Atividade 06

Eduardo Satiro da Cruz
PPGMMC
CEFET-MG
Belo Horizonte, Brasil
eduardo.satiro@gmail.com

***Abstract*—Esse trabalho informa passos que foi feito para utilização do algoritmo *K Neighbors Classifier* no conjuntos de dados *Airlines*.**

***Index Terms*—pré-processamento, conjunto de dados, classificação, KNN.**

I. Introdução

Na disciplina de Inteligência Computacional, ministrada pelo dr. Alisson Marques da Silva no CEFET-MG, foi solicitado uma atividade que consiste em escolher um conjunto de dados e realizar as etapas do pré-processamento até a utilização de um algoritmo de inteligência artificial. O *dataset* escolhido foi através do Kaggle [1]. Para os procedimentos foi utilizado a linguagem *Python* com as bibliotecas: *pandas, numpy, seaborn, matplotlib, sklearn* e *scipy*.

II. Conjunto de Dados: *Airlines*

O conjunto de dados é o *Airlines* [2] e o objetivo é prever se o voo irá atrasar, dada a informação da partida programada.

Inicialmente foi criado um *DataFrame* da bibioteca *pandas* para a manipulação dos dados. A função utilizada foi *read_csv* que carrega o arquivo baixado com extensão CSV para o *DataFrame* criado.

	Flight	Time	Length	Airline	AirportFrom	AirportTo	DayOfWeek	Class
0	2313.0	1296.0	141.0	DL	ATL	HOU	1	0
1	6948.0	360.0	146.0	OO	COS	ORD	4	0
2	1247.0	1170.0	143.0	B6	BOS	CLT	3	0
3	31.0	1410.0	344.0	US	OGG	PHX	6	0
4	563.0	692.0	98.0	FL	BMI	ATL	4	0

Fig. 1. Cinco primeiros registros do conjunto de dados *Airlines*

Na Fig. 1 consta o comando *head* da biblioteca *pandas* que apresenta os cinco primeiros registros do *dataset*. Existe seis atributos sendo o *Flight* descrito como ID do voo, *Time* como tempo da partida, *Length* como a distância do voo, *Airline* como identificação da companhia aérea, *AirportFrom* de qual aeroporto partiu o voo, *AirportTo* para destino do voo, *DayOfWeek* dia da semana do voo. A saída da previsão é a última coluna *Class* que contém dois valores: 1 para atrasado e 0 para pontual.

Utilizando a função *shape* da biblioteca *numpy* retorna como valor: "(539382, 8)", isso significa que existe 539382 linhas e 8 colunas no conjunto de dados. Na Fig. 2 retorna o resultado

```
Flight         0
Time           0
Length         0
Airline        0
AirportFrom    0
AirportTo      0
DayOfWeek      0
Class          0
dtype: int64
```

Fig. 2. Soma dos valores nulos de cada atributo.

da função *isnull* com a *sum* que mostra se há algum valor nulo no conjunto de dados, não há valores vazios.

	Flight	Time	Length	DayOfWeek	Class
count	539382.000000	539382.000000	539382.000000	539382.000000	539382.000000
mean	2427.927988	802.728161	132.202104	3.929666	0.445443
std	2067.431700	278.045546	70.117045	1.914666	0.497015
min	1.000000	10.000000	0.000000	1.000000	0.000000
25%	712.000000	565.000000	81.000000	2.000000	0.000000
50%	1809.000000	795.000000	115.000000	4.000000	0.000000
75%	3745.000000	1035.000000	162.000000	5.000000	1.000000
max	7814.000000	1439.000000	655.000000	7.000000	1.000000

Fig. 3. Análise estatística do *dataset*.

A Fig. 3 apresenta a descrição estatística dos dados, isso foi retorno da função *describe* da biblioteca *pandas*.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 539382 entries, 0 to 539381
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Flight       539382 non-null  float64
 1   Time         539382 non-null  float64
 2   Length       539382 non-null  float64
 3   Airline      539382 non-null  object
 4   AirportFrom  539382 non-null  object
 5   AirportTo    539382 non-null  object
 6   DayOfWeek    539382 non-null  int64
 7   Class        539382 non-null  int64
dtypes: float64(3), int64(2), object(3)
memory usage: 32.9+ MB
```

Fig. 4. Informações sobre o *DataFrame Airlines*.

A Fig. 4 apresenta informações sobre o *DataFrame*, incluindo o tipo do dado e colunas, valores não nulos e uso de memória. O conjunto de dados é relativamente pequeno consumindo pouca memória. Além disso, demostra que existem três variáveis não numéricas que precisam serem modificadas

pois o algoritmo de classificação exige que os atributos sejam apenas numéricos.

Para a transformação das variáveis alfanuméricas em numéricas foi usando a função *LabelEncoder* da biblioteca *sklearn*, ela normaliza os rótulos de forma que contenham apenas valores entre 0 e número de classes menos um. Essa técnica foi utilizada nas três variáveis: *Airline*, *AirportFrom* e *AirportTo*.

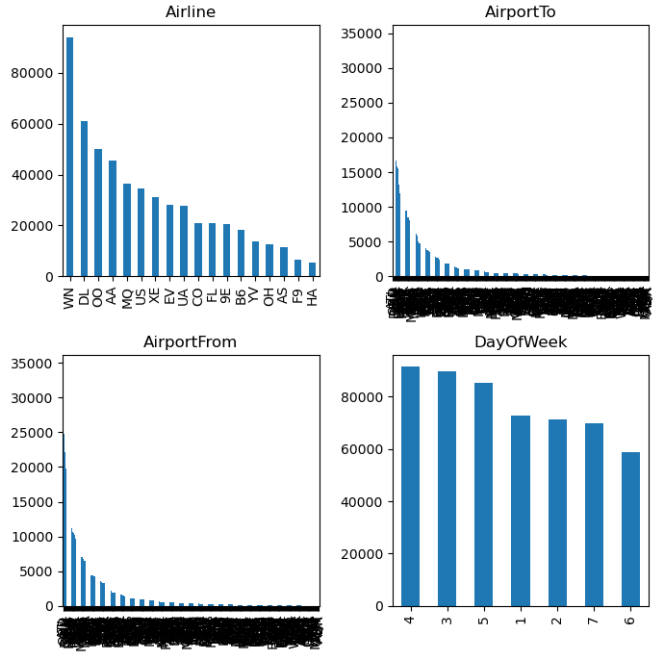

Fig. 5. Gráficos de barra de quatro atributos do *Airlines*.

A Fig. 5 possui o quatro gráficos de barras e nota-se que os gráficos do *AirportTo* e *AirportFrom* ficam difíceis de analisar pois existem vários locais de origem e destino. Sobre o *DayOfWeek* nota-se que não há diferença significativa entre os dias da semana, porém pelo gráfico da *Airline* há companhias que apresentam muitos registros de voos em relação a outras.

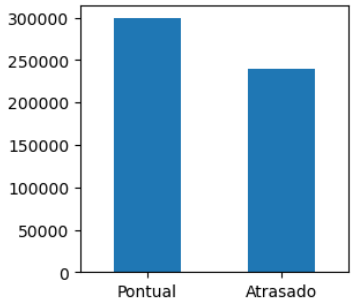

Fig. 6. Gráfico de barras da variável alvo do *Airlines*.

Na Fig. 6 consta um gráfico de barras da variável alvo *Class* e observa-se um equilíbrio entre os resultados.

Com essas análises e algumas modificações, o algoritmo do *Nearest Neighbors* da biblioteca *sklearn* pode ser utilizado.

III. Metodologia

A. Airlines

Para o *dataset Airlines* foi utilizado a técnica de validação cruzada *k-fold*. Essa técnica consiste em dividir o conjunto total em **k** subconjuntos de tamanhos iguais (se possível) e um subconjunto é utilizado para teste e os **k - 1** restantes são utilizados para dados de treinamento [4]. A função de previsão é aprendida usando **k - 1** *folds*, e a *fold* deixada de fora é usada para teste. O algoritmo utilizado foi o *K Neighbors Classifier* da biblioteca *SciKit Learn*. Esse algoritmo possui oito parâmetros:

- *n_neighbors* que significa o número de vizinhos a serem usados
- *weights* que é o peso usado na previsão
- *algorithm* que é o algoritmo usado para calcular os vizinhos mais próximos que pode ser: *ball_tree, kd_tree, brute* ou *auto*
- *leaf_size* que é o tamanho da folha passado para o algoritmo *BallTree* ou *KDTree*. Isso pode afetar a velocidade da construção e consulta, bem como a memória necessária para armazenar a árvore e o valor ótimo depende da natureza do problema.
- *p* parâmetro de potência para a métrica de *Minkowsk*. Quando p = 1, isso é equivalente a usar *manhattan_distance* e para p = 2 *euclidean_distanc*
- *metric* métrica a seu usada para cálculo de distância
- *metric_params* argumentos adicionais para a função métrica
- *n_jobs* números de tarefas paralelas a serem executadas para pesquisa de vizinhos

Como esse trabalho não tem objetivo de ter uma análise profunda e nem testar a eficiência do algoritmo, foi utilizado os valores *default* para os parâmetros do algoritmo afim de obter uma acurácia maior que 65%. Foi utilizado todas os atributos como entrada para o algoritmo.

Para a avaliação dessa técnica foi utilizada a matriz de confusão, taxa de verdadeiros positivos (TVP), taxa de falsos positivos (TFP), acurácia (ACC), erro (E), precisão (Pr), revocação (Re), F1 *Score*.

IV. Experimentos

Foi utilizado o algoritmo KNN com os parâmetros *default* com o *k-fold* e obteve uma acurácia de 64,29%. Na tentativa de obter um resultado melhor, foi feito vários modelos alterando os parâmetros, para o parâmetro *n_neighbors* foram testado com os valores 2, 5 e 10, para o parâmetro *weights* foram utilizados os pesos *uniform e distance*, *algorithm* foram os *ball_tree, kd_tree e brute* e as *metric* teve *cityblock, euclidean, l2 e manhattan*. Para os atributos *leaf_size e p* se mantiveram os valores padrões, 30 e 2 respectivamente. Duas combinações tiveram o melhor resultado e parâmetros foram *uniform, ball_tree, 30, 2, cityblock* e *uniform, ball_tree, 30, 2, manhattan* com acurácia de 65,36%. Para a avaliação do modelo tiveram os resultados de acordo com a tabela

TVP	79.80%
TFP	52.62%
ACC	65.36%
E	34,64
Pr	65.37
Re	79.80
F1 Score	71.87

V. Considerações e Discussões

As variações que foram obtidos dos modelos teve acurácia entre 61.20% e 65.35%, demonstrando que houve pouca diferença entre eles, assim, com esse algoritmo de classificação, não é possível obter um modelo com desempenho significamente diferente desses. Poderia em trabalhos futuros testar outros modelos afim de tentar uma melhor performace.

References

[1] "Kaggle" https://www.kaggle.com (accessed Mar. 21, 2023)

[2] Pedersen, U. T. (2023, February). Airlines Delay, Version 1. Retrieved March 21, 2023 from https://www.kaggle.com/datasets/ulrikthygepedersen/airlines-delay.

[3] Choi, M. (2018, February). Medical Cost Personal Datasets, Version 1. Retrieved March 21, 2023 from https://www.kaggle.com/datasets/mirichoi0218/insurance.

[4] "Scikit Learn" https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html (accessed Abr. 7, 2023)

[5] "Scikit Learn" https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html#sklearn.neural_network.MLPRegressor (accessed Abr. 7, 2023)