**SPBD 2018/2019**
MAEBD - NOVA FCT
Relatório - 2º Projeto

**João Palmeiro, 56036**
**Mário Gomes, 54667**

# 1 Introdução

Neste trabalho foi-nos prosposto, dado o conhecimento de um *dataset* contendo informação acerca das viagens de táxi na cidade de Nova York segundo um conjunto de 17 variáveis, incluindo as coordenadas GPS (latitude e longitude) da posição onde o indivíduo apanhou o táxi, a despesa da viagem e a forma de pagamento, que identificássemos as rotas mais frequentes bem como as áreas que geravam um maior lucro. No término deste trabalho é-nos também proposto que fossem obtidas as localizações de possíveis *stands* de táxis, por forma a minimizar a distância que os clientes teriam de se deslocar no sentido de apanhar um táxi.

No sentido de proceder à obtenção dos resultados pretendidos, foi necessário, através das coordenadas GPS das posições onde começou e terminou uma viagem, obter um conjunto de zonas não sobrepostas, a partir das quais fosse possível executar as computações posteriores. Assim, a estratégia adoptada foi proceder a um arredondamento a duas casa decimais das coordenadas. Com este procedimento, é garantido não só que cada posição pertence a uma e uma só zona, bem como pontos suficientemente próximos pertencem à mesma zona. Em particular, o número de zonas será igual ao número distinto de coordenadas GPS anteriormente arredondadas.

# 2 Pré-processamento

Uma vez que o *dataset* utilizado contém diversas viagens que podemos considerar como "ruído", procedemos a um pré-processamento dos dados com o intuito de ignorar as entradas com duração de viagem nula e/ou com coordenadas geográficas (latitude e longitude) fora dos limites considerados para os dados recolhidos. Deste modo, antes de realizarmos cada uma das *queries* solicitadas, são realizadas duas filtrações dos dados:

- Remoção das viagens cuja duração é menor ou igual a 0 (com base na variável *trip_time_in_secs*).
- Remoção das viagens cujas latitudes e longitudes, quer de entrada, quer de saída, não se encontram nos limites definidos (com base nas variáveis *pickup_longitude*, *pickup_latitude*, *dropoff_longitude* e *dropoff_latitude*).

# 3 Query A

Nesta tarefa, o nosso objetivo era a identificação das rotas mais populares, isto é, quais os conjuntos de zonas em que o maior número de viagens de táxi se verificava entre as mesmas. Assim, a estratégia adoptada foi a seguinte:

- Inicialmente, foi obtido um novo *dataset* a partir do primeiro, no sentido de garantir que neste não existem viagens de táxi de duração nula, bem como que as viagens de táxi se verificavam de uma zona para outra que não a primeira, uma vez que não interessam viagens que se verifiquem dentro da mesma zona.
- Seguidamente, foram identificadas e agregadas as viagens em função do dia da semana e da hora, através das coordenadas GPS das zonas em que se verifica a viagem, bem como a sua contagem.
- Finalmente, segue-se a obtenção, obviamente que por dia da semana e hora, dos pares de zonas entre as quais se verificava um maior número de viagens.

# 4 Query B

Relativamente à *Query B*, o objetivo passava por descobrir as áreas (conforme definidas inicialmente no nosso contexto) mais rentáveis, apresentando-as organizadas por dia da semana e hora. Tendo em mente esta particularidade ao nível da apresentação e a fórmula proposta para o cálculo do lucro (que envolve considerar os últimos 15 e 30 minutos com base num determinado momento), adoptámos a seguinte estratégia interativa e a seguinte assumpção para a nossa implementação:

- Uma vez que a janela de tempo de interesse coincide com o período de uma determinada hora, considerámos o minuto coincidente com a metade de cada hora (minuto 30) como o nosso ponto de partida para a computação pedida, ou seja, para cada uma das 24 horas, considerámos apenas as viagens que aconteceram no primeiro período de 30 minutos de cada hora para realizar os cálculos necessários para o lucro.
- Contudo, dado que a assumpção apresentada não nos permite chegar a uma solução ótima, mas sim a uma espécie de estimativa, seguimos uma abordagem interativa, no sentido de permitirmos, caso seja interessante considerar outro minuto como ponto de partida, uma adaptação rápida da nossa solução ao ser necessário, apenas, alterar a marca temporal nas UDFs definidas em complemento à implementação da *query*.

Posto isto, a nossa solução desenrola-se da seguinte forma:

1. Em primeiro lugar, definimos 4 UDFs: *getTimestamp*, para definir o ponto de partida, para cada hora, das viagens que iremos considerar para a computação, *getTime30*, para facilitar a definição do período de 30 minutos para o qual queremos extrair o número de táxis livres numa determinada área, *getTime15*, para facilitar a definição do período de 15 minutos para o qual queremos extrair a média da tarifa e da gorjeta associada às viagens que começaram numa determinada área e terminaram neste período, e *getTop10*, para obtermos o top 10 de áreas mais rentáveis para cada dia da semana e hora no final.
2. Posto isto, em complemento ao pré-processamento já apresentado, procede-se à definição do (Spark) DataFrame de trabalho, que contará apenas com as variáveis originais necessárias e algumas novas, nomeadamente o dia da semana, a data e a hora de cada viagem. Optámos por utilizar esta tecnologia devido ao foro manipulativo de dados armazenados numa tabela que encontrámos neste problema.
3. Uma vez que esta *query* está dividida em três fases, começámos, primeiramente, por definir um DataFrame que contará com as viagens, para cada hora, cujo tempo de saída coincide com o primeiro período de 30 minutos de uma determinada hora. Este DataFrame, em conjunto com o próximo que será apresentado, serve para descobrirmos os táxis que podemos considerar como "livres"consoante o enunciado. Seguidamente, são removidos os táxis com o mesmo identificador único, para uma determinada hora e data (sem estas duas variáveis auxiliares poderíamos remover táxis com o mesmo identificador, mas em períodos de tempo diferentes), de modo a agilizar a computação necessária.
4. De forma semelhante ao ponto anterior, procedemos à definição de um novo DataFrame, sendo que, contrariamente ao anterior, iremos considerar as viagens que começaram num determinado período de 30 minutos.
5. Com base nestes dois DataFrames, define-se um novo ao juntá-los em relação ao identificador único de cada táxi. Com este DataFrame, podemos facilmente identificar os táxis que, num determinado período de 30 minutos, terminaram uma viagem e que começaram, ou não, uma nova viagem. Esta condição é rapidamente verificada ao comparar se o tempo de entrada é maior que o tempo de saída, permitindo-nos definir uma nova variável binária, *taxi_busy*, que identifica os táxis que deveremos considerar para o cálculo dos táxis livres caso esta seja definida com o valor 0.

6. Filtrando o DataFrame anterior, considerámos apenas as entradas com os táxis que não estão ocupados e procedeu-se ao agrupamento e contagem destes para cada dia da semana, hora e área (sendo "área"definida por um par latitude-longitude).

7. Terminada a primeira fase, define-se um novo DataFrame que nos permite considerar o último período de 15 minutos, em relação ao ponto de partida inicial, para o qual iremos contabilizar a tarifa e a gorjeta conforme o necessário para o cálculo do lucro.

8. Mais uma vez, teremos um DataFrame agrupado por dia da semana, hora e área, com a diferença de este conter uma nova variável, *profit*, que contém o resultado do cálculo mencionado anteriormente.

9. Posto isto, inicia-se a última fase desta *query* com a definição da condição que queremos seguir aquando a última junção de DataFrames. Esta última junção permite-nos obter, para cada dia da semana, hora e área. o lucro associado. Como nota lateral, é importante destacar que foi acrescentada uma condição que considera o valor 1 quando estamos na situação em que não temos táxis livres, de modo a evitar divisões por 0. Para além disso, também se ordenou este DataFrame por *profit* para que as entradas seguissem uma ordem descrescente face a este valor, algo que facilitou a definição do DataFrame final.

10. Por fim, de modo a apresentar os resultados conforme solicitado no enunciado, resta-nos agrupar as entradas por dia da semana e hora, e agregar, numa só célula, as 10 áreas mais rentáveis para cada uma das entradas.

# 5 Query C

Numa última fase deste trabalho, pretende-se obter a localização de possíveis zonas para a construção de *stands* de táxis, de forma a minimizar a distância que cada cliente necessita de percorrer com o intuito de iniciar a sua viagem de táxi.

Assim, a nossa estratégia passou pela utilização do algortimo de *clustering* K-Means (através da implementação presente na biblioteca MLlib), dado que este se adequa ao problema em questão. Deste modo, o algoritmo irá agrupar num *cluster* $C_i$ todos os pontos cujas distâncias dos mesmos ao centróide de $C_i$ sejam mínimas relativamente aos restantes *clusters* $C_k$, por forma a minimizar a soma de quadrados interno característica de $C_i$, sendo que todos os pontos pertencem a um e só um cluster. Fazendo o paralelo com a presente situação, é fácil de observar que os pontos serão as coordenadas GPS (funcionando como os nossos dados de treino do modelo considerado) dos pontos onde se iniciaram as viagens e os centróides irão corresponder às localizações dos *stands*.

No sentido de escolher o número ótimo de *clusters* de entre um intervalo pré-selecionado de $k = [2, 50]$, procedemos ao cálculo da soma das distâncias quadradas internas (entre os pontos e os centros dos respetivos clusters) bem como ao cálculo do *silhouette score*, um indicador que toma valores entre -1 e 1 e que permite ter uma ideia do grau de afastamento entre os *clusters*. São de seguida mostrados os respetivos gráficos:

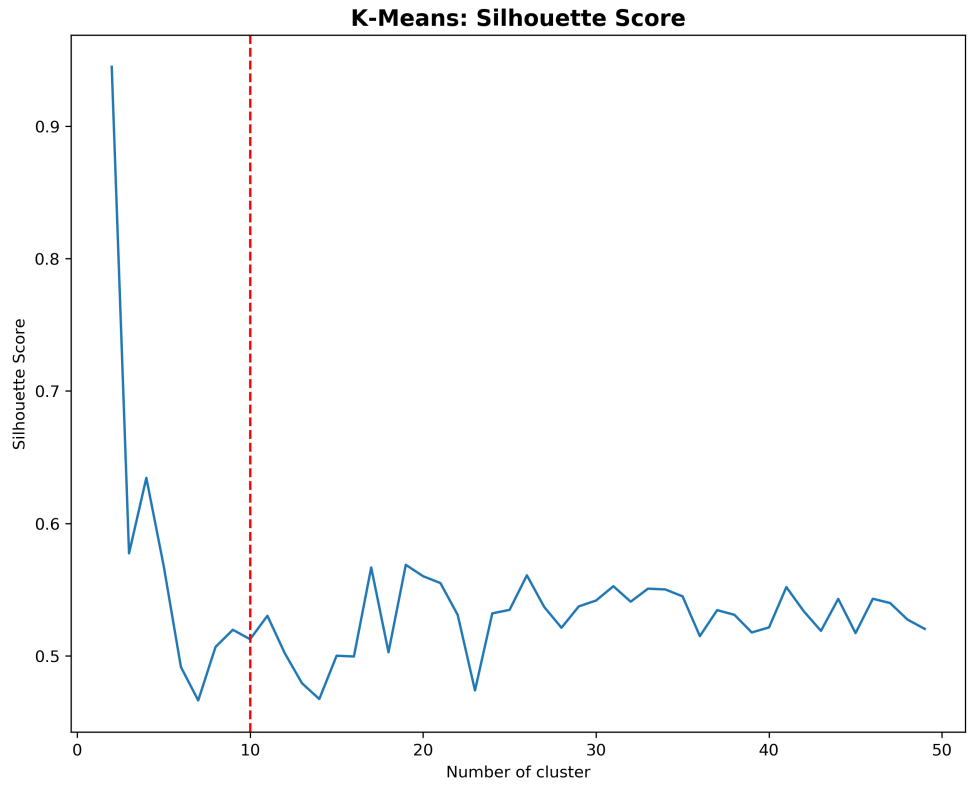


Figura 1: *Silhouette Score* para cada valor de $k$

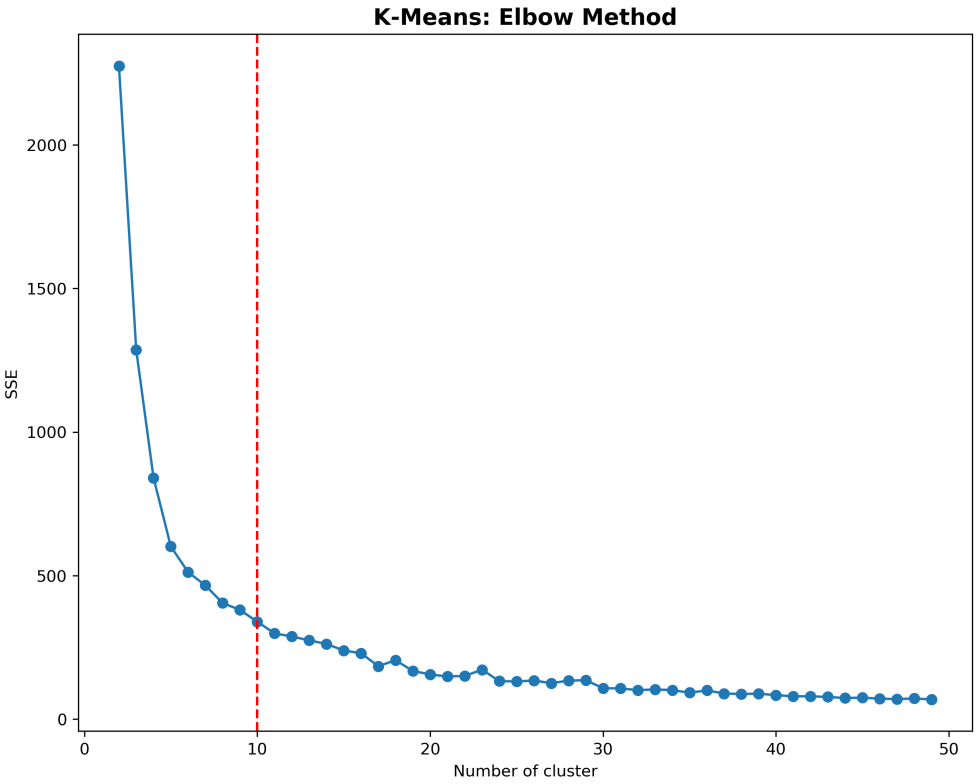


Figura 2: Soma das Distâncias Quadradas para cada valor de $k$

Tendo em particular atenção a Figura 2 e aplicando o método de *Elbow* (método de interpretação e validação de consistência dentro da análise de clusters projetada para ajudar a encontrar o número apropriado de clusters consoante um determinado conjunto de dados), chegámos à conclusão que o melhor número de clusters será provavelmente $k = 10$, uma vez que a partir deste valor não será

compensatório a construção de mais *stands*, já que as distâncias quadradas dos pontos alocados a um cluster aos seus centróides não tendem a diminuir. Note-se igualmente que para $k = 10$, o *silhouette score* é superior a 0.5 e desta forma sabemos que os *clusters* obtidos se encontram significativamente separados um dos outros, sendo também este um ponto positivo, uma vez que não haveria interesse em ter *stands* muito próximos uns dos outros. Assim, as coordenadas GPS de cada um dos *stands* serão:

1. (40.75844809, -73.97475739)
2. (40.64672616, -73.78498119)
3. (40.76946522, -73.8710348)
4. (40.73116434, -73.9943157)
5. (40.71614047, -73.94780892)
6. (40.70684687, -74.00614255)
7. (40.80582246, -73.95559961)
8. (40.75062002, -73.99008315)
9. (40.78058132, -73.9793871)
10. (40.77415471, -73.95485623)

Finalmente, tendo em mente este contexto, também será interessante considerar outro valor para $k$, como $k = 20$, uma vez que, com este número de clusters, a soma das distâncias quadradas entre os pontos e os centros dos respetivos clusters será bastante baixa (e não diminuirá significativamente caso consideremos um maior número de clusters), o que poderá evidenciar, caso se pretenda (e exista essa possibilidade num contexto real) um maior número de *stands*, que este número é mais interessante do que o anterior, sendo que um valor maior de $k$ implicará um maior investimento sem o aumento do benefício pretendido.