Projeto de Curso
Estatística e Modelos Probabilísticos
2023.2

Aluno: Nick Hermogenio Correa

DRE: 119024656

Rio de Janeiro, RJ

1. Introdução

Projeto realizado como parte da avaliação da disciplina de Estatística e Modelos Probabilísticos oferecida na UFRJ no período de 2023.2 para o curso de Engenharia de Computação e Informação. O código utilizado na realização do projeto pode ser encontrado na íntegra no seguinte link: https://github.com/NickHCorrea/TrabalhoProbest. O projeto foi feito utilizando a linguagem python e bibliotecas math, numpy, pandas, matplotlib e scipy

2. Estatísticas Gerais

Dataset

Definição dos datasets utilizando a biblioteca Pandas

```
dataset_chromecast = pd.read_csv('dataset_chromecast.csv')
down_cc = dataset_chromecast['bytes_down'].apply(lambda x: np.log10(x) if x != 0 else 0)
up_cc = dataset_chromecast['bytes_up'].apply(lambda x: np.log10(x) if x != 0 else 0)
n_cc = int(dataset_chromecast.size / 4)

dataset_smartTv = pd.read_csv('dataset_smart-tv.csv')
down_tv = dataset_smartTv['bytes_down'].apply(lambda x: np.log10(x) if x != 0 else 0)
up_tv = dataset_smartTv['bytes_up'].apply(lambda x: np.log10(x) if x != 0 else 0)
n_tv = int(dataset_smartTv.size / 4)
```

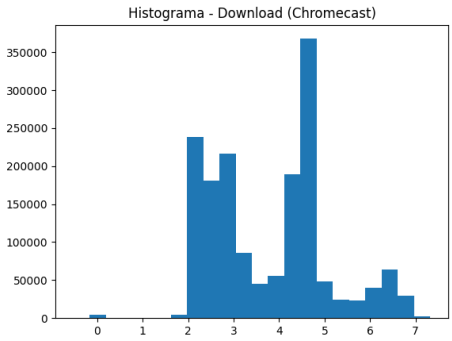
Histogramas

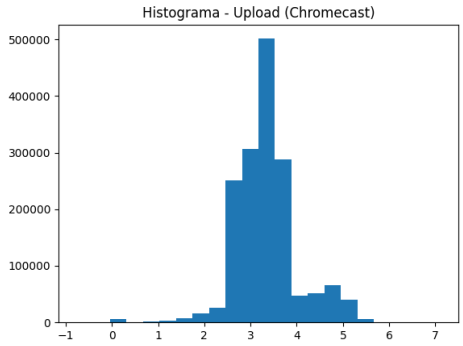
- Obtenção

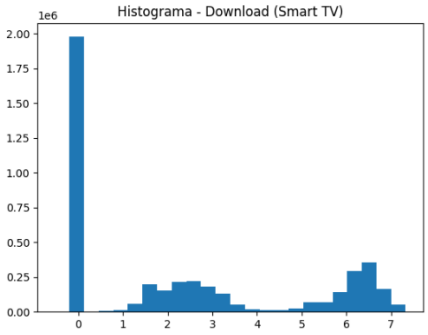
```
# Histograma
binsNumber_cc = int(math.ceil(1 + math.log(n_cc, 10) * 3.322))

_, downBins_cc = np.histogram(down_cc, bins=binsNumber_cc)
_, upBins_cc = np.histogram(up_cc, bins=binsNumber_cc)

plt.figure()
plt.hist(down_cc, downBins_cc)
plt.title("Histograma - Download (Chromecast)")

plt.figure()
plt.hist(up_cc, upBins_cc)
plt.title("Histograma - Upload (Chromecast)")
```

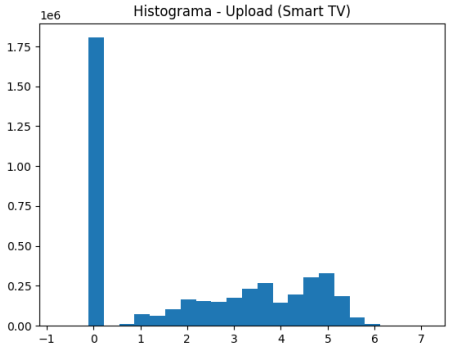
Foi calculado o número de bins utilizando o método de Sturges. Então a função histograma da biblioteca numpy foi utilizada passando como parâmetro o número de bins. O objetivo era receber em uma variável os intervalos correspondentes no dataset ao número de bins. Então foi plotado um resultado utilizando a biblioteca matplotlib.

- Resultados









- Análise

Podemos observar que para a Smart TV é mais frequente ter taxa de upload/download baixa ou nula, enquanto para o Chrome Cast temos a concentração de download por volta de 10^4~10^5 bps e de upload por volta de 10^3~10^4 bps.

Uma possível melhoria para o provedor seria focar em prover uma maior qualidade de banda para Chromecasts do que para Smart Tvs, pois eles são os mais utilizados.

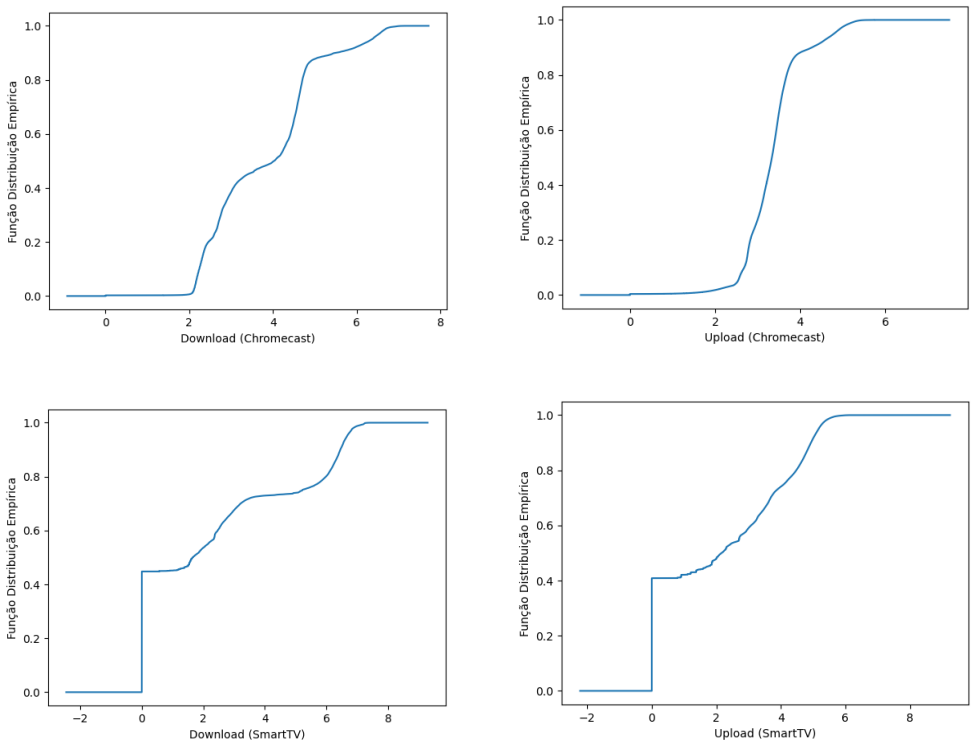
Funções Distribuição Empírica

- Obtenção

```
#Função Distribuição Empirica
res_down_cc = stats.ecdf(down_cc)
plt.figure()
ax = plt.subplot()
res_down_cc.cdf.plot(ax)
ax.set_xlabel('Download (Chromecast)')
ax.set_ylabel('Função Distribuição Empírica')
```

A função distribuição empírica foi obtida utilizando a função ecdf da biblioteca scipy. O resultado então foi plotado utilizando a biblioteca matplotlib.

- Análise



Podemos observar que há um pico tanto para download quanto upload da Smart Tv quando a taxa é da ordem de 10^0, de forma que aproximadamente 40% de todos os dados estão naquela faixa. Para o Chromecast temos um pico em 10^4 para o upload que representa por volta de 90% de todos os dados, ou seja, há uma concentração de dados para taxas a partir desse valor. Para o download o pico se desloca para 10^5.

Como há uma concentração de taxas maiores para Chromecasts, o provedor pode se preocupar em disponibilizar uma largura de banda maior para esses dispositivos. Enquanto isso, as taxas usadas pelas Smart Tvs são mais distribuídas, então há um uso de taxas pequenas também, ou seja, não terá tanto acúmulo para taxas mais altas quanto os Chromecasts.

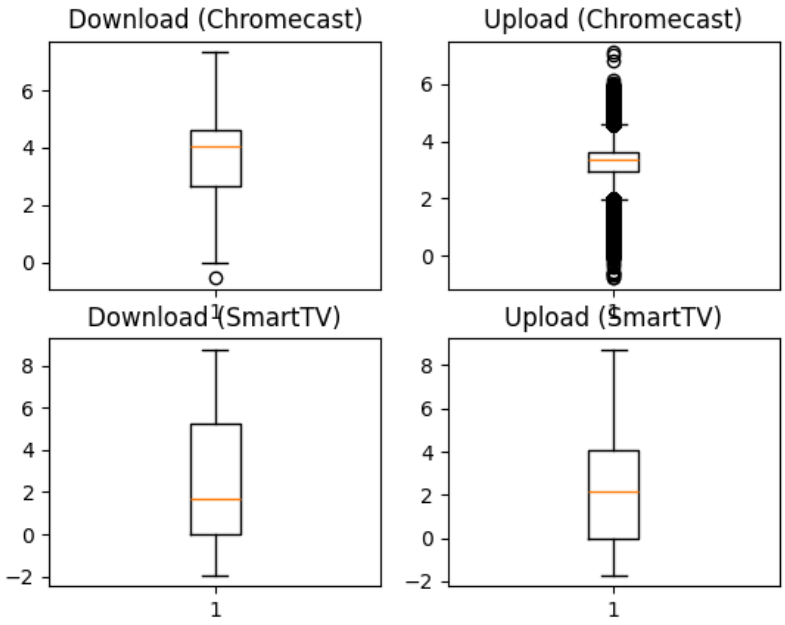
Box Plot

- Obtenção

```
#Boxplot

fig, axs = plt.subplots(2, 2)

axs[0, 0].boxplot(down_cc)
axs[0, 0].set_title('Download (Chromecast)')

axs[0, 1].boxplot(up_cc)
axs[0, 1].set_title('Upload (Chromecast)')

axs[1, 0].boxplot(down_tv)
axs[1, 0].set_title('Download (SmartTV)')

axs[1, 1].boxplot(up_tv)
axs[1, 1].set_title('Upload (SmartTV)')
```

Os gráficos de boxplot foram obtidos utilizando a função boxplot da biblioteca matplotlib.

- Análise



Observando os boxes plot podemos ver que as taxas do Chromecast possuem menos variância e médias mais altas do que as taxas da Smart TV. Entretanto vemos que o upload do Chromecast possui muitos outliers, o que diz que essa variância na verdade não é tão pequena assim. Os dados de Smart TVs mostram uma média baixa com uma variância maior.

Esse gráfico endorsa as análises previamente realizadas.

Média, variância e desvio padrão

- Obtenção

```
#Média
down_mean_cc = down_cc.mean()
up_mean_cc = up_cc.mean()

down_mean_tv = down_tv.mean()
up_mean_tv = up_tv.mean()

#Variância
down_var_cc = down_cc.var()
up_var_cc = up_cc.var()

down_var_tv = down_tv.var()
up_var_tv = up_tv.var()

#Desvio Padrão
down_std_cc = down_cc.std()
up_std_cc = up_cc.std()

down_std_tv = down_tv.std()
up_std_tv = up_tv.std()
```

- Análise
  Chromecast
  Download
  Média: 3.799335488086478
  Variância: 1.665979814323226
  Desvio padrão: 1.2907284045542757
  Upload
  Média: 3.3496717251158694
  Variância: 0.46160016892790384
  Desvio padrão: 0.679411634377793
  Smart TV
  Download
  Média: 2.350172639895904
  Variância: 6.723920763635554
  Desvio padrão: 2.5930524027939645
  Upload
  Média: 2.1565902037563727
  Variância: 4.1130827523161635
  Desvio padrão: 2.0280736555451244

Os dados numéricos obtidos confirmam as hipóteses levantadas analisando os gráficos.

3. Estatísticas por Horário

Dataset

Definição dos datasets utilizando a biblioteca Pandas

```
dataset_chromecast = pd.read_csv('dataset_chromecast.csv')
dataset_chromecast['downLog'] = dataset_chromecast['bytes_down'].apply(lambda x: np.log10(x) if x != 0 else 0)
dataset_chromecast['upLog'] = dataset_chromecast['bytes_up'].apply(lambda x: np.log10(x) if x != 0 else 0)
dataset_chromecast['hour'] = dataset_chromecast["date_hour"].apply(lambda x: int(x.split(" ")[1].split(":")[0]))

dataset_smartTv = pd.read_csv('dataset_smart-tv.csv')
dataset_smartTv['downLog'] = dataset_smartTv['bytes_down'].apply(lambda x: np.log10(x) if x != 0 else 0)
dataset_smartTv['upLog'] = dataset_smartTv['bytes_up'].apply(lambda x: np.log10(x) if x != 0 else 0)
dataset_smartTv['hour'] = dataset_smartTv["date_hour"].apply(lambda x: int(x.split(" ")[1].split(":")[0]))
```

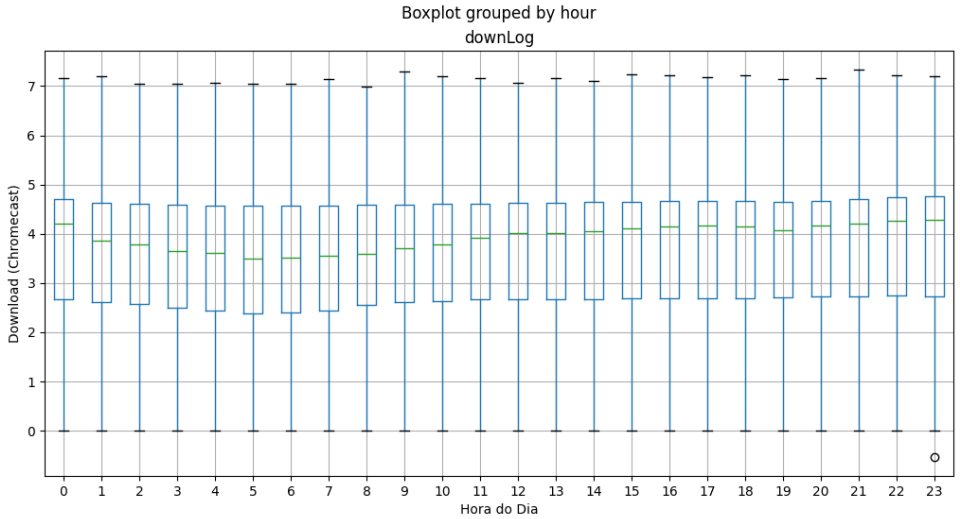
Boxes plot

- Obtenção

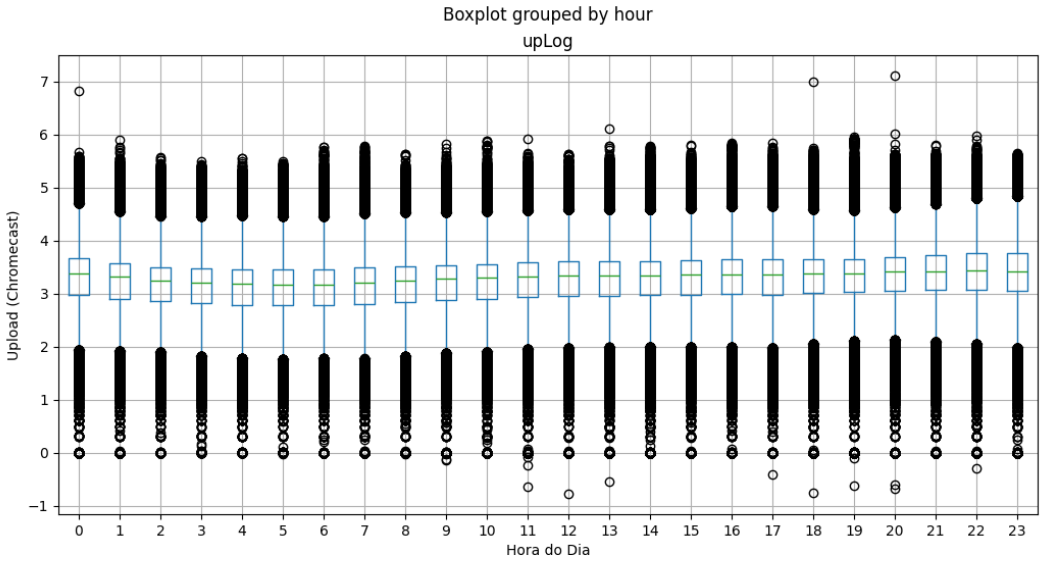
```
#Boxplot
fig, ax = plt.subplots(figsize=(12, 6))

boxplot = dataset_chromecast.boxplot(column='downLog', by='hour', ax=ax)

plt.xlabel('Hora do Dia')
plt.ylabel('Download (Chromecast)')

fig, ax = plt.subplots(figsize=(12, 6))

boxplot = dataset_chromecast.boxplot(column='upLog', by='hour', ax=ax)

plt.xlabel('Hora do Dia')
plt.ylabel('Upload (Chromecast)')

fig, ax = plt.subplots(figsize=(12, 6))

boxplot = dataset_smartTv.boxplot(column='downLog', by='hour', ax=ax)

plt.xlabel('Hora do Dia')
plt.ylabel('Download (Smart TV)')

fig, ax = plt.subplots(figsize=(12, 6))

boxplot = dataset_smartTv.boxplot(column='upLog', by='hour', ax=ax)

plt.xlabel('Hora do Dia')
plt.ylabel('Upload (Smart TV)')
```

Os gráficos de boxplot foram obtidos utilizando a função boxplot da biblioteca matplotlib.

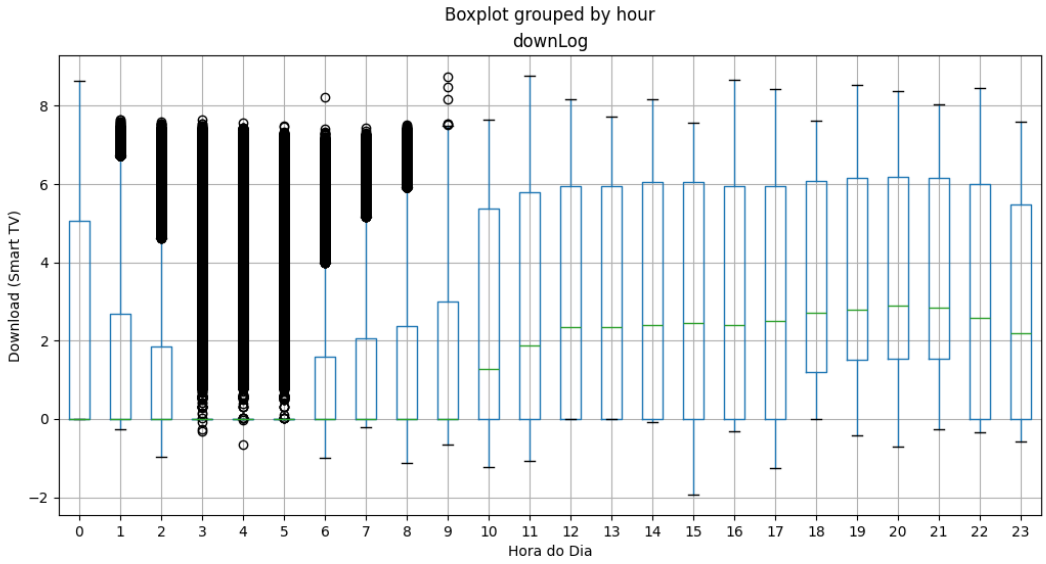
- Análise



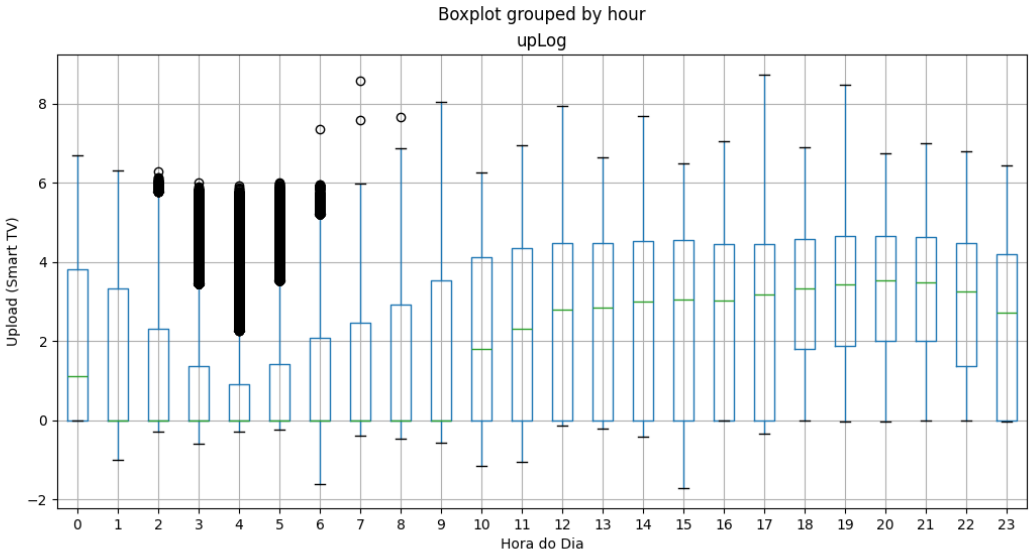
Download (Chromecast) – Podemos perceber que os dados apresentam uma variância grande, sem muita presenta de dados outliers. Durante a madrugada percebemos uma baixa na média da taxa de download, mas ainda assim por volta de 10^3~10^4 bps. Pode-se dizer que durante o dia inteiro há um uso constante de rede de download, já que todos os horários mantêm a média no mesmo intervalo.



Upload (Chromecast) – Esses dados, ao contrário dos dados de download, possuem uma variância muito pequena, mas um número alto de dados outliers. Considerando o grande número de outliers, acredito que a média e variância encontradas não são uma boa representação de todo o conjunto de dados.



Download (Smart TV) – Podemos perceber que, em comparação aos dados obtidos para o Chromecast, a variância dos dados é bem maior e a média geral bem menor, ficando ao redor de 10^2 bps. Durante o período da madrugada temos a presença de muitos outliers junto com médias 0. Isso mostra que a maior parte dos usuários não utiliza o serviço nessas horas, então todos que utilizam se tornam outliers.



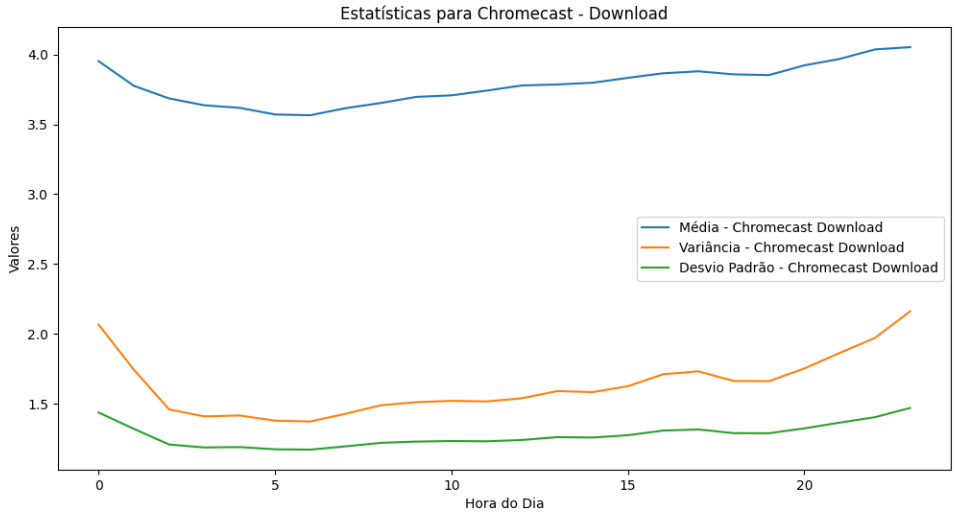
Upload (Smart TV) – Percebemos o mesmo padrão do gráfico anterior: variância maior, médias menores e presença de outliers na parte da madrugada. Também podemos perceber, assim como no gráfico anterior, que entre as 18h e 22h o valor do primeiro quartil aumenta, o que mostra que temos a maior parte dos usuários utilizando mais a rede naquele momento.

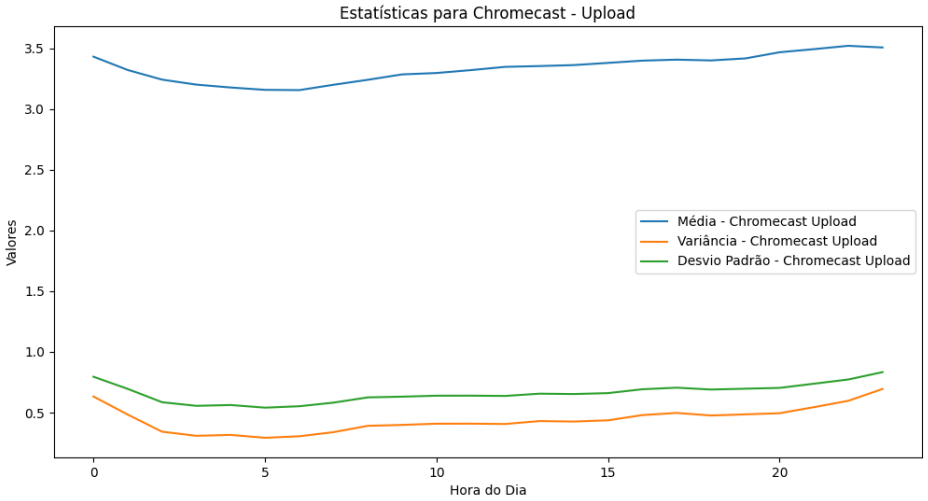
Média, variância e desvio padrão

- Obtenção

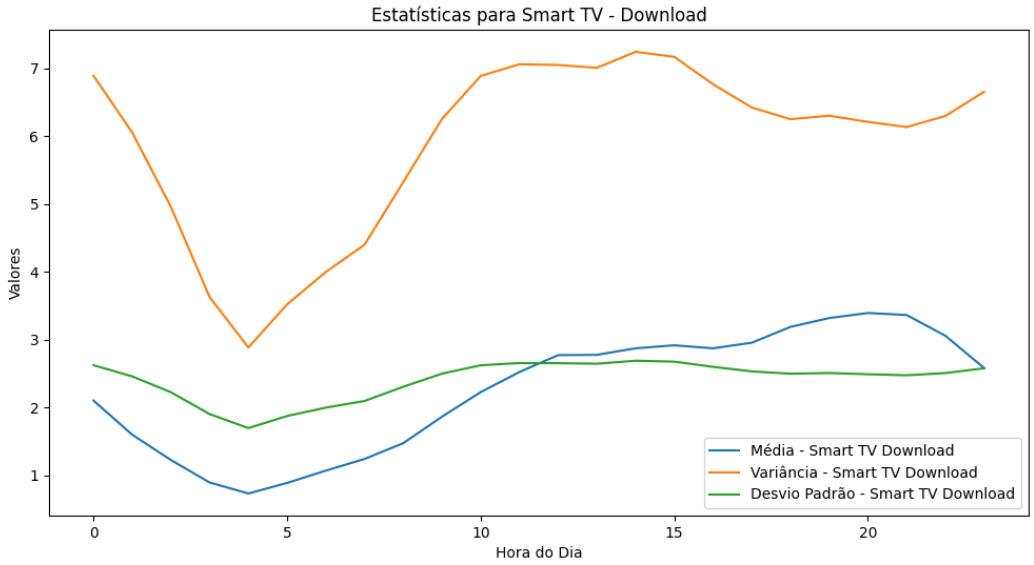
```python
#Média
down_mean_cc = dataset_chromecast.groupby('hour')['downLog'].mean()
up_mean_cc = dataset_chromecast.groupby('hour')['upLog'].mean()

down_mean_tv = dataset_smartTv.groupby('hour')['downLog'].mean()
up_mean_tv = dataset_smartTv.groupby('hour')['upLog'].mean()

#Variância
down_var_cc = dataset_chromecast.groupby('hour')['downLog'].var()
up_var_cc = dataset_chromecast.groupby('hour')['upLog'].var()

down_var_tv = dataset_smartTv.groupby('hour')['downLog'].var()
up_var_tv = dataset_smartTv.groupby('hour')['upLog'].var()

#Desvio Padrão
down_std_cc = dataset_chromecast.groupby('hour')['downLog'].std()
up_std_cc = dataset_chromecast.groupby('hour')['upLog'].std()

down_std_tv = dataset_smartTv.groupby('hour')['downLog'].std()
up_std_tv = dataset_smartTv.groupby('hour')['upLog'].std()

# Gráficos para Chromecast - Download
plt.figure(figsize=(12, 6))
plt.plot(dataset_chromecast.groupby('hour')['downLog'].mean(), label='Média - Chromecast Download')
plt.plot(dataset_chromecast.groupby('hour')['downLog'].var(), label='Variância - Chromecast Download')
plt.plot(dataset_chromecast.groupby('hour')['downLog'].std(), label='Desvio Padrão - Chromecast Download')
plt.title('Estatísticas para Chromecast - Download')
plt.xlabel('Hora do Dia')
plt.ylabel('Valores')
plt.legend()
```

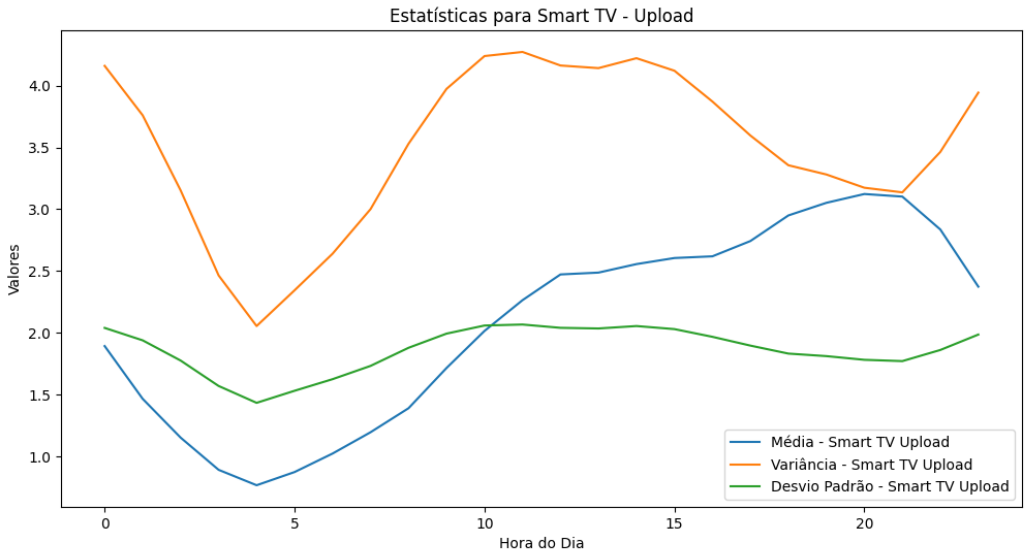
- Análise





Chromecast – Para o Chromecast percebemos os seguintes pontos: aumento da variância na madrugada, média maior na parte da noite, tendo seu ponto mínimo na madrugada, média de download por volta de 10^4 bps e média de upload por volta de $10^{3.5}$ bps.





Smart TV – Assim como dito anteriormente na análise dos Boxes Plot, podemos ver que para a Smart TV temos uma variância significantemente grande. Podemos perceber também: baixa da média durante a madrugada, alta da taxa durante o período da noite (18h até 22h), baixa da variância durante a madrugada e a noite (o que indica que os usuários tem o comportamento mais parecido entre si do que em momentos onde a variância está alta).

4.Caracterizando horários com maior valor de tráfego

Dataset

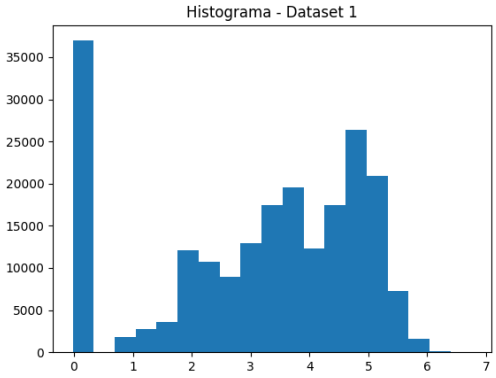
```
dataset_chromecast = pd.read_csv('dataset_chromecast.csv')
dataset_chromecast['downLog'] = dataset_chromecast['bytes_down'].apply(lambda x: np.log10(x) if x != 0 else 0)
dataset_chromecast['upLog'] = dataset_chromecast['bytes_up'].apply(lambda x: np.log10(x) if x != 0 else 0)
dataset_chromecast['hour'] = dataset_chromecast["date_hour"].apply(lambda x: int(x.split(" ")[1].split(":")[0]))

dataset_smartTv = pd.read_csv('dataset_smart-tv.csv')
dataset_smartTv['downLog'] = dataset_smartTv['bytes_down'].apply(lambda x: np.log10(x) if x != 0 else 0)
dataset_smartTv['upLog'] = dataset_smartTv['bytes_up'].apply(lambda x: np.log10(x) if x != 0 else 0)
dataset_smartTv['hour'] = dataset_smartTv["date_hour"].apply(lambda x: int(x.split(" ")[1].split(":")[0]))

DATASET1 = dataset_smartTv[dataset_smartTv['hour'] == 20][['upLog']]
DATASET2 = dataset_smartTv[dataset_smartTv['hour'] == 20][['downLog']]

DATASET3 = dataset_chromecast[dataset_chromecast['hour'] == 22][['upLog']]
DATASET4 = dataset_chromecast[dataset_chromecast['hour'] == 23][['downLog']]
```

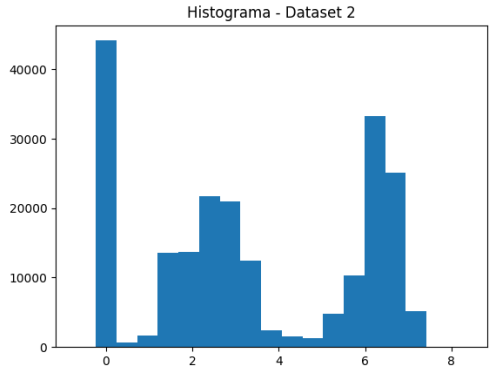
Histogramas

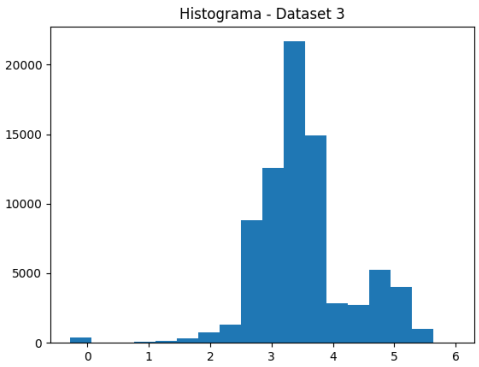
- Obtenção

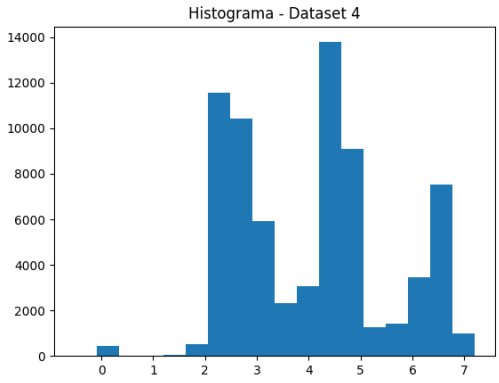
```
#Histogramas

n_1 = DATASET1.size
binsNumber_1 = int(math.ceil(1 + math.log(n_1, 10) * 3.322))
_, bins_1 = np.histogram(DATASET1, bins=binsNumber_1)
plt.figure()
plt.hist(DATASET1, bins_1)
plt.title("Histograma - Dataset 1")

n_2 = DATASET2.size
binsNumber_2 = int(math.ceil(1 + math.log(n_2, 10) * 3.322))
_, bins_2 = np.histogram(DATASET2, bins=binsNumber_2)
plt.figure()
plt.hist(DATASET2, bins_2)
plt.title("Histograma - Dataset 2")
```

- Análise







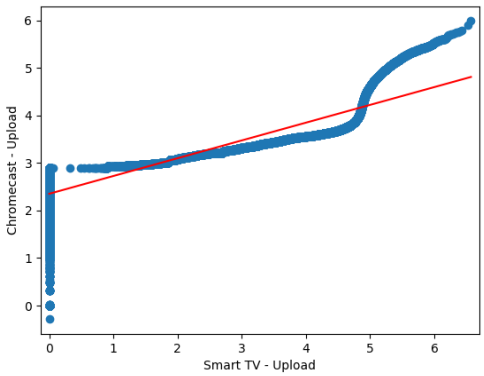


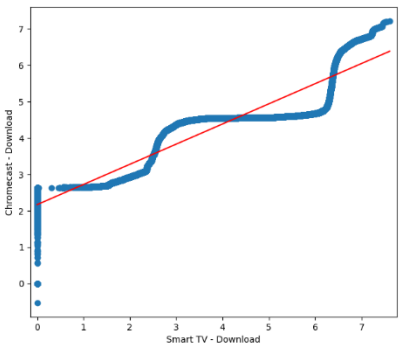
QQPlot

- Obtenção

```
DATA1_sorted = np.sort(DATASET1, axis=0)
DATA2_sorted = np.sort(DATASET2, axis=0)
DATA3_sorted = np.sort(DATASET3, axis=0)
DATA4_sorted = np.sort(DATASET4, axis=0)

qqplot_2samples(DATA1_sorted, DATA3_sorted, xlabel="Smart TV - Upload", ylabel="Chromecast - Upload", line='r')
qqplot_2samples(DATA2_sorted, DATA4_sorted, xlabel="Smart TV - Download", ylabel="Chromecast - Download", line='r')
plt.show()
```

- Análise





Conclusões

1. Quais foram os horários escolhidos para cada dataset?

20h, 20h, 22h e 23h respectivamente

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

Os gráficos referentes as taxas da Smart TV apresentam mais variância e médias menores, enquanto os gráficos referentes as taxas do Chromecast são mais concentradas e com médias maiores.

3. Comente sobre as diferenças e/ou similaridades entre os datasets 1, 2, 3 e 4.

Podemos observar que tanto o dadaset 1 quanto o 2 possuem um número alto para 10^0 bps, o que significa que suas médias são mais baixas. Já nos datasets 3 e 4 temos a maior concentração por volta de $10^{3.5}$ e $10^{4.5}$, respectivamente.

4. O que você pôde observar a partir dos gráficos QQ Plot?

Upload – a taxa de upload do Chromecast se mantém por volta de 10^3 grande parte do tempo, enquanto a taxa da Smart TV aumenta de 10^0 até 10^4, aproximadamente. Isso forma aquele comportamento semelhante a uma linha horizontal que representa a maior parte do gráfico.

Download – Podemos observar que para a taxa de download o gráfico se aproxima mais da reta inclinada, de forma que os valores sobem em proporções semelhantes para os dois aparelhos.

5.Análise da correlação entre as taxas de upload e download para os horários com o maior valor de tráfego

Dataset

```
dataset_chromecast = pd.read_csv('dataset_chromecast.csv')
dataset_chromecast['downLog'] = dataset_chromecast['bytes_down'].apply(lambda x: np.log10(x) if x != 0 else 0)
dataset_chromecast['upLog'] = dataset_chromecast['bytes_up'].apply(lambda x: np.log10(x) if x != 0 else 0)
dataset_chromecast['hour'] = dataset_chromecast["date_hour"].apply(lambda x: int(x.split(" ")[1].split(":")[0]))

dataset_smartTv = pd.read_csv('dataset_smart-tv.csv')
dataset_smartTv['downLog'] = dataset_smartTv['bytes_down'].apply(lambda x: np.log10(x) if x != 0 else 0)
dataset_smartTv['upLog'] = dataset_smartTv['bytes_up'].apply(lambda x: np.log10(x) if x != 0 else 0)
dataset_smartTv['hour'] = dataset_smartTv["date_hour"].apply(lambda x: int(x.split(" ")[1].split(":")[0]))

DATASET1 = dataset_smartTv[dataset_smartTv['hour'] == 20][['upLog']]
DATASET2 = dataset_smartTv[dataset_smartTv['hour'] == 20][['downLog']]

DATASET3 = dataset_chromecast[dataset_chromecast['hour'] == 23][['upLog']]
DATASET4 = dataset_chromecast[dataset_chromecast['hour'] == 23][['downLog']]
```

Coeficiente de Correlação Amostral

- Obtenção

```
#Calculo do coeficiente de correlação amostral

correlation_coefficient, p_value = pearsonr(DATASET1['upLog'], DATASET2['downLog'])
print(f"Coeficiente de Correlação: {correlation_coefficient}")
print(f"P-value: {p_value}")

correlation_coefficient, p_value = pearsonr(DATASET3['upLog'], DATASET4['downLog'])
print(f"Coeficiente de Correlação: {correlation_coefficient}")
print(f"P-value: {p_value}")
```

- Análise

1-2
Coeficiente de Correlação: 0.9154767447400012
P-value: 0.0

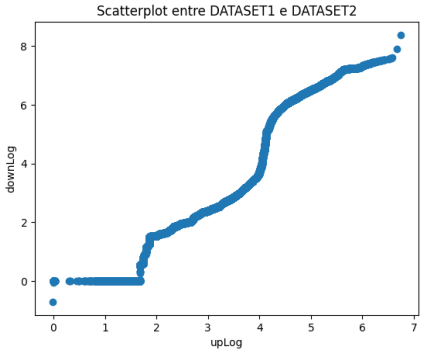
3-4
Coeficiente de Correlação: 0.7919586388344998
P-value: 0.0

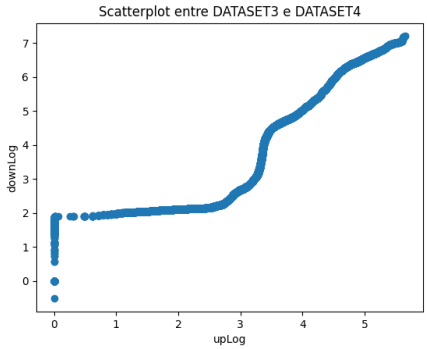
Scatter Plot

- Obtenção

```
#Gráfico ScatterPlot

DATA1_sorted = np.sort(DATASET1, axis=0)
DATA2_sorted = np.sort(DATASET2, axis=0)
DATA3_sorted = np.sort(DATASET3, axis=0)
DATA4_sorted = np.sort(DATASET4, axis=0)

plt.scatter(DATA1_sorted, DATA2_sorted)
plt.xlabel('upLog')
plt.ylabel('downLog')
plt.title('Scatterplot entre DATASET1 e DATASET2')
plt.show()

plt.scatter(DATA3_sorted, DATA4_sorted)
plt.xlabel('upLog')
plt.ylabel('downLog')
plt.title('Scatterplot entre DATASET3 e DATASET4')
plt.show()
```

- Análise





Conclusões

Percebemos que para a SmartTV temos uma relação um pouco mais linear a partir de 10^2 bps de upload. Ou seja, a taxa de download e a de upload tendem a aumentar em taxas semelhantes.

Para o Chromecast vemos que a reta formada até 10^3 de upload é horizontal, de forma que a taxa de upload aumenta muito mais rápido do que a de download. Após 10^4 bps de download, temos um comportamento um pouco mais linear, semelhante ao encontrado para a SmartTV.

6.Conclusão Geral

Analisando todas as estatísticas obtidas na realização desse relatório posso realizar as seguintes recomendações para o provedor:

- Como muitos dos usuários de Smart TV utilizam por volta de 10^0 bps, pode-se disponibilizar uma banda menor para esses usuários. Entretanto é importante que no período da noite (18h ~ 22h) haja um suporte a um número maior de usuários com necessidade de banda um pouco maior também.
- Os usuários de Chromecast tendem a utilizar mais banda, então seria importante disponibilizar uma rede que consiga suprir essas demandas. Também pudemos perceber que há um grande uso da taxa de upload para uma mesma taxa de download, o que diz que é importante manter as velocidades de upload na rede altas também, não só as de download.
- De maneira geral os dois dispositivos têm uma queda em seus usos no período de madrugada, de forma que seria possível disponibilizar menos banda no geral para economia do provador.
- Também de maneira geral, os dois dispositivos têm um aumento nos usos durante o período da noite, o que mostra que é imprescindível que o provedor tenha uma rede capaz de sustentar múltiplos acessos com altas taxas de download/upload.