Relatório técnico final de projeto

Previsão do tempo de chegada dos ônibus

Universidade: PUC-Rio
Alunos participantes: Igor Rigolon, Rafael Palis
Professora: Nathalie Gimenes
Secretaria envolvida: Secretaria Municipal de Transportes

1. Introdução

A Secretaria Municipal de Transportes do Rio de Janeiro (SMTR) apresentou a professores do Departamento de Economia da PUC-Rio o projeto de gerar modelos capazes de prever o tempo de chegada dos ônibus aos pontos. Os alunos do Departamento de Economia Igor Rigolon (mestrado) e Rafael Palis (graduação), compuseram a equipe participante do projeto. Com a colaboração do Escritório de Dados, tivemos acesso, pelo Datalake da prefeitura, aos dados históricos de GPS dos ônibus da SMTR.

Com esses dados, treinamos modelos de Machine Learning capazes de prever em quantos minutos cada ônibus chegará em determinado ponto, até 20 pontos à frente. O modelo de melhor desempenho obteve um erro absoluto médio de menos de 2 minutos, e um erro percentual médio inferior a 20%. Esse modelo pode ser usado para prever os tempos de chegada dos ônibus em tempo real, sendo de imensa serventia para os usuários da rede de ônibus do Rio de Janeiro.

2. Objetivos

Para chegar em previsões de tempo de chegada, tivemos como passos intermediários:

1. Montar uma base de validação:
    - Calcular, nos dados históricos, os tempos de chegada que de fato ocorreram, para que os modelos possam aprender com o passado.
2. Estimar os modelos mais utilizados na literatura
3. Validar o desempenho dos modelos:
    - Treinar os modelos com dados apenas mais antigos, e medir quão bem eles prevêem os tempos de chegada nos dados mais recentes, para simular o desempenho na prática.

3. Dados

Utilizamos os dados de GPS dos ônibus e do GTFS da SMTR no Datalake da Prefeitura. A tabela abaixo resume os dados encontrados em cada uma das bases de dados.

Base de dados	Informação
GPS dos ônibus	Coordenadas dos ônibus a cada 30 segundos, com identificador do veículo, código do serviço, e horário
GTFS	Itinerários possíveis de cada serviço e todas as suas paradas

Fizemos o cruzamento entre as bases de dados do GTFS para coletar essa informações de itinerários e paradas, e então unimos esses dados aos de GPS.

4. Tratamento dos dados

Para conseguir treinar os modelos, precisamos montar uma base de validação que incluísse alguns dados adicionais. Foi necessário detectar qual itinerário cada ônibus estava percorrendo a cada momento, e detectar quando cada ponto foi ultrapassado. Esta seção descreve as escolhas que fizemos para construir essa base de dados. Todos os códigos de manipulação dos dados foram feitos em SQL, podendo ser implementados diretamente pelo BigQuery.

4.1. Identificação das shape_ids

Grande parte dos serviços de ônibus tem mais de itinerário possível descrito nos dados do GTFS – em maioria, um de ida e um de volta. Cada itinerário é acompanhado de um código, registrado na variável shape_id. Para prever os tempos de chegada, foi necessário identificar corretamente a shape_id de cada observação.

Usamos a função `ST_LINELOCATEPOINT`, no SQL do BigQuery, para projetar as coordenadas de cada ônibus em todos os itinerários possíveis de seu serviço. A função retorna um número entre 0 e 1, medindo a proporção do caminho que já foi viajada. Pela distância total de cada itinerário, calculamos quantos metros cada ônibus viajou desde seu ponto de partida.

No nosso código, declaramos que uma shape_id está correta se a distância viajada pelo ônibus cresce por 5 observações seguidas. Isso indica que o ônibus está, de fato, indo na direção prescrita por esse itinerário. Removemos observações em que ainda havia ambiguidade, restando mais de um itinerário possível para o ônibus.

Também removemos observações em que (i) o ônibus está a mais de 500 metros do itinerário, (ii) o ônibus não tem observações nos 3 minutos seguintes, ou (iii) a distância viajada pelo ônibus permanece constante nas 5 observações passadas.

4.2. Identificação das paradas

O GTFS registra todos as paradas de cada itinerário possível para um determinado serviço, com a variável `dist_traveled_stop`, que mede a quantos metros (ao longo do itinerário) do ponto de partida fica a parada. Realizamos um join entre o GPS com shape_ids identificados e a base de paradas do GTFS, mantendo apenas as paradas seguintes: paradas tais que a `dist_traveled_stop` é maior do que a distância projetada que ônibus viajou, ao todo, no passo anterior. Também calculamos a variável `stop_order`, que conta quantos pontos à frente a parada está: é 1 para o ponto seguinte, 2 para o depois dele, e assim por diante. Para reduzir o volume de dados e ter um horizonte de previsão mais próximo, restringimos a base a até 20 pontos à frente, com `stop_order` ≤ 20.

4.3. Cálculo dos tempos de chegada

Sabendo todas as paradas que cada ônibus tem pela frente, calculamos os tempos de chegada da seguinte forma: consideramos que um ônibus chegou ao ponto quando sua distância projetada viajada supera a `dist_traveled_stop`. Então registramos o horário em que observamos o ônibus tendo ultrapassado o ponto, e trouxemos esse valor para as observações anteriores. Assim, pudemos subtrair o horário anterior do horário de chegada, para obter o tempo de chegada.

4.4. Estrutura da base de validação

A tabela abaixo apresenta as variáveis presentes na base de validação construída por meio dos passos acima, que utilizamos para aplicar os modelos.

Nome	Detalhes	Código
Data	data	data
Serviço	código da linha	servico
Posição do ônibus	longitude e latitude	posicao_veiculo_geo
Velocidade instantânea	km/h	velocidade_instantanea
Velocidade média	km/h, média dos últimos 10mins	velocidade_estimada_10_min
Número do ponto	de 1, ..., N	stop_sequence
Distância ao ponto	m	dist_to_stop
Distância viajada	m	dist_traveled_shape
Quantos pontos à frente	de 1 a 20	stop_order
Tempo de chegada ao ponto	quantos mins até chegar	arrival_time
ID do Itinerário	código shape_id	shape_id
Hora	12h, 13h, ...	hora
Dia da semana	de 1 a 7	day_of_week

5. Modelos de previsão

O problema de previsão que enfrentamos é tratado com modelos de regressão: modelos que estimam uma variável y – no caso, o tempo de chegada – com base em diversas variáveis x – posição do ônibus, distância ao ponto, horário, velocidade, etc. Testamos algumas classes de modelos diferentes, descritos abaixo:

- Médias históricas: utilizamos como benchmark o modelo mais simples, que calcula a média de tempo que cada serviço levou para chegar a uma certa parada, quando ela estava algum número de pontos a frente, a depender do horário e do dia da semana.
- Regressão linear: modelos de média condicional ainda simples. Testamos diversas especificações, incluindo efeitos fixos de serviço-hora e coeficientes variáveis por serviço-hora. Optamos por não utilizar esses modelos na base completa, porque eles tinham poucos ganhos em relação às médias históricas.
- Floresta Aleatória: modelos de regressão não-linear, calculados a partir da média de muitas árvores de regressão.
- Redes Neurais: classe de modelos mais comum na literatura.

Todos os modelos foram treinados utilizando dados de 01/05/2024 até 21/05/2024, separados por linha. Então testamos o desempenho dos modelos na base de teste, de 22/05/2024 até 31/05/2024, e calculamos as seguintes medidas de erro de previsão, onde y é o tempo de chegada observado, e $\widehat{y}$ é o tempo de chegada previsto pelos modelos. Agregamos os erros de cada linha por médias ponderadas pelo número de observações. Para ser computacionalmente viável, usamos uma amostra aleatória de 10% das observações no período.

1. Raiz do erro quadrático médio: medida canônica para avaliação de modelos

$$\text{RMSE} = \frac{1}{N}\sqrt{\sum_i (y_i - \widehat{y}_i)^2}$$

2. Erro absoluto médio: menos sensível a outliers

$$\text{MAE} = \sum_i \frac{1}{N}|y_i - \widehat{y}_i|$$

3. Erro percentual absoluto médio: em termos relativos

$$\text{MAPE} = \frac{1}{N}\sum_i \frac{|y_i - \widehat{y}_i|}{y_i}$$

4. Desvio absoluto mediano: uma medida de estabilidade dos erros de previsão.

$$\text{MAPE} = \text{mediana}\Big(\big|(y_i - \widehat{y}) - \text{mediana}(y_i - \widehat{y})\big|\Big)$$

6. Resultados

As medidas de desempenho de previsões fora da amostra estão compiladas na tabela abaixo.

Modelo	RMSE	MAE	MAPE	MAD
Médias históricas	4,26	2,71	0,33	1,58
Random Forest	2,51	1,55	0,17	0,87
Neural Network	4,86	3,44	0,57	2,38

O modelo de melhor desempenho foi o Random Forest. Ele obteve um erro quadrático médio com raiz abaixo de 3 minutos, erro absoluto médio inferior a 2 minutos, e erro percentual de menos de 20%. O desvio absoluto mediano significa que, em 50% das observações, o erro desviou menos de 0,9 minutos de sua mediana. Isso indica que o modelo tem excelentes previsões na maior parte do tempo. O gráfico a seguir mostra como a qualidade das previsões varia com quantos pontos à frente a previsão é feita.

Vemos que o Random Forest tem sempre erros menores do que os demais modelos. Prevendo até 5 pontos à frente, o RMSE é inferior a 2 minutos; 10 pontos à frente, inferior a 2,5 minutos; e, mesmo 20 pontos à frente, é inferior a 4 minutos. O gráfico abaixo mostra os erros em termos percentuais. Omitimos o modelo de Rede Neural, que tinha erros muito elevados.

Novamente, o modelo de Random Forest tem desempenho melhor em todos os horizontes. Seu erro percentual é um pouco mais alto para distâncias próximas – já que cada minuto de erro representa uma porcentagem maior – mas fica sempre abaixo de 20% a partir de 5 pontos à frente. Mais de 10 pontos à frente, o erro cai abaixo de 10%.

7. Desafios

As maiores dificuldades para o projeto foram:

- Identificar as shape_ids corretas:
  - Como essa variável não está presente na base de dados original, temos que identificá-la pelas posições dos ônibus.

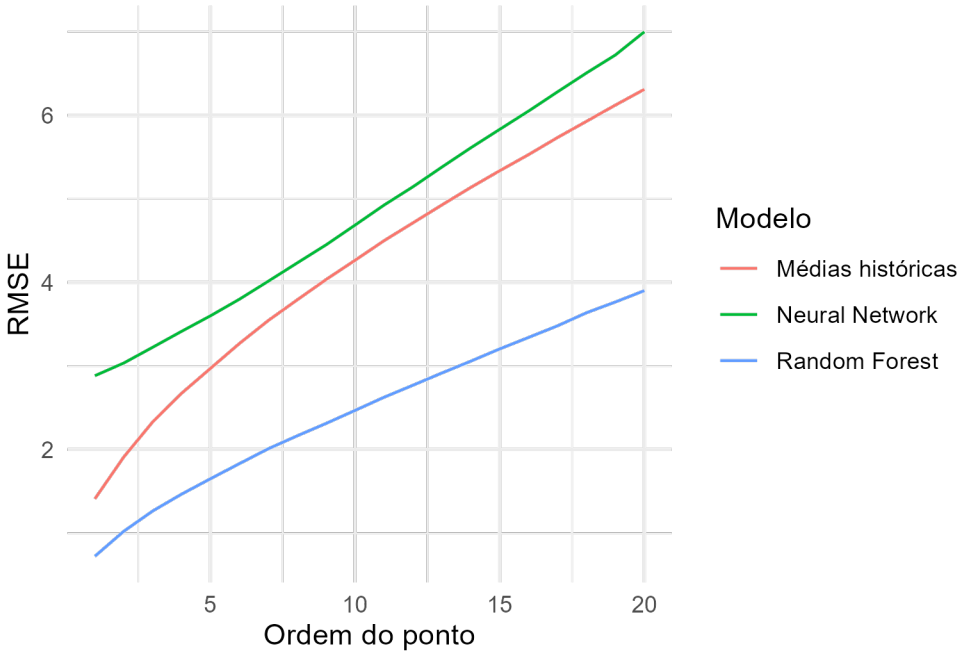

Figure 1: Raiz do erro quadrático médio

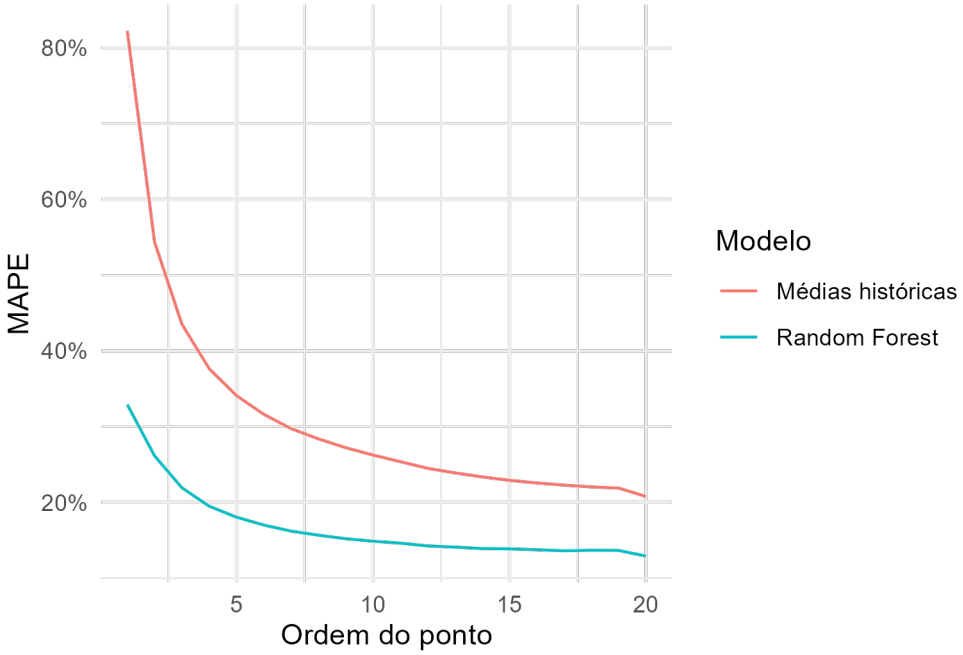

Figure 2: Erro percentual absoluto médio

  - Perdemos cerca de 25% de observações em que não conseguimos identificar a shape_id ou o tempo de chegada corretamente.
  - Há algumas anomalias no GTFS de `ShapesGeom`. Para a maioria dos serviços, a variável de `shape`, que contém o itinerário, é uma `linestring`. Mas em uma parcela delas, é em formato `multilinestring`, com várias `linestrings` agrupadas. Não é claro se é uma sequência de `linestrings` que formam o trajeto completo, ou se são caminhos alternativos. Por causa disso, perdemos uma parte das observações.

- Volume de dados:
  - A base de validação completa para o mês de maio tem 3 bilhões de observações (porque cada observação original tem até 20 pontos à frente, que se tornam observações separadas).
  - Mesmo trabalhando com uma amostra aleatória de 10%, os modelos de Random Forest e de Rede Neural demoraram pouco mais de 24h para serem treinados e terem o desempenho avaliado.
  - Como tivemos que trabalhar com os dados em nossos computadores pessoais, a restrição de tempo não permitiu que treinássemos uma maior variedade de modelos, e que aprimorássemos a Rede Neural.

8. Próximos passos

Para que as previsões sejam aplicadas na prática, essa manipulação de dados e os modelos devem ser implementados na arquitetura da Prefeitura, para se tornarem parte do site Mobilidade Rio. Trazemos abaixo algumas possíveis mudanças que facilitariam essa implementação.

- Incorporar a variável shape_id na base de validação: talvez seja possível que o dispositivo de GPS transmita diretamente essa informação. Isso agilizaria o tempo de execução e aumentaria a precisão das estimativas.
- O código de SQL de manipulação de dados, na prática, tendo os modelos treinados e agindo nos dados em tempo real, precisa apenas identificar os pontos seguintes, sem a parte de calcular tempos de chegada – já que, no mundo real, não os saberemos.
- Enquando o código ainda precisa identificar shape_ids, a inconsistência das `multilinestrings` descrita na seção de Desafios precisa ser investigada.
- Podem ser necessários recursos de computação em nuvem para treinar os modelos com grande volume de dados.
- Diferentes arquiteturas de Rede Neural e de Random Forest ainda podem ser testadas. Em particular, pode-se treinar os modelos alimentados apenas das variáveis originais do GPS – longitude, latitude, serviço, horário, e velocidade. Esses modelos podem ser treinados com custo computacional muito mais baixo. Além de serem mais simples, os dados em tempo real não precisarão passar por pré-processamento para que eles dêem uma previsão, o que deve facilitar a implementação.