UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA

Felipe Trindade Radovanovic

Desenvolvimento de pipelines para padronização, limpeza e enriquecimento de dados veiculares

Florianópolis
2024

Felipe Trindade Radovanovic

Desenvolvimento de pipelines para padronização, limpeza e enriquecimento de dados veiculares

Trabalho de Conclusão de Curso submetido ao curso de Graduação em Engenharia Eletrônica do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador: Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho

Florianópolis

2024

Radovanovic, Felipe Trindade
Desenvolvimento de pipelines para padronização, limpeza e enriquecimento de dados veiculares / Felipe Trindade Radovanovic ; orientador, André Carlos Ponce de Leon Ferreira de Carvalho, 2024.
60 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Eletrônica, Florianópolis, 2024.

Inclui referências.

1. Engenharia Eletrônica. 2. Dados veiculares. 3. Padronização de dados. 4. Limpeza de dados. 5. Enriquecimento de dados. I. de Carvalho, André Carlos Ponce de Leon Ferreira . II. Universidade Federal de Santa Catarina. Graduação em Engenharia Eletrônica. III. Título.

Felipe Trindade Radovanovic

Desenvolvimento de pipelines para padronização, limpeza e enriquecimento de dados veiculares

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Bacharel em Engenharia Eletrônica e aprovado em sua forma final pelo Curso de Graduação em Engenharia Eletrônica.

Florianópolis, 12 de março de 2024.

Documento assinado digitalmente
Daniela Ota Hisayasu Suzuki
Data: 21/03/2024 13:45:19-0300
CPF: ***.397.389-**
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Curso

Banca examinadora

Documento assinado digitalmente
ANDRE CARLOS PONCE DE LEON F...CARVALHO
Data: 19/03/2024 22:06:03-0300
CPF: ***.418.884-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho

Orientador

Documento assinado digitalmente
ARTUR SABINO DE ANDRADE
Data: 19/03/2024 10:20:40-0300
CPF: ***.632.969-**
Verifique as assinaturas em https://v.ufsc.br

Artur Sabino de Andrade

Mobway

Documento assinado digitalmente
Danilo Silva
Data: 19/03/2024 17:17:52-0300
CPF: ***.557.924-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. Danilo Silva

Universidade Federal de Santa Catarina

Florianópolis, 2024

Este trabalho é dedicado aos meus amigos e familiares,
os quais me apoiaram durante toda minha trajetória.

AGRADECIMENTOS

Agradeço a todos que estiveram ao meu lado ao longo desta jornada acadêmica, contribuindo de maneira significativa para a conclusão deste trabalho.

Aos meus pais, Eduardo Radovanovic e Cremilde Aparecida Trindade Radovanovic, que sempre me apoiaram e incentivaram. Sem a paciência e compreensão de vocês, eu não teria alcançado este feito. Obrigado por acreditarem em mim incondicionalmente.

Aos meus amigos e demais familiares, que estiveram ao meu lado nos momentos de estudo, de dificuldades e de descontração, agradeço por tornarem essa jornada mais leve e divertida. O companheirismo de vocês foi fundamental.

À minha namorada, Bruna Just Meller, que me acompanhou em todos os momentos, agradeço por seu amor, compreensão e incentivo constantes. Você foi meu alicerce emocional e minha maior motivação.

Ao meu orientador, Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho, agradeço por sua orientação, sabedoria e dedicação ao longo deste trabalho. Suas sugestões e feedbacks foram cruciais para o sucesso deste projeto. Agradeço aos membros da banca por aceitarem avaliar esse trabalho.

Por fim, agradeço a todos que, de alguma forma, contribuíram para esta conquista. Este trabalho não seria possível sem o apoio e o carinho de cada um de vocês. Muito obrigado.

"Não podemos resolver nossos problemas com o mesmo pensamento que tínhamos quando os criamos." (Einstein)

RESUMO

Esse projeto visa desenvolver uma *pipeline* de processamento de dados veiculares, que inclui processos de padronização, limpeza e enriquecimento. O mesmo foi realizado dentro da mobway, uma startup que conecta montadoras com veículos conectados a empresas nos diversos setores da mobilidade, permitindo que utilizem dados veiculares para otimizar sua operação e serviços. Esta *pipeline* será a base para o todo o desenvolvimento relacionado a tratamento de dados ao longo dos próximos anos. Desta forma, um conjunto pequeno de dados foi selecionado para figurar a primeira versão da *pipeline.* O projeto foi desenvolvido em Python, utilizando principalmente a biblioteca Pandas, e implementado em um orquestrador de dados chamado Mage. Os resultados obtidos foram satisfatórios, e serão a base para o MVP da mobway.

Palavras-chave: Dados veiculares; Padronização de dados; Limpeza de dados; Enriquecimento de dados.

ABSTRACT

This project aims to develop a vehicle data processing pipeline, which includes standardization, cleaning, and enrichment processes. The same was carried out within mobway, a startup that connects OEMs with connected vehicles to companies in different mobility sectors, allowing them to use vehicle data to optimize their operations and services. This pipeline will be the basis for all development related to data processing over the next few years. In this way, a small set of data was selected to represent the first version of the pipeline. The project was developed in Python, using mainly the Pandas library, and implemented in a data orchestration tool called Mage. The results obtained were satisfactory and will be the basis for mobway MVP.

Keywords: Vehicle data; Data standardization; Data cleaning; Data enrichment; Data orchestration.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IBGE	Instituto Brasileiro de Geografia e Estatística
ETL	*Extract, Transform, and Load*
ELT	*Extract, Load, and Transform*
IoT	*Internet of Things*
AWS	*Amazon Web Services*
LGPD	Lei Geral de Proteção de Dados Pessoais
VIN	*Vehicle Identification Number*
WMI	*World Manufacturer Identifier*
RPM	Rotações por Minuto
LV	*Low Voltage*
HV	*High Voltage*
S3	*Simple Storage Service*
ECS	*Elastic Container Service*
CPU	*Central Processing Unit*
SQL	*Structured Query Language*
dbt	*Data build tool*
GPU	*Graphics processing unit*
TPU	*Tensor processing unit*

SUMÁRIO

1 INTRODUÇÃO

No contexto atual, o surgimento de um mundo conectado, onde pessoas, dispositivos e serviços estão interligados, está se tornando uma realidade palpável. O termo "conectividade" tornou-se comum no campo da engenharia, refletindo o avanço das telecomunicações e o desenvolvimento de tecnologias relacionadas a sistemas de informação e tratamento de dados. Esses avanços já possibilitam uma ampla gama de produtos, sistemas e serviços com diversas aplicações. Nesse novo cenário, as indústrias têm a oportunidade de buscar soluções para aumentar sua competitividade e produtividade, aproveitando essa conectividade para impulsionar a inovação de produtos, serviços e processos internos (Sugayama; Negrelli, 2015).

Dados do último Censo do Instituto Brasileiro de Geografia e Estatística (IBGE, 2023) apontam que a frota brasileira foi de cerca de 45 milhões, em 2006, para mais de 115 milhões de veículos em 2022. Sendo assim, houve um aumento de 155,65% em 16 anos, o que represente em média 4,3 milhões de veículos por ano. Nesse contexto de expansão veicular, tem-se os veículos conectados, que são dotados de tecnologias que permitem coleta, compartilhamento e transmissão de dados em tempo real, tanto com o fabricante do veículo quanto com outros dispositivos conectados.

Veículos conectados geram e enviam em tempo real mais de 200 pontos de dados para as montadoras, incluindo dados como velocidade, geolocalização e estado de carga da bateria. Embora esses dados sejam valiosos para uma ampla gama de prestadores de serviços em setores relacionados à mobilidade, como locadoras, seguradoras e empresas de gestão de frota, o acesso a essas informações é geralmente limitado e os dados não estão de maneira homologada. Muitas vezes, essas empresas precisam recorrer a sistemas de telemetria terceirizados, uma solução que tende a ser menos eficaz, pois captura apenas uma fração dos dados disponíveis e com qualidade inferior em comparação com as informações geradas internamente pelos próprios veículos.

O acesso por parte dos prestadores de serviço aos dados gerados pelos próprios veículos representa grande potencial para todas as partes envolvidas. Ao terem acesso a serviços nativamente conectados, os proprietários se beneficiam ao terem uma experiência com o veículo melhorada, que podem contribuir para redução

do consumo de combustível, aumento da segurança, preservação do veículo e até mesmo melhorar a experiência de recarga, em casos de elétricos.

Por outro lado, os prestadores de serviço têm a oportunidade de substituir a instalação de uma telemetria terceirizada, o que reduziria os custos associados aos processos de instalação e manutenção, além de diminuir o tempo de ociosidade gerado por esses processos. Além disso, os custos também reduzem com a otimização da sua operação, gerada pelo uso estratégico dos dados. Por fim, ao oferecerem serviços mais atraentes aos clientes em comparação com outras opções, eles podem aumentar seu faturamento.

Da mesma forma, as montadoras também se beneficiam ao tornar seus veículos mais atrativos, por apresentarem compatibilidade com serviços conectados, o que potencialmente aumenta o número de vendas. Além disso, geram uma nova fonte de monetização com a comercialização dos dados gerados pelos veículos.

Apesar de beneficiar toda a cadeia automotiva, esse processo apresenta grandes desafios de implementação. De acordo com Bertoncello *et al.* (2023) a falta de padronização dos dados entre diferentes modelos e marcas de veículos é o principal desafio tecnológico, seguido da necessidade de ter as ferramentas e habilidades para o uso efetivo e estratégico dos dados.

Para alcançar esse objetivo, é possível empregar uma pipeline. No contexto de tratamento de dados, uma *pipeline* se refere a uma série de etapas ou processos sequenciais que são aplicados aos dados, geralmente de forma automatizada. Cada etapa na *pipeline* realiza uma função específica, como padronização, limpeza ou enriquecimento dos dados, e os dados fluem de uma etapa para a próxima de maneira ordenada. Uma pipeline de tratamento de dados é projetada para processar grandes volumes de dados de forma eficiente e consistente, garantindo que os dados estejam prontos para análise ou uso em outras aplicações.

Dessa forma, o propósito desse projeto é de atuar nestes dois principais desafios, implementando um processo de padronização, limpeza e enriquecimento dos dados.

1.1 OBJETIVO GERAL

O objetivo geral deste projeto é desenvolver uma pipeline de tratamento de dados veiculares abrangente, englobando processos de padronização, limpeza e

enriquecimento. O foco é contribuir para a integração de dados homologados e nativos dos veículos com empresas do setor de mobilidade, beneficiando toda a cadeia automotiva. Além disso, busca estabelecer uma base para futuras implementações da mobway na área de tratamento de dados, impulsionando o desenvolvimento contínuo e a inovação no campo da mobilidade conectada.

1.2 OBJETIVOS ESPECÍFICOS

Objetivos específicos foram traçados a fim de guiar o desenvolvimento do projeto e garantir que o objetivo geral seja alcançado:

- Definir um conjunto de dados a serem alvo do processo de tratamento com base no que é disponibilizado pelas montadoras e o que entrega valor aos prestadores de serviço;
- Definir e implementar um conjunto de regras de padronização de dados, garantindo a facilidade da integração desses dados aos sistemas dos prestadores de serviço;
- Definir e implementar um conjunto de regras de limpeza de dados, garantindo que dados errôneos sejam corrigidos, antes de enviados às empresas;
- Definir e implementar um conjunto de regras de enriquecimento de dados, facilitando a interpretação por parte dos prestadores de serviço;
- Avaliar e otimizar o tempo de processamento, garantindo que os consumidores de dados, tenham acesso a eles pouco tempo após terem sido gerados.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ABORDAGENS DE PROCESSAMENTO DE DADOS

Os processos de extração, transformação e carregamento (ETL) e extração, carregamento e transformação (ELT) são as principais abordagens para processamento de dados. As duas abordagens incluem as três mesmas etapas, porém em ordens diferentes, o que gera vantagens e desvantagens para ambas (AWS, 2023).

2.1.1 Extração

A fase inicial de ambas as abordagens é conhecida como extração, etapa que desempenha a função da obtenção de dados brutos, que podem vir de diversas fontes. Estas fontes podem ser bancos de dados convencionais, arquivos, aplicações de software como serviço (SaaS), dispositivos IoT e eventos de aplicações. Durante esta etapa, é possível coletar dados semiestruturados, estruturados ou não estruturados.

2.1.2 Transformação

Na metodologia ETL, a etapa subsequente à extração é conhecida como transformação, enquanto no ELT ela é a terceira fase. Esta etapa visa converter os dados brutos de sua estrutura original para um formato compatível com os requisitos do sistema de destino, onde os dados serão armazenados para análise posterior. Durante a transformação, uma série de operações são realizadas, tais como citadas por Densmore (2021):

- Alteração de tipos ou formatos de dados.
- Remoção de dados inconsistentes ou imprecisos.
- Eliminação de duplicatas.
- Aplicação de regras e funções para limpeza e preparação dos dados para análise no sistema de destino.

Essas ações são essenciais para garantir que os dados estejam prontos e adequados para análise e interpretação, contribuindo assim para a qualidade e eficácia do processo de tomada de decisões.

2.1.3 Carregamento

Durante esta fase, os dados são armazenados no banco de dados de destino. Nos processos de ETL, o carregamento dos dados é a etapa final, permitindo que ferramentas gerem relatórios e insights. Por outro lado, no ELT, esta é a segunda etapa, sendo ainda necessário realizar transformações adicionais nos dados extraídos.

2.1.4 Extração, Transformação e Carregamento

Nesta abordagem, os dados são transformados em um servidor de processamento secundário, uma vez que eles ainda não foram carregados no banco de dados. Este processo funciona melhor com dados estruturados, uma vez que os dados já são carregados no banco de destino em um formato predeterminado. Em relação a velocidade, custos e segurança, esta abordagem também apresenta desvantagens em relação ao ELT.

Apesar do ETL apresentar diversas desvantagens, em algumas situações especificas, ele pode ser uma boa opção, como em:

- Bancos de dados antigos: integração de bancos de dados legados ou fontes de terceiros com formatos de dados fixos pode ser mais eficientemente realizada com ETL. Uma vez transformados, os dados podem ser usados de forma eficaz para análises futuras.
- Experimentação: em grandes organizações, durante experimentos de dados para descobrir fontes ocultas ou testar novas ideias, o ETL pode ser útil para entender o banco de dados e sua utilidade em cenários específicos.
- Análise complexa: em casos de análises complexas que envolvem múltiplos formatos de dados de fontes variadas, a combinação de ETL e ELT pode ser vantajosa. Os cientistas de dados podem configurar pipelines de ETL a partir

de algumas fontes e usar ELT com outras, aumentando a eficiência da análise e a performance da aplicação.

Em cenários de IoT que envolvem fluxos de dados de sensores, o uso de ETL em vez de ELT pode ser mais apropriado, tendo como exemplos:

- Conversão de dados de diferentes protocolos em formatos padrão para uso em *workloads* na nuvem.
- Filtragem de dados de alta frequência e execução de funções de agregação em grandes conjuntos de dados antes do carregamento.
- Cálculo de valores locais em dispositivos IoT e envio de valores filtrados para o *back-end* da nuvem.
- Limpeza, eliminação de duplicação ou preenchimento de elementos ausentes em séries temporais de dados.

2.1.5 Extração, carregamento e transformação

Nesta abordagem, os dados são transformados no banco de dados de destino, após o seu carregamento. Este processo funciona bem com dados estruturados, semiestruturados e não estruturados. Em relação a velocidade, o ELT é mais rápido pois pode utilizar de recursos internos do banco de dados. Pelo mesmo motivo a segurança é superior do que o modelo ETL. Por obter diversas vantagens, esta abordagem é a escolha padrão para análises modernas.

2.1.6 Comparação entre as abordagens

O Quadro 1 resume os pontos comparativos elencados nos tópicos anteriores.

Quadro 1 – Comparação entre as abordagens extração, transformação e carregamento (ETL) e extração, carregamento e transformação (ELT)

Abordagem	ETL	ELT
Processo	Extrai os dados brutos, os transforma em um formato predeterminado e os carrega no banco de dados.	Extrai os dados brutos, os carrega no banco de dados e os transforma.
Local de transformação	Servidor de processamento secundário.	Banco de dados de destino.
Compatibilidade	Dados estruturados.	Dados estruturados, semiestruturados e não estruturados.
Velocidade	Mais lento.	Mais rápido, pois tem à disposição recursos do banco de dados.
Custo	Mais cara.	Mais barata.
Segurança	Menos seguro.	Mais seguro, pois tem à disposição recursos do banco de dados.

Adaptado de: AWS, 2024

2.2 LIMPEZA DE DADOS

O processo de limpeza de dados envolve várias etapas para identificar e corrigir problemas nos conjuntos de dados. Inicialmente, é crucial analisar os dados para identificar erros, o que pode ser feito utilizando ferramentas de análise qualitativa que empregam regras, padrões e restrições predefinidas. Em seguida, os erros identificados devem ser removidos ou corrigidos. Segundo a AWS (2024), as etapas de limpeza de dados geralmente incluem:

- Identificação e remoção de dados duplicados.
- Eliminação de dados irrelevantes para a análise específica.
- Identificação e tratamento de *outliers* que possam afetar alguma análise ou representar um erro de medida.
- Sinalização e tratamento de dados ausentes.
- Correção de erros estruturais, como erros tipográficos e inconsistências na formatação.

3 METODOLOGIA

3.1 PRIVACIDADE DE DADOS

A Lei Geral de Proteção de Dados (LGPD) é uma legislação brasileira que visa proteger a privacidade e a segurança dos dados pessoais dos cidadãos. Ela estabelece regras claras sobre como organizações podem coletar, armazenar, processar e compartilhar informações pessoais. Entre os principais princípios da LGPD estão a transparência no uso de dados, o consentimento do titular, a segurança da informação e a responsabilidade das empresas na gestão adequada desses dados.

Para o desenvolvimento deste projeto, serão utilizados *datasets* disponibilizados à mobway, como amostras, por diferentes montadoras. A fim de garantir a privacidade dos proprietários dos veículos, os VINs serão anonimizados logo após o carregamento dos dados.

O VIN é o número de identificação de um veículo, também conhecido como número do chassi. Ele possui 17 caracteres que podem ser letras ou números, que são dispostos da seguinte forma:

- 1º ao 3º caractere: São chamados de *World Manufacturer Identifier* (WMI), eles indicam a divisão específica do fabricante do veículo;
- 4º caractere: Indica características específicas do modelo e características de segurança;
- 5º caractere: Indica a série do modelo específico do veículo;
- 6º e 7º caracteres: Indicam o tipo de carroceria do veículo;
- 8º caractere: Indica o tamanho do motor;
- 9º caractere: É o código de segurança, utilizado para identificar a autenticidade do VIN;
- 10º caractere: Indica o ano de fabricação do veículo;
- 11º caractere: Indica em qual planta do fabricante o veículo foi feito;
- 12º ao 17º caractere: É o número de série do veículo.

Afim de anonimizar os identificadores de cada veículo, eles serão substituídos por uma sequência de 17 caracteres gerados aleatoriamente com o seguinte padrão:

- 1º caractere: Dígito
- 2º ao 5º caractere: Letras
- 6º ao 7º caractere: Dígitos
- 8º ao 11º caractere: Letras
- 12º ao 17º caractere: Dígitos

As montadoras que disponibilizaram os dados e também os clientes que foram envolvidos no projeto também terão sua identidade preservada ao longo de todo este trabalho.

Por fim, buscando preservar os segredos tecnológicos da mobway, os códigos desenvolvidos para este projeto não serão apresentados, entretanto toda a sua lógica de funcionamento será bem detalhada.

3.2 PLATAFORMAS UTILIZADAS

Inicialmente o desenvolvimento foi realizado no Google Colab, uma plataforma que permite aos usuários escrever e executar código Python em um ambiente de notebook baseado na web. A principal vantagem do Colab é que ele fornece acesso gratuito a recursos de computação, incluindo GPUs (Unidades de Processamento Gráfico) e TPUs (Unidades de Processamento Tensorial), que são úteis para realizar tarefas intensivas de processamento de dados.

Esta plataforma foi escolhida por apresentar diversas características que facilitam o rápido desenvolvimento, como:

- Notebooks interativos: Os notebooks Colab permitem escrever e executar código em células individuais, o que facilita o desenvolvimento iterativo e a experimentação.
- Integração com Google Drive: Os notebooks Colab estão integrados ao Google Drive, o que permite armazenar, compartilhar e colaborar em projetos de forma fácil.
- Bibliotecas pré-instaladas: O Colab vem pré-instalado com várias bibliotecas populares de Python, como TensorFlow, PyTorch, NumPy e pandas, tornando-o adequado para uma variedade de tarefas de análise de dados.

- Suporte a markdown: Além de código, os notebooks Colab também suportam células de texto formatado usando Markdown, o que permite adicionar documentação, explicações e gráficos explicativos aos seus projetos.

Após a finalização do desenvolvimento inicial, a plataforma Colab foi substituída pelo Mage, uma plataforma orquestradora de dados, que permite transformar a solução em comercializável. Uma plataforma orquestradora de dados é um sistema de software projetado para gerenciar e coordenar o fluxo de dados em um ambiente de computação distribuída ou em nuvem. Algumas das características do Mage que o fizeram ser selecionado são:

- Gerenciamento de fluxo de dados: Coordena o movimento de dados entre diferentes sistemas, aplicativos e serviços. Isso pode envolver a ingestão de dados de diversas fontes, o processamento desses dados em diferentes estágios e a entrega dos resultados aos destinos apropriados.
- Agendamento e coordenação de tarefas: A plataforma é capaz de agendar e coordenar a execução de diversas tarefas de processamento de dados, garantindo que elas sejam executadas de forma eficiente e no momento adequado. Isso pode incluir a execução de pipelines de dados, transformações de dados, análises e outros processamentos.
- Monitoramento e gerenciamento de recursos: A plataforma fornece recursos para monitorar o desempenho do sistema, o status das tarefas em execução e o uso de recursos de computação, armazenamento e rede. Isso permite identificar e resolver problemas rapidamente, otimizar o desempenho do sistema e garantir a disponibilidade e confiabilidade dos dados.
- Interface e usabilidade: O Mage possui uma interface e usabilidade que permitem um aprendizado mais rápido em relação à outros orquestradores de dados.

3.3 CONJUNTO DE DADOS

Um conjunto pequeno de dados foi escolhido para ser alvo do tratamento de dados. A escolha foi com base em dois fatores: a disponibilidade dos dados nas montadoras de veículos conectados e a atratividade para os prestadores de serviço.

A disponibilidade dos dados foi avaliada em três montadoras diferentes. Foram selecionados dados que possuem disponibilidade em pelo menos uma das montadoras, dando prioridade àqueles com a maior disponibilidade, mostrados no Quadro 2. O dado foi considerado disponível quando havia pelo menos uma leitura presente no dataset disponibilizado pela montadora.

Quadro 2 – Disponibilidade dos dados selecionados em diferentes montadoras

Dado	Montadora 1	Montadora 2	Montadora 3
VIN	Presente	Presente	Presente
Data e hora	Presente	Presente	Presente
Velocidade	Presente	Presente	Presente
Odômetro	Presente	Presente	Presente
Latitude	Presente	Presente	Presente
Longitude	Presente	Presente	Presente
Orientação	Ausente	Presente	Presente
RPM	Presente	Ausente	Ausente
Nível de óleo	Ausente	Ausente	Presente
Temperatura do líquido de arrefecimento do motor	Ausente	Ausente	Presente
Nível de combustível	Ausente	Presente	Presente
Autonomia	Ausente	Presente	Presente
Pressão do pneu dianteiro esquerdo	Ausente	Ausente	Presente
Pressão do pneu dianteiro direito	Ausente	Ausente	Presente
Pressão do pneu traseiro esquerdo	Ausente	Ausente	Presente
Pressão do pneu traseiro direito	Ausente	Ausente	Presente
Tensão da bateria LV	Presente	Ausente	Presente
Tensão da bateria HV	Ausente	Ausente	Presente
Estado de carga da bateria HV	Ausente	Ausente	Presente
Estado de carregamento da bateria HV	Ausente	Ausente	Presente
Estado do cinto de segurança	Ausente	Presente	Presente
Estado da ignição	Presente	Presente	Presente
Evento	Presente	Ausente	Presente

Apesar de dados com grande disponibilidade terem sido priorizados, dados com pouca disponibilidade também foram selecionados por serem estratégicos para prestadores de serviço.

Em entrevistas realizadas com clientes da mobway foi possível constatar que os dados selecionados possuem os seguintes casos de uso:

- VIN: identificação do veículo.
- Data e hora: identificação do horário em que os dados foram gerados.
- Velocidade: avaliação da condução em relação à segurança, consumo de combustível e eficiência operacional.
- Odômetro: avaliação do veículo para revenda.

- Latitude e longitude: rastreamento de veículos de frota e recuperação de veículos roubados.
- Orientação: utilizado para auxiliar na definição da rota de um veículo, em conjunto da latitude e longitude.
- RPM: avaliação da condução em relação ao consumo de combustível.
- Nível de óleo: manutenção preditiva e emissão de avisos para o motorista.
- Temperatura do líquido de arrefecimento do motor: manutenção preditiva e emissão de avisos para o motorista.
- Nível de combustível: avaliação da condução em relação ao consumo de combustível.
- Autonomia: utilizado na definição e otimização de rotas.
- Pressão do pneu: auxiliar a manter a pressão adequada para eficiência no consumo de combustível.
- Tensão da bateria LV: manutenção preditiva e emissão de avisos para o motorista.
- Tensão da bateria HV: manutenção preditiva e emissão de avisos para o motorista. Utilizado na definição e otimização de rotas. Personalização da experiência de carregamento.
- Estado de carga da bateria HV: manutenção preditiva e emissão de avisos para o motorista. Utilizado na definição e otimização de rotas. Personalização da experiência de carregamento.
- Estado de carregamento da bateria HV: personalização da experiência de carregamento.
- Estado do cinto de segurança: garantir segurança do motorista e redução de multas.
- Estado da ignição: identificação de veículos ligados fora de uso.
- Evento: avaliação da condução em relação à segurança. Personalização da experiência de carregamento.

Todos os datasets são estruturados em uma única tabela. Cada linha destas tabelas faz referência a um veículo em uma determinada data e hora, e várias informações são transmitidas de uma única vez por meio dos demais dados. A Tabela 1 exemplifica a estrutura dos dados.

Tabela 1 – Exemplo de estrutura das tabelas de dados

VIN	Data e Hora	Velocidade	Odômetro	Estado de ignição
12345678	17/02/2024 22:03:42	54	45610	Ignition on
23456789	17/02/2024 22:03:53	109	23879	Ignition on
12345678	17/02/2024 22:04:09	57	45611	Ignition on

3.4 PADRONIZAÇÃO DE DADOS

Para o processo de padronização, foram definidas características que seriam tratadas, sendo as seguintes:

- Nome: nome do dado utilizado no banco de dados, API e documentação.
- Formato: *float*, *object* ou *datetime*.
- Precisão: Número de casas decimais.
- Unidade: V, km e km/h por exemplo.
- Categorias: possíveis valores de um dado categórico.

As características definidas para cada um dos dados são mostradas nas seguintes subseções. Por questão de organização, os dados serão apresentados separados em quatro tipos, os quais recebem tipos de processamento diferentes, sendo eles: numéricos, categóricos, VIN e data e hora.

3.4.1 Numéricos

Este tipo, possui uma quantidade considerável de dados, todos com o formato *float*, variando a precisão entre 2 ou 6 dígitos, dependendo da aplicação dos dados. Suas características são as seguintes:

- Velocidade
    - Nome: *speed*
    - Formato: *float*
    - Precisão: 2
    - Unidade: quilômetros por hora (km/h)
    - Categorias: não aplicável
- Odômetro
    - Nome: *odometer*

    - Formato: *float*
    - Precisão: 2
    - Unidade: quilômetro (km)
    - Categorias: não aplicável
- Latitude
    - Nome: latitude
    - Formato: *float*
    - Precisão: 6
    - Unidade: grau (º)
    - Categorias: não aplicável
- Longitude
    - Nome: longitude
    - Formato: *float*
    - Precisão: 6
    - Unidade: grau (º)
    - Categorias: não aplicável
- Orientação
    - Nome: *heading*
    - Formato: *float*
    - Precisão: 2
    - Unidade: grau (º)
    - Categorias: não aplicável
- RPM
    - Nome: rpm
    - Formato: *float*
    - Precisão: 2
    - Unidade: rotações por minuto (rpm)
    - Categorias: não aplicável
- Nível de óleo
    - Nome: *oil_level*
    - Formato: *float*
    - Precisão: 2
    - Unidade: percentual (%)

    - Categorias: não aplicável
- Temperatura do líquido de arrefecimento do motor
    - Nome: *engine_cooling_temperature*
    - Formato: *float*
    - Precisão: 2
    - Unidade: graus Celcius (ºC)
    - Categorias: não aplicável
- Nível de combustível
    - Nome: *fuel_level*
    - Formato: *float*
    - Precisão: 2
    - Unidade: litros (L)
    - Categorias: não aplicável
- Autonomia
    - Nome: *range*
    - Formato: *float*
    - Precisão: 2
    - Unidade: quilômetro (km)
    - Categorias: não aplicável
- Pressão do pneu dianteiro esquerdo
    - Nome: *right_front_tire_pressure*
    - Formato: *float*
    - Precisão: 2
    - Unidade: libra-força por polegada quadrada (psi)
    - Categorias: não aplicável
- Pressão do pneu dianteiro direito
    - Nome: *left_front_tire_pressure*
    - Formato: *float*
    - Precisão: 2
    - Unidade: libra-força por polegada quadrada (psi)
    - Categorias: não aplicável
- Pressão do pneu traseiro esquerdo
    - Nome: *right_rear_tire_pressure*

    - Formato: *float*
    - Precisão: 2
    - Unidade: libra-força por polegada quadrada (psi)
    - Categorias: não aplicável
- Pressão do pneu traseiro direito
    - Nome: *left_rear_tire_pressure*
    - Formato: *float*
    - Precisão: 2
    - Unidade: libra-força por polegada quadrada (psi)
    - Categorias: não aplicável
- Tensão da bateria LV
    - Nome: *lv_battery_voltage*
    - Formato: *float*
    - Precisão: 2
    - Unidade: volt (V)
    - Categorias: não aplicável
- Tensão da bateria HV
    - Nome: *hv_battery_voltage*
    - Formato: *float*
    - Precisão: 2
    - Unidade: volt (V)
    - Categorias: não aplicável
- Estado de carga da bateria HV
    - Nome: *hv_battery_soc*
    - Formato: *float*
    - Precisão: 2
    - Unidade: percentual (%)
    - Categorias: não aplicável

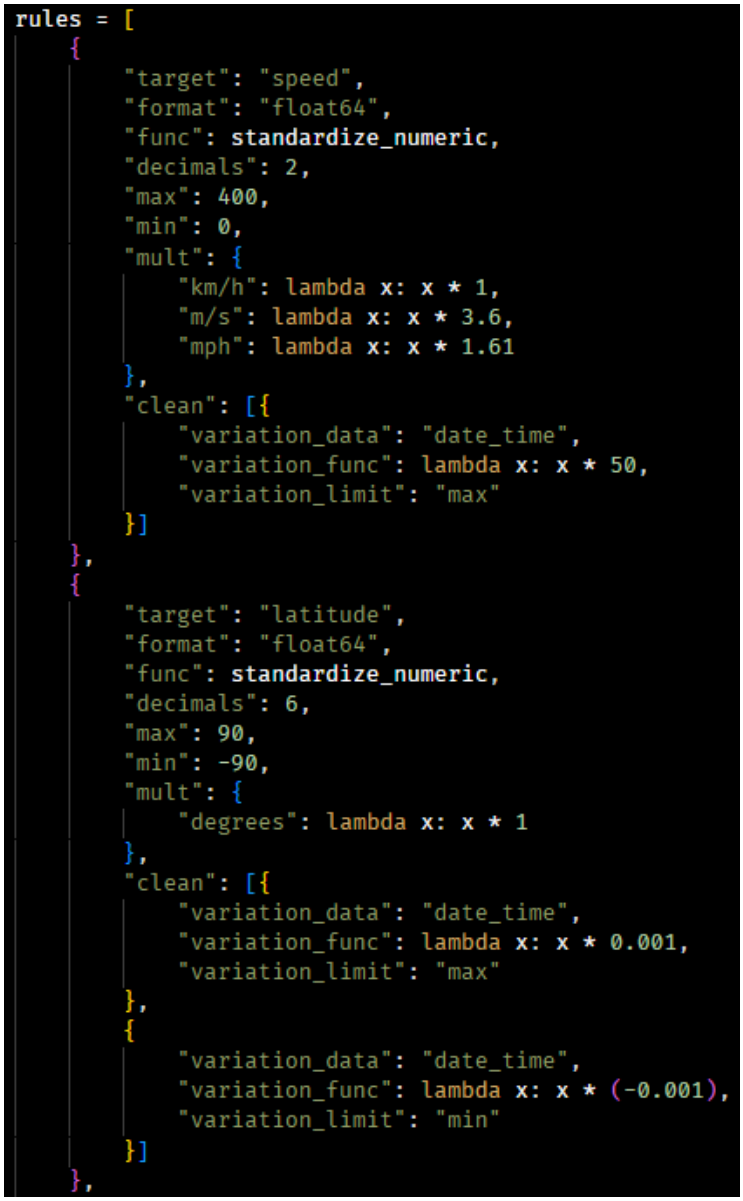
A definição destas características durante a implementação do processo de padronização no Mage foi feita em uma lista de dicionários, como mostrado na Figura 1, em que cada dicionário é responsável por um dado, definido pela chave “*target*”. A chave “*func*” define o tipo deste dado, desta forma definindo também qual

será a função que realizará seu tratamento. As chaves “*format*”, “*decimals*” e “*mult*” são responsáveis por definir respectivamente o formato, precisão e unidade.

O valor da chave “*mult*” é na verdade outro dicionário, que define as transformações de diferentes possíveis unidades a serem encontradas em *datasets* para a unidade padrão. É possível identificar qual é a unidade padrão analisando as funções de transformação, e verificando qual se trata de uma multiplicação por 1. No caso do dado “*speed*”, a unidade padrão é km/h.

As chaves “*max*”, “*min*” e “*clean*” são utilizadas no processo de limpeza e serão explicadas no tópico “limpeza de dados”.

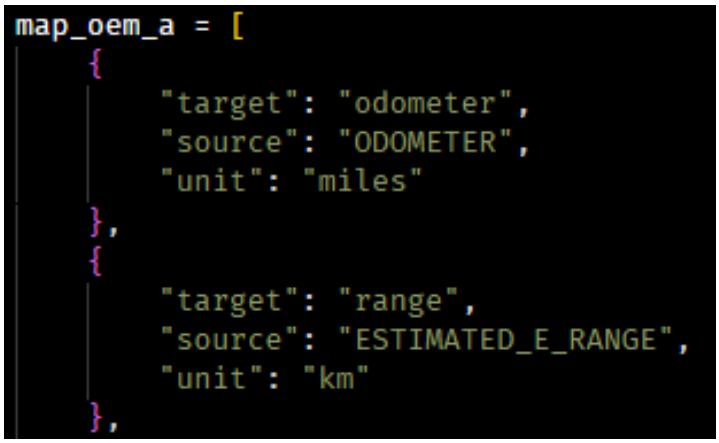
Figura 1 – Dicionário de regras de padronização para dados numéricos



Fonte: Autor

Outra lista de dicionários é utilizada para informar algumas informações a respeito do estado original dos dados, como mostra a Figura 2. Novamente, a chave "*target*" define o dado, para que a função de padronização possa conectar as informações do *dataset* original, com as características desejadas para o *dataset* padronizado. Já a chave "*source*" informa como este dado é nomeado no *dataset* original e a "*unit*" define a unidade do dado original. Essa unidade será utilizada para definir qual transformação definida no dicionário anterior será utilizada.

Figura 2 – Dicionário com a definição de características de dados numéricos do *dataset* original



Fonte: Autor

3.4.2 Categóricos

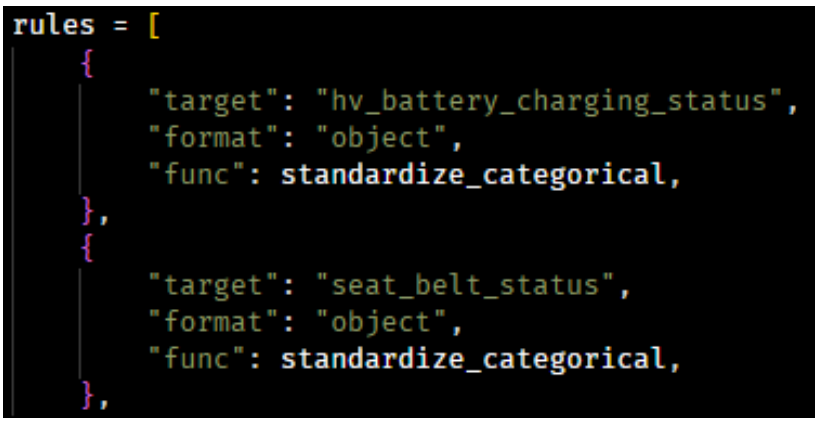
Este tipo possui uma quantidade menor de dados, todos com o formato *object*, e com diferentes "categorias", que definem os possíveis valores deste dado. Suas características são as seguintes:

- Estado de carregamento da bateria HV
  - Nome: *hv_battery_charging_status*
  - Formato: *object*
  - Precisão: não aplicável
  - Unidade: não aplicável
  - Categorias: *charged*, *charging*, *not charging*, *charging error* e *other*
- Estado do cinto de segurança

  - Nome: *seat_belt_status*
  - Formato: *object*
  - Precisão: não aplicável
  - Unidade: não aplicável
  - Categorias: *buckled* e *not buckled*
- Estado da ignição
  - Nome: *ignition_status*
  - Formato: *object*
  - Precisão: não aplicável
  - Unidade: não aplicável
  - Categorias: *ignition on* e *ignition off*
- Evento
  - Nome: *event*
  - Formato: *object*
  - Precisão: não aplicável
  - Unidade: não aplicável
  - Categorias: *periodical, charging status change, ignition on, ignition off, charging plug status change, abrupt acceleration, abrupt brake, direction change, lv battery low voltage* e *Other*

A definição destas características também foi feita em uma lista de dicionários. Porém, neste tipo de dado, apenas as chaves “*target*”, “*format*” e “*func*” são necessárias, como visto na Figura 3.

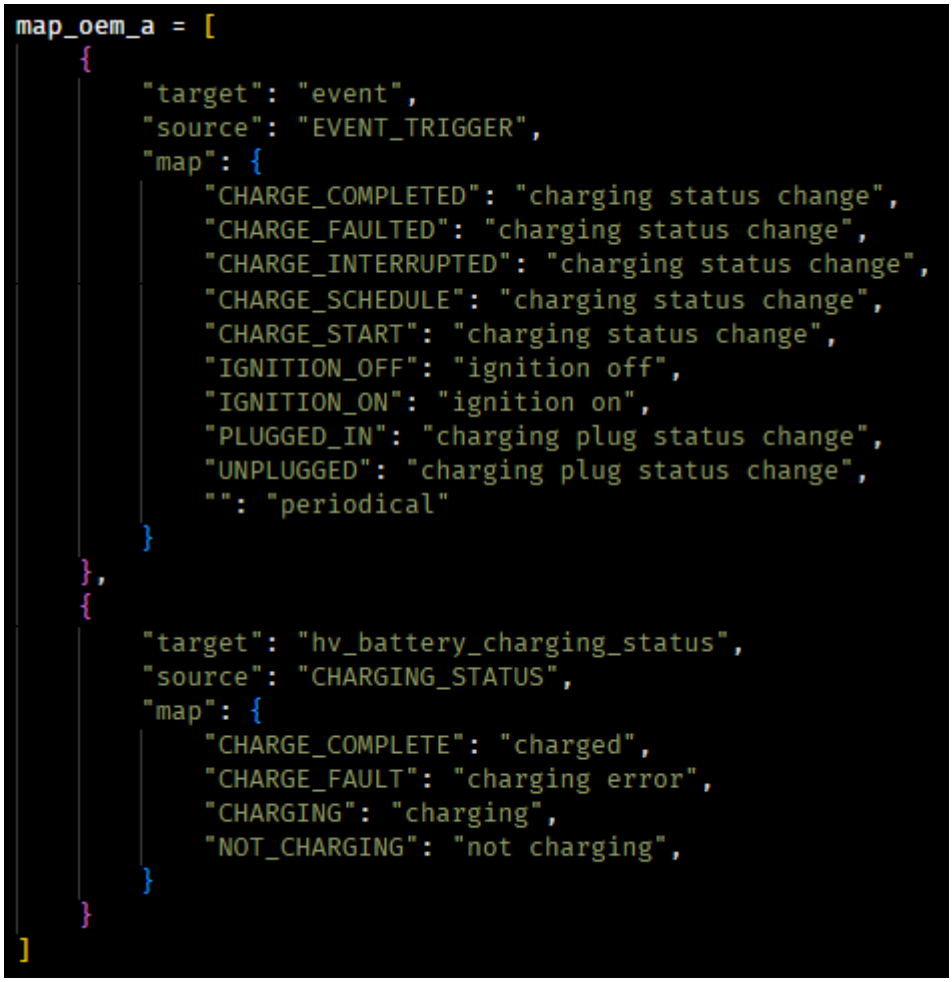
Figura 3 – Dicionário de regras de padronização para dados categóricos



Fonte: Autor

Assim como para o tipo numérico, também existe uma segunda lista de dicionários que é utilizada para definir as características do *dataset* original. O valor da chave “*map*” é um dicionário que define a correlação entre as categorias originais e as definidas como padrão, como visto no exemplo da Figura 4.

Figura 4 – Dicionário com a definição de características de dados categóricos do *dataset* original



Fonte: Autor

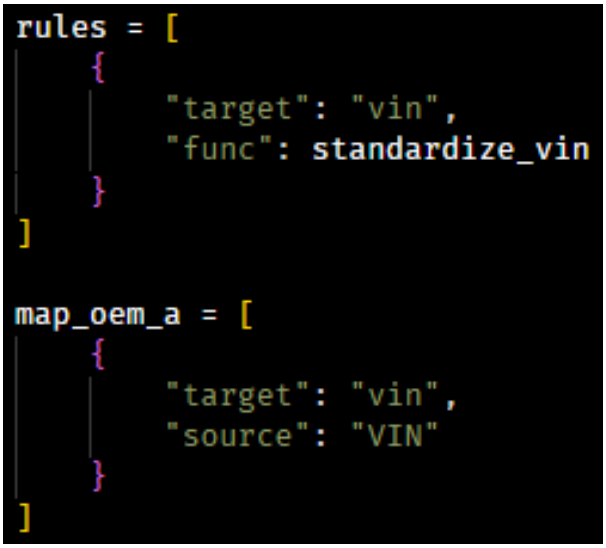
3.4.3 VIN

Este tipo possui apenas um dado, o próprio VIN, por ser a identificação única do veículo, ele recebe um processamento diferente dos demais dados. Suas características são as seguintes:

- Nome: *vin*
- Formato: *object*
- Precisão: não aplicável
- Unidade: não aplicável
- Categorias: não aplicável

A mesma estrutura se repete para este tipo de dado, mas como o VIN será anonimizado, apenas o nome da coluna é padronizado. A Figura 5 mostra os dois dicionários responsáveis pela padronização do VIN.

Figura 5 – Dicionário de regras de padronização e dicionário com a definição de características de dados de VIN do *dataset* original



Fonte: Autor

3.4.4 Data e hora

Este tipo também possui apenas um dado, o *date_time*, que por ser o único dado no formato *datetime*, recebe um processamento diferente dos demais dados. Suas características são as seguintes:

- Nome: *date_time*
- Formato: *datetime*
- Precisão: não aplicável

- Unidade: não aplicável
- Categorias: não aplicável

A mesma estrutura dos demais tipos de dados é utilizada aqui. Porém, desta vez a chave “*unit*” se refere ao formato da data e hora, como mostra na Figura 6, e por isso não possui uma função de transformação, mas assim como anteriormente, na segunda lista, o formato do *dataset* original é informado, e este será transformado para o formato padrão DD/MM/AAAAThh:mm:ss.

Figura 6 – Dicionário de regras de padronização e dicionário com a definição de características de dados de data e hora do *dataset* original

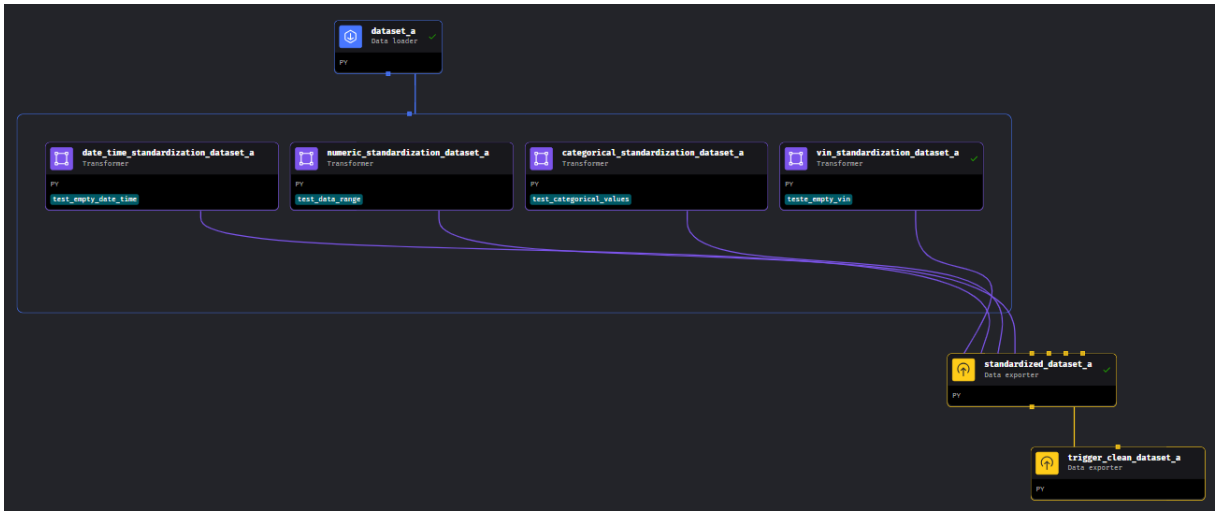
```
rules = [
    {
        "target": "date_time",
        "func": standardize_date_time
    }
]

map_oem_a = [
    {
        "target": "date_time",
        "source": "TIMESTAMP_EVENT",
        "unit": "%d/%m/%Y %H:%M"
    }
]
```

Fonte: Autor

3.4.5 Pipeline de padronização de dados

Com o intuito de preservar a propriedade intelectual da mobway, as funções de padronização que implementam as características definidas anteriormente não serão expostas em detalhes, porém é possível mostrar a estrutura da pipeline responsável por isto. A Figura 7 mostra como ficou a pipeline de padronização de dados implementada no Mage, seguindo uma estrutura ETL, que foi escolhida por ter sua implementação mais simples e rápida, que são características importantes, pois se trata de uma primeira versão que será utilizada para validar o negócio da mobway.

Figura 7 – Estrutura da pipeline de padronização de dados na plataforma Mage



Fonte: Autor

O primeiro bloco chamado “*dataset_a*” é responsável pela extração dos dados. Durante o desenvolvimento do projeto, por questão de praticidade, os dados eram extraídos de um arquivo local, porém com a consolidação da *pipeline*, os dados passaram a ser extraídos de um *bucket* do S3 da mobway, um serviço de armazenamento da AWS.

Os próximos 4 blocos realizam a padronização dos dados de forma paralela, cada um sendo responsável por um dos 4 tipos de dados, com a seguinte correlação:

- Data e hora: “*date_time_standardization_dataset_a*”
- Numéricos: “*numeric_standardization_dataset_a*”
- Categóricos: “*categorical_standardization_dataset_a*”
- VIN: “*vin_standardization_dataset_a*”

Na parte inferior de cada um dos quatro blocos é possível notar textos em verde. Estes representam testes que são realizados nos dados de saída de cada um dos blocos, garantindo que a qualidade dos dados, e que todo o processo de padronização funcionou adequadamente. O funcionamento de cada teste não será explicado aqui, visto que não faz parte do escopo principal deste projeto.

Em seguida, no bloco “*standardized_dataset_a*” é realizado o carregamento dos dados de volta para o S3 da mobway. E por fim, o bloco

“*trigger_clean_dataset_a*” aciona a próxima pipeline, que será responsável pela limpeza dos dados.

3.5 LIMPEZA DE DADOS

A respeito da limpeza de dados, foram definidas três frentes para serem tratadas: valores fora da faixa aceitável, valores com variação acima da aceitável e dados faltantes. Nos três casos, o valor é substituído por um novo gerado pela média do valor anterior e posterior.

Apenas os dados numéricos serão tratados aqui, pois foi considerado que a interpolação de dados categóricos não iria gerar resultados confiáveis, com métodos como substituir o valor pela moda ou pelo mesmo valor do dado anterior.

A seguir são mostrados os valores definidos em relação a faixa e variação mínima e máxima para todos os dados, junto de uma justificativa para sua escolha. Alguns dados não possuem variação mínima ou máxima por não existir um valor confiável para esta definição. Um exemplo de dado que não possui variação mínima ou máxima é o RPM, uma vez que seu valor pode variar de forma muito rápida. Outro exemplo é o nível de combustível, que apesar de ser possível definir um consumo máximo, um valor acima disso pode representar um vazamento, ao invés de um erro no sensoriamento.

- Velocidade
    - Faixa: 0 a 400 km/h
        - A faixa de velocidade foi definida com base no valor máximo que os veículos conectados à mobway podem atingir, considerando uma margem de segurança.
    - Variação máxima: 50 km/h por segundo
        - A variação máxima de velocidade foi definida com base na aceleração máxima que os veículos conectados à mobway podem atingir, considerando uma margem de segurança.
- Odômetro
    - Faixa: 0 a 1000000 km
        - A faixa do odômetro foi definida com base no valor atingido por veículos durante seu ciclo de vida, considerando uma margem de segurança

    - Variação mínima e máxima: 0 km e 0.111 km por segundo
        - A variação mínima do odômetro foi definida pelo fato dele nunca diminuir de valor. A variação máxima do odômetro foi definida com base na distância percorrida por um veículo a 400 km/h, considerando uma margem de segurança
- Latitude
    - Faixa: -90º a 90º
        - A faixa de latitude foi definida com base nos valores limite por definição
    - Variação mínima e máxima: - 0.001º e 0.001º por segundo
        - As variações mínima e máxima de latitude foram definidas com base na distância percorrida por um veículo a 400 km/h, considerando uma trajetória no sentido norte-sul da terra, considerando uma margem de segurança
- Longitude
    - Faixa: -180º a 180º
        - A faixa de longitude foi definida com base nos valores limite por definição
    - Variação mínima e máxima: - 0.001º e 0.001º por segundo
        - As variações mínima e máxima de longitude foram definidas com base na distância percorrida por um veículo a 400 km/h, considerando uma trajetória no sentido leste-oeste da terra, considerando uma margem de segurança
- Orientação
    - Faixa: 0º a 360º
        - A faixa do sentido de orientação foi definida com base nos valores limite por definição
- RPM
    - Faixa: 0 a 10000 rpm
        - A faixa de rpm foi definida com base no valor máximo que os veículos conectados à mobway podem atingir, considerando uma margem de segurança
- Nível de óleo

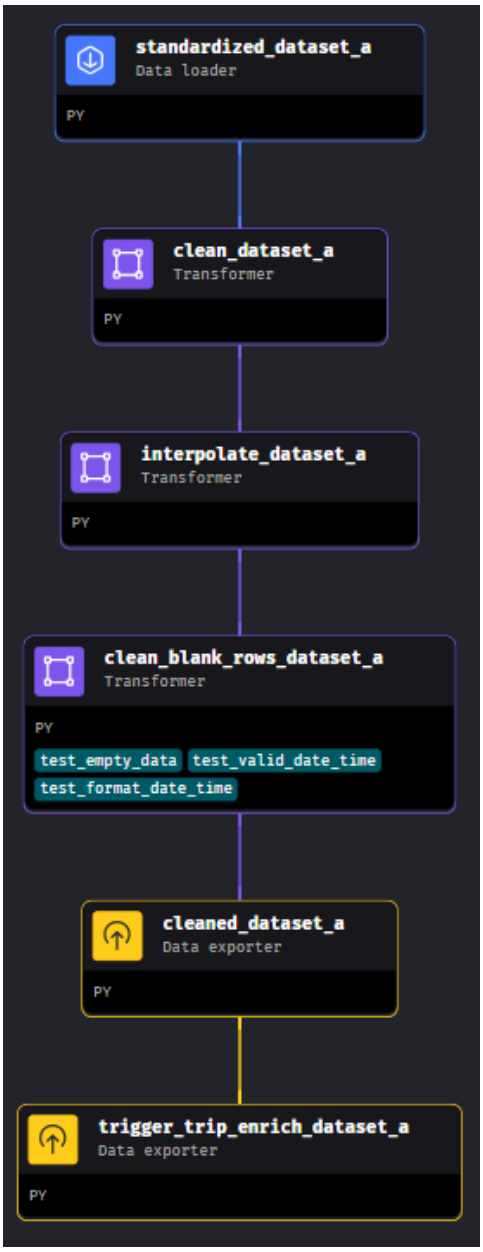
    - Faixa: 0 a 100%
        - A faixa do nível de óleo foi definida com base nos valores limite por definição
- Temperatura do líquido de arrefecimento do motor
    - Faixa: -15°C a 150°C
        - O valor mínimo da temperatura do refrigerador do motor foi definido com base na temperatura do ambiente mínima atingida no Brasil, considerando uma margem de segurança. O valor máximo da temperatura do refrigerador do motor foi definido com base nos valores máximos comumente atingidos, considerando uma margem de segurança
- Nível de combustível
    - Faixa: 0 a 80 L
        - A faixa do nível de combustível foi definida com base no valor máximo que os veículos conectados à mobway podem possuir
- Autonomia
    - Faixa: 0 a 1500 km
        - A faixa de autonomia foi definida com base no valor máximo que os veículos conectados à mobway podem possuir
    - Variação mínima: - (quilômetros rodados * 2 + 5 km)
        - A variação mínima da autonomia foi definida considerando que a mesma deve variar de acordo com o odômetro, considerando uma margem de erro
- Pressão do pneu dianteiro esquerdo
    - Faixa: 14 a 50 psi
        - O valor mínimo da pressão de pneu foi definido com base na pressão atmosférica, considerando uma margem de erro. O valor máximo da pressão de pneu foi definido com base no valor máximo que os veículos conectados à mobway podem atingir, considerando uma margem de segurança
- Pressão do pneu dianteiro direito
    - Faixa: 14 a 50 psi

        - O valor mínimo da pressão de pneu foi definido com base na pressão atmosférica, considerando uma margem de erro. O valor máximo da pressão de pneu foi definido com base no valor máximo que os veículos conectados à mobway podem atingir, considerando uma margem de segurança
- Pressão do pneu traseiro esquerdo
    - Faixa: 14 a 50 psi
        - O valor mínimo da pressão de pneu foi definido com base na pressão atmosférica, considerando uma margem de erro. O valor máximo da pressão de pneu foi definido com base no valor máximo que os veículos conectados à mobway podem atingir, considerando uma margem de segurança
- Pressão do pneu traseiro direito
    - Faixa: 14 a 50 psi
        - O valor mínimo da pressão de pneu foi definido com base na pressão atmosférica, considerando uma margem de erro. O valor máximo da pressão de pneu foi definido com base no valor máximo que os veículos conectados à mobway podem atingir, considerando uma margem de segurança
- Tensão da bateria LV
    - Faixa: 0 a 16 V
        - A faixa de tensão da bateria lv foi definida com base nos valores limite atingidos por uma bateria de 12 V, considerando uma margem de segurança
- Tensão da bateria HV
    - Faixa: 0 a 1000 V
        - A faixa de tensão da bateria hv foi definida com base no valor máximo que os veículos conectados à mobway podem atingir, considerando uma margem de segurança
- Estado de carga da bateria HV
    - Faixa: 0 a 100%
        - A faixa do estado de carga da bateria hv foi definida com base nos valores limite por definição

3.5.1 Pipeline de limpeza de dados

Da mesma forma que na *pipeline* de padronização, a pipeline de limpeza não terá suas funções expostas, porém é possível explicar de forma pouco detalhada o que cada bloco faz. A Figura 8 mostra a estrutura da *pipeline* de limpeza, que é acionada após a finalização da *pipeline* de padronização. A estrutura novamente é ETL, buscando manter a facilidade e rapidez de implementação.

Figura 8 – Estrutura da pipeline de limpeza de dados na plataforma Mage



Fonte: Autor

O primeiro bloco, chamado “*standardized_dataset_a*”, e os dois últimos, “*cleaned_dataset_a*” e “*trigger_trip_enrich_dataset_a*”, têm a função de realizar a extração, carregamento e acionamento da próxima *pipeline.*

Os blocos centrais realizam a limpeza dos dados. O primeiro, chamado “*clean_dataset_a*”, deleta os dados fora da faixa defina anteriormente e com variação acima da máxima ou abaixo da mínima definida. Especificamente para a limpeza dos dados com variação fora do esperado a interpolação é realizada em conjunto, visto que o processo precisa ser iterativo, já que uma interpolação pode gerar novas variações não aceitáveis. Para explicar com clareza este ponto, é necessário dar um exemplo. O exemplo usará dados de odômetro, que possui como possui uma variação mínima de zero quilômetros.

A Tabela 1 mostra um exemplo em que não seria possível realizar a limpeza adequada, caso ela fosse feita separadamente da interpolação. Neste caso, apenas o dado com índice 2 seria identificado como errado, uma vez que apesar o dado com índice 3 ser menor que o de índice 1, ele é maior que o de índice 2, que é a referência para a definição da variação. O resultado seria o mostrado na Tabela 2.

Tabela 2 – Exemplo de conjunto de dados em que a para limpeza de dados com variação fora do esperado não funcionária com a primeira errônea

Índice	**Odômetro**
1	1000 km
2	998 km
3	999 km
4	1003 km
5	1004 km

Tabela 3 – Exemplo de resultado para a limpeza de dados com variação fora do esperado com abordagem errônea

Índice	**Odômetro**
1	1000 km
2	999,5 km
3	999 km
4	1003 km
5	1004 km

Na Tabela 2, observa-se que os dados com índice 2 e 3 continuam errados, uma vez que são menores que o de índice 1. Com a integração da limpeza e

interpolação em um processo iterativo é possível ter o resultado mostrado na Tabela 3. Na primeira iteração da Tabela 3, o resultado como esperado seria o mesmo, porém agora é possível identificar os dois dados como errôneos, já que o dado de índice 2 é menor que o de índice 1 e dado de índice 3 é menor que o de índice 2. A Tabela 4 mostra o resultado, em que os dados de índice 2 e 3 foram corretamente limpos e interpolados a partir dos dados de índice 1 e 4.

Tabela 4 - Exemplo de resultado para a primeira iteração da limpeza de dados com variação fora do esperado com abordagem correta

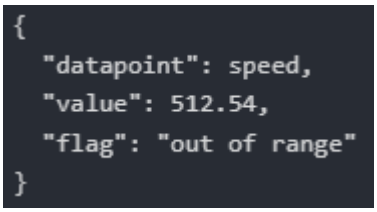
Índice	**Odômetro**
1	1000 km
2	999,5 km
3	999 km
4	1003 km
5	1004 km

Tabela 5 - Exemplo de resultado para a primeira final da limpeza de dados com variação fora do esperado com abordagem correta

Índice	**Odômetro**
1	1000 km
2	1001 km
3	1002 km
4	1003 km
5	1004 km

No próximo bloco, chamado “interpolate_dataset_a”, a interpolação dos demais dados é realizada, isto é, aqueles que estavam fora da faixa definida e os faltantes. Neste bloco, e no anterior, também é gerada uma *flag* para os dados interpolados, avisando o cliente que o dado foi modificado, por ser considerado errôneo. A Figura 9 mostra o formato da *flag* gerada, trata-se de um dicionário. O primeiro valor identifica a respeito de que dado a *flag* se trata. O segundo informa o valor original e o último o motivo pelo qual o dado foi modificado, neste caso, porque o valor estava fora da faixa defina de 0 a 400 km/h.

Figura 9 – Estrutura de *flag* para dados interpolados para o caso de um dado identificado como fora da faixa definida



Fonte: Autor

O último bloco de limpeza, chamado "*clean_blank_rows_dataset_a*" é responsável por deletar as linhas que não possuem informação suficiente. Existem três condições independentes para isso acontecer:

- VIN faltante
- Data e hora faltante
- Todos os demais dados faltantes

3.6 ENRIQUECIMENTO DE DADOS

O enriquecimento de dados foi feito baseado no conceito de viagem. Essa é definida da seguinte forma:

- Possível identificação do início de uma viagem:
  - Veículo é ligado
  - Fim de uma parada para reabastecimento ou carregamento

- Possível identificação do fim de uma viagem:
  - Veículo é desligado
  - Início de uma parada para reabastecimento ou carregamento

- Para identificar estes casos os seguintes dados são utilizados:
  - Evento: *ignition on* -> Veículo é ligado
  - Evento: *ignition off* -> Veículo é desligado
  - Velocidade > 0 -> Fim de uma parada para reabastecimento ou carregamento

- Evento: *charging status change* -> Início de uma parada para carregamento
- Evento: *charging plug status change* -> Início de uma parada para carregamento
- Estado de carregamento da bateria HV: *charging* -> Início de uma parada para carregamento
- Nível de combustível (x) > Nível de combustível (x-1) -> Início de uma parada para reabastecimento

Antes de identificar estes casos, é necessário organizar o *dataset* temporalmente e agrupar os VINs, para isso, foram utilizadas funções básicas da biblioteca Pandas.

Como o início e fim de uma viagem podem ser identificados de diversas formas, é necessário selecionar apenas o primeiro sinal que identifica um início e o primeiro sinal que identifica o fim, para cada viagem. A Tabela 5 mostra um exemplo de como os inícios e fins são identificados. É possível notar que a identificação inicial não permite traçar o início e fim da viagem, para isso é necessário filtrar os dados.

Tabela 6 – Exemplo de conjunto de dados com identificação de viagens antes da aplicação do filtro

Índice	**Identificação de Viagem**
1	Início
2	-
3	Início
4	-
5	Início
6	Fim
7	Fim
8	Início
9	Fim
10	Fim

A Tabela 6 mostra o resultado após os identificadores serem filtrados de forma que apenas o primeiro “Início” e “Fim” de cada viagem fossem selecionados. Agora é possível identificar que uma viagem tem início no índice 1 e fim no índice 6 e uma segunda viagem tem início no índice 8 e fim no índice 10.

Tabela 7 – Exemplo de conjunto de dados com identificação de viagens depois da aplicação do filtro

Índice	Identificação de Viagem
1	Início
2	-
3	-
4	-
5	-
6	Fim
7	-
8	Início
9	-
10	Fim

Após a identificação das viagens, informações sobre elas foram calculadas. Essas informações foram selecionadas de forma a entregar o mínimo valor necessário para o cliente, buscando validar a comercialização de dados enriquecidos. As informações selecionadas foram:

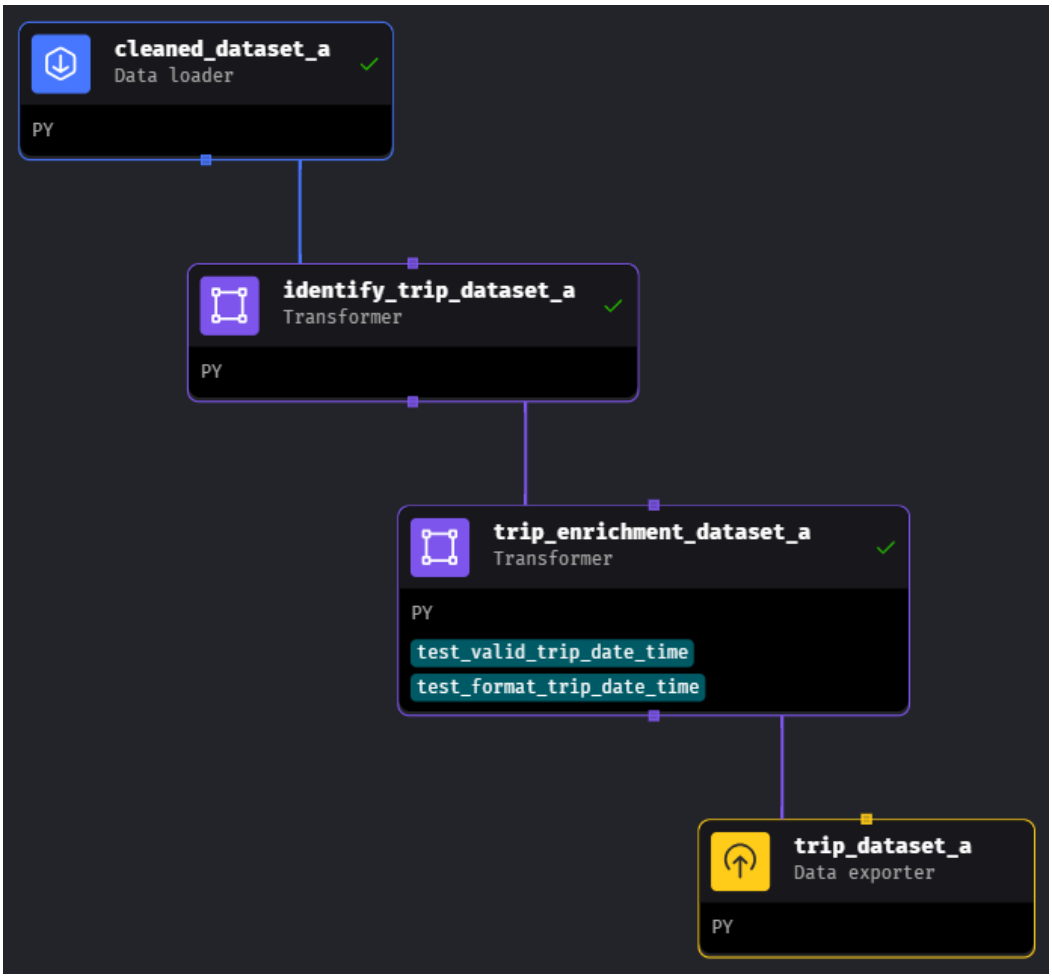
- Data e horário de início
- Data e horário de fim
- Duração
- Velocidade média
- Velocidade máxima
- Odômetro inicial
- Odômetro final
- Distância
- Consumo de combustível
- Latitude inicial
- Latitude final
- Longitude inicial
- Longitude final

As características padrão de todas estas informações como formato, número de casas decimais e unidade foram derivadas das definições feitas para os dados originais.

3.6.1 Pipeline de enriquecimento de dados

Da mesma forma que nas *pipelines* apresentadas anteriormente, esta não terá suas funções expostas, porém é possível explicar de forma pouco detalhada o que cada bloco faz. A Figura 10 mostra a *pipeline* de enriquecimento. O bloco inicial e o final, chamados "*cleaned_dataset_a*" e "*trip_dataset_a*", como padrão são utilizados para extrair e carregar os dados, mantendo então a estrutura ETL. Desta vez não existe um bloco que aciona uma próxima *pipeline*, já que se trata da última.

Figura 10 - Estrutura da pipeline de enriquecimento de dados na plataforma Mage



Fonte: Autor

Os blocos centrais são responsáveis pelo enriquecimento. O primeiro deles, chamado "*identify_trip_dataset_a*" é responsável pela identificação do início e fim de cada viagem. Já o "*trip_enrichment_dataset_a*" possui as funções que irá gerar as informações listadas anteriormente a respeito das viagens.

4 RESULTADOS E DISCUSSÔES

Após a finalização da implementação das *pipelines* foi realizado um processo de otimização do código, tanto em questão de documentação e qualidade de escrita, seguindo princípios do *clean code*, mas principalmente em relação ao desempenho. O primeiro passo foi a identificação de funções que estavam levando bastante tempo para serem executadas e que tinham abertura para otimização. Foi identificado que o bloco “*clean_dataset_a*" estava demandando bastante tempo para ser executado, em relação aos demais blocos. Avaliando as funções separadamente, ficou claro que a limpeza de dados com variação fora da faixa esperada era a responsável.

A função identificada como gargalo utilizava laços do tipo *while* que em muitos casos, gerava iterações desnecessárias para o funcionamento da função. Buscando resolver este problema, foi implementado uma nova condição dinâmica baseada no número de iterações para a permanência no laço, reduzindo então o tempo de processamento. Antes da otimização, testes com um *dataset* de 75 mil linhas e 8 colunas levavam cerca de 10 minutos para rodar as 3 pipelines, e após a otimização o tempo diminuiu para cerca de 7 minutos.

Todo o desenvolvimento foi realizado com o Mage hospedado localmente, buscando tornar a solução comercializável, o último passo foi alterar a hospedagem para um serviço da AWS. O serviço utilizado foi o Elastic Container Service (ECS). Foram realizados diversos testes para entender como o tempo de processamento varia de acordo com a quantidade de linhas e colunas do *dataset*, horário e dia, quantidade de memória e CPUs de máquina e número de *datasets* sendo tratados ao mesmo tempo em paralelo. A seguir será detalhado como o tempo de processamento e custo varia de acordo com estas variáveis individualmente e em conjunto.

4.1 VARIAÇÃO DE LINHAS E COLUNAS DE UM DATASET

Buscando entender a como o tempo de execução varia de acordo com o tamanho do *dataset*, tanto em questão de linhas quanto de colunas, diversos testes em diferentes cenários foram feitos. A Tabela 7 mostra as condições de cada um dos oito cenários de teste que foram realizados, sendo cada um deles executado

diversas vezes e extraído o tempo médio das pipelines individualmente e o tempo médio total.

Tabela 8 – Tempo de execução para diferentes cenários variando o *dataset*, número de linhas e de colunas

Dataset	**Linhas**	**Colunas**	**Padronização**	**Limpeza**	**Enriquecimento**	**Total**
A	75k	8	31s	5m18s	1m21s	7m10s
A	75k	4	19s	3m52s	1m20s	5m32s
A	37,5k	4	20s	3m15s	1m01s	4m36s
H	50k	10	30s	3m10s	Não aplicável	3m40s
H	100k	10	50s	6m29s	Não aplicável	7m19s
H	200k	10	1m50s	16m21s	Não aplicável	18m11s
H	400k	10	3m50s	40m40s	Não aplicável	44m30s
H	750k	10	10m	Erro	Não aplicável	Erro

Dois *datasets* diferentes foram utilizados, o *dataset* A foi escolhido para os testes, pois foi o *dataset* utilizado para testes durante o desenvolvimento, desta forma já existia uma referência para comparação. Já o *dataset* H, apesar de não possuir alguns dados necessários para a aplicação do enriquecimento, não tornando possível então executá-lo, é o maior *dataset* disponível, sendo possível então testar o tempo de execução com um número grande de linhas. Quando o teste foi aplicado para o número total de linhas (750 mil) deste *dataset*, ocorreu um erro pois a máquina não foi capaz de processar esta grande quantidade de dados. A máquina utilizada para todos os testes possuía dois giga bytes de memória e uma unidade central de processamento (CPU).

Analisando os dados, foi possível gerar uma função para estimar o tempo de processamento de acordo com o número de linhas e colunas. A função gerada com base nos dados do *dataset* A está mostrada na Equação 1.

$$Ta = \left(\left(\frac{7\text{m}10\text{s}}{5\text{m}32\text{s}}\right)^{log_2\left(\frac{\text{C}}{8}\right)}\right) * \left(\left(\frac{7\text{m}10\text{s}}{4\text{m}36\text{s}}\right)^{log_2\left(\frac{\text{L}}{75000}\right)}\right) * 7\text{m}10s \quad (1)$$

Onde:

Ta é o tempo de processamento do *dataset* A (segundos);

C é o número de colunas (adimensional);

L é o número de linhas (adimensional).

Para desenvolver esta função, o primeiro passo foi criar uma hipótese sobre seu comportamento. A hipótese criada, e que posteriormente se provou verdadeira, para os cenários analisados, foi que o tempo varia com um fator fixo, quando o tamanho da dataset varia com um mesmo fator. Ou seja, quando o tamanho do dataset dobra, o tempo de processamento, independentemente do tamanho inicial do dataset, aumentará em 30%, por exemplo.

O primeiro fator numérico da Equação 1 é a divisão do tempo de processamento para oito colunas por quatro colunas. Ele está elevado ao logaritmo na base dois do número de colunas dividido por oito, que é o número de colunas referente. O mesmo raciocínio é aplicado agora em relação ao número de linhas, tendo 75 mil como valor de referência. O último valor é o tempo de processamento para o cenário de referência, ou seja, oito colunas e 75 mil linhas.

Já a função gerada com base nos dados do *dataset* H está mostrada na Equação 2.

$$Th = \left(\left(\frac{7\text{m}10\text{s}}{5\text{m}32\text{s}}\right)^{log_2\left(\frac{\text{C}}{10}\right)}\right) * \left(\left(\frac{7\text{m}19\text{s}}{3\text{m}40\text{s}}\right)^{log_2\left(\frac{\text{L}}{50000}\right)}\right) * 3\text{m}40\text{s} * 1{,}24 \quad (2)$$

Onde:
Th é o tempo de processamento do *dataset* H (segundos);
C é o número de colunas (adimensional);
L é o número de linhas (adimensional).

A estrutura da Equação 2 é semelhante à da Equação, porém os valores são diferentes, já que os cenários de teste são diferentes. A única diferença é o último fator de 1,24 que é utilizado para compensar a ausência da aplicação da *pipeline* de enriquecimento.

Para obtenção de uma função geral que possa aproximar o tempo de processamento para qualquer *dataset*, a média das duas funções foi realizada, resultando na Equação 3.

$$Tm = \frac{Ta+Th}{2} \quad (3)$$

Onde:
Tm é o tempo de processamento médio (segundos);
Ta é o tempo de processamento do *dataset* A (segundos);
Th é o tempo de processamento do *dataset* H (segundos).

A Tabela 8 mostra a comparação do tempo executado para o tempo calculado para cada uma das 3 funções. Como já citado, o tempo executado para os cenários com o *dataset* H (4 a 7) foi corrigido pelo fator 1,24 para compensar a ausência da aplicação da *pipeline* de enriquecimento. As funções específicas para cada *dataset* foram aplicadas somente para os cenários com seus respectivos *datasets*.

Tabela 9 – Comparação do tempo executado e calculado pelas fórmulas Ta, Th e Tm para diferentes cenários variando o *dataset*, número de linhas e de colunas

Cenário	**Tempo executado**	**Ta**	**Th**	**Tm**
1	7m10s	7m10s		7m13s
2	5m32s	5m32s		5m34s
3	4m36s	4m36s		3m11s
4	4m33s		5m17s	5m39s
5	9m04s		10m32s	9m57s
6	22m33s		21m01s	17m47s
7	55m11s		41m56s	32m17s

É possível notar que a equação Ta foi capaz de calcular sem erros para todos os cenários, isso devido ao fato de ela ter sido criada utilizando os mesmos três cenários como referência. Já a equação Th, por ter que realizar algumas estimativas, obteve erros que variam entre 16,2% e 24,0%. A função Tm obteve bons resultados em alguns cenários, como no cenário 1 que apresentou um erro de 0,7%. Porém para outros casos obteve erros significativo, como o cenário 7 que apresentou um erro de -41,5%.

4.2 VARIAÇÃO DA MEMÓRIA E NÚMERO DE CPUS

Mais testes em diferentes cenários foram realizados para entender como o tempo de execução varia de acordo com a memória e quantidade de CPUs da máquina. A Tabela 9 mostra quatro combinações diferentes de memória e

quantidade de CPUs, já as demais variáveis se mantem, o *dataset* A foi utilizado, com 75 mil linhas e oito colunas.

Tabela 10 - Tempo de execução para diferentes cenários variando o número de CPUs e a quantidade de memória

CPUs	Memória	Padronização	Limpeza	Enriquecimento	Total
1	2	31s	5m18s	1m21s	7m10s
1	4	30s	5m20s	1m20s	7m10s
2	4	30s	4m40s	1m29s	6m39s
4	8	20s	4m29s	1m19s	6m09s

Pode-se notar que o desempenho não é limitado pela memória, uma vez que o aumento exclusivo dela não resulta na redução do tempo de processamento. Portanto, nos cenários 1, 3 e 4, a quantidade de memória utilizada é a mínima disponível para a respectiva quantidade de CPUs.

A Figura 11 mostra as possíveis configurações de máquina para o serviço ECS.

Figura 11 – Tamanhos de máquinas disponíveis no serviço Amazon Elastic Container Service da AWS

CPU value	Memory value	Operating systems supported for AWS Fargate
256 (.25 vCPU)	512 MiB, 1 GB, 2 GB	Linux
512 (.5 vCPU)	1 GB, 2 GB, 3 GB, 4 GB	Linux
1024 (1 vCPU)	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB	Linux, Windows
2048 (2 vCPU)	Between 4 GB and 16 GB in 1 GB increments	Linux, Windows
4096 (4 vCPU)	Between 8 GB and 30 GB in 1 GB increments	Linux, Windows
8192 (8 vCPU) **Note** This option requires Linux platform `1.4.0` or later.	Between 16 GB and 60 GB in 4 GB increments	Linux
16384 (16vCPU) **Note** This option requires Linux platform `1.4.0` or later.	Between 32 GB and 120 GB in 8 GB increments	Linux

Fonte: AWS (2024b)

Entretanto, com o aumento do número de CPUs o tempo de processamento é otimizado. Foi desenvolvida uma nova equação (Equação 4) para entender como o tempo varia com o aumento de CPUs para valores não testados e com diferentes configurações das demais variáveis. A Equação 4 utiliza a mesma estrutura das

desenvolvidas anteriormente, porém agora a variável é a quantidade de CPUs e a referência é teste com uma CPU.

$$\mathrm{Tc} = \left(\left(\frac{6\mathrm{m}39\mathrm{s}}{7\mathrm{m}10\mathrm{s}}\right)^{log_2\left(\frac{\mathrm{CPU}}{1}\right)}\right) * 7\mathrm{m}10\mathrm{s} \quad (4)$$

Onde:
Tc é o tempo de processamento (segundos);
CPU é o número de CPÚs (adimensional).

A Tabela 10 mostra que um resultado bastante satisfatório, uma vez que o erro para o cenário com quatro CPUs é de 0,27%. Já o erro para os dois primeiros cenários é nulo, já que eles foram utilizados como referência para a função. Além disso, a Tabela 10 traz a estimativa de tempo de processamento para oito e dezesseis CPUs, cenários não testados.

Tabela 11 - Comparação do tempo executado e calculado pela fórmula Tc para diferentes cenários variando o número de CPUs

CPUs	**Tempo executado**	**Tc**
1	7m10s	7m10s
2	6m39s	6m39s
4	6m09s	6m10s
8		5m44s
16		5m19s

Analisando a Tabela 10, mostra-se que, entre o primeiro e o último cenário houve uma redução de tempo de apenas 25,8%. Esse resultado sugere que, com o crescimento da mobway e a necessidade de processamento de dados de milhões de veículos simultaneamente, seria necessárias grandes otimizações. Isso pode incluir a alteração da linguagem e do serviço de nuvem utilizados, uma vez que a melhor máquina disponível no ECS é a com 16 CPUs.

4.3 VARIAÇÃO DO NÚMERO DE DATASETS EM PARALELO

Outros cenários foram testados para entender como o tempo de execução varia de acordo com a quantidade de *datasets* sendo processados em paralelo. A

Tabela 11 mostra nove combinações diferentes de quantidade de *datsets* em paralelo e quantidade de CPUs. Já as demais variáveis se mantem: utilizou-se o *dataset* A com 75 mil linhas e oito colunas, e a memória foi utilizada a mínima disponível em cada tipo de máquina.

Tabela 12 - Tempo de execução para diferentes cenários variando o número de CPUs e a quantidade de *datasets* em tratados em paralelo

CPUs	*Datasets*	Padronização	Limpeza	Enriquecimento	Total	Total/*datasets*
1	1	31s	5m18s	1m21s	7m10s	7m10s
2	1	30s	4m40s	1m29s	6m39s	6m39s
2	2	41s	7m25s	2m	10m06s	5m03s
2	3	51s	11m25s	2m50s	15m06s	5m02s
4	1	20s	4m29s	1m19s	6m09s	6m09s
4	2	25s	4m39s	1m20s	6m24s	3m12s
4	3	31s	5m56s	1m36s	8m03s	2m41s
4	4	40s	7m14s	1m44s	9m38s	2m25s
4	5	49s	8m35s	2m10s	11m34s	2m19s

Apesar de um tempo aumentar consideravelmente com o aumento do número de *datasets* sendo processados, a relação entre o tempo total e o número de *datasets* diminuiu, ou seja, para uma mesma quantidade de dados sendo processados, o tempo necessário para o processamento diminuiu.

Outra conclusão é que a relação entre o tempo e o número de *datasets* diminui de forma satisfatória, contanto que a quantidade não ultrapasse o número de CPUs. Isso ocorre porque uma única CPU precisa processar paralelamente dois *datasets*. No entanto, é importante destacar que o código não foi desenvolvido considerando esse tipo de paralelização, o que resulta em uma eficiência reduzida.

A Equação 5 foi desenvolvida para analisar o tempo de processamento nestes cenários. Embora sua estrutura seja diferente desta vez, ela guarda semelhanças, pois a redução de tempo varia em proporções distintas para diferentes quantidades de CPUs.

$$Td = \left(1 + 1{,}2 * 0{,}38^{log_2\left(\frac{\mathrm{CPU}}{1}\right)}\right)^{\mathrm{D}-1} * 7\mathrm{m}10\mathrm{s} \quad (5)$$

Onde:

Td é o tempo de processamento (segundos);

CPU é o número de CPUs (adimensional);

D é o número de *datasets* (adimensional).

A Tabela 12 mostra que a equação obteve sucesso em calcular os tempos de execução com baixos erros, sendo o cenário com quatro CPUs e dois *datasets* em paralelo o que apresentou maior erro, de 13%.

Tabela 13 - Comparação do tempo executado e calculado pela fórmula Td para diferentes cenários variando o número de CPUs

CPUs	***Datasets***	**Tempo executado**	**Td**
1	1	7m10s	7m10s
2	1	6m39s	6m39s
2	2	10m06s	9m41s
2	3	15m06s	14m06s
4	1	6m09s	6m10s
4	2	6m24s	7m14s
4	3	8m03s	8m30s
4	4	9m38s	9m58s
4	5	11m34s	11m42s

4.4 VARIAÇÃO DO HORÁRIO E DIA DA SEMANA

Desta vez, todos os testes foram realizados com as mesmas variáveis. O *dataset* A foi utilizado, contendo 75 mil linhas e oito colunas, em uma máquina equipada com um CPU e 2GB de memória, processando apenas um *dataset* em paralelo. A análise foi realizada em relação ao horário em que a pipeline foi executada, com o objetivo de determinar se isso interfere no tempo de processamento.

A Tabela 13 mostra a média geral dos tempos de execução, o desvio padrão e as médias para cada um dos quatro períodos do dia: manhã, tarde, noite e madrugada. Estes períodos foram definidos da seguinte forma:

- Manhã: Entre 6h00 e 11h59
- Tarde: Entre 12h00 e 17h59
- Noite: Entre 18h00 e 23h59
- Madrugada: Entre 00h00 e 5h59

Tabela 14 - Tempo de execução para diferentes períodos do dia

Dado	Padronização	Limpeza	Enriquecimento	Total
Média	31s	5m18s	1m21s	7m10s
Desvio padrão	1s	12s	4s	12s
Média de manhã	31s	5m16s	1m22s	7m09s
Média de tarde	31s	5m21s	1m21s	7m13s
Média de noite	31s	5m21s	1m21s	7m12s
Média de madrugada	30s	5m14s	1m21s	7m05s

Ao analisar a variação do tempo de execução entre diferentes períodos do dia e a média geral, percebe-se que, embora os dados indiquem uma pequena diminuição do tempo de execução durante a madrugada e um aumento durante a tarde, a diferença é mínima. Durante a madrugada, o tempo de execução é 1,2% mais rápido em média, enquanto durante a tarde é apenas 0,7% mais lento. Essa diferença é quase insignificante.

A mesma análise foi realizada em relação aos dias da semana, como pode ser visto na Tabela 14. Da mesma forma, as variações foram pequenas. O dia mais lento foi a sexta-feira, que apresentou um tempo de execução 1,2% mais lento que a média geral, enquanto o dia mais rápido foi o domingo, sendo 1,2% mais rápido que a média.

Tabela 15 - Tempo de execução para diferentes dias da semana

Dado	Padronização	Limpeza	Enriquecimento	Total
Média	31s	5m18s	1m21s	7m10s
Desvio padrão	1s	12s	4s	12s
Média da segunda	30s	5m22s	1m20s	7m12s
Média do domingo	31s	5m12s	1m23s	7m05s
Média do sábado	31s	5m20s	1m21s	7m11s
Média da sexta	31s	5m23s	1m21s	7m15s

Por apresentar variações baixas em relação ao dia e horário, não foi desenvolvida uma equação para analisar o tempo de execução em relação a estas variáveis.

4.5 EQUAÇÃO GERAL

Após gerar as equações que avaliam a interferência das variáveis apresentadas anteriormente separadamente, é possível integrar essas fórmulas de forma a avaliar o tempo de execução para qualquer cenário. A Equação 6 pode ser

construída pela multiplicação das três funções desenvolvidas anteriormente, divididas pelo tempo de execução de referência de 7m10s. O resultado é, então, multiplicado novamente pelo tempo de referência.

$$T = \left(\frac{\mathrm{Tm}}{7\mathrm{m10s}}\right) * \left(\frac{\mathrm{Tc}}{7\mathrm{m10s}}\right) * \left(\frac{\mathrm{Td}}{7\mathrm{m10s}}\right) * 7\mathrm{m10s} \quad (6)$$

Onde:
T é o tempo de execução (segundos);
Tm é o tempo de processamento médio (segundos);
Tc é o tempo de processamento calculado pela Equação 4 (segundos);
Td é o tempo de processamento calculado pela Equação 5 (segundos).

A Tabela 15 mostra a comparação entre o tempo de execução e o tempo calculado para diversos cenários variando todas as variáveis. A função obteve resultado satisfatórios para a maioria dos casos, porém o último cenário apresentou um erro de -49,5%, o que pode indicar que a função não funciona bem para *datasets* com tamanhos muito menores ou muito maiores que o de referência.

Tabela 16 - Comparação do tempo executado e calculado pela fórmula T para diferentes cenários variando o *dataset*, número de linhas e colunas, quantidade de CPUs e *datasets* sendo tratados em paralelo

Dataset	**Linhas**	**Colunas**	**CPUs**	***Datasets***	**Tempo executado**	**T**
A	75k	8	1	1	7m10s	7m13s
A	37,5k	8	1	1	4m36s	4m08s
A	37,5k	8	1	2	10m31s	9m05s
A	75k	4	1	2	11m29s	12m15s
A	75k	8	4	4	9m38s	10m03s
H	50k	10	4	1	4m33s	4m52s
H	100k	10	4	1	9m04s	8m34s
H	400k	10	4	1	55m11s	27m49s

4.6 ESTABILIDADE

Os testes apresentados anteriormente, além de serem importantes para avaliar o desempenho das *pipelines* em diversos cenários, também contribuíram para avaliar a estabilidade. Mais de 100 testes em cada uma das três *pipelines* foram realizados ao longo de algumas semanas, inclusive automatizações foram

utilizadas para iniciar o processamento uma vez a cada hora ao longo de todo o dia. Excluindo os erros causados por falha humana ao adaptar o código para trocar as variáveis apresentadas anteriormente, apenas um erro ocorreu. Por algum problema interno ao Mage, alguns dos blocos das *pipelines* começou a ser executado repetidamente, sem que o próximo bloco fosse acionado, impedindo que o processamento fosse finalizado. Por se tratar de uma primeira versão, e apresentar uma taxa de erros de menos de 1%, é possível considerar que a estabilidade do sistema é satisfatória. Além disso, todo o sistema possui logs, e diversos testes e alertas que enviam notificações, permitindo a identificação e correção dos possíveis problemas de forma rápida.

5 CONCLUSÃO

Os resultados desse estudo mostram que foi possível desenvolver *pipelines* de padronização, limpeza e enriquecimento implementando todos as funcionalidades e requisitos definidos inicialmente, concluindo que o sistema pode ser utilizado como uma primeira versão comercializável pela mobway, para validar a sua proposta de valor aos seus clientes. Durante o desenvolvimento, também foi possível identificar novas implementações que podem aumentar o valor entregue ao cliente, como:

- Utilização de inteligência artificial para o enriquecimento de dados, permitindo gerar informações como:
    - Perfil de condução do motorista em relação a agressividade
    - Perfil de condução do motorista em relação ao consumo de combustível
    - Perfil de condução do motorista em relação à pegada ecológica
    - Manutenção preditiva
    - Nível de conservação do veículo
    - Identificação de furto de veículo
    - Identificação de mudanças no perfil de condução possivelmente causadas por problemas de saúde, cansaço ou uso de drogas.
- Utilização de especificações técnicas dos veículos para parametrizar a limpeza de dados, sendo possível alterar da velocidade máxima de 400 km/h, para a velocidade máxima de fato do veículo, por exemplo.
- Criação de alertas parametrizáveis pelo cliente para, por exemplo, identificar com mais facilidade quando um motorista está acima da velocidade máxima definida como segura pela empresa.

A respeito da performance, os resultados foram satisfatórios considerando o cenário atual da mobway, em que os dados de apenas dezenas ou centenas de veículos serão tratados. Porém, para viabilizar a escalabilidade da empresa, otimizações terão que ser feitas para que milhares ou milhões de veículos possam ter seus dados tratados, sem que o tempo de processamento se torne um gargalo. A otimização pode acontecer em diversas frentes, como a otimização individual das fórmulas, adaptação do código para outras linguagens e tecnologias mais eficientes,

como SQL ou dbt, alteração do orquestrador Mage ou até a alteração do serviço de hospedagem da AWS, buscando máquinas com melhor performance.

Ainda sobre a performance, foi possível caracterizar com exatidão satisfatória o tempo de processamento em diversos cenários variando parâmetros como quantidade de linhas e colunas, tipo de máquina utilizada e quantidade de pipelines rodando em paralelo. Isso é importante para entender as limitações do sistema, e traçar planos de otimização permitindo a escalabilidade da empresa.

Se tratando da estabilidade, os resultados novamente foram satisfatórios, apresentando uma taxa de erros, não causada por falha humana, abaixo de 1%. Além disso, a identificação e correção de erros pode ser feita de forma bastante eficiente devido aos *logs*, testes e alertas implementados nas *pipelines*.

REFERÊNCIAS

AWS – AMAZON WEB SERVICES. **Qual é a diferença entre ETL e ELT?**. Disponível em: https://aws.amazon.com/pt/compare/the-difference-between-etl-and-elt/. Acesso em: 31 out. 2023.

AWS – AMAZON WEB SERVICES. **O que é limpeza de dados?**. Disponível em: https://aws.amazon.com/pt/what-is/data-cleansing/. Acesso em: 12 jan. 2024a.

AWS – AMAZON WEB SERVICES. **Task definition parameters**. Disponível em: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definition_parameters.html#task_size. Acesso em: 19 jan. 2024b.

BERTONCELLO, Michele; MARTENS, Christopher; SCHNEIDERBAUER, Tobias; ZEDELIUS, Kilian. **Corporate business building to unlock value in automotive connectivity**. 2023. Disponível em: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/corporate-business-building-to-unlock-value-in-automotive-connectivity#/. Acesso em: 12 nov. 2023.

DENSMORE, James. **Data pipelines pocket reference**. O'Reilly Media, 2021.

MARTIN, Robert. **Clean Code: A Handbook of Agile Software Craftsmanship**. Robert C. Martin Series, 2008.

IBGE – INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. Censo Brasileiro de 2022. Rio de Janeiro: IBGE, 2023.

SUGAYAMA, Ricardo; NEGRELLI, Evaldir. **Connected vehicle on the way of Industry 4.0**. 2015. 16 f. Monografia (Especialização) - Curso de Especialização em Engenharia Automotiva, Universidade Tecnológica Federal do Paraná, Curitiba, 2015.