PREDIÇÃO DE FALHAS NA LOGÍSTICA DE ENTREGAS EM E-COMMERCE

Prof. MSc. Delermando Branquinho Filho

Gerente de Projetos em Desenvolvimento de Sistemas e Infraestrutura Tecnológica. Especialista em gestão de pessoas, Sistemas Inteligentes como Cientista de Dados. Especialização em Data Science - Johns Hopkins University. Mestrado em Engenharia Elétrica e Engenharia da Computação – UFG. MBA - Desenvolvimento de Executivos em Gestão e Economia Empresarial – UFRJ. Física – Pontifícia Universidade Católica de Goiás – PUC Goiás.

delermando@gmail.com

RESUMO

Com a Pandemia do COVID-19 as empresas de logística, relacionadas às entregas do comércio eletrônico, em 2020 tiveram um aumento significativo em seus pedidos de transporte. Um dos fatores fundamentais para essas empresas é a entrega no prazo. Determinar previamente quando uma entrega vai atrasar pode significar uma economia de muito dinheiro e esforço para essas empresas. Neste artigo fornecemos recursos e métodos em Aprendizado de Máquina para fazer predições individuais de cada volume envolvido em uma entrega com a sua probabilidade de atrasar ou não. Faremos uma apresentação teórica e logo após descrevemos os métodos e modelos usados para ilustrar a solução, bem como os algoritmos e a avaliação do seu desempenho. Ao final exibiremos os resultados obtidos na predição de falhas das entregas.

Palavras-Chave

Anomalia; Predição; Logística; Comércio Eletrônico.

ABSTRACT

With the COVID-19 Pandemic, logistics companies, related to e-commerce deliveries in 2020, had a significant increase in their transport orders. One of the fundamental factors for these companies is on-time delivery. Determining in advance when a delivery will be delayed can mean a lot of money and effort for these companies. In this article we provide Machine Learning resources and methods for making individual predictions of each volume involved in a delivery with its likelihood of being late or not. We will make a theoretical presentation and soon after we describe the methods and models used to illustrate the solution, as well as the algorithms and the evaluation of its performance. At the end, we will show the results obtained in the prediction of delivery failures.

Keywords

Anomaly; Prediction; Logistics; E-commerce. r.

INTRODUÇÃO

Em um ambiente com mudanças ininterruptas, o controle do estado de um pacote (unidade de entrega) na malha logística parece algo inviável (PAURA 2016, p.73). Queremos encontrar uma solução, baseada no histórico de dados, criando um ambiente que mostre o estado do pacote e sua probabilidade de prejudicar uma ou mais métricas de desempenho. Um problema que enfrentamos em logística é quando lidamos com métricas de qualidade, como por exemplo, quando um pacote está fora do prazo, onde não somos capazes de determinar qual parte do processo foi a principal causa do atraso de cada um dos pacotes. Por exemplo, um pacote pode ser rotulado como 'Não despachado' como a situação igual a fora do prazo. A razão real pode ser devido a um erro operacional em XD (acrônimo do Inglês Cross Docking que significa Sistema de Distribuição), mas como conseguimos chegar à agência antes do prazo final, então ele é rotulado como atrasado (APTE; VISWANATHAN 2000).

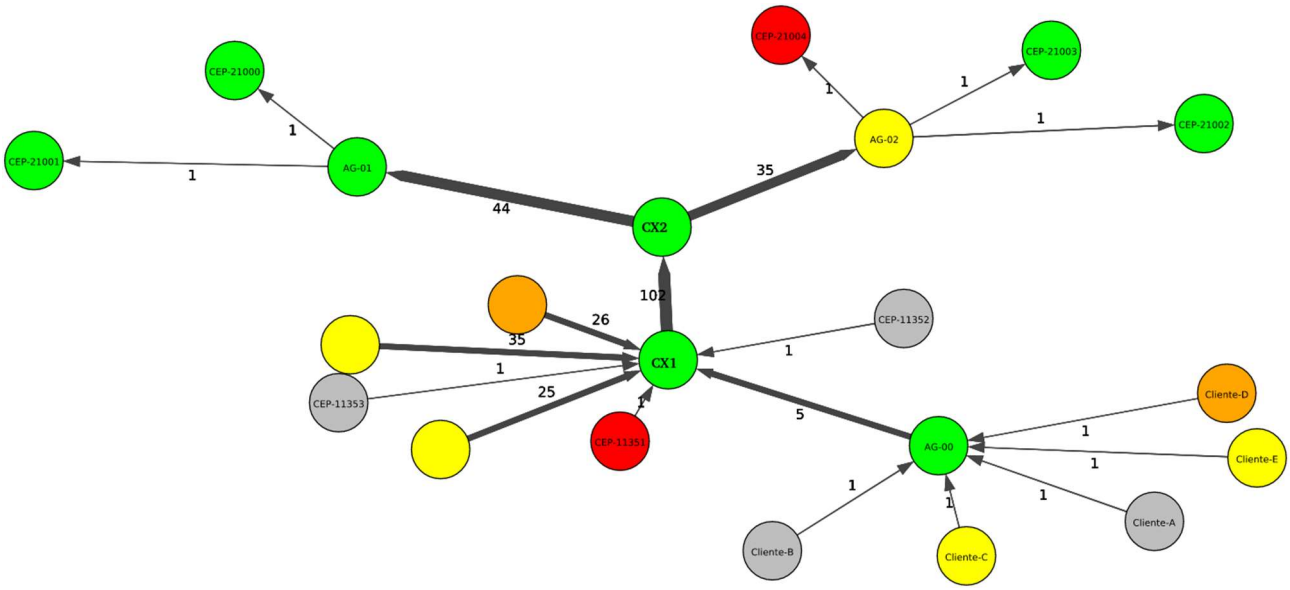
Portanto, construir um modelo que possa realmente entender o comportamento do pacote em sua jornada, prever como ele pode afetar nossas métricas de desempenho é um grande passo para nossa cultura baseada em dados. A solução em uma visão simplificada é emitir alertas para a operação quando a probabilidade de atraso atingir limites inaceitáveis.

Antes de avançarmos é preciso estabelecer algumas definições do mecanismo de roteamento. Um Grafo de contato de múltiplos destinos (do Inglês *multi-destination contact graph routing* - MD-CGR) em um grafo direcionado e ponderado $G=(V, E, T, c)$, onde:

> V: O conjunto de vértices (nós) na rede
>
> E: os contatos na rede que representam a capacidade de envio entre os nós ao longo do tempo. A notação $e(i, j)$ representa a aresta entre os nós $i,j \in V$.
>
> T: Um subconjunto de vértices na rede que deve ser incluído em um caminho ideal. Isso inclui, no mínimo, a origem e o destino do pacote.
>
> c: Um mapeamento de função de custo $E \Rightarrow R$, associando um custo de um número real para atravessar um contato particular na rede. $c(i, j)$ representa o custo de uma aresta particular entre os nós $i,j \in V$
>
> (BIRRANE; BURLEIGH; KASCH, 2012, p.110, tradução nossa).

A Figura 1 mostra o nosso interesse na roteirização, onde os pontos de contato podem receber cores para determinar se há uma entrega atrasada ou não. Se os pacotes pertencentes aquele ponto de contato está sem atrasos significativos ele recebe uma cor verde, caso contrário amarela ou vermelha, dependendo da proporção ou limites aceitáveis que podem ser configurados pelos operadores.

Figura 1. Logística de distribuição e roteirização.



Fonte: próprio autor

O importante aqui é entender que um pacote pode sair de um ponto qualquer (cep_origin) passar por vários pontos intermediários até chegar ao seu destino final (cep_destination). Não estamos preocupados com os pacotes que foram devolvidos ou que não foram entregues. Apenas pacotes entregues, com ou sem atraso, foram submetidos ao treinamento dos modelos.

Queremos ilustrar os múltiplos pontos de contato de um determinado pacote na rede em sua rota. A partir desse entendimento podemos avançar para o contexto e escopo desse experimento.

CONTEXTO E ESCOPO

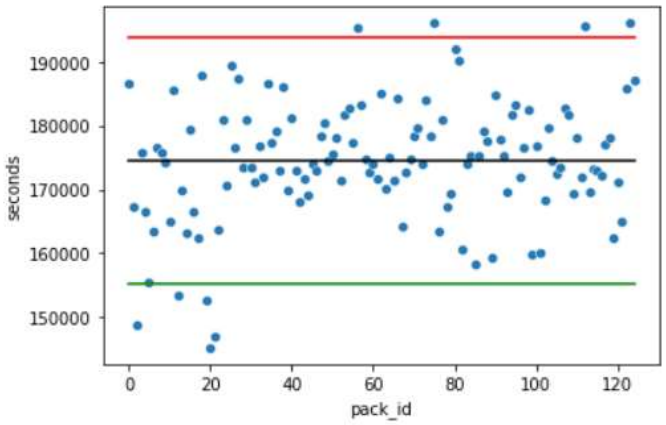
O principal objetivo desse artigo é demonstrar **como criar um ambiente preditivo para prevenir atrasos nas entregas em e-Commerce?** Dentre os objetivos específicos, podemos destacar:

- Quais métricas provavelmente serão ofendidas;
- Rastreabilidade preventiva dos pacotes;
- Alarmes para os pacotes com alta probabilidade de atraso;
- Sugerir dados preventivos para retroalimentação de outros sistemas: Capacidade, Transporte, Otimização e muito mais.

Não estamos interessados aqui no mapeamento de processos em XD (Cross Docking), desenvolvimento e otimização das rotas ou na capacidade de ramificação da malha logística. Esta seção explica o design, as transformações dos dados brutos e suas consequências conforme apropriado. Isso deve, portanto, ser o mais curto possível no menor tempo possível. Vamos pensar no caminho das embalagens, desde a integração até o fim do ciclo de vida, ou seja, até a entrega.

O primeiro passo é analisar o tempo gasto para cada evento sistemático. Vamos pensar estatisticamente sobre a mediana para este tempo, vamos adicionar o desvio padrão a essa mediana. Agora temos um gráfico de controle da seguinte forma:

Figura 2. Distribuição dos pacotes e o tempo gasto desde a coleta até a entrega.



Fonte: próprio autor

A linha preta central é a mediana de todos os dados entre uma faixa de tempo, a linha vermelha é o limite superior com duas vezes desvio padrão. A linha verde é o limite inferior com duas vezes desvio padrão.

Os pontos que estão fora dos limites superior e inferior são pontos anômalos. Pontos acima do limite superior significam que o pacote é uma anomalia em relação aos outros pacotes entregues na mesma origem e no mesmo destino, por outro lado, uma entrega tardia. Os pacotes abaixo do limite inferior também são considerados anomalias sistêmicas porque foram entregues em um tempo extremamente curto. Eles não foram registrados corretamente ou há valores indevidos em seu rastreamento.

O gráfico da Figura-1 é apenas ilustrativo e estatisticamente inválido, pois a distribuição não é gaussiana (como veremos mais tarde) e as medidas de tendência central e dispersão não são válidas. A ilustração é meramente didática para entender as anomalias resultantes do processo de coleta até a entrega.

A partir desses dados históricos tentamos prever quando um novo pacote irá atrasar sua entrega. A Tabela 1 mostra o modelo de dados que foi anonimizado preservar dados sensíveis da empresa que forneceu a base para esse experimento. O modelo de dados pode ser descrito da seguinte forma:

Tabela 1. Modelo de dados original.

ID	package_id	cep_origin	cep_destination	points	elapsed_time	outcome
1	000001	12345123	22345130	18	12351	1
2	000002	12345124	22345101	15	25354	0
3	000003	12345231	22345102	22	14233	1
4	000004	12345100	22345103	12	16000	1

Fonte: próprio autor

A primeira coluna é a identificação da observação começando em 1 até *n*. Cada pacote tem uma identificação única do sistema que chamaremos aqui de *package_id*. Todo pacote integrado ainda tem pontos de Partida e Chegada que foram caracterizados com os seus respectivos CEPs (Código de Endereçamento Posta – Correios) aqui exemplificados com números aleatórios para fins de demonstração. A coluna de *points* é a contagem dos pontos de contato percorridos pelo pacote, ou seja, quantas vezes esse pacote foi registrado dentro do sistema computacional da empresa de logística. O tempo total gasto desde o início até a entrega do pacote em segundos (*timetamp*). A última coluna da tabela é o rótulo da observação que queremos classificar. Presumimos que o tempo não é uma distribuição normal. Validamos essa hipótese realizando um simples teste de normalidade. Para a nossa amostra temos:

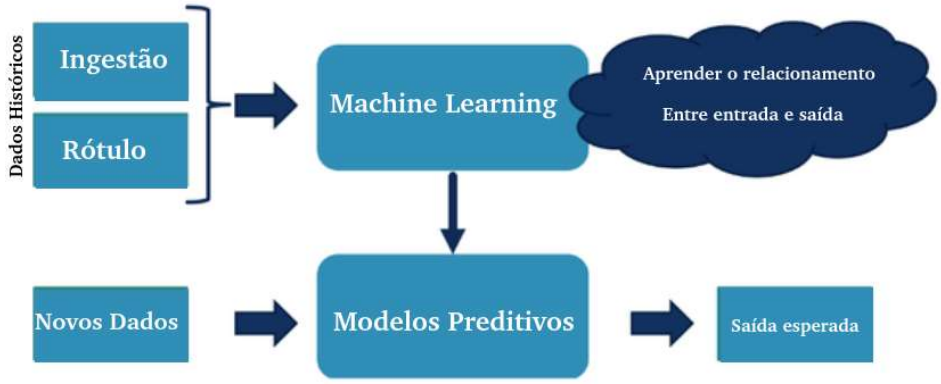
ShapiroResult (estatística=0,9772894978523254, pvalue=0,03340694680809975)

A chance de rejeitar a hipótese nula quando é verdadeira é próxima de 5% independentemente do tamanho da amostra.

EXTRAÇÃO E PREPARAÇÃO DOS DADOS

Através dos dados históricos extraídos da base de dados da Classificação de Registros e do respectivo Cep (Cep), será possível criar uma estrutura de dados que explique os tempos e interações dos pacotes ao longo da rota de entrega.

Figura 3. Fluxo de dados (*Pipeline*).



Fonte: próprio autor

A ingestão de dados através do histórico armazenado em um banco de dados relacional, fonte escolhida para este experimento, fornece uma abordagem previamente estruturada com todos os eventos ao longo da vida dos pacotes. Dessa forma, será possível enviar esses dados para o treinamento de vários algoritmos, como podemos ver nas próximas seções. Os dados originais extraídos serão modificados para uma única linha contendo as características descritas na Tabela 1.

O nosso alvo está na última coluna rotulado como *outcome*. Se o pacote foi entregue atrasado seu valor será preenchido como zero, caso contrário o valor será um. O número de pontos de contato e o tempo gasto são calculados na fase de transformação dos dados.

Transformação dos Dados

Após as transformações o nosso Conjunto de Dados ficou com valores de duas colunas trucados: a origem e o destino. Uma vez que os endereços postais muito próximos têm tempos de entrega semelhantes, como por exemplo, os mesmos percentis dos prazos de entrega e sua agregação não mudou significativamente (no teste de hipóteses) esses valores. A tabela a seguir mostra a nova configuração de dados.

Tabela 2. Modelo de dados transformado (CEP).

ID	package_id	cep_origin	cep_destination	points	elapsed_time	outcome
1	000001	12345	22345	18	12351	1
2	000002	12345	22345	15	25354	0
3	000003	12345	22345	22	14233	1
4	000004	12345	22345	12	16000	1

Fonte: próprio autor

Nosso resultado, bem como as características para a transformação do modelo de dados, como a soma dos pontos de contato e o tempo gasto, foram baseados no artigo Monitoramento de Processo Preditivo Orientado para resultados: Revisão e Benchmark. A redução do CEP de oito dígitos para apenas quatro, é decorrente de uma maior aproximação e volumetria de origens e destinos. Observem que há, nessa tabela exemplo, as mesmas origens e os mesmos destinos quando truncamos esses valores e três pacotes chegaram em tempo, mas um foi entregue com atraso. Isso Pode dizer muito sobre os processos intermediários (TEINEMAA et al, 2019).

Forma dos conjuntos de dados

Neste experimento, utilizamos um fragmento do banco de dados que totaliza 7.381.957 observações separadas em 7.015.017 entregues no prazo e 365.452 entregues tardiamente. Com classes tão desequilibrados sugerimos a leitura da seção de hiper parâmetros dos modelos cuidadosamente para uma melhor compreensão da solução encontrada.

O ID do pacote foi removido do modelo. Os códigos postais foram convertidos em características categóricas, deixando os pontos de contato e o tempo decorrido como variáveis numéricas do tipo inteiras.

Tabela 3: Base de Treinamento/Teste e Validação:

	Classe	Observações
Treino e Teste	1	6.839.642
	0	356.316
Validação	1	175.375
	0	9.136

Fonte: próprio autor

Reservamos 67,5% de toda a amostra para treinamento, 27,5% para testes e 5% para validação. A validação são observações que o algoritmo nunca viu. "Se quiser medir o desempenho como uma maneira de selecionar uma boa hipótese, então deve-se dividir os dados disponíveis em um conjunto de treinamento, testes e um conjunto de validação" (RUSSELL, 2004; KUHN, 2013). Dividimos os dados assim:

- Conjunto de dados de treinamento: A amostra de dados usados para se adequar ao modelo.
- Conjunto de dados do teste: A amostra de dados usados para fornecer uma avaliação imparcial de um modelo encaixado no conjunto de dados de treinamento enquanto ajusta hiper parâmetros de modelo. A avaliação torna-se mais tendenciosa à medida que a habilidade no conjunto de dados de validação é incorporada à configuração do modelo.
- Conjunto de dados de validação: A amostra de dados utilizada para fornecer uma avaliação imparcial de um modelo final encaixado no conjunto de dados de treinamento.

RECURSOS E DESENHO DO EXEPRIMENTO

Para este experimento, usamos o pacote H2O (https://www.h2o.ai). H2O é uma iniciativa de físicos, matemáticos, cientistas da computação. Pesquisadores acadêmicos e cientistas de dados industriais colaboram com essa iniciativa. Stephen Boyd, Trevor Hastie e Rob Tibshirani da Universidade de Stanford aconselham a equipe H2O para construir algoritmos escaláveis de aprendizagem de máquina. O H2O tornou-se um fenômeno de crescimento entre a comunidade de dados usando R, Python, Hadoop e Spark. Algoritmos avançados, como Deep Learning, Boosting e Bagging Ensembles estão disponíveis para aplicativos mais inteligentes através de APIs, como pode ser visto no experimento nas seções a seguir.

H2O AutoML

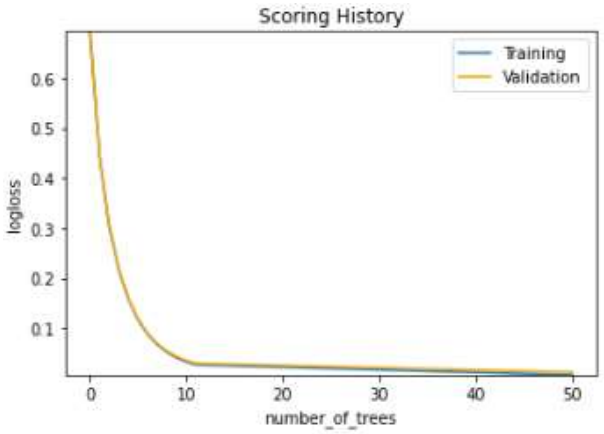
AutoML (Auto Machine Learning) é uma função em H2O que automatiza o processo de construção de um grande número de modelos, com o objetivo de encontrar o "melhor" modelo sem qualquer conhecimento ou esforço prévio dos dados. A versão atual do AutoML usada nesse experimento (H2O 3.16) treina e valida um padrão para Random Forest, um gride para o Gradient Boosting Machines (GBMs), um gride aleatório para Deep Neural Nets, uma grade fixa de GLMs, e depois treina dois modelos em Stacked Ensemble no final. Um conjunto contém todos os modelos (otimizados para o desempenho do modelo), e o segundo conjunto contém apenas o modelo de melhor desempenho de cada classe/família de algoritmos (WOLPERT 1992).

Hiperparâmetros

Esta Seção fornece algumas descrições de parâmetros especificados nos algoritmos. Além disso, cada parâmetro também inclui se o parâmetro é um hiper-parâmetro (usado na pesquisa de grade também). Os principais parâmetros que tiveram seus valores padrão alterados foram:

- **categorical_encoding**: Especifique um dos seguintes esquemas de codificação para lidar com recursos categóricos: sort_by_response Reordena os níveis pela resposta média (por exemplo, o nível com menor resposta -> 0, o nível com segunda menor resposta -> 1, etc.). Isso é útil no GBM/DRF, por exemplo, quando você tem mais níveis do que nbins_cats, e onde o nível superior se divide agora tem a chance de separar os dados comuns à divisão. Observe que isso requer uma coluna de resposta especificada;
- **balance_classes**: Supersample as classes minoritárias para equilibrar a distribuição da classe;
- **fold_assignment**: Especifique o esquema de atribuição de dobra de validação cruzada;
- **keep_cross_validation_models**: Especifique se deve manter os modelos validados cruzadamente;
- **keep_cross_validation_predictions**: Mantenha a previsão de validação cruzada;
- **ntrees**: Especificar o número de árvores (usamos 200). O melhor valor encontrado para o nosso modelo foi 50 e pode ser visto no gráfico abaixo.

Figura 4: Número ideal de árvores para os modelos baseados em Decision Trees



Fonte: próprio autor

Os resultados do AutoML

Os modelos são classificados por uma métrica padronizada com base no tipo de problema, em nosso caso, a segunda coluna da Tabela 4. Em problemas de classificação binária como o nosso, a métrica mais adequada é AUC (HOSSIN; SULAIMAN, 2015). Na tabela acima temos xGBoost como vencedor, mas na próxima seção faremos a modelagem individual dos principais algoritmos usados neste AutoML.

Tabela 4: Resultados do Processamento em Auto Machine Learning

model_id	auc	logloss	aucpr	mean_per_class_error	rmse	mse	training_time_ms	predict_time_per_row_ms
XGBoost_grid__1_AutoML_20201124_144750_model_1	0.993292	0.0277407	0.999835	0.159886	0.0894191	0.00799577	5352	0.002391
StackedEnsemble_BestOfFamily_AutoML_20201124_144750	0.993196	0.0365935	0.999829	0.182427	0.093415	0.00872636	2181	0.003281
StackedEnsemble_AllModels_AutoML_20201124_144750	0.991782	0.0365219	0.999772	0.163859	0.093419	0.0087271	2204	0.006192
XGBoost_3_AutoML_20201124_144750	0.99149	0.0338313	0.999779	0.193762	0.0916761	0.00840451	853	0.001375
DeepLearning_1_AutoML_20201124_144750	0.975569	0.0447155	0.999229	0.263823	0.106525	0.0113475	394	0.002063
XGBoost_2_AutoML_20201124_144750	0.970811	0.0715979	0.999069	0.194049	0.110599	0.0122321	572	0.001129
XGBoost_1_AutoML_20201124_144750	0.966716	0.0801211	0.998965	0.202587	0.1169	0.0136656	621	0.000481
GLM_1_AutoML_20201124_144750	0.964273	0.0529767	0.998959	0.309863	0.11587	0.0134258	3738	0.000505
XGBoost_grid__1_AutoML_20201124_144750_model_2	0.963231	0.448694	0.99862	0.245123	0.361852	0.130937	98	0.000209
XRT_1_AutoML_20201124_144750	0.953832	0.100481	0.99788	0.197604	0.0957738	0.00917263	581	0.000821
DeepLearning_grid__1_AutoML_20201124_144750_model_1	0.952885	0.0626127	0.998597	0.302995	0.122166	0.0149246	1973	0.004954
DRF_1_AutoML_20201124_144750	0.944734	0.123832	0.997437	0.179298	0.0971027	0.00942893	464	0.000708

Fonte: próprio autor

Conjuntos empilhados (*Stacked Ensembles*)

Os métodos de *Machine Learning* usam algoritmos de aprendizagem múltiplos, assim conseguem obter melhor desempenho preditivo do que poderia ser obtido de qualquer um dos algoritmos de aprendizagem separadamente. Muitos dos algoritmos populares modernos de aprendizado de máquina são na verdade conjuntos. Por exemplo, R*andom Forest* (RF) e *Gradient Boosting Machine* (GBM) são ambos conjuntos chamados de *Ensemble Learners*. Tanto o Bagging (por exemplo, Random Forest) quanto o Gradiente Boost (por exemplo, GBM) são métodos para ensembling que tomam uma coleção de aprendizados fracos (por exemplo, árvore de decisão) e juntos formam um único e forte conjunto de aprendizado (AKOUR; ALSMADI; ALAZZAM, 2017).

H2O usa a regra trapezoidal para aproximar a área sob a curva (ROC). A AUC geralmente não é a melhor métrica para um alvo binário desequilibrado (como no nosso caso), porque um alto número de True Negatives pode fazer com que a AUC pareça inflada. Para um alvo binário desequilibrado, recomendamos AUCPR ou MCC (HAND 2009). Os resultados do AutoML estão na Tabela 5.

Tabela 5: Resultados do Stacked Ensemble

Modelo de Stacked Ensemble	**Resultados**
Melhor AUC em testes de *Base-learner*	0.9867
Teste de *Ensemble* AUC	0.9923
Melhor teste de aprendizado de base F1	0.9936
Precisão do teste do Stacled Ensemble	0.9875
Conjunto de testes AUCPR	0.9994

Fonte: próprio autor

Construção dos Modelos Individuais

Com os resultados do AutoML, utilizamos apenas três dos modelos mais performáticos que são eles: XGBoost, GBM e DRF. Esta seção fornece uma visão geral de cada algoritmo disponível para este experimento (VAN DER LAAN; POLLEY; HUBBARD, 2007).

xGBoost é um algoritmo de aprendizagem supervisionado que implementa um processo chamado boosting para produzir modelos precisos. O aumento refere-se à técnica de aprendizagem do conjunto de construção de muitos modelos sequencialmente, com cada novo modelo tentando corrigir as deficiências do modelo anterior. No aumento de árvores, cada novo modelo que é adicionado ao conjunto é uma árvore de decisão. O XGBoost fornece um reforço paralelo de árvores (também conhecido como GBDT, GBM) que resolve muitos problemas de ciência de dados de forma rápida e precisa. Para muitos problemas, o XGBoost é uma das melhores estruturas de máquina de aumento de gradiente (GBM) atualmente.

Gradiente Boosting Machine (para regressão e classificação) é um método de conjunto de aprendizagem para a frente. A heurística orientadora é que bons resultados preditivos podem ser obtidos através de aproximações cada vez mais refinadas. O GBM do H2O constrói sequencialmente árvores de regressão em todas as características do conjunto de dados de forma totalmente distribuída - cada árvore é construída em paralelo (BREIMAN 1996).

O Dynamic Random Dorest (DRF) é uma poderosa ferramenta de classificação e regressão. Quando dado um conjunto de dados, a DRF gera uma floresta de árvores de classificação ou regressão, em vez de uma única árvore de classificação ou regressão. Cada uma dessas árvores é um aprendiz fraco construído sobre um subconjunto de linhas e colunas. Mais árvores reduzirão a variância. Tanto a classificação quanto a regressão levam a previsão média sobre todas as suas árvores para fazer uma previsão final, seja prevendo uma classe ou valor numérico. (Nota: Para uma coluna de resposta categórica, a DRF mapeia fatores (por exemplo, cep_origin e cep_destination).

Construindo Modelos Individuais – Resultados

Os resultados individuais dos três algoritmos estão divididos em sete métricas de avaliação de desempenho que estão descritas na Tabela 6 (LEDELL 2015).

Tabela 6: Métricas de avaliação de desempenho dos modelos

nameModel	HammingLoss	ACC	MCC	RCI	CSI	F1	Cost
Model_XGB_Score	0.004136	0.995864	0.879886	0.736749	0.880650	0.967698	0.004136
Model_GBM_Score	0.004547	0.995453	0.867789	0.717758	0.868569	0.964464	0.004523
Model_DRF_Score	0.005321	0.994679	0.840360	0.653938	0.844429	0.956341	0.005321

Fonte: próprio autor

Hamming Loss - É a fração de rótulos que são incorretamente previstos. A perda média de Hamming ou a distância de Hamming entre o alvo e a previsão. Em primeiro lugar, consideramos o problema de classificação binária para comparar a qualidade entre os modelos.

ACC - A acurácia é uma métrica para avaliar modelos de classificação. Informalmente, a acurácia é a fração das previsões que nosso modelo acertou.

MCC - O coeficiente de correlação de Matthews é usado no aprendizado de máquina como uma medida da qualidade das classificações binárias (duas classes), introduzidas pelo bioquímico Brian W. Matthews em 1975. Leva em conta verdadeiros e falsos positivos e negativos e é geralmente considerado como uma medida equilibrada que pode ser usada mesmo que as classes sejam de tamanhos muito diferentes.

RCI - Informações relativas do classificador. O desempenho de diferentes classificadores no mesmo domínio pode ser medido comparando informações relativas do classificador, enquanto as informações do classificador (informações mútuas) podem ser usadas para comparação entre diferentes problemas de decisão.

CSI - Índice de sucesso de classificação. O Índice de Sucesso de Classificação (CSI) é uma medida global definida pela média do ICSI em todas as classes.

F1 - Na análise estatística da classificação binária, o escore F ou f-medida é uma medida da precisão de um teste. É calculado a partir da precisão e recall do teste, onde a precisão é o número de resultados positivos corretamente identificados divididos pelo número de todos os resultados positivos, incluindo aqueles não identificados corretamente, e o recall é o número de resultados positivos corretamente identificados dividido pelo número de todas as amostras que deveriam ter sido identificadas como positivas.

Custo - Matriz de custos é semelhante à matriz de confusão, exceto o fato de que estamos calculando o custo da previsão errada ou previsão correta.

Variáveis mais importantes

A Importância de uma Variável representa a significância estatística de cada variável nos dados em relação ao seu efeito no modelo gerado. Importância da Variável é, na verdade, cada ranking preditor com base na contribuição que os preditores fazem ao modelo. Nossos resultados são os seguintes:

Tabela 7: Variáveis mais importantes

variable	relative_importance	scaled_importance	percentage
elapsed_time	10344.422852	1.000000	0.491519
cep_destination	6393.197266	0.618033	0.303775
points	3760.372070	0.363517	0.178675
cep_origin	547.842712	0.052960	0.026031

Fonte: próprio autor

Configuração do Experimento

Este experimento foi realizado em um computador com as seguintes configurações de Hardware e Software:

- Linux Ubuntu 20.04 (AWS)
- HVM de virtualização
- RAM 64 GB
- Número de CPU 32
- SSD 256 GB

Tempo total gasto processando os três algoritmos: tempo: 3h 45min 7s

Pacotes e Instalações usados no Linux (comandos para instalar):

sudo apt-get atualização

sudo apt-get instalar software-propriedades-comuns

sudo add-apt-repository ppa:deadsnakes/ppa

sudo apt-get atualização

sudo apt-get instalar python3.8

sudo apt-get instalar python3-pip

pip3 instalar pandas

pip3 instalar pyspark

pip3 instalar pycm

pip3 instalar h2o

pip3 instalar joblib

pip3 instalar ipywidgets

sudo apt-get instalar default-jre

sudo apt-get instalar dateutils

pip3 instalar jupyterlab

Previsão do Modelo

A modelagem preditiva na negociação é um processo de modelagem em que a probabilidade de um resultado é prevista usando um conjunto de variáveis preditoras. O alvo é a variável do nosso Dataset original que foi calculada entre a data de entrega e a data de promessa, sendo a primeira a maior o pacote atrasado. A coluna de *target_cal* é a coluna calculada pelo modelo dizendo se naquele ponto o pacote está atrasado. A coluna de *target_bin* é a coluna original rotulada que serviu de treinamento, mas que o modelo preditor não viu.

Tabela 8: Probabilidade de atraso a cada observação temporal do pacote

cep_origin	cep_destination	points	elapsed_time	target	score	target_cal	target_bin
0504	7900	1	0	0	0.000005	1	0
0504	7900	2	601043	0	0.069190	1	0
0504	7900	3	601381	0	0.069190	1	0
0504	7900	4	1207238	0	0.995051	0	0
0504	7900	5	1207246	0	0.995051	0	0
0504	7900	6	1211166	0	0.995051	0	0
0504	7900	7	1213191	0	0.995051	0	0
0504	7900	8	1213196	0	0.995051	0	0
0504	7900	9	1218178	0	0.995051	0	0
0504	7900	10	1218536	0	0.995051	0	0
0504	7900	11	1347894	0	0.995051	0	0
0504	7900	12	1381326	0	0.991031	0	0
0504	7900	13	1387183	0	0.991031	0	0
0504	7900	14	1459801	0	0.998086	0	0

Fonte: próprio autor

A coluna de score é a probabilidade de o pacote estar atrasado no ponto de contato na linha específica. Podemos exemplificar, neste caso, no terceiro BIP a operadora saberia que o pacote tem 99,5% de atraso e poderia fornecer ações para mitigar essa situação. O Registro de Classificação original tem os seguintes dados anonimizados:

Tabela 9: Registro sistêmico das observações a cada ponto de contato

ID	Local	Package_id	Operation	Created_time
1	Local#01	12345	Receiveing	2020-09-12 12:23:25
2	Local#01	12345	BAG	2020-09-19 11:20:48
3	Local#02	12345	Transport	2020-09-19 11:44:03
4	**Local#03**	**12345**	**Transport**	**2020-09-26 13:26:00**
5	Local#04	12345	Transport	2020-09-26 13:46:23
6	Local#05	12345	Receiving	2020-09-27 10:05:20
7	Local#05	12345	Transport	2020-09-27 11:23:12

O pacote foi entregue tardiamente. Não inserimos na Tabela 9 todas alinhas descritas na Tabela 8. Ao observarmos a linha quatro fica evidente o motivo pelo qual o pacote atrasou. Por alguma razão ele levou uma semana em um transporte sem receber qualquer atualização sistêmica. Assim como esse evento, outros eventos podem ser detectados antecipadamente.

CONCLUSÕES

Podemos concluir com esse experimento que é possível, preventivamente, observar eventos temporais e prever seu estado binário de atrasado ou não dentro de uma cadeia logística. A eficácia dos modelos, DRF, xGBoost e GBM demonstram que os algoritmos com regressão entregam

resultados mais robustos. Não há evidências significativas de que outras características sejam necessárias para determinar se um pacote vai atrasar. Observamos que dependendo da amostra, os três algoritmos se revezam no topo da lista com os melhores resultados na classificação, portanto, nosso sistema grava a lista da Tabela 6 para ser usada nas predições futuras. Concluímos ainda que não é possível ter apenas uma ou duas métricas para a avaliação do melhor algoritmo, pois dependendo dos resultados do Custo e Hamming Loss podem alterar a decisão de uso de um ou outro.

REFERÊNCIAS BIBLIOGRÁFICAS

AKOUR, Mohammed; ALSMADI, Izzat; ALAZZAM, Iyad. Software fault proneness prediction: a comparative study between bagging, boosting, and stacking ensemble and base learner methods. International Journal of Data Analysis Techniques and Strategies, v. 9, n. 1, p. 1-16, 2017.

APTE, Uday M.; VISWANATHAN, S. Effective cross docking for improving distribution efficiencies. International Journal of Logistics, v. 3, n. 3, p. 291-302, 2000.

BIRRANE, Edward; BURLEIGH, Scott; KASCH, Niels. Analysis of the contact graph routing algorithm: Bounding interplanetary paths. Acta Astronautica, v. 75, p. 108-119, 2012.

BREIMAN, Leo. Stacked regressions. Machine learning, v. 24, n. 1, p. 49-64, 1996.

HAND, David J. Measuring classifier performance: a coherent alternative to the area under the ROC curve. Machine learning, v. 77, n. 1, p. 103-123, 2009.

HOSSIN, Mohammad; SULAIMAN, M. N. A review on evaluation metrics for data classification evaluations. International Journal of Data Mining & Knowledge Management Process, v. 5, n. 2, p. 1, 2015.

KUHN, Max et al. Applied predictive modeling. New York: Springer, 2013.

LIMA, Carla Augusta Evangelista et al. Análise de anomalias: métodos simplificados. 2009.

LEDELL, Erin. Scalable Ensemble Learning and Computationally Efficient Variance Estimation. 2015. Tese de Doutorado. UC Berkeley.

PAURA, Glávio Leal. Fundamentos da logística. 2016.

RUSSELL, Stuart J.; NORVIG, Peter. Inteligência artificial. Elsevier, 2004.

TEINEMAA, Irene et al. Outcome-oriented predictive process monitoring: Review and benchmark. ACM Transactions on Knowledge Discovery from Data (TKDD), v. 13, n. 2, p. 1-57, 2019.

VAN DER LAAN, Mark J.; POLLEY, Eric C.; HUBBARD, Alan E. Super learner. Statistical applications in genetics and molecular biology, v. 6, n. 1, 2007.

WOLPERT, David H. Stacked generalization. Neural networks, v. 5, n. 2, p. 241-259, 1992.