Um Modelo para Análise Preditiva do Tempo de Deslocamento de Viagens

Felipe Soares Costa
Programa de Pós-Graduação em Modelagem e Matemática Computacional
CEFET-MG
felipesibh@gmail.com

Abstract

Estimar a duração de uma viagem é um problema comum em diversas situações. Saber quando um táxi estará disponível para receber uma nova viagem, ou incluir o deslocamento no cálculo do tempo de execução de ordens de serviço são exemplos comuns. Existem vários fatores que podem exercer influência no cálculo do tempo de deslocamento, tais como: clima, horário da viagem, dia da semana em que a viagem foi realizada. Esse artigo visa propor um modelo de estimativa do tempo de duração de viagens utilizando algoritmos de Machine Learning. Para isso serão analisados dados históricos de viagens de táxi realizadas em Nova York a partir do ano de 2009, bem como dados climáticos. Serão comparados diversos modelos e algoritmos a fim de encontrar aquele que produza as melhores estimativas.

1 Introdução

Tecnologias de gerenciamento de localização (como o GPS, por exemplo), exercem um papel fundamental em diversas áreas. Desde o gerenciamento de rotas de grandes frotas de veículos, até mesmo no cálculo de rota de uma viagem de transporte coletivo, são exemplos de situações onde essas tecnologias são utilizadas. Aplicativos de transporte como Uber e Cabify, são também exemplos de processos que dependem em sua essência de serviços de localização. Esses aplicativos crescem a cada dia em número de usuários e ganham cada vez mais espaço na economia.

Analisar a trajetória, o tempo total de deslocamento e a influência de fatores externos nesses eventos são tarefas de grande importância, e que permitem auxiliar na otimização de diversos processos. Estimar a duração total de uma viagem de táxi, por exemplo, permite que sistemas de despacho de serviços saibam com antecedência quando um veículo estará disponível para receber uma nova viagem. Estimativas de duração do tempo de viagens também são úteis no planejamento do trasporte coletivo urbano: calcular a duração das viagens auxilia no mapeamento de frotas a serem disponibilizadas para cada linha.

Encontrar um modelo que seja capaz de estimar a duração total de uma viagem com um certo nível de precisão é uma tarefa complexa, dada a interferência de inúmeros fatores externos. A aplicação de Algoritmos de Machine Learning nesse processo de estimativa, auxilia na detecção de padrões e permite aumentar a precisão consideravelmente. Existem diversos algortimos, modelos e técnicas que podem ser aplicadas na previsão do tempo de deslocamento de viagens, entretanto, encontrar o melhor modelo é uma tarefa complexa.

A proposta desse artigo consiste em analisar e comparar diversos modelos e algoritmos de Machine Learning com o objetivo de encontrar aquele que efetue as estimativas mais precisas. Serão aplicadas diversas métricas e técnicas a fim de comparar a qualidade dos modelos. Será desenvolvida uma classe utilizando a linguagem de programação Python a fim de auxiliar na análise e seleção do modelo.

Esse estudo está organizado da seguinte forma: a serção 2 apresenta outros trabalhos relacionados ao tema. A Seção 3 apresenta uma definição dos dados utilizados durante o estudo. A seção 4 explica a metodologia utilizada durante os experimentos. A seção 5 apresenta os resultados encontrados e a seção 6 as conclusões.

2 Trabalhos Relacionados

Exisem diversas pesquisas que se propõe a prever o tempo de deslocamento de viagens e comparar a perfomance de diferentes algoritmos nessas estimativas. Masiero L. P. et al [1] estudou o padrão de movimentação dos veículos ao longo do tempo, analisando dados históricos de trajetórias a fim de definir um modelo preditivo. Uma vez que durante o trajeto a velocidade média do veículo muda constantemente, existem porções da trajetória em que veículo esteve mais lento que em outras. O estudo realizado se propôs a segmentar essas porções de movimento dos veículos a fim de representar essa diferença de velocidade ao longo do tempo.

No estudo realizado por Cho Y. et al [2] diversos modelos de predição foram analisados tais como Historical Means, Current Time Predictor, Time varying Linear Regression, Time varying Linear Regression per Segments, Days in a Week, entre outros. Esse estudo identificou que o algoritmo *Days in a Week*, que considera basicamente um modelo diferente para cada dia da semana reduz significatiamente os erros nas estimativas. Entretanto, o TVLR (Time varying Linear Regression), é sugerido ao final do estudo devido á perfomance e simplicidade apresentadas.

Monreale A. et al [3] estudou os padrões de trajetória de objetos através de uma representação concisa das regiões visitadas por esse objeto ao longo do tempo. O estudo propôs um método chamado WhereNext que permite prever a próxima posição de um objeto com um certo nível de acurácia.

Wu, C. et al [4] estudou a utilização do algoritmo SVR (Support Vector Regression), para estimativa de tempo de viagens. O estudo comparou o SVR com outros modelos e concluiu que o SVR conseguiu efetuar estimativas mais precisas.

3 Explorando os Dados

Nesse estudo foram utilizados dois datasets disponbilizados através da plataforma Kaggle [5]. O primeiro dataset contém os dados de viagens de táxi realizadas em Nova York a partir do ano de 2009, disponibilizados pelo NYC Taxi and Limousine Commission. Cada linha do arquivo representa uma viagem de táxi realizada, e cada viagem está associada a uma série de características: um identificador numérico único para cada viagem, data e hora em que a viagem iniciou, data e hora do término da viagem, número de pasageiros no veículo (incluindo o motorista), coordenadas de início da viagem (latitude e longitude), coordenadas de término da viagem (latitude e longitude) e a duração da viagem em segundos. A Figura 1 apresenta algumas linhas desse Dataset.

	pickup_datetime	dropoff_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
0	2016-03-14 17:24:55	2016-03-14 17:32:30	-73.982155	40.767937	-73.964630	40.765602
1	2016-06-12 00:43:35	2016-06-12 00:54:38	-73.980415	40.738564	-73.999481	40.731152
2	2016-01-19 11:35:24	2016-01-19 12:10:48	-73.979027	40.763939	-74.005333	40.710087
3	2016-04-06 19:32:31	2016-04-06 19:39:40	-74.010040	40.719971	-74.012268	40.706718
4	2016-03-26 13:30:55	2016-03-26 13:38:10	-73.973053	40.793209	-73.972923	40.782520

Figure 1: Dataset com dados de viagens.

O segundo dataset contém informações climáticas da cidade de Nova York no período de dezembro de 2015 à dezembro de 2016 coletados através da Wundergrounds API [6]. Cada linha corresponde a dados climáticos coletados em um momento do dia sendo: a data e hora do evento, a temperatura em graus Celsius, a temperatura em escala Fahrenheit, o ponto de condensação em Celsius e em

Fahrenheit, velocidade do vento, a umidade em porcentagem, a visiblidade, a precipitação. Possui também colunas binárias com os seguintes indicadores: chuva, neve, tornado e trovão. Na Figura 2 é possível observar algumas linhas desse Dataset.

	timestamp	temp	windspeed	humidity	precip	pressure	conditions	dailyprecip	dailysnow	fog	rain	snow
0	2016-01-01 00:51:00	42.1	4.6	51.0	0.0	30.06	Overcast	0.00	0.00	0	0	0
1	2016-01-01 01:51:00	41.0	3.5	53.0	0.0	30.06	Overcast	0.00	0.00	0	0	0
2	2016-01-01 02:51:00	41.0	4.6	55.0	0.0	30.06	Overcast	0.00	0.00	0	0	0
3	2016-01-01 03:51:00	41.0	9.2	55.0	0.0	30.06	Overcast	0.00	0.00	0	0	0
4	2016-01-01 04:51:00	39.9	10.4	58.0	0.0	30.04	Overcast	0.00	0.00	0	0	0

Figure 2: Dataset com dados de clima.

Uma vez que o dataset de clima foi coletado em um período menor do que o dataset de viagens, foi necessário primeiramente selecionar dados do dataset de viagens correspondentes apenas ao período coletado no dataset de clima. A data de início da viagem foi convertida para um valor inteiro a fim de utilizá-la no conjunto de features. A distância entre as coordendas de início e de término da viagem foi calculada utilizando a fórmula Haversine. A fim de garantir uma mesma escala de valores, foi calculado o logaritmos de alguns dados, tais como: latitude e longitude de início e término, data de início e duração da viagem. Dos dados de clima foram extraídas as seguintes informações: temperatura, velocidade do vento, chuva e neve.

Para representar os dias da semana será utilizado um vetor binário. Para cada dia da semana uma coluna foi criada no dataset. Cada uma dessa colunas foi preenchida com o valor 1, caso a viagem tenha sido iniciada nesse dia da semana, e com o valor 0 caso tenha sido iniciada em qualquer outro dia.

Ao ordenar o dataset final de forma decrescente pela duração da viagens um ponto interessante foi constatado: as viagens mais longas do dataset, ou seja, as viagens com maior tempo de duração, possuíam uma distância muito baixa entre as coordenadas. Claramente trata-se de algum erro nos dados, uma vez que a distância é um fator que influencia diretamente na duração de uma viagem: longas distâncias produzem uma tempo de duração maior. Portanto foi necessário remover essas linhas a fim de que não influenciassem negativamente nesse estudo.

4 Metodologia

Para fins de representação matemática as características do modelo serão representadas da seguinte forma: p irá representar a data de início da viagem, l_i a latitude inicial, l_f latitude final, g_i a longitude inicial,g_f a longitude final. A distância entre os dois serviços será representada por d, a temperatura será representada por t, a velocidade do vento por v, a ocorrência de chuva por r e a neve por s. O tempo total da viagem será representado por c e o vetor binário com os dias da semana será representado por w. Serão definidos três conjuntos de fetarues para serem comparadas e aplicadas em cada modelo sendo elas:

1. Grupo01: Composto apenas pelas características do dataset principal, sendo elas:
   (a) TimeStamp do Início da Viagem
   (b) Latitude Inicial
   (c) Longitude Inicial
   (d) Latitude Final
   (e) Longitude Final
   (f) Distância entre os dois pontos

   Esse modelo pode ser representado pela seguinte equação:

$$logt = \beta_0 + \beta_1 logl\_i + \beta_2 logl\_f + \beta_3 logg\_i + \beta_4 + logg\_f + \beta_5 logd \quad (1)$$

2. Grupo02: Componsto pelas características descritas no Grupo01 e acrescidas de dados climáticos descritos abaixo:
   (a) Temperatura
   (b) Velocidade do Vento
   (c) Ocorrência de Chuva
   (d) Ocorrência de Neve

   Esse modelo pode ser representado pela seguinte equação:

$$logt = \beta_0 + \beta_1 logl\_i + \beta_2 logl\_f + \beta_3 logg\_i + \beta_4 + logg\_f + \beta_5 logd + \beta_6 logc + \beta_7 v + \beta_8 r + \beta_9 s \quad (2)$$

3. Grupo03: Composto pelas características descritas nos Grupo01 e Grupo02, acrescidos de um vetor binário para os dias da semana. O modelo desse grupo pode ser representado pela seguinte equação:

$$logt = \beta_0 + \beta_1 logl\_i + \beta_2 logl\_f + \beta_3 logg\_i + \beta_4 + logg\_f + \beta_5 logd + \beta_6 logc + \beta_7 v + \beta_8 r + \beta_9 s + \beta_{10} w \quad (3)$$

4. Grupo04: Composto pelas características descritas nos Grupos01 acrescidos de um vetor binário para os dias da semana. É possível utilizar a seguinte equação pra representar o modelo desse grupo:

$$logt = \beta_0 + \beta_1 logl\_i + \beta_2 logl\_f + \beta_3 logg\_i + \beta_4 + logg\_f + \beta_5 logd + \beta_{10} w \quad (4)$$

Fora escolhidos também dois grupos de algoritmos que serão comparados. O primeiro grupo conta com um conjunto de algoritmos de Regressão implementados pela biblioteca *Scikit-Learn* [7], sendo eles:

1. Linear Regression: Modelo que utiliza o OLS (Ordinary Least Squares).
2. Ridge: Esse algoritmo utiliza o Linear Least Square como função de erro e a norma L2 para regularização.
3. Lasso: Esse algoritmo utiliza a norma L1 para regularização.
4. ElasticNet: Esse algoritmo combina as normas L1 e L2 para regularização.
5. LassoLars: Utiliza o Least Angle Regression pra treinar o modelo e a norma L1 para regularização.
6. BayesianRidge: Esse modelo é similar ao Ridge. Comparado ao OLS (Ordinary Least Squares), os pesos dos coeficientes são ligeiramente deslocados em direção a zero o que os estabiliza.
7. SVR (Support Vector Regression): Uma extensão do classificador SVC (Support Vector Classification) para resolver problemas de Regressão.

O segundo grupo a ser comparado conta com um conjunto de *Ensembles*. Segundo Moreira, J. et al [8], *Ensembles* se caracterizam por métodos que combinam diferentes tipos de modelo para aumentar a precisão das estimativas. Os seguintes *Ensembles* foram escolhidos para o estudo:

1. AdaBoostRegressor: Esse algoritmo treina o modelo no dataset original e depois treina cópias adicionais do modelo no mesmo dataset, porém com os pesos dos parâmetros ajustados de acordo com o erro atual. A implementação utilizada faz parte do *Sikit-Learn*.
2. BaggingRegressor: Esse algoritmo treina vários modelos em partes individais do dataset, e depois sumariza essas estimativas, através de média ou votação.
3. GradientBoostingRegressor: Esse algoritmo utiliza o Gradiente Descendente para minimizar uma função de erro. A cada iteração modelos diferentes são utilizados para minimizar essa função.
4. RandomForestRegressor: Esse algoritmo treina vários modelos em diferentes amostras do dataset para melhorar a precisão e controlar o overfitting.

Cada um dos três conjuntos de parâmetros será utilizado para treinar cada um dos modelos descritos acima utilizando a técnica de K-Fold Cross-Validation. Segundo Zheng, A. [10], nesse procedimento o dataset de treino é dividio em k folds. Um dos folds é separado para teste e o modelo é treinado nos folds restantes. Esse procedimento é repetido para todos os folds. O score final da validação é dado pela média das perfomances medidas em cada uma das iterações. Para os experimentos descritos nesse artigo serão utilizados 10 k-folds.

Será calculado então o RMSE (Root Mean Squared Error) para cada modelo. O RMSE pode ser descrito como a distância Euclidiana entre os valores reais e os valores previstos pelo modelo [9]. Serão calculados também o Desvio Padrão dos resíduos e o Score R2 á cada iteração. Para Schneider, A. et al [9], o coeficiente R2 é uma métrica que indica o quanto o modelo é capaz de descrever os dados observados. O valor do coeficiente varia de 0 a 1, onde 0 indica que o modelo não consegue explicar os dados observados e 1 indica que o modelo consegue explicar perfeitamente esses dados. Ao final, será selecionado o modelo com o menor RMSE, maior R2 Score e menor Desvio Padrão.

5 Resultados

Para automatizar a aplicação dos modelos foi criada uma bibiloteca em Python chamada *ChooseTheModel*. Essa biblioteca possui uma classe chamada *Chooser* responsável por avaliar uma lista de modelos e uma lista de conjunto de parâmetros, sumarizar os dados e retornar dados e estatísticas que auxiliem na avaliação e seleção de modelos. Essa classe depende das bibliotecas *Scikit-Learn* [7] para execução dos procedimentos de Cross-Validation e para calcular as métricas de avaliação. Utiliza também a biblioteca *Pandas* [11] para sumarização e manipulação dos dados. O método *choose* da classe percorre os modelos definidos, e executa um procedimento de Cross-Validation para cada um dos grupos de parâmetros. A classe retorna uma tabela contendo o RMSE, o Score R2 e o Desvio Padrão para cada combinação de Modelo e Parâmetos, e a data e hora de início e término de execução do procedimento de Cross-Validation. A Figura 3 exemplifica a utilização dessa classe.

```
models = []
models.append((AdaBoostRegressor()))
models.append((BaggingRegressor()))
models.append((GradientBoostingRegressor()))
models.append((RandomForestRegressor()))

chooser = chooseTheModel.Chooser(models,x,y)

x name,name,df = chooser.choose()
```

Figure 3: Classe ChooseTheModel.

Aplicando a comparação no grupo de modelos com algoritmos simples, o algoritmo de melhor perfomance foi o SVR (Support Vector Regression) em conjunto com o primeiro grupo de parâmetros (Grupo01), ou seja o algoritmo teve uma perfomance melhor utilizando o modelo mais simples. A Figura 4 apresenta um exemplo do DataFrame retornado pela classe *ChooseTheModel*.

	Grupo	R2 Score	RMSE	Standard Deviation
SVR	GRUPO01	0.980465	0.224623	0.008278
Lasso	GRUPO01	0.205495	1.432510	0.805945
ElasticNet	GRUPO01	0.201814	1.435825	0.884976
Ridge	GRUPO01	0.194065	1.442776	1.030597
LinearRegression	GRUPO01	0.175307	1.459463	1.172953

Figure 4: Comparação entre modelo simples.

Esse modelo obteve um RSME muito baixo (0.224623), o que indica que os erros nas estimativas são baixos. O score R2 é bem próximo de 1 (0.980465), o que indica uma boa capacidade preditiva. É importante ressaltar também que o SVR foi o algoritmo que levou mais tempo pra executar as estimativas. A Figura 5 apresenta uma comparação entre os valores previstos e reais onde é possível avaliar a proximidade dos valores. Os demais modelos tiveram perfomances muito ruins com altos valores de RMSE e baixos valores no score R2, o que indica uma distância muito grande das estimativas em relação aos valores reais e uma capacidade preditiva muito ruim.

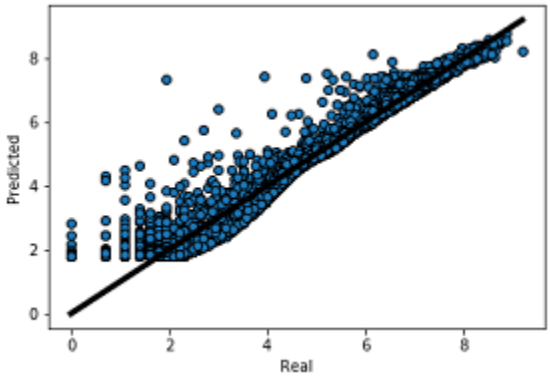

Figure 5: SVR - Previsto X Real.

A Figura 6 apresenta uma comparação entre as avaliações de cada algoritmo do grupo composto pelos *Ensembles*. Analisando esse grupo, o algoritmo que obetve a melhor perfomance foi GradientBoostingRegressor, em conjunto com o terceiro grupo de parâmetros (Grupo03). Esse modelo obteve um RMSE ainda mais baixo que o SVR (0.206476). O score R2 também foi maior utilizando o GradientBoostingRegressor: 0.983480. Diferente dos algoritmos do primeiro grupo, nesse grupo o restante dos algoritmos tiveram também uma ótima perfomance. O pior algoritmo desse grupo foi o AdaBoostRegressor e mesmo assim ainda obteve uma perfomance muito maior que os demais algoritmos do primeiro grupo.

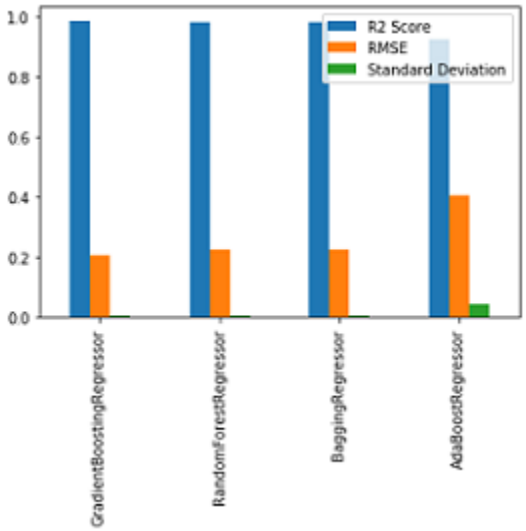

Figure 6: Ensembles - Comparação.

A Figura 7 mostra o gráfico de comparação entre os valores previstos e os valores reais estimados pelo algoritmos GradientBoostingRegressor.

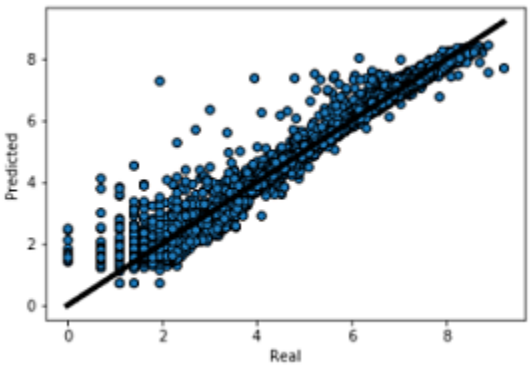

Figure 7: GradientBoostingRegressor - Previsto X Real.

6 Conclusão

Com base nos resultados dos experimentos foi possível identificar que a combinação que resulta no modelo com maior capacidade preditiva é composta pelo *ensemble* GradientBoostingRegressor e o pelo terceiro grupo de parâmetros (Grupo03), que é o grupo composto pelos dados do dataset principal, pelos dados do dataset de clima e pelo vetor binário com os dias da semana.

O SVR (Support Vector Regression) obteve também uma perfomance muito superior aos demais algoritmos do primeiro grupo e se aproximou bastante da perfomance obtida pelo grupo composto por *Ensembles*. No geral todos os *Ensembles* obtiveram uma perfomance aceitável.

Para trabalhos futuros espera-se aprofundar no estudo de seleção e avaliação de modelo, e evoluir a implmentação da classe *ChooseTheModel* adicionando suporte á algoritmos de classificação e mas métricas e possibilidades de análise e comparação.

Referências

[1] Masiero, L. P. & Casanova, M. A. & Carvalho, M. T. (2011) *Travel Time Prediction using Machine Learning* Rio de Janeiro, Brasil.

[2] Cho, Y. & Kwac, J. (2007) *A Travel Time Prediciton with Machine Learning.* Stanford, CA: Stanford University.

[3] Monreale, A. & Pinelli, F. & Trasarti R. (2009) *WhereNext: a Location Predictor on Trajectory Pattern Mining* Paris, France.

[4] Wu, C. & Wei, C. & Su D. & Chang, M. & Ho, J. (2003) Travel Time Prediction with Support Vector Regression *IEEE* **0-7803-8125-4/03.**

[5] Kaggle, *New York City Taxi Trip Duration*. Disponível em <https://www.kaggle.com/c/nyc-taxi-trip-duration>. Acesso em 7 de setembro de 2017.

[6] Wunderground API, *Phrase Glossary*. Disponível em <https://www.wunderground.com/weather/api/d/docs?d=resources/phrase-glossary>. Acesso em 7 de setembro de 2017.

[7] Scikit-Learn: Machine Learning in Python. Disponível em <http://scikit-learn.org/stable/>. Acesso em 26 de novembro de 2017.

[8] Moreira J. M. & Soares, C. & Jorge, A. M. & Sousa, J. F. (2007) *Ensemble Approaches for Regression: a Surveyg* Porto, Portugal.

[9] Zheng, A. (2015) *Evaluating Machine Learning Models: A Beginners Guide to Key Concepts and Pitfalls.* Sebastopol, CA: OReilly Media, Inc.

[10] Schneider, A. & Hommel, G. & Blettner, M (2017) *Linear Regression Analysis.* Germany, Mainz: Deutsches rzteblatt International.

[11] Pandas, *Python Data Analysis Library*. Disponível em <https://pandas.pydata.org/>.