Projeto II – WeRateDogs Tweeter

Por Leandro Almeida

O objetivo do projeto é apresentar análises e visualizações das mensagens no tweeter do usuário WeRateDogs. A principal intenção, na realidade, é aplicar os conceitos de data wrangling. As análises e visualizações necessárias para conclusão são uma forma de motivar além de mostrar a importância do processo de data wrangling na análise de dados.

Esse relatório tem a finalidade de documentar as ações requeridas e dificuldades encontradas durante de coletar, avaliar e limpar.

- Coletar

No processo de coleta, foram executados códigos para obter arquivos de fontes diferentes.

A primeira delas foi a simulação de um arquivo em mãos. O documento que continha posts da conta estava no formato de arquivo simples *".csv"* cujas informações foram carregadas utilizando o método *"read_csv()"*.

A segunda fonte foi um arquivo com predições de raças de cachorro a partir da análise das fotos postadas. O arquivo estava disponível nos servidores da Udacity e foi necessário baixar por meio de programação. Nessa ocasião, utilizamos a biblioteca "*Requests*" para completar a tarefa.

A terceira fonte foi a API do Twitter. Nessa parte houve uma pequena dificuldade com as chaves de acesso. Para evitar escreve-las no Jupyter Notebook, foi escolhido criar um arquivo "*.txt"* com as chaves e importa-las através do método "*.readline()*". Entretanto, foi cometido um erro conceitual, pois foi esquecido a existência do caractere de nova linha (***\n)*** que não é exibido. A solução foi encontrada em uma postagem no *StackOverflow* [1].

- Avaliar

A parte de avaliação foi concluída com apenas uma dificuldade: visualização campos com textos longos. As colunas text, source e extended_url foram avaliadas com auxílio do Microsoft Excel. O motivo foi a necessidade de ler o texto completo para entender como a Expressão Regular (regex) deveria ser escrita para retirar as informações de nome e classificação do cachorro e fonte da postagem. A coluna com endereço URL foi avaliado para avaliar se havia um padrão de construção e foi percebido que é uma concatenação de stings: *'https://twitter.com/'*, *usuário*, *'/status/'* e id do tweet.

- Limpar

Como já era esperado, essa etapa foi a mais demorada e com maior dificuldade de ser concluída. Alguns códigos necessários para executar as limpezas não eram conhecidos e foi necessário pesquisa e ler a documentação para utilizar os métodos que produzissem o resultado esperado.

Uma das ações de limpeza que houve mais dificuldade de implementar foi a necessária para substituir valores ausentes com valores de outra coluna, porém sem perder dados não-nulos. Foi necessário executar a rotina para os nomes e as classificações dos cachorros.

No caso dos nomes dos cachorros, a informação era extraída executando três expressões regulares em sequência. A primeira expressão executada, simplesmente criava a coluna com a informação dos nomes dos cachorros. A partir da segunda expressão, foi escolhido criar um novo dataframe para guardar os resultados das duas expressões separados e agregar os valores da segunda consulta através do método ***.fillna()***, porém com o parâmetro ***method='backfill'*** . Esse parâmetro permite que os valores NaN de uma série sejam preenchidos com o primeiro valor *não-NaN* encontrado nas séries a esquerda. Essa solução foi encontrada no fórum *StackOverflow* [2].

Na coluna de estágios de cachorro, como o interesse era manter todos os registros de todas as colunas que não fosses nulos, o método utilizado foi o ***.str.cat()***. O resultado é a concatenação de todos os valores, sendo que os valores nulos foram representados por uma string vazia para não ter algum valor no resultado. Contudo, essa opção resultou na necessidade de substituir as ocorrências onde todos os caracteres eram vazios para serem considerados como valor nulo.

Referências:

[1] https://stackoverflow.com/questions/3277503/how-to-read-a-file-line-by-line-into-a-list

[2] https://stackoverflow.com/questions/24015379/row-by-row-fillna-with-respect-to-a-specific-column