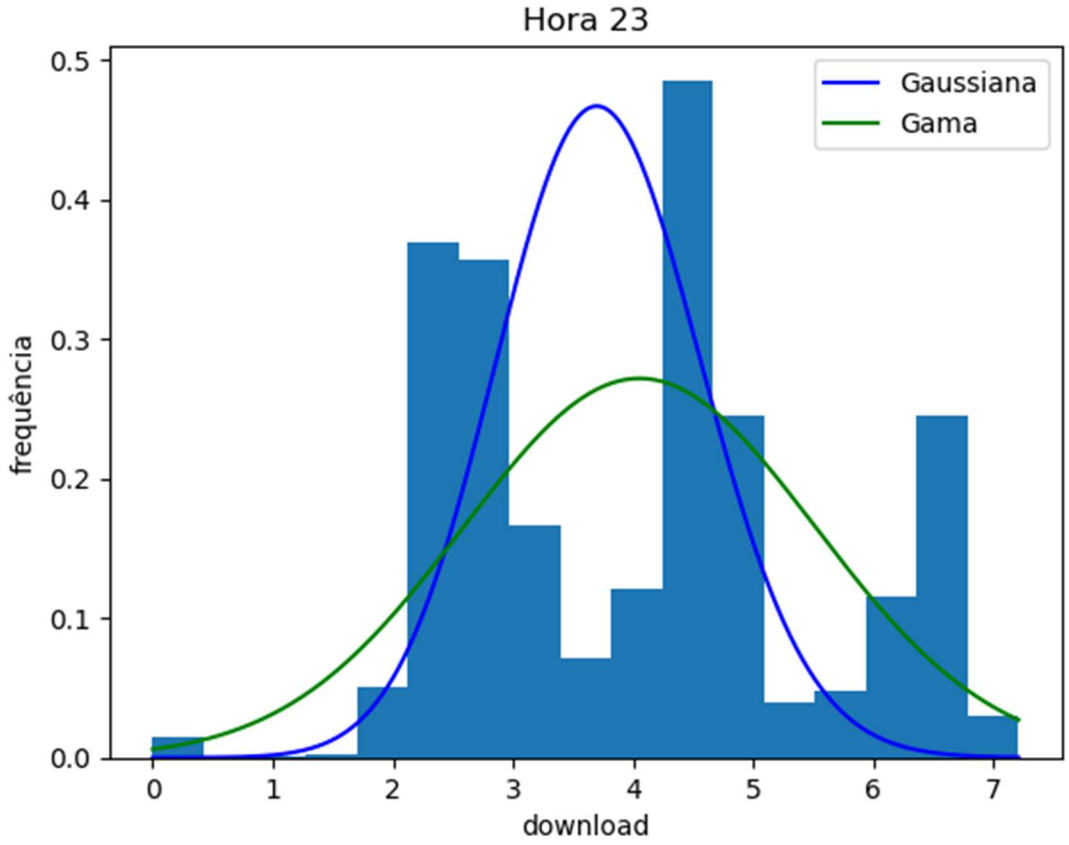
```
eixox = np.linspace(dataset1['bytes_up'].min(), dataset1['bytes_up'].max(), len(dataset1['bytes_up']))

gama = gamma.pdf(eixox, float(alfa1up), float(posicao1up), float(beta1up))

gaussiana = norm.pdf(eixox, float(media1up), float(desvio1up))

plt.title('Hora 20 - Upload')
plt.plot(eixox, gama, label = 'Gama', color = 'b')
plt.plot(eixox, gaussiana, label = 'Gaussiana', color = 'g')
plt.xlabel('upload')
plt.ylabel('frequência')
plt.legend(['Gaussiana', 'Gama'])
histograma = plt.hist(dataset1['bytes_up'], bins = int(1 + 3.3*math.log(212608, 10)), density=True)

plt.show()
```

Os gráficos gerados são:




Hora 20 - Download
Gaussiana
Gama
frequência
download
0.0
0.1
0.2
0.3
0.4
0
2
4
6
8

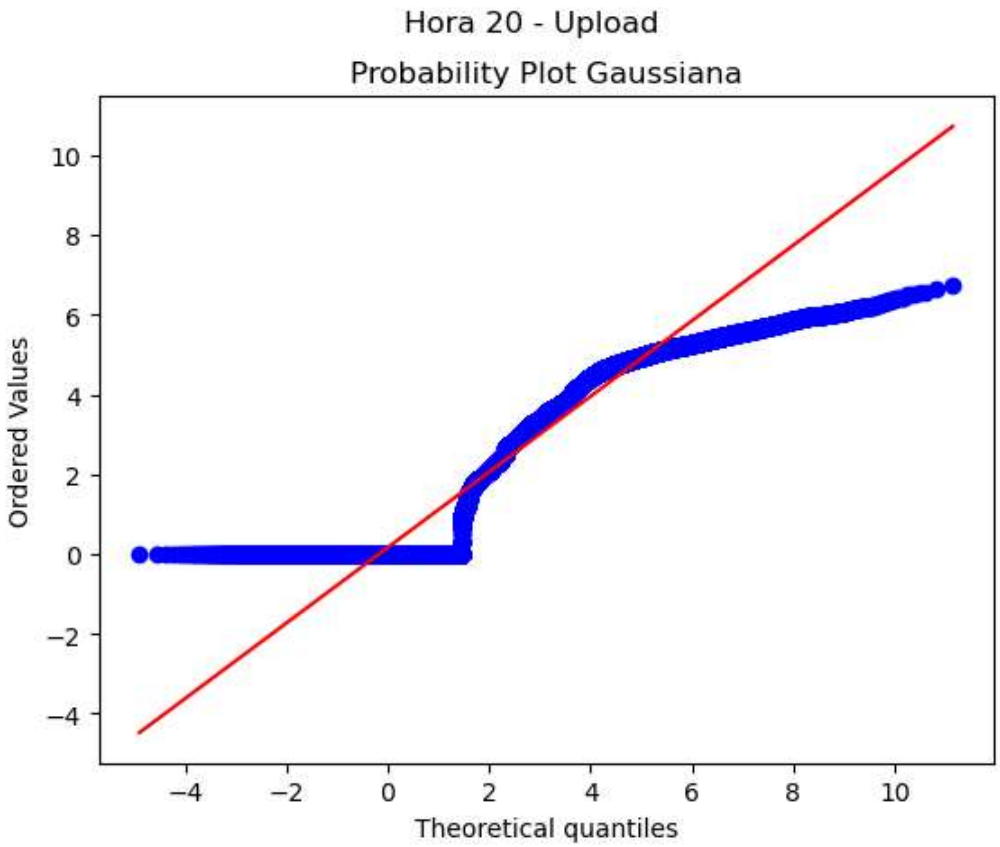

Hora 22
Gaussiana
Gama
frequência
upload
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
5
6

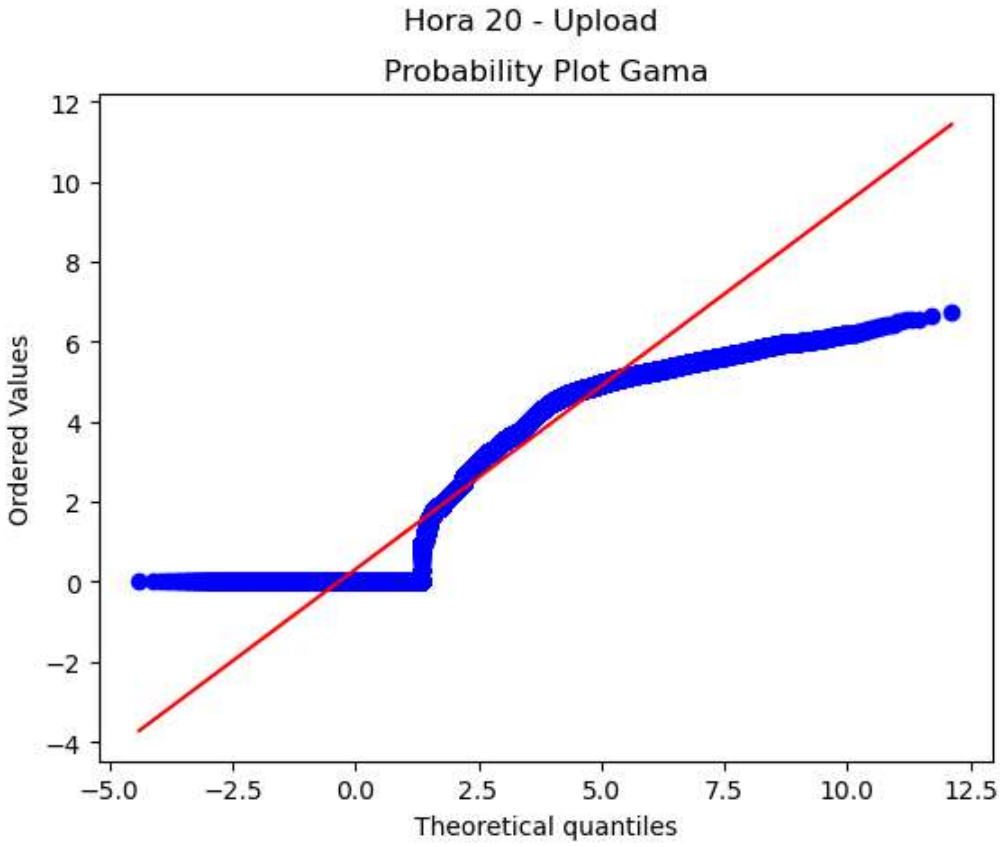


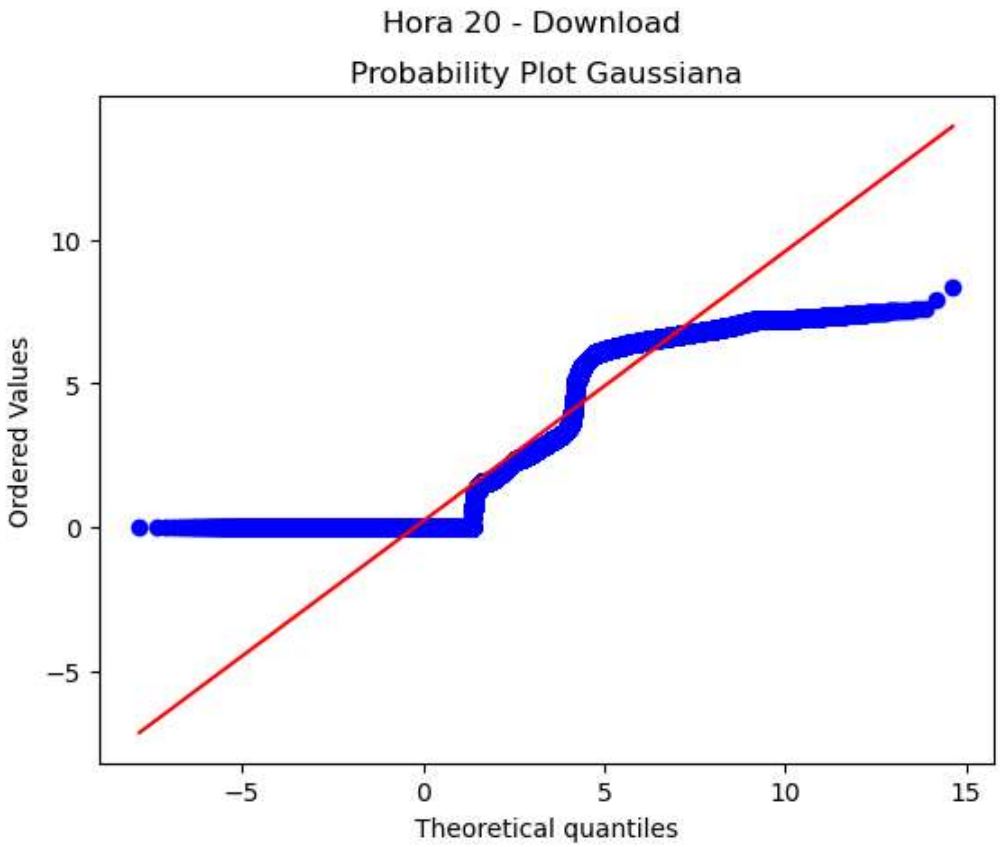
Por fim, são traçados probability plots, comparando os dados dos datasets com cada uma das distribuições parametrizadas. Esses gráficos são plotados com a ajuda do método probplot da biblioteca scipy, como demonstrado abaixo.

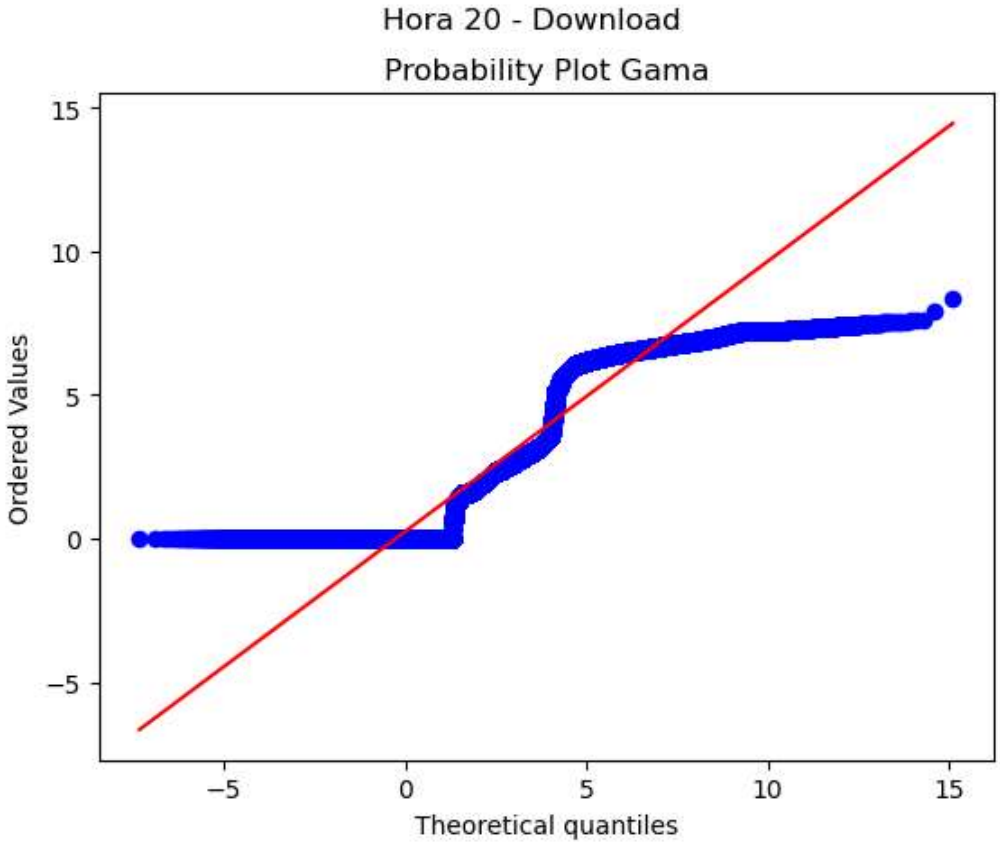
```
eixox = dataset1['bytes_up']

probplot(eixox, dist=norm, sparams=(float(media1up), float(desvio1up)), plot = plt)

plt.suptitle('Hora 20 - Upload')
plt.title('Probability Plot Gaussiana')
plt.show()
```

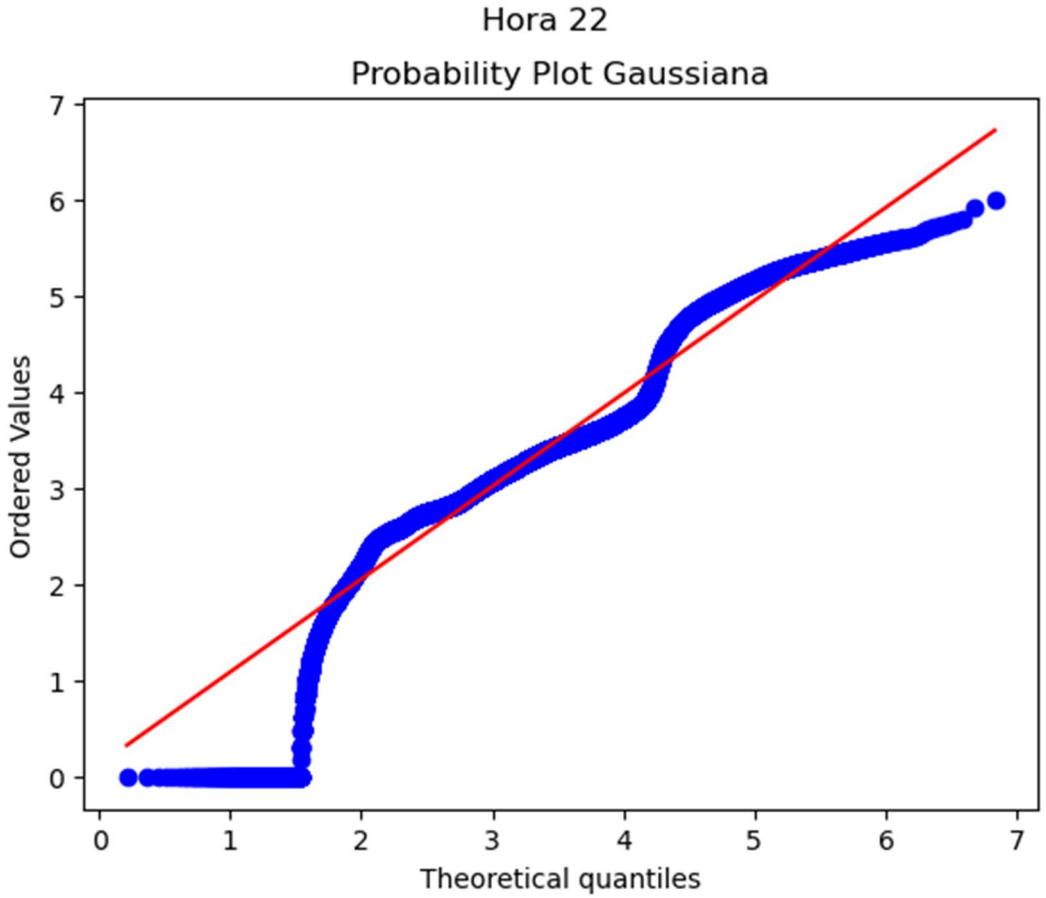
Os gráficos gerados são:

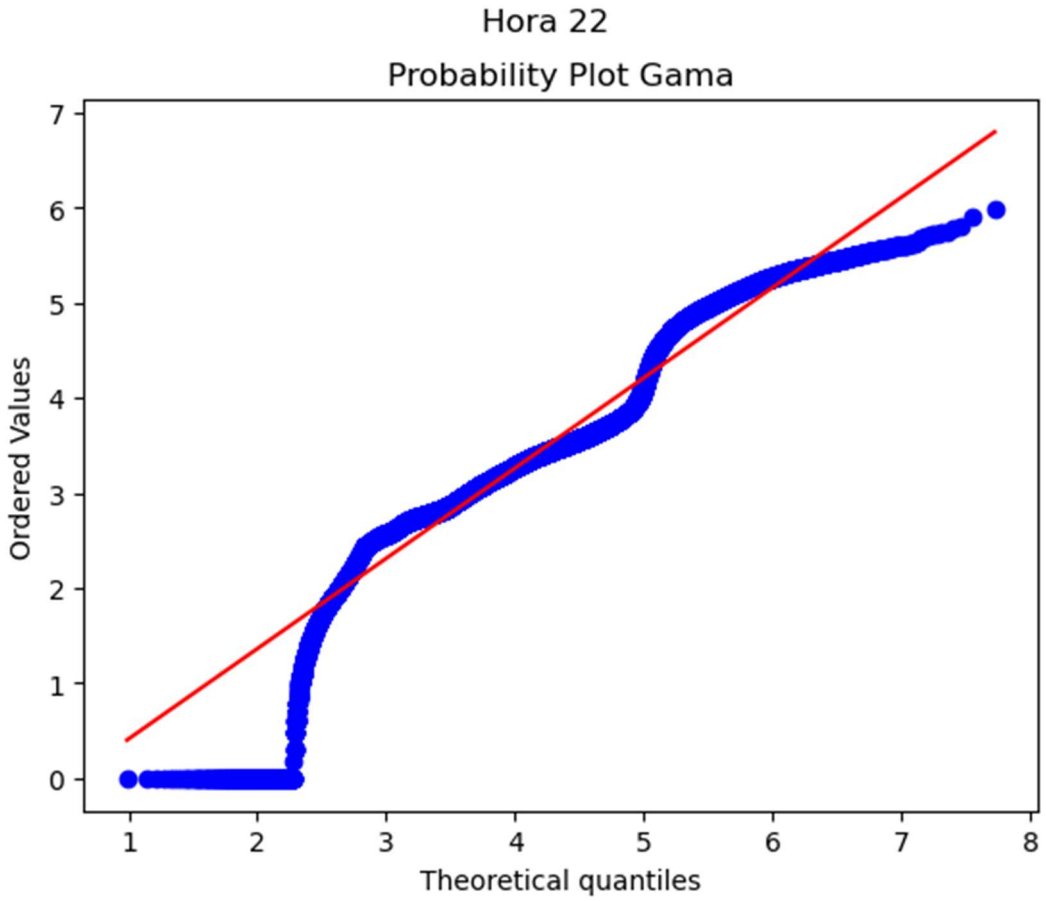

Hora 20 - Upload
Probability Plot Gaussiana
Ordered Values
Theoretical quantiles
10
8
6
4
2
0
−2
−4
−4
−2
0
2
4
6
8
10

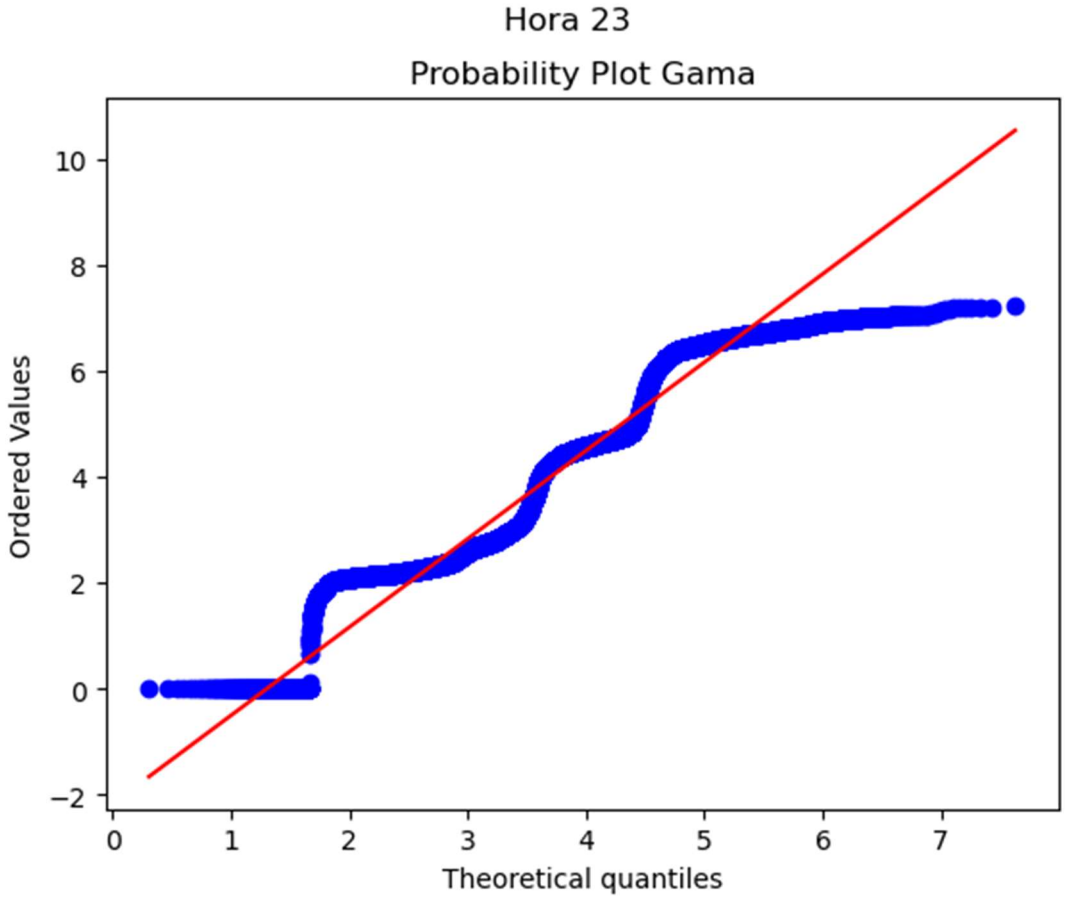

Hora 20 - Upload
Probability Plot Gama
Ordered Values
Theoretical quantiles
12
10
8
6
4
2
0
−2
−4
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5

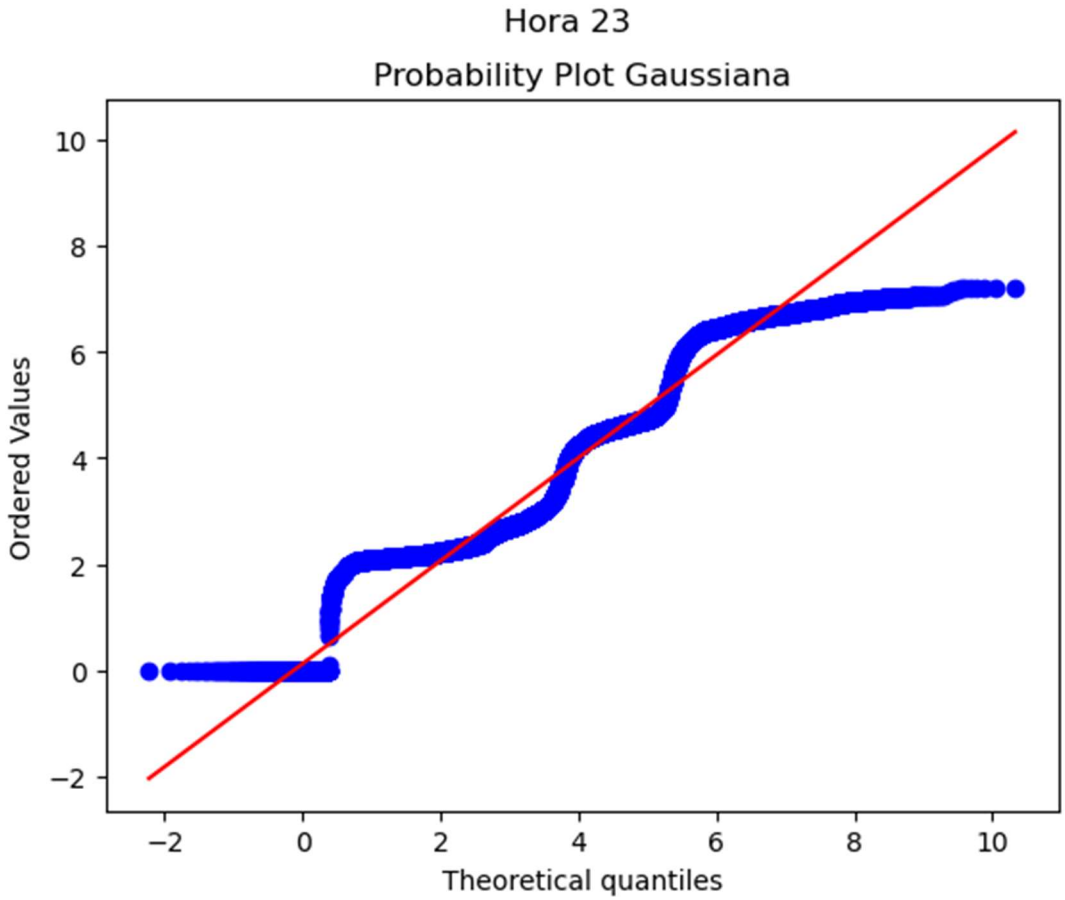

Hora 20 - Download
Probability Plot Gaussiana
Ordered Values
Theoretical quantiles
10
5
0
−5
−5
0
5
10
15


Hora 20 - Download
Probability Plot Gama
Ordered Values
Theoretical quantiles
15
10
5
0
−5
−5
0
5
10
15


Hora 22
Probability Plot Gaussiana
Ordered Values
Theoretical quantiles
0
1
2
3
4
5
6
7


Hora 22
Probability Plot Gama
Ordered Values
Theoretical quantiles
0
1
2
3
4
5
6
7
8


Hora 23
Probability Plot Gama
Ordered Values
Theoretical quantiles
10
8
6
4
2
0
−2
0
1
2
3
4
5
6
7


Hora 23
Probability Plot Gaussiana
Ordered Values
Theoretical quantiles
10
8
6
4
2
0
−2
−2
0
2
4
6
8
10

Respondendo diretamente a algumas questões do relatório:

1. Quais foram os horários escolhidos para cada dataset?

   1/2/3/4 – 20.
   5/6 - 22.
   7/8 – 23

2. O que você pôde observar a partir dos histogramas dos datasets?

   Com o histograma de upload do horário 20, é possível notar uma distribuição bem regular, se assemelhando à uma gaussiana. Já a distribuição de download deste horário parece se assemelhar a uma distribuição bimodal.
   Para ambos nota-se uma quantidade alta de valores próximos a zero, demonstrando um nível relativamente alto de dispositivos com pouco uso do serviço de Internet para dispositivos Smart-TV mesmo em horário de pico.

   No histograma da hora 22 se demonstra uma alta concentração de upload ao redor de 3,5, enquanto para o histograma da hora 23 é difícil de se tirar qualquer tipo de conclusão, talvez sinalizando a existência de vários tipos de usuários distintos que utilizam o dispositivo em horário de pico.
   Para ambos se nota uma quantidade baixa de valores próximos a zero, que sugere um nível baixo de ociosidade em relação aos serviços de Internet por dispositivos do tipo Chromecast em horário de pico.

3. Comente sobre as diferenças e/ou similaridades entre os datasets 1 e 2, 3 e 4, 5 e 6, 7 e 8. O objetivo é comparar as características dos datasets com a maior mediana em um determinado horário com os datasets com a maior média em um determinado horário.
   Para o caso em questão todos os datasets a serem comparados coincidiram, isto é, os datasets de maior mediana eram os datasets de maior média para cada instância.

Isto reflete a constatação feita anteriormente sobre os gráficos das médias e medianas por tempo, em que se afirma que estes apresentam comportamentos similares, o que indica um baixo nível de skewness.

4. É possível caracterizar os datasets acima por uma variável aleatória da literatura? Se a resposta for não, qual o motivo? O que você pôde observar a partir dos gráficos Probability Plot?

   Observando os resultados, creio que tanto a distribuição gama quanto a normal atendem bem aos dados de upload, das horas 20 e 22 e download da hora 23.

   No entanto, não me parece que nenhuma das distribuições foi capaz de mapear bem os dados de download, da hora 20. Creio que para esta talvez uma distribuição bimodal seja mais interessante.

- Análise de correlação entre upload e download

Os coeficientes de correlação são calculados com o método pearsonr da biblioteca scipy e os gráficos scatterplot são gerados com o método scatter da biblioteca matplotlib, como demonstrado abaixo.

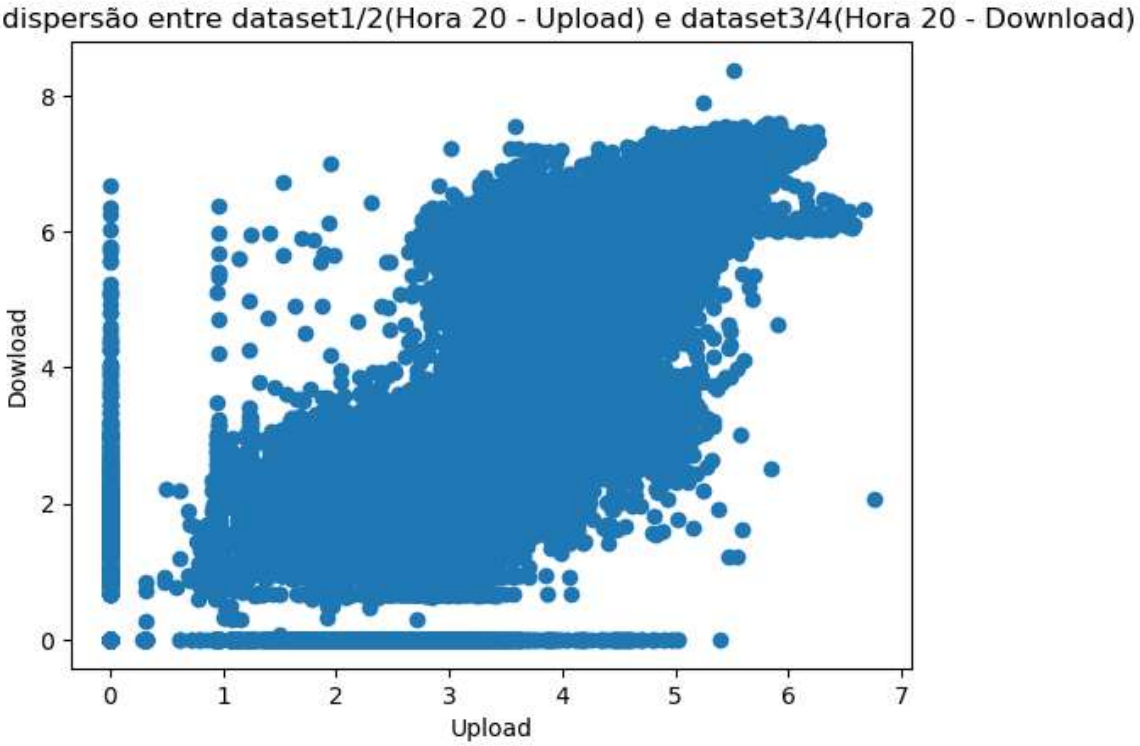
Vale ressaltar que como os datasets 1, 2 e 3, 4 se referem aos mesmos dados, as comparações entre dataset1 e dataset3, bem como dataset2 e dataset4 se tratam da mesma comparação.

Além disso, como os horários devem ser os mesmos para efeito de comparação e para isto se pede que o horário utilizado seja o de download, as comparações de 5 com 7 e de 6 com 8, uma vez que 5 e 6 são o mesmo dataset, bem como 7 e 8, ambas serão feitas na forma da comparação das taxas de upload e download do dataset 7, que equivale ao 8.

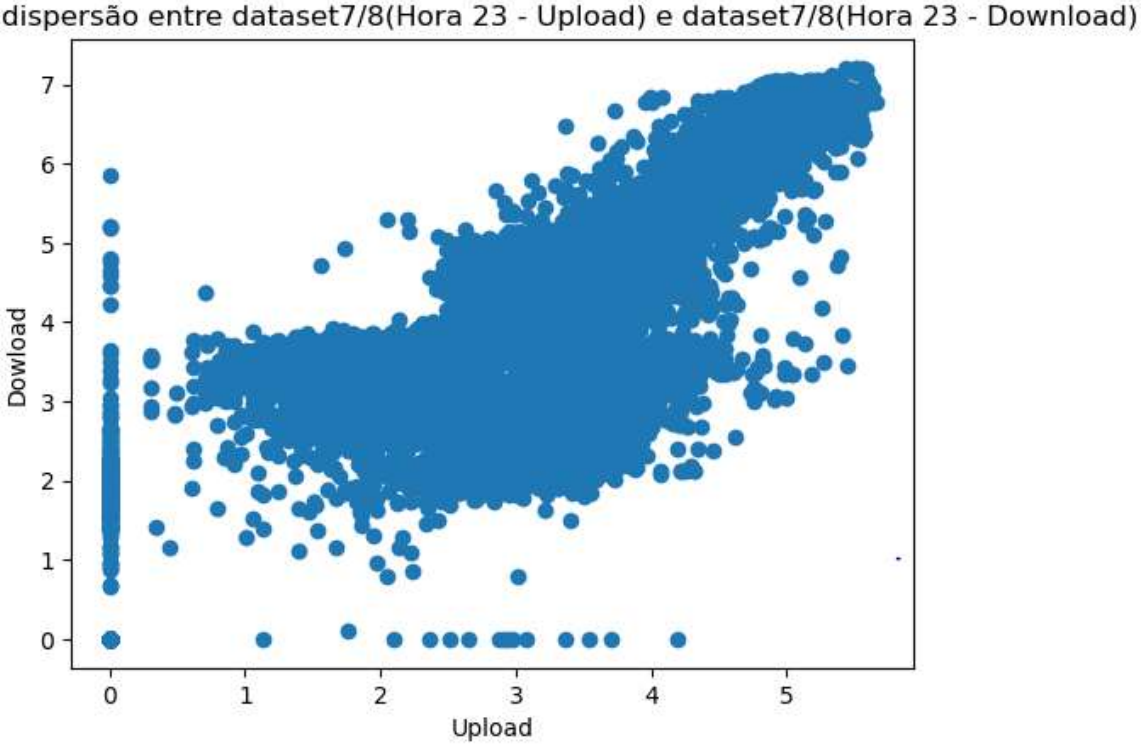
```
pr13 = pearsonr(dataset1['bytes_up'], dataset1['bytes_down'])
correlacao = pr13[0]
correlacao = np.format_float_positional(correlacao, precision=3)
print(f'Correlação entre entre dataset1/2(Hora 20 - Upload) e dataset3/4(Hora 20 - Download): {correlacao}')
plt.scatter(dataset1['bytes_up'], dataset1['bytes_down'])


plt.xlabel('Upload')
plt.ylabel('Dowload')
plt.title('Gráfico de dispersão entre dataset1/2(Hora 20 - Upload) e dataset3/4(Hora 20 - Download)')
plt.show()
```

Os resultados são:

```
Correlação entre entre dataset1/2(Hora 20 - Upload) e dataset3/4(Hora 20 - Download): 0.916
```




```
Correlação entre entre dataset7/8(Hora 23 - Upload) e dataset7/8(Hora 23 - Download): 0.793
```



Pelos coeficientes de correlação é possível observar uma forte correlação entre as taxas de download e upload na hora 20, bem como uma correlação um pouco menor entre os datasets 7 e 8. Baseado nesses gráficos, e com a quantidade de dados sendo analisada é possível afirmar que há uma alta probabilidade de haver uma correlação entre as taxas de download e upload para estes horários.

- Comparação entre os tipos de dispositivos Smart-TV e Chromecast

Inicialmente se define um histograma para os datasets a serem comparados, utilizando o método de Sturges com o número de entradas do dataset com menor número de entradas dos dois para calcular as bins.

Em seguida, se determinam as frequências observadas de cada bin, que são utilizadas para formar a matriz que é utilizada para fazer o G-test. O G-test é feito com o método chi2_contingency da biblioteca scipy, com o parâmetro lambda determinado como log-likelihood, uma vez que o valor do G-test pode ser encontrado pelo log-likelihood ratio test. Isto é feito da seguinte forma:

```
bins1 = pd.cut(dataset1['bytes_up'], bins = int(1 + 3.3*math.log(76738, 10)), include_lowest = True)
bins5 = pd.cut(dataset5['bytes_up'], bins = int(1 + 3.3*math.log(76738, 10)), include_lowest = True)

obs1 = bins1.value_counts().sort_index()/bins1.value_counts().sort_index().sum()
obs5 = bins5.value_counts().sort_index()/bins5.value_counts().sort_index().sum()

gtest = chi2_contingency([obs1, obs5], lambda_="log-likelihood")
```

Como os datasets 1,2,3,4 se referem ao mesmo dataset, bem como os datasets 5,6 e 7,8, os testes a serem realizados são entre o dataset 1 e o dataset 5, bem como entre o dataset 3 e o dataset 7:

```
G-test entre dataset1/2 e 5/6:

G-test: 0.659, P-valor: 0.9999999974


G-test entre dataset3/4 e 7/8:

G-test: 1.126, P-valor: 0.99999985
```

Como o P-valor é extremamente próximo de 1 em ambos os casos, pode se afirmar com grande nível de confiança que a hipótese nula não será rejeitada, isto é, que os dispositivos seguem distribuições distintas.

Os datasets não estão contidos no GitHub devido ao tamanho, é recomendado que eles sejam inseridos no mesma pasta que o notebook para avaliação.

Link para o código: https://github.com/DanielrfGit/AnaliseDeDados-Probest