act_report

March 1, 2019

1 Act Report

Sumário: Eu tenho em mãos um produto melhorado (não perfeito) e talvez possa tirar algumas conclusões interessantes dele.

1.1 Parte I - Consistência e preparação dos dados

O que foi recebido

Um arquivo em formato **.csv** com algum trabalho já realizado sobre ele. O arquivo contém um conjunto de Twitagens sobre um grupo Twitter chamado **WeRateDogs**. O grupo compartilha fotos e textos sobre cães fofos. Eles atribuiem notas aos cães e usam um jargão próprio de adoradores de cães. Algum trabalho já foi realizado sobre ele, como busca dos nomes dos cães.

Um segundo arquivo e um conjunto de fotos, nas quais foi feito um trabalho de **machine learning**, atribuindo nomes aos cães. O conjunto de fotos foi baixado para a máquina virtual.

Uma sugestão de se atualizar as Twitagens recebidas, via o servidor Twitter, a fim de puxar novos dados interessantes sobre as postagens.

Problemas identificados

Os arquivo original **.csv** recebido aparentou inicialmente melhor qualidade do que ofereceu. Aparentemente todos os dados do dataset estão preenchidos. No entanto, ao se realizar algumas sondagens, se pôde descobrir algumas incoerências, como:

- Problemas de **qualidade** (quality)
- twitagens sem foto (material sem valor para este trabalho)
- twitagens de resposta a outra twitagem (idem)
- retwitagens (idem)
- **tweet_id** formatado como Integer Longo

- **timestamp** não segue o formato Date
- campos **puppo**, **pupper**, **doggo** e floofer incompletos
- campo **name** contém nomes incorretos
- **campos rating_numerator** contém entradas incorretas
- **rating_denominator** contém entradas incorretas
- **retweet** formatado como Float
- **favorites** formatado como Float
- Problemas de **organização** (tidyness)
- campo **retweet** se encontra no novo arquivo vindo do servidor do Twitter
- campo **favorites** se encontra no novo arquivo vindo do servidor do Twitter
- **link** da foto dos cães se encontra no arquivo vindo do trabalho de Machine Learning

Propostas de solução

Algumas ações são propostas a fim de melhorar a qualidade dos dados, a saber:

- Problemas de **qualidade**
- twitagens sem foto ← **eliminar linhas**
- twitagens de resposta a outra twitagem ← **eliminar linhas**
- retwitagens ← **eliminar linhas**
- **tweet_id** formatado como Integer Longo ← corrigir para **String Numérica**
- **timestamp** não segue o formato Date ← converter para **Date**
- campos **puppo**, **pupper**, **doggo** e floofer incompletos ← buscar no texto e **preencher**, deixando os não encontrados como **vazios** (NaN)
- campo **name** contém nomes incorretos ← idem
- **campos rating_numerator** contém entradas incorretas ← idem
- **rating_denominator** contém entradas incorretas ← preencher todos como **10** (solução alternativa: **eliminar coluna**)
- **retweet** formatado como Float ← converter para Integer
- **favorites** formatado como Float ← converter para Integer
- Problemas de **organização** (tidyness)

- campo **retweet** se encontra no novo arquivo vindo do servidor do Twitter ← trazer para o **arquivo principal**, quando a entrada existir
- campo **favorites** se encontra no novo arquivo vindo do servidor do Twitter ← idem
- **link** da foto dos cães se encontra no arquivo vindo do trabalho de Machine Learning ← idem

Soluções implantadas

Estão listadas a seguir as operações realizadas no dataset.

- Problemas de **qualidade**:
- twitagens sem foto → linhas eliminadas
- twitagens de resposta a outra twitagem → linhas eliminadas
- retwitagens → linhas eliminadas
- tweet_id formatado como Integer Longo *X* não implementado
- timestamp não segue o formato Date → convertido para Date
- campos puppo, pupper, doggo e floofer incompletos *X* não implementado
- campo name contém nomes incorretos → registros incorretos identificados e eliminados + nova busca realizada no campo de twitagem, identificando **23 novos** nomes corretos, acrescentados
- campos rating_numerator contém entradas incorretas → registros incorretos eliminados + nova busca realizada no campo de twitagem identificou **1 novo** registro correto, acresentado
- rating_denominator contém entradas incorretas → preenchidos todos com o valor **10**
- retweet em Float *X* não implementado
- favorites em Float *X* não implementado
- Problemas de **organização**
- campo **retweet** se encontra no novo arquivo vindo do servidor do Twitter → realizada a consulta ao servidor Twitter e coluna trazida para o arquivo principal sem eliminar linhas do dataset original, quando a entrada não existir
- campo **favorites** se encontra no novo arquivo vindo do servidor do Twitter → idem
- campo **link** para a foto dos cães se encontra no arquivo vindo do trabalho de Machine Learning *X* não implementado

1.2 Parte II - Exibição de alguns resultados

Existe um tutorial do **SKlearn** muito bom disponível no Youtube. Ao final do tutorial, existe um pequeno trabalho com o recorte em fatias e a visualização de alguns dados sobre **qualidade de vinhos**. As ferramentas utilizadas foram basicamente **Pandas** e **Seaborn**. O tutorial encontra-se aqui

Eu não possuo quase nenhuma experiência em **Seaborn**. No entanto, como a sintaxe dele é muito prática e compativel com os trabalhos realizados em **Pandas**, resolvi experimentar!

Parece ser muito divertido separar cães em basicamente três grupos:

- bem classificados (algo como notas e 8 a 14)
- medianamente classificados (notas de 5 a 7)
- mal classificados (notas de 0 a 4)

E pensando bem, provavelmente cães fofinhos e **bem classificados** devem ser alvo de maior número de retuitagem e de favoritagem... mas será que é só isso?

E se cães **mal classificados** também forem **nasty** e bem retuitados e e favoritados? Como será que isso ficaria em gráfico?

Para não me perder, vou seguir basicamente os mesmos passos do tutor no vídeo indicado. A maioria do trabalho dele é em outras ferramentas mais avançadas, no **sklearn**. Este exercício mais singelo aparece ao final, lá pelos dois minutos finais do vídeo... (mais precisamente, em 43:12)

*segundo o Domingão do Faustão, **se vira nos 30!** e como diria Raul Seixas, **quem não tem colírio, usa óculos escuros**... então é isso, melhor seguir os passos de alguém a se perder!*

```
In [1]: import pandas as pd
        import seaborn as sns

In [12]: dfiniciallimpo2 = pd.read_csv('twitter-archive-enhancedNEW2.csv',
                                       sep='\t', encoding='utf-8', index_col='tweet_id')

In [3]: dfiniciallimpo2.info()
        dfiniciallimpo2.head(2)

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2094 entries, 892420643555336193 to 666020888022790149
Data columns (total 13 columns):
timestamp             2094 non-null object
source                2094 non-null object
text                  2094 non-null object
expanded_urls         2094 non-null object
rating_numerator      2077 non-null float64
rating_denominator    2094 non-null int64
name                  1412 non-null object
```

```
doggo                2094 non-null object
floofer              2094 non-null object
pupper               2094 non-null object
puppo                2094 non-null object
retweet_count        2092 non-null float64
favorite_count       2092 non-null float64
dtypes: float64(3), int64(1), object(9)
memory usage: 229.0+ KB
```

Out[3]:

tweet_id	timestamp	source	text	expanded_urls	rating_numerator	rating_denominator	name	doggo	floofer	pupper	puppo	retweet_count	favorite_count
892420643555336193	2017-08-01 16:23:56	<a href="http://twitter.com/download/iphone" r...	This is Phineas. He's a mystical boy. Only eve...	https://twitter.com/dog_rates/status/892420643...	13.0	10	Phineas	None	None	None	None	8265.0	37870.0
892177421306343426	2017-08-01 00:17:27	<a href="http://twitter.com/download/iphone" r...	This is Tilly. She's just checking pup on you...	https://twitter.com/dog_rates/status/892177421...	13.0	10	Tilly	None	None	None	None	6107.0	32537.0

Eu pretendo trabalhar apenas dados com todos os campos preenchidos, então para evitar muitas exclusões, eu eliminei do meu novo dataset o **nome** do cão (e que pensando bem, para o que eu quero fazer, seria inútil!)

```
In [4]: dfapresenta = dfiniciallimpo2[['rating_numerator',
                                    'retweet_count', 'favorite_count']].dropna(axis=0)
        print(dfapresenta.info())
        dfapresenta
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2075 entries, 892420643555336193 to 666020888022790149
Data columns (total 3 columns):
rating_numerator    2075 non-null float64
retweet_count       2075 non-null float64
favorite_count      2075 non-null float64
dtypes: float64(3)
memory usage: 64.8 KB
None
```

Out[4]:

tweet_id	rating_numerator	retweet_count	favorite_count
892420643555336193	13.0	8265.0	37870.0
892177421306343426	13.0	6107.0	32537.0
891815181378084864	12.0	4043.0	24496.0
891689557279858688	13.0	8411.0	41228.0
891327558926688256	12.0	9110.0	39401.0
891087950875897856	13.0	3028.0	19807.0
890971913173991426	13.0	2002.0	11576.0
890729181411237888	13.0	18346.0	63899.0
890609185150312448	13.0	4162.0	27213.0
890240255349198849	14.0	7177.0	31219.0
890006608113172480	13.0	7128.0	29998.0
889880896479866881	13.0	4837.0	27203.0
889665388333682689	13.0	9762.0	47046.0
889638837579907072	12.0	4403.0	26538.0
889531135344209921	13.0	2187.0	14785.0
889278841981685760	13.0	5215.0	24673.0
888917238123831296	12.0	4381.0	28486.0
888804989199671297	13.0	4167.0	25006.0
888554962724278272	13.0	3445.0	19390.0
888078434458587136	12.0	3396.0	21278.0
887705289381826560	13.0	5242.0	29527.0
887517139158093824	14.0	11373.0	45257.0
887473957103951883	13.0	17682.0	67492.0
887343217045368832	13.0	10141.0	32946.0
887101392804085760	12.0	5801.0	29908.0
886983233522544640	13.0	7559.0	34371.0
886736880519319552	13.0	3185.0	11801.0
886680336477933568	13.0	4343.0	21950.0
886366144734445568	12.0	3112.0	20739.0
886258384151887873	13.0	6123.0	27392.0
...	...	...	...
666411507551481857	2.0	322.0	434.0
666407126856765440	7.0	40.0	106.0
666396247373291520	9.0	83.0	160.0
666373753744588802	11.0	89.0	182.0

```
666362758909284353                6.0          560.0          759.0
666353288456101888                8.0           71.0          215.0
666345417576210432               10.0          132.0          286.0
666337882303524864                9.0           90.0          191.0
666293911632134144                3.0          344.0          490.0
666287406224695296                1.0           63.0          143.0
666273097616637952               11.0           76.0          170.0
666268910803644416               10.0           35.0          100.0
666104133288665088                1.0         6388.0        14030.0
666102155909144576               11.0           11.0           78.0
666099513787052032                8.0           66.0          151.0
666094000022159362                9.0           72.0          161.0
666082916733198337                6.0           44.0          113.0
666073100786774016               10.0          159.0          315.0
666071193221509120                9.0           58.0          142.0
666063827256086533               10.0          213.0          465.0
666058600524156928                8.0           57.0          109.0
666057090499244032                9.0          138.0          289.0
666055525042405380               10.0          235.0          428.0
666051853826850816                2.0          831.0         1195.0
666050758794694657               10.0           57.0          130.0
666049248165822465                5.0           42.0          106.0
666044226329800704                6.0          136.0          292.0
666033412701032449                9.0           43.0          123.0
666029285002620928                7.0           46.0          125.0
666020888022790149                8.0          498.0         2529.0

[2075 rows x 3 columns]
```

Mais um detalhe, isso não ficará realmente bom se os números não forem convertidos aos seus formatos originais:

```
In [6]: import numpy as np

In [7]: dfapresenta['rating_numerator'] = dfapresenta[
                                    'rating_numerator'].astype(np.int64)
        dfapresenta['retweet_count'] = dfapresenta[
                                    'retweet_count'].astype(np.int64)
        dfapresenta['favorite_count'] = dfapresenta[
                                    'favorite_count'].astype(np.int64)
        dfapresenta.info()
        dfapresenta.head(2)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2075 entries, 892420643555336193 to 666020888022790149
Data columns (total 3 columns):
rating_numerator    2075 non-null int64
retweet_count       2075 non-null int64
favorite_count      2075 non-null int64
```

```
dtypes: int64(3)
memory usage: 64.8 KB
```

Out[7]:

tweet_id	rating_numerator	retweet_count	favorite_count
892420643555336193	13	8265	37870
892177421306343426	13	6107	32537

Preprocessamento dos dados

Segundo a documentação do Pandas, tuplas é o melhor para inteiros:

```
In [8]: #fatias = (0, 4, 8, 14) #é um corte em inteiros
        fatias = pd.IntervalIndex.from_tuples([(0, 4), (5, 7), (8, 14)])
        nomesgrupos = ['mal', 'medianamente', 'bem']
        fatias

Out[8]: (0, 4, 8, 14)

In [13]: dfapresenta['rating_numerator']

Out[13]: tweet_id
         892420643555336193    13
         892177421306343426    13
         891815181378084864    12
         891689557279858688    13
         891327558926688256    12
         891087950875897856    13
         890971913173991426    13
         890729181411237888    13
         890609185150312448    13
         890240255349198849    14
         890066608113172480    13
         889880896479866881    13
         889665388333682689    13
         889638837579907072    12
         889531135344209921    13
         889278841981685760    13
         888917238123831296    12
         888804989199671297    13
         888554962724278272    13
         888078434458587136    12
         887705289381826560    13
         887517139158093824    14
         887473957103951883    13
         887343217045368832    13
```

```
887101392804085760    12
886983233522544640    13
886736880519319552    13
886680336477933568    13
886366144734445568    12
886258384151887873    13
                      ..
666411507551481857     2
666407126856765440     7
666396247373291520     9
666373753744588802    11
666362758909284353     6
666353288456101888     8
666345417576210432    10
666337882303524864     9
666293911632134144     3
666287406224695296     1
666273097616637952    11
666268910803644416    10
666104133288665088     1
666102155909144576    11
666099513787052032     8
666094000022159362     9
666082916733198337     6
666073100786774016    10
666071193221509120     9
666063827256086533    10
666058600524156928     8
666057090499244032     9
666055525042405380    10
666051853826850816     2
666050758794694657    10
666049248165822465     5
666044226329800704     6
666033412701032449     9
666029285002620928     7
666020888022790149     8
Name: rating_numerator, Length: 2075, dtype: int64
```

In [14]:
```
dfapresenta['rating_numerator'] = pd.cut(dfapresenta['rating_numerator'],
                                         bins=fatias,
                                         labels=nomesgrupos)
#dfapresenta['rating_numerator'].unique()
dfapresenta['rating_numerator']
```

Out[14]:
```
tweet_id
892420643555336193            bem
892177421306343426            bem
```

```
8918151813780848 64            bem
8916895572798586 88            bem
8913275589266882 56            bem
8910879508758978 56            bem
8909719131739914 26            bem
8907291814112378 88            bem
8906091851503124 48            bem
8902402553491988 49            bem
8900066081131724 80            bem
8899808964798668 81            bem
8899653883336826 89            bem
8896388375799070 72            bem
8895311353442099 21            bem
8892788419816857 60            bem
8889172381238312 96            bem
8888049891996712 97            bem
8885549627242782 72            bem
8880784344585871 36            bem
8877052893818265 60            bem
8875171391580938 24            bem
8874739571039518 83            bem
8873432170453688 32            bem
8871013928040857 60            bem
8869832335225446 40            bem
8867368805193195 52            bem
8866803364779335 68            bem
8863661447344455 68            bem
8862583841518878 73            bem
                    ...
6664115075514818 57            mal
6664071268567654 40   medianamente
6663962473732915 20            bem
6663737537445888 02            bem
6663627589092843 53   medianamente
6663532884561018 88   medianamente
6663454175762104 32            bem
6663378823035248 64            bem
6662939116321341 44            mal
6662874062246952 96            mal
6662730976166379 52            bem
6662689108036444 16            bem
6661041332886650 88            mal
6661021559091445 76            bem
6660995137870520 32   medianamente
6660940000221593 62            bem
6660829167331983 37   medianamente
6660731007867740 16            bem
6660711932215091 20            bem
```

```
666063827256086533             bem
666058600524156928    medianamente
666057090499244032             bem
666055525042405380             bem
666051853826850816             mal
666050758794694657             bem
666049248165822465    medianamente
666044226329800704    medianamente
666033412701032449             bem
666029285002620928    medianamente
666020888022790149    medianamente
Name: rating_numerator, Length: 2075, dtype: category
Categories (3, object): [mal < medianamente < bem]
```

In [17]:
```
dfapresenta
```

Out[17]:

tweet_id	rating_numerator	retweet_count	favorite_count
892420643555336193	bem	8265	37870
892177421306343426	bem	6107	32537
891815181378084864	bem	4043	24496
891689557279858688	bem	8411	41228
891327558926688256	bem	9110	39401
891087950875897856	bem	3028	19807
890971913173991426	bem	2002	11576
890729181411237888	bem	18346	63899
890609185150312448	bem	4162	27213
890240255349198849	bem	7177	31219
890066608113172480	bem	7128	29998
889880896479866881	bem	4837	27203
889665388333682689	bem	9762	47046
889638837579907072	bem	4403	26538
889531135344209921	bem	2187	14785
889278841981685760	bem	5215	24673
888917238123831296	bem	4381	28486
888804989199671297	bem	4167	25006
888554962724278272	bem	3445	19390
888078434458587136	bem	3396	21278
887705289381826560	bem	5242	29527
887517139158093824	bem	11373	45257
887473957103951883	bem	17682	67492
887343217045368832	bem	10141	32946
887101392804085760	bem	5801	29908
886983233522544640	bem	7559	34371
886736880519319552	bem	3185	11801
886680336477933568	bem	4343	21950
886366144734445568	bem	3112	20739
886258384151887873	bem	6123	27392

```
...                                 ...             ...             ...
666411507551481857                  mal             322             434
666407126856765440      medianamente                 40             106
666396247373291520                  bem              83             160
666373753744588802                  bem              89             182
666362758909284353      medianamente                560             759
666353288456101888      medianamente                 71             215
666345417576210432                  bem             132             286
666337882303524864                  bem              90             191
666293911632134144                  mal             344             490
666287406224695296                  mal              63             143
666273097616637952                  bem              76             170
666268910803644416                  bem              35             100
666104133288665088                  mal            6388           14030
666102155909144576                  bem              11              78
666099513787052032      medianamente                 66             151
666094000022159362                  bem              72             161
666082916733198337      medianamente                 44             113
666073100786774016                  bem             159             315
666071193221509120                  bem              58             142
666063827256086533                  bem             213             465
666058600524156928      medianamente                 57             109
666057090499244032                  bem             138             289
666055525042405380                  bem             235             428
666051853826850816                  mal             831            1195
666050758794694657                  bem              57             130
666049248165822465      medianamente                 42             106
666044226329800704      medianamente                136             292
666033412701032449                  bem              43             123
666029285002620928      medianamente                 46             125
666020888022790149      medianamente                498            2529

[2075 rows x 3 columns]
```

```
In [47]: import seaborn as sns
         import matplotlib.pyplot as plt

         %matplotlib inline
```

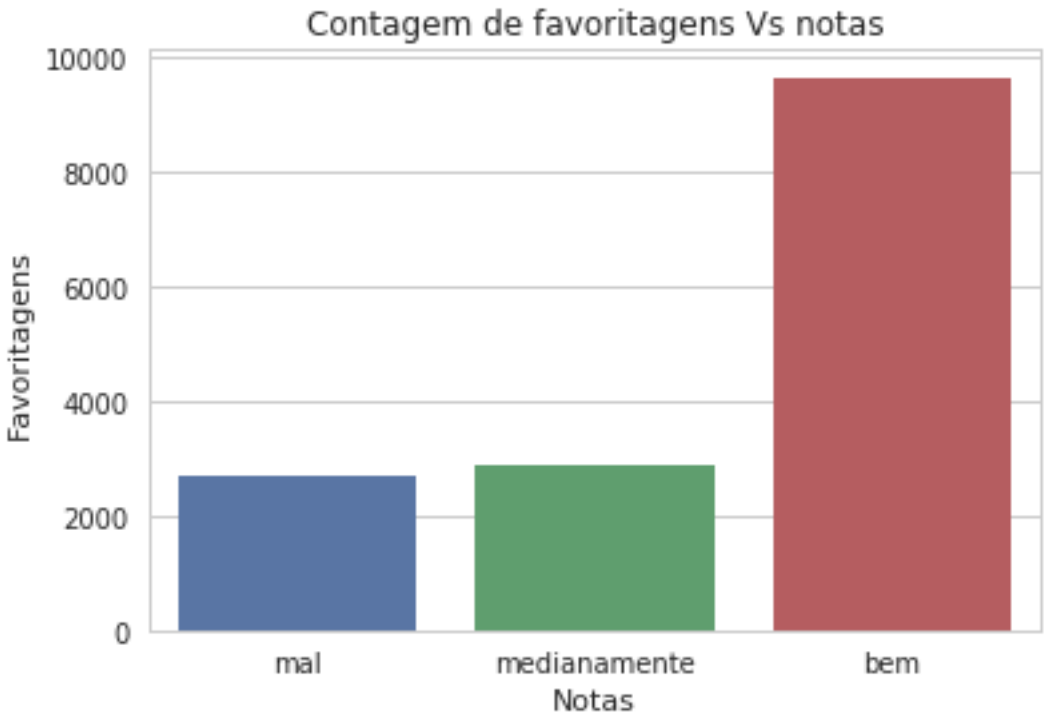
Intuição: Cães com **ótimas notas** (bem pontuados) serem muito mais frequentemente retwitados é algo já esperado... mas surpresa! As pessos tendem a retwitar também cães **mal pontuados** (as fotos podem ser realmente engraçadas!)

Favoritar um cão bem pontuado, todo fofinho... é bastante **comum** pelo visto. Pessoalmente, imaginei que cães mal pontuados talvez fossem mais favoritados do que os cães da média... não foi

o que ocorreu, mas tudo bem! Gráficos nos mostram coisas que às vezes dados não condensados não conseguem nos apresentar direito:

```
In [46]: sns.set(style="whitegrid")

         ax = sns.barplot(x="rating_numerator", y="favorite_count",
                          data=dfapresenta, ci=None)
         ax.set_title("Contagem de favoritagens Vs notas")
         ax.set_xlabel('Notas')
         ax.set_ylabel('Favoritagens')
         plt.show()
```



Uma observação final sobre o projeto Sofri um bocado para colocar essas coisas no lugar. Levei mais tempo do que esperava e pensei que não fosse dar conta do projeto.

Depois percebi que estava emperrando em alguns defeitos conceituais da minha parte no **Pandas**. Correu tudo bem e aprendi muitas coisas. A ferramenta **SQL** é ótima e muito eficiente para algumas tarefas (consultas e reparos de dados em lote), mas não é muito eficiente em tarefas especializadas (como aquele meu filtro de nome do cão em um texto). Então valeu à pena brigar com o **Pandas**!

O que será daqui para frente? Vou implantar a ferramenta **Pandas** + **Jupyter Notebook** para diversas coisas que fazemos aqui na ANA (Agência Nacional de Águas)

Nota final:

- inicialmente achei super estranho, um banco chamado **We Rate Dogs**?
- nunca havia usado Twitter. Abri minha conta e adicionei o grupo
- realmente foi um projeto muito, mas muito **fofinho**! Posso dizer que no final, simplesmente **amei**!

*Mais uma nota: por favor evitem nos próximos projetos usarem temas pesados como o **Cancer Data**. Tive muitos familiares mortos deste mal e é um pouco traumatizante. Acaba não sendo divertido fazer exercícios do tipo: qual a probabilidade de você ter **câncer**, dado que fez um teste e deu **positivo**...*