UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
INTRODUÇÃO A INTELIGENCIA ARTIFICIAL

PROVA 2

THIAGO ALVES DE ARAUJO

Sumário

1 – Classificação

A classificação é um método de aprendizagem de máquina supervisionada. Ela consiste em tomar algum tipo de entrada e atribuir um rótulo a ela. Sistemas de classificação são usados geralmente quando as previsões são de natureza distinta, ou seja, um simples “sim ou não”.

Neste trabalho, utilizaremos dois algoritmos de aprendizagem de máquina, ambos algoritmos de aprendizagem supervisionada, para classificar a emoção de uma música.

1.1 – Pré-processamento

Vamos iniciar a análise da nossa base de dados realizando uma série de técnicas matemáticas/estatísticas para preparar os dados antes de aplicar os algoritmos de aprendizagem de máquina. Abaixo podemos ver um trecho da nossa base de dados. Ela inicialmente possui 593 linhas e 78 colunas.

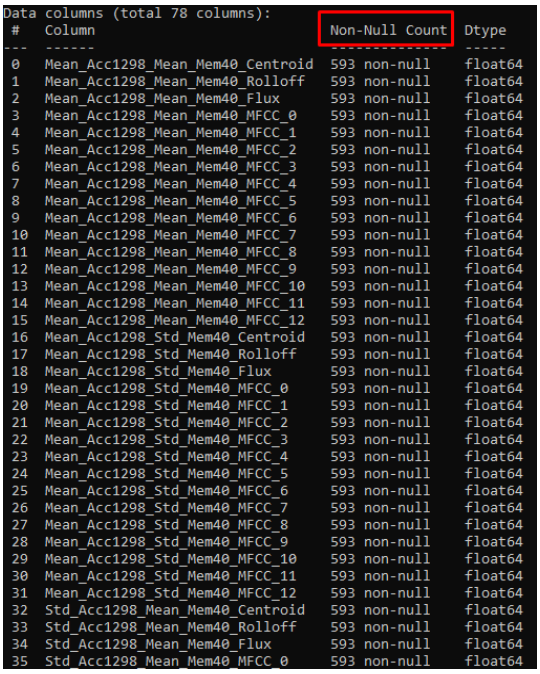
	Mean_Acc1298_Mean_Mem40_Centroid	Mean_Acc1298_Mean_Mem40_Rolloff	Mean_Acc1298_Mean_Mem40_Flux	Mean_Acc1298_Mean_Mem40_MFCC_0
0	0.034741	0.089665	0.091225	-73.302422
1	0.081374	0.272747	0.085733	-62.584437
2	0.110545	0.273567	0.084410	-65.235325
3	0.042481	0.199281	0.093447	-80.305152
4	0.074550	0.140880	0.079789	-93.697749
...	...	...	...	...
588	0.027142	0.047551	0.072043	-79.881347
589	0.094829	0.204498	0.082824	-61.364436
590	0.035169	0.065403	0.075227	-81.750533
591	0.054276	0.238158	0.095935	-71.009724
592	0.073194	0.140733	0.080545	-74.517081

593 rows × 78 columns

Figura 1 Trecho do Dataframe emtions.arff

1.1.1 – Verificando atributos incompletos

Existem diversas maneiras de verificar a presença de instancias com atributos incompletos em um dataframe. Uma delas é utilizando a função *info.* Com ela podemos observar a contagem de valores não nulos em um atributo (coluna). Abaixo podemos ver que todas as colunas possuem 593 elementos não nulos; este é exatamente o número de instancias (linhas) presentes no dataframe.



```
Data columns (total 78 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Mean_Acc1298_Mean_Mem40_Centroid  593 non-null    float64
 1   Mean_Acc1298_Mean_Mem40_Rolloff   593 non-null    float64
 2   Mean_Acc1298_Mean_Mem40_Flux      593 non-null    float64
 3   Mean_Acc1298_Mean_Mem40_MFCC_0    593 non-null    float64
 4   Mean_Acc1298_Mean_Mem40_MFCC_1    593 non-null    float64
 5   Mean_Acc1298_Mean_Mem40_MFCC_2    593 non-null    float64
 6   Mean_Acc1298_Mean_Mem40_MFCC_3    593 non-null    float64
 7   Mean_Acc1298_Mean_Mem40_MFCC_4    593 non-null    float64
 8   Mean_Acc1298_Mean_Mem40_MFCC_5    593 non-null    float64
 9   Mean_Acc1298_Mean_Mem40_MFCC_6    593 non-null    float64
 10  Mean_Acc1298_Mean_Mem40_MFCC_7    593 non-null    float64
 11  Mean_Acc1298_Mean_Mem40_MFCC_8    593 non-null    float64
 12  Mean_Acc1298_Mean_Mem40_MFCC_9    593 non-null    float64
 13  Mean_Acc1298_Mean_Mem40_MFCC_10   593 non-null    float64
 14  Mean_Acc1298_Mean_Mem40_MFCC_11   593 non-null    float64
 15  Mean_Acc1298_Mean_Mem40_MFCC_12   593 non-null    float64
 16  Mean_Acc1298_Std_Mem40_Centroid   593 non-null    float64
 17  Mean_Acc1298_Std_Mem40_Rolloff    593 non-null    float64
 18  Mean_Acc1298_Std_Mem40_Flux       593 non-null    float64
 19  Mean_Acc1298_Std_Mem40_MFCC_0     593 non-null    float64
 20  Mean_Acc1298_Std_Mem40_MFCC_1     593 non-null    float64
 21  Mean_Acc1298_Std_Mem40_MFCC_2     593 non-null    float64
 22  Mean_Acc1298_Std_Mem40_MFCC_3     593 non-null    float64
 23  Mean_Acc1298_Std_Mem40_MFCC_4     593 non-null    float64
 24  Mean_Acc1298_Std_Mem40_MFCC_5     593 non-null    float64
 25  Mean_Acc1298_Std_Mem40_MFCC_6     593 non-null    float64
 26  Mean_Acc1298_Std_Mem40_MFCC_7     593 non-null    float64
 27  Mean_Acc1298_Std_Mem40_MFCC_8     593 non-null    float64
 28  Mean_Acc1298_Std_Mem40_MFCC_9     593 non-null    float64
 29  Mean_Acc1298_Std_Mem40_MFCC_10    593 non-null    float64
 30  Mean_Acc1298_Std_Mem40_MFCC_11    593 non-null    float64
 31  Mean_Acc1298_Std_Mem40_MFCC_12    593 non-null    float64
 32  Std_Acc1298_Mean_Mem40_Centroid   593 non-null    float64
 33  Std_Acc1298_Mean_Mem40_Rolloff    593 non-null    float64
 34  Std_Acc1298_Mean_Mem40_Flux       593 non-null    float64
 35  Std_Acc1298_Mean_Mem40_MFCC_0     593 non-null    float64
```

Figura 2 Trecho das informações do dataframe

1.1.2 – Normalização

A primeira técnica utilizada será a normalização. Com ela, vamos "equalizar" todos os valores (numéricos) deixando-os entre o intervalo [0,1].

Abaixo podemos ver os valores *máximo* e *mínimo* de cada atributo. Podemos observar que os dados não estão normalizados e apresentam valores em intervalos diferentes.

	Mean_Acc1298_Mean_Mem40_Centroid	Mean_Acc1298_Mean_Mem40_Rolloff	Mean_Acc1298_Mean_Mem40_Flux	...
Min	0.010201	0.038286	0.070932	...
Max	0.195412	0.698277	0.159460	...

Figura 3 Dados sem normalização

Após a normalização, podemos observar que os valores se encontram no intervalo desejado.

	Mean_Acc1298_Mean_Mem40_Centroid	Mean_Acc1298_Mean_Mem40_Rolloff	Mean_Acc1298_Mean_Mem40_Flux	...
Min	0.0	0.0	0.0	...
Max	1.0	1.0	1.0	...

Figura 4 Dados normalizados

1.1.3 – Matriz de correlação

Agora vamos gerar a matriz de correlação. Com ela, podemos verificar a correlação entre os diversos atributos. Abaixo podemos ver o resultado obtido.

	Mean_Acc1298_Mean_Mem40_Centroid	Mean_Acc1298_Mean_Mem40_Rolloff	Mean_Acc1298_Mean_Mem40_Flux	Mean_
Mean_Acc1298_Mean_Mem40_Centroid	1	0.63	0.16	
Mean_Acc1298_Mean_Mem40_Rolloff	0.63	1	0.58	
Mean_Acc1298_Mean_Mem40_Flux	0.16	0.58	1	
Mean_Acc1298_Mean_Mem40_MFCC_0	0.37	0.29	0.057	
Mean_Acc1298_Mean_Mem40_MFCC_1	0.64	0.71	0.36	
Mean_Acc1298_Mean_Mem40_MFCC_2	0.21	0.068	0.15	
Mean_Acc1298_Mean_Mem40_MFCC_3	0.29	0.32	0.098	
Mean_Acc1298_Mean_Mem40_MFCC_4	0.2	0.065	0.18	
Mean_Acc1298_Mean_Mem40_MFCC_5	0.27	0.21	0.039	
Mean_Acc1298_Mean_Mem40_MFCC_6	0.071	0.11	0.21	
Mean_Acc1298_Mean_Mem40_MFCC_7	0.05	0.04	0.16	

Figura 5 Trecho da matriz de correlação

Agora vamos correlacionar os atributos utilizando um fator de correlação de 0,80. Após a correlação, podemos observar que o número de atributos diminuiu de 78 para 59.

	Mean_Acc1298_Mean_Mem40_Centroid	Mean_Acc1298_Mean_Mem40_Rolloff	Mean_Acc1298_Mean_Mem40_Flux	...	quiet-still	sad-lonely	angry-aggresive
0	0.034741	0.089665	0.091225	...	b'0'	b'0'	b'0'
1	0.081374	0.272747	0.085733	...	b'0'	b'0'	b'1'
2	0.110545	0.273567	0.084410	...	b'0'	b'0'	b'1'
3	0.042481	0.199281	0.093447	...	b'0'	b'0'	b'0'
4	0.074550	0.140880	0.079789	...	b'1'	b'0'	b'0'
..	...	...	...	...	...	...	...
588	0.027142	0.047551	0.072043	...	b'1'	b'1'	b'0'
589	0.094829	0.204498	0.082824	...	b'0'	b'1'	b'1'
590	0.035169	0.065403	0.075227	...	b'1'	b'1'	b'0'
591	0.054276	0.238158	0.095935	...	b'0'	b'0'	b'0'
592	0.073194	0.140733	0.080545	...	b'0'	b'0'	b'0'

[593 rows x 59 columns]

Figura 6 Dataframe correlacionado

1.1.4 – Outliers

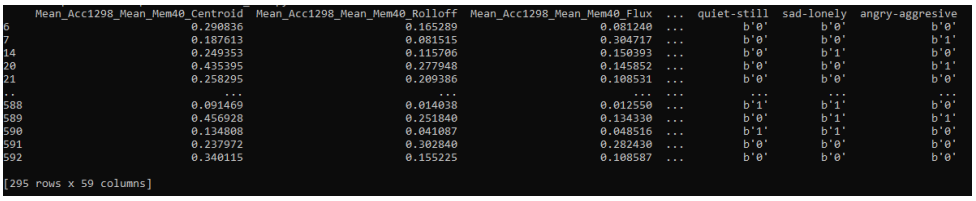
Na estatística, os outliers são valores atípicos que apresentam um grande afastamento dos demais valores. Os estatísticos desenvolveram diversas maneiras de identificar o e mensurar matematicamente o afastamento necessário para que um valor seja considerado um outlier. Este afastamento pode ser calculado pela regra do *1.5*FIQ*.

Utilizando a função *describe* podemos obter diversas informações sobre cada atributo do dataframe como podemos observar abaixo. Com ela, podemos obter os valores do primeiro quartil (25%) e do terceiro quartil (75%). Com eles, podemos calculara a faixa interquartil (FIQ)

	Mean_Acc1298_Mean_Mem40_Centroid	Mean_Acc1298_Mean_Mem40_Rolloff	Mean_Acc1298_Mean_Mem40_Flux	...	BH_LowPeakBPM	BH_HighPeakBPM	BH_HighLowRatio
count	593.000000	593.000000	593.000000	...	593.000000	593.000000	593.000000
mean	0.319141	0.238053	0.166757	...	0.647452	0.654919	0.699831
std	0.168961	0.178193	0.104596	...	0.127045	0.125772	0.105347
min	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000
25%	0.186706	0.105238	0.091338	...	0.556522	0.556962	0.666667
50%	0.297229	0.199201	0.143514	...	0.617391	0.632911	0.666667
75%	0.410823	0.332402	0.230334	...	0.730435	0.759494	0.666667
max	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000

Figura 7 Informações do dataframe

Utilizando dessa regra, foram identificados 298 instancias com a presença de outliers em algum dos seus atributos.

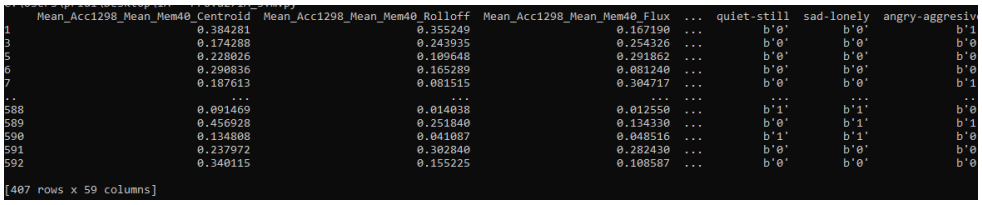


	Mean_Acc1298_Mean_Mem40_Centroid	Mean_Acc1298_Mean_Mem40_Rolloff	Mean_Acc1298_Mean_Mem40_Flux	...	quiet-still	sad-lonely	angry-aggresive
6	0.290836	0.165289	0.081240	...	b'0'	b'0'	b'0'
7	0.187613	0.081515	0.304717	...	b'0'	b'0'	b'1'
14	0.249353	0.115706	0.150393	...	b'0'	b'1'	b'0'
20	0.435395	0.277948	0.145852	...	b'0'	b'0'	b'1'
21	0.258295	0.209386	0.108531	...	b'0'	b'0'	b'0'
..	...	...	...	...	...	...	...
588	0.091469	0.014038	0.012550	...	b'1'	b'1'	b'0'
589	0.456928	0.251840	0.134330	...	b'0'	b'1'	b'1'
590	0.134808	0.041087	0.048516	...	b'1'	b'1'	b'0'
591	0.237972	0.302840	0.282430	...	b'0'	b'0'	b'0'
592	0.340115	0.155225	0.108587	...	b'0'	b'0'	b'0'

[295 rows x 59 columns]

Figura 8 Base sem outliers com fator de 1,5

Como a base resultante possui menos de 50% da quantidade original de instancias, alteramos o fator de 1.5 para 2.5 (ou seja, toleramos outliers mais distantes dos quartis). Com isso, a base resultante possui 407 instancias e 59 atributos.



	Mean_Acc1298_Mean_Mem40_Centroid	Mean_Acc1298_Mean_Mem40_Rolloff	Mean_Acc1298_Mean_Mem40_Flux	...	quiet-still	sad-lonely	angry-aggresiv
1	0.384281	0.355249	0.167190	...	b'0'	b'0'	b'1
3	0.174288	0.243935	0.254326	...	b'0'	b'0'	b'0
5	0.228026	0.109648	0.291862	...	b'0'	b'0'	b'0
6	0.290836	0.165289	0.081240	...	b'0'	b'0'	b'0
7	0.187613	0.081515	0.304717	...	b'0'	b'0'	b'1
..	...	...	...	...	...	...	..
588	0.091469	0.014038	0.012550	...	b'1'	b'1'	b'0
589	0.456928	0.251840	0.134330	...	b'0'	b'1'	b'1
590	0.134808	0.041087	0.048516	...	b'1'	b'1'	b'0
591	0.237972	0.302840	0.282430	...	b'0'	b'0'	b'0
592	0.340115	0.155225	0.108587	...	b'0'	b'0'	b'0

[407 rows x 59 columns]

Figura 9 Base sem outliers com fator de 2,5

1.1.5 – Balanceamento

Abaixo podemos observar o balanceamento dos atributos relacionados a emoção da música. Como podemos ver, existe um leve desbalanceamento entre as classes.

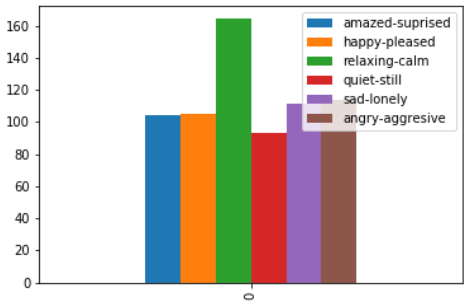


Figura 10 Dados desbalanceados

Após eliminar algumas instâncias, podemos observar que os dados se encontram mais balanceados.

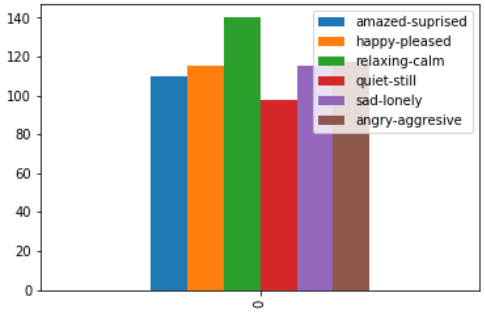


Figura 11 Dados balanceados

1.2 – KNN

KNN (K — Nearest Neighbors) é um dos muitos algoritmos (de aprendizagem supervisionada) usado no campo da machine learning. Ele é um classificador onde o aprendizado é baseado “no quão similar” é um dado (um vetor) do outro.

Utilizando funções da biblioteca scikit-learn (biblioteca de aprendizado de máquina de código aberto para a linguagem de programação Python) podemos obter dados como acurácia, matriz de confusão, score f1, sensibilidade e especificidade. Abaixo podemos ver os resultados obtidos.

Iniciamos a analise dos resultados obtidos observando a matriz de confusão. Para a nossa aplicação, podemos notar que a classe “happy-pleased” obteve o melhor resultado, obtendo 15 previsões corretas de 18 amostras. Já a classe “angry-aggresive” obteve o pior resultado, onde o algoritmo não previu nenhuma das três amostras.

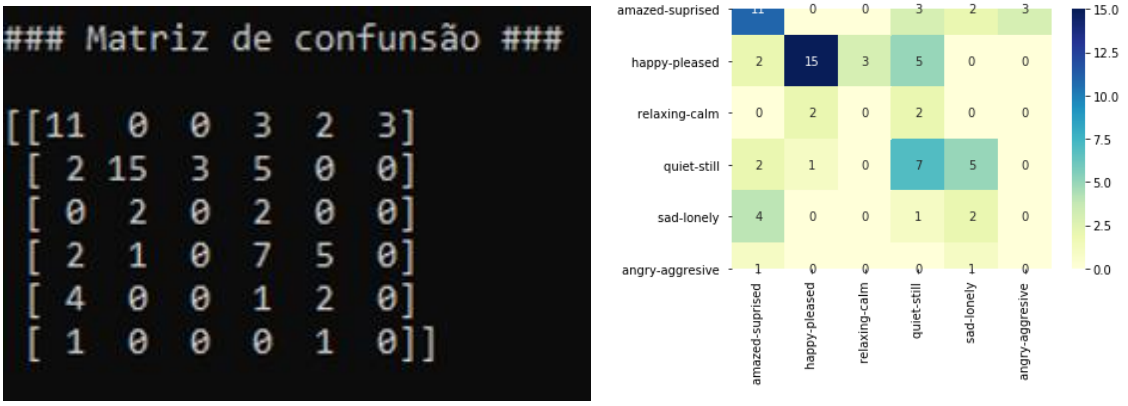


Figura 12 KNN - Matriz de confusão

A próxima medida a ser analisada é a acurácia. Com ela, podemos mensurar a precisão da nossa medição em relação os valores reais (valores de referência). Como podemos observar, a acurácia obtida foi consideravelmente baixa e o melhor resultado obtido foi de 36%

```
Acurácia : 36.11%
```

Figura 13 KNN – Acurácia

A próxima analisada é o score f1. Com ela podemos mensurar a precisão do nosso teste. É valido lembrar que uma pontuação f1 atinge seu melhor valor em 1 (precisão e recuperação perfeitas) e pior em 0.

```
Score : 0.71
```

Figura 14 KNN - Score f1

Por fim, vamos analisar a sensibilidade e a especificidade do nosso teste. A **sensibilidade** diz respeito a capacidade do teste de identificar corretamente os indivíduos que *pertencem* a classe analisada e a **especificidade** diz respeito a capacidade do teste de identificar corretamente os indivíduos que *não pertencem* a classe analisada. Abaixo podemos ver os resultados obtidos.

Como podemos observar, o algoritmo possui uma sensibilidade baixa para ambas as classes, porém possui uma especificidade razoável, principalmente para as classes "angry-aggressive" e "relaxing-calm".

Out[193]:

	amazed-suprised	happy-pleased	relaxing-calm	quiet-still	sad-lonely	angry-aggresive
Sensibilidade	0.58	0.6	0.0	0.47	0.29	0.0

Figura 13 KNN – Sensibilidade

Out[192]:

	amazed-suprised	happy-pleased	relaxing-calm	quiet-still	sad-lonely	angry-aggresive
Especificidade	0.73	0.87	0.92	0.72	0.8	0.92

Figura 14 KNN - Especificidade

1.3 – SVM

Support Vector Machine" (SVM) é um algoritmo de aprendizado de máquina supervisionado que pode ser usado tanto para problemas de classificação como para problemas de regressão. Resumidamente, o SVM busca encontrar uma linha de separação entre dados de duas classes. Essa linha busca maximizar a distância entre os pontos mais próximos em relação a cada uma das classes.

Para este algoritmo também foram utilizadas funções da biblioteca scikit-learn. Como as definições dos resultados obtidos já foram discutidas previamente anteriormente, aqui apresentaremos apenas os resultados em si, sem nos aprofundarmos em suas definições. Abaixo podemos ver os resultados obtidos.

Figura 15 SVM - Matriz de confusão

Como podemos observar, novamente a classe "happy-pleased" obteve o melhor resultado, com 11 previsões corretas de 15 amotras. As classes "sad-lonely" e "angry-aggresive" obtiveram péssimos resultados.

Abaixo podemos ver os valores de acurácia e score obtidos. Como podemos observar, o SVM obteve resultados muito semelhantes aos obtidos com o KNN.

```
Acurácia : 38.89%
```

Figura 16 SVM – Acurácia

```
Score : 0.73
```

Figura 17 SVM - Score f1

Por fim, obtemos a sensibilidade e a especificidade do SVM. Como podemos ver, a classe "amazed-suprised" obteve uma sensibilidade consideravelmente maior quando comparado com a sensibilidade do KNN. Além disso, com exceção da classe "amazed-suprised", todos os resultados de especificidade obtidos foram superiores aos do KNN.

Out[202]:

	amazed-suprised	happy-pleased	relaxing-calm	quiet-still	sad-lonely	angry-aggresive
Sensibilidade	0.91	0.55	0.22	0.6	0.0	0.0

Figura 18 SVM - Sensibilidade

Out[201]:

	amazed-suprised	happy-pleased	relaxing-calm	quiet-still	sad-lonely	angry-aggresive
Especificidade	0.65	0.89	0.98	0.75	0.98	0.98

Figura 19 SVM - Especificidade

1.4 – Conclusão

Após a análise dos dados, podemos concluir que ambos os algoritmos utilizados não apresentaram bons resultados. Isso se dá pelo fato de que tanto o KNN como o SVM são bons para casos onde os dados possuem classes bem definidas, o que não é o caso da base de dados analisada, tendo em vista que uma determinada música pode possuir mais de uma classificação no que diz respeito a sua emoção.

2 – Regressão

A regressão é um método de aprendizagem e maquina supervisionada. Ela consiste em prever um valor numérico especifico. Neste trabalho, iremos testar dois métodos de aprendizagem de máquina para identificar o que apresenta melhores resultados na previsão do volume de tráfego.

2.1 – Pré-processamento

Abaixo podemos ver algumas informações da nossa base de dados. A base de dado possui 48203 instancias e 9 atributos.

Out[13]:

	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	date_time	traffic_volume
0	None	288.28	0.0	0.0	40	Clouds	scattered clouds	2012-10-02 09:00:00	5545
1	None	289.36	0.0	0.0	75	Clouds	broken clouds	2012-10-02 10:00:00	4516
2	None	289.58	0.0	0.0	90	Clouds	overcast clouds	2012-10-02 11:00:00	4767
3	None	290.13	0.0	0.0	90	Clouds	overcast clouds	2012-10-02 12:00:00	5026
4	None	291.14	0.0	0.0	75	Clouds	broken clouds	2012-10-02 13:00:00	4918
...	...	...	...	...	...	...	...	...	...
48199	None	283.45	0.0	0.0	75	Clouds	broken clouds	2018-09-30 19:00:00	3543
48200	None	282.76	0.0	0.0	90	Clouds	overcast clouds	2018-09-30 20:00:00	2781
48201	None	282.73	0.0	0.0	90	Thunderstorm	proximity thunderstorm	2018-09-30 21:00:00	2159
48202	None	282.09	0.0	0.0	90	Clouds	overcast clouds	2018-09-30 22:00:00	1450
48203	None	282.12	0.0	0.0	90	Clouds	overcast clouds	2018-09-30 23:00:00	954

48204 rows × 9 columns

Figura 20 Metro_Interstate_Traffic_Volume.csv

Também podemos observar que nenhum dos atributos possui instancias com valores incompletos (nulos).

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48204 entries, 0 to 48203
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   holiday              48204 non-null  object
 1   temp                 48204 non-null  float64
 2   rain_1h              48204 non-null  float64
 3   snow_1h              48204 non-null  float64
 4   clouds_all           48204 non-null  int64
 5   weather_main         48204 non-null  object
 6   weather_description  48204 non-null  object
 7   date_time            48204 non-null  object
 8   traffic_volume       48204 non-null  int64
dtypes: float64(3), int64(2), object(4)
memory usage: 2.6+ MB
None
```

Figura 21 Informações da base de dados

Verificando os valores máximo e mínimo de cada atributo, podemos observar que os dados não estão normalizados.

```
### Min - Max ###

                holiday    temp  ...            date_time  traffic_volume
Min         Christmas Day    0.00  ...  2012-10-02 09:00:00               0
Max  Washingtons Birthday  310.07  ...  2018-09-30 23:00:00            7280

[2 rows x 9 columns]

--------------------
```

Figura 22 Dados não normalizados

Após realizar a normalização, podemos observar que agora os dados se encontram entre o intervalor desejado.

```
### Min - Max ###

     temp  rain_1h  snow_1h  ...  weather_main  weather_description            date_time
Min   0.0      0.0      0.0  ...         Clear              SQUALLS  2012-10-02 09:00:00
Max   1.0      1.0      1.0  ...  Thunderstorm      very heavy rain  2018-09-30 23:00:00

[2 rows x 9 columns]

--------------------
```

Figura 23 Dados normalizados

Após normalizarmos os dados, vamos analisar a matriz de correlação. Como podemos notar, os atributos numéricos possuem uma baixa correlação. Assim, após aplicarmos o algoritmo utilizado anteriormente, nenhuma coluna será correlacionada.

Out[17]:

	temp	rain_1h	snow_1h	clouds_all	traffic_volume
temp	1	0.0091	0.02	0.1	0.13
rain_1h	0.0091	1	9e-05	0.0048	0.0047
snow_1h	0.02	9e-05	1	0.028	0.00073
clouds_all	0.1	0.0048	0.028	1	0.067
traffic_volume	0.13	0.0047	0.00073	0.067	1

Figura 24 Matriz de correlação

Por fim, iremos tratar os outliers utilizando a regra do 1.5*FIQ. Após aplicarmos o algoritmo, 7% das instancias foram identificadas como outlier.

	temp	rain_1h	snow_1h	clouds_all	traffic_volume
count	48204.000000	48204.000000	48204.000000	48204.000000	48204.000000
mean	0.906911	0.000034	0.000436	0.493622	0.447777
std	0.043017	0.004556	0.016015	0.390158	0.272920
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.877737	0.000000	0.000000	0.010000	0.163874
50%	0.910923	0.000000	0.000000	0.640000	0.464286
75%	0.941097	0.000000	0.000000	0.900000	0.677610
max	1.000000	1.000000	1.000000	1.000000	1.000000

Figura 25 df.describle()

	temp	rain_1h	snow_1h	clouds_all	traffic_volume	holiday	weather_main	weather_description	date_time
0	0.929726	0.0	0.0	0.40	0.761676	None	Clouds	scattered clouds	2012-10-02 09:00:00
1	0.933209	0.0	0.0	0.75	0.620330	None	Clouds	broken clouds	2012-10-02 10:00:00
2	0.933918	0.0	0.0	0.90	0.654808	None	Clouds	overcast clouds	2012-10-02 11:00:00
3	0.935692	0.0	0.0	0.90	0.690385	None	Clouds	overcast clouds	2012-10-02 12:00:00
4	0.938949	0.0	0.0	0.75	0.675549	None	Clouds	broken clouds	2012-10-02 13:00:00
...	...	...	...	...	...	...	...	...	...
48199	0.914148	0.0	0.0	0.75	0.486676	None	Clouds	broken clouds	2018-09-30 19:00:00
48200	0.911923	0.0	0.0	0.90	0.382005	None	Clouds	overcast clouds	2018-09-30 20:00:00
48201	0.911826	0.0	0.0	0.90	0.296566	None	Thunderstorm	proximity thunderstorm	2018-09-30 21:00:00
48202	0.909762	0.0	0.0	0.90	0.199176	None	Clouds	overcast clouds	2018-09-30 22:00:00
48203	0.909859	0.0	0.0	0.90	0.131044	None	Clouds	overcast clouds	2018-09-30 23:00:00

[44669 rows x 9 columns]

Figura 26 Base resultante

2.2 – KNN

Vamos realizar dois testes utilizando o KNN como regressão. A base será dívida em 70% para treinamento e 30% para testes (ambos os dados serão selecionados de forma aleatória). Após executarmos o primeiro teste, obtemos os seguintes resultados.

```
Out[122]: 0.9076104690048761
```

Figura 27 teste 1 - r2_score

```
Out[128]: <matplotlib.axes._subplots.AxesSubplot at 0x1f254400d88>
```

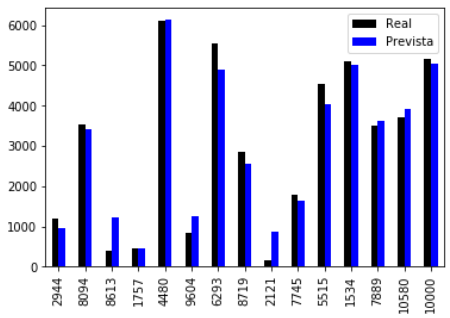


Figura 28 teste 1 - Valores reais x valores previstos

A função r2 score é utilizada como métrica para problemas de regressão. Quanto mais próxima de 1.0, melhor é seu resultado. Após isso, executamos novamente nossos testes. Abaixo podemos ver os resultados obtidos.

```
Out[129]: 0.9064389449171398
```

Figura 29 teste 2 - r2 score

```
Out[130]: <matplotlib.axes._subplots.AxesSubplot at 0x1f254549848>
```

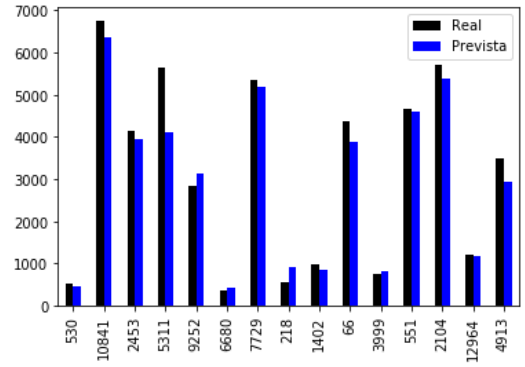


Figura 30 teste 2 - Valores reais x valores previstos

2.3 – KNN validação cruzada

A validação cruzada é uma técnica utilizada para avaliar a capacidade de generalização de um modelo, a partir de um conjunto de dados. Utilizando a biblioteca *sklearn* podemos testar o quão eficaz o nosso modelo gerado pelo KNN é. Abaixo podemos observar os *scores* obtidos.

```
## Score validação cruzada

[0.91230946, 0.90724507, 0.90557015, 0.90810872, 0.90461817, 0.90847381, 0.90671871, 0.90993087, 0.91679691, 0.90476481]
```

Figura 31 Score - Validação cruzada

3 – Clusterização

A clusterização é o agrupamento de instâncias similares, utilizada para classificação não-supervisionada dos dados. Os dados são classificados em conjuntos que "se assemelham" de alguma forma, independentemente de classes predefinidas.

3.1 – Pré-processamento

Assim como as analises anteriores, vamos iniciar o tratamento dos dados realizando o pré-processamento. Abaixo podemos ver um trecho da base de dados

Out[35]:

	status_id	status_type	status_published	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	nu
0	246675545449582_1649696485147474	video	4/22/2018 6:00	529	512	262	432	92	3	
1	246675545449582_1649426988507757	photo	4/21/2018 22:45	150	0	0	150	0	0	
2	246675545449582_1648730588577397	video	4/21/2018 6:17	227	236	57	204	21	1	
3	246675545449582_1648576705259452	photo	4/21/2018 2:29	111	0	0	111	0	0	
4	246675545449582_1645700502213739	photo	4/18/2018 3:22	213	0	0	204	9	0	
...	...	...	...	...	...	...	...	...	...	
7045	1050855161656896_1061863470556065	photo	9/24/2016 2:58	89	0	0	89	0	0	
7046	1050855161656896_1061334757275603	photo	9/23/2016 11:19	16	0	0	14	1	0	
7047	1050855161656896_1060126464063099	photo	9/21/2016 23:03	2	0	0	1	1	0	
7048	1050855161656896_1058663487542730	photo	9/20/2016 0:43	351	12	22	349	2	0	
7049	1050855161656896_1050858841656528	photo	9/10/2016 10:30	17	0	0	17	0	0	

7050 rows × 16 columns

Figura 32 Trecho Live.csv

Iniciamos removendo 4 colunas que possuem todos os valores nulos e a coluna *status_id* que possui informações que não serão utilizadas. Além disso algumas instancias duplicadas foram identificadas e removidas. Abaixo está a base resultante.

Out[47]:

	status_type	status_published	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys
0	video	4/22/2018 6:00	529	512	262	432	92	3	1	1	0
1	photo	4/21/2018 22:45	150	0	0	150	0	0	0	0	0
2	video	4/21/2018 6:17	227	236	57	204	21	1	1	0	0
3	photo	4/21/2018 2:29	111	0	0	111	0	0	0	0	0
4	photo	4/18/2018 3:22	213	0	0	204	9	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
7045	photo	9/24/2016 2:58	89	0	0	89	0	0	0	0	0
7046	photo	9/23/2016 11:19	16	0	0	14	1	0	1	0	0
7047	photo	9/21/2016 23:03	2	0	0	1	1	0	0	0	0
7048	photo	9/20/2016 0:43	351	12	22	349	2	0	0	0	0
7049	photo	9/10/2016 10:30	17	0	0	17	0	0	0	0	0

6996 rows × 11 columns

Figura 33 Trecho Live.csv

Além disso, vamos remover as colunas *num_reactions* (pois ela apresenta a mesma informação presente nas demais colunas referentes as reações) e a coluna status_*published* (não apresenta informações relevantes). Por fim, temos a seguinte base de dados.

Out[59]:

	status_type	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys
0	3	512	262	432	92	3	1	1	0
1	1	0	0	150	0	0	0	0	0
2	3	236	57	204	21	1	1	0	0
3	1	0	0	111	0	0	0	0	0
4	1	0	0	204	9	0	0	0	0
...	...	...	...	...	...	...	...	...	...
6974	1	0	0	89	0	0	0	0	0
6975	1	0	0	14	1	0	1	0	0
6976	1	0	0	1	1	0	0	0	0
6977	1	12	22	349	2	0	0	0	0
6978	1	0	0	17	0	0	0	0	0

6979 rows × 9 columns

Figura 34 Live.csv final

3.2 – Agrupamento K-means

O k-means é um algoritmo do tipo não supervisionado. O objetivo desse algoritmo é encontrar similaridades entre os dados e agrupá-los conforme o número de clusters disponíveis, utilizando o conceito de distância.

Vamos iniciar nosso teste utilizando 4 clusters. É valido ressaltar que os centroides foram iniciados de forma randômica. Abaixo podemos ver o resultado obtido.

Out[65]:

	nInstancias
0	6064
1	435
2	376
3	104

Figura 35 k-means com 4 clusters

Após isso, vamos aumenta o número de clusters para 15. Abaixo podemos ver o resultado obtido.

Out[66]:

	nInstancias
0	4772
1	771
2	451
3	226
4	223
5	181
6	101
7	57
8	52
9	50
10	35
11	26
12	22
13	11
14	1

Figura 36 k-means com 15 clusters

3.3 – Agrupamento hierárquico

Agora vamos analisar os dados utilizando o agrupamento hierárquico. Iniciamos os testes utilizando o single linkage que, processo de concatenação, utiliza a menor distancia existente entre dois clusters. Abaixo podemos ver o primeiro teste utilizando 4 clusters.

Out[60]:

	nInstancias
0	6976
1	1
2	1
3	1

Figura 37 single com 4 clusters

Logo em seguida, aumentamos o número de clusters para 15. Abaixo podemos observar o resultado obtido.

Out[61]:

	nInstancias
0	6964
1	2
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1

Figura 38 single com 15 clusters

Agora vamos analisar utilizando o complete linkage. Nele, o processo de concatenação utiliza a maior distância existente entre dois clusters. Abaixo podemos ver o primeiro teste utilizando 4 clusters.

Out[63]:

	nInstancias
0	6716
1	230
2	32
3	1

Figura 39 complete com 4 clusters

Logo em seguida, aumentamos o número de clusters para 15. Abaixo podemos observar o resultado obtido.

Out[64]:

	nInstancias
0	6288
1	239
2	158
3	101
4	53
5	28
6	25
7	24
8	23
9	15
10	13
11	6
12	4
13	1
14	1

Figura 40 complete com 15 clusters

3.4 – Conclusão

Como podemos observar, o agrupamento hierárquico single linkege obteve o pior resultado entre os testes realizados. O agrupamento hierárquico complete linkege também obteve resultados consideravelmente ruins para este conjunto de dados. Por fim, o algoritmo k-mens apresentou resultados mais equilibrados, porém apenas um especialista pode afirmar se os clusters criados fazem ou não sentido.