
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO


Luiz Felipe Ribeiro de Arruda

# Uso da ciência de dados para estudo de falhas e fraudes dos Abastecimentos de Postos de Gasolina

Recife

2019

Luiz Felipe Ribeiro de Arruda

# Uso da ciência de dados para estudo de falhas e fraudes dos Abastecimentos de Postos de Gasolina

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Estatística e Informática

Curso de Bacharelado em Sistemas de Informação

Orientador: Gabriel Alves

Coorientador: Ana Roullier

Recife

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

A779u Arruda, Luiz Felipe Ribeiro
Uso da ciência de dados para estudo de falhas e fraudes dos Abastecimentos de Postos de Gasolina / Luiz Felipe Ribeiro Arruda. - 2019.
56 f. : il.

Orientador: Gabriel Alves de Albuquerque Junior.
Coorientadora: Ana Roullier.
Inclui referências.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Sistemas da Informação, Recife, 2019.

1. Ciência de dados. 2. Aprendizado de máquina. 3. Posto de gasolina. I. Junior, Gabriel Alves de Albuquerque, orient. II. Roullier, Ana, coorient. III. Título

CDD 004

LUIZ FELIPE RIBEIRO DE ARRUDA

**Uso da ciência de dados para estudo de falhas e fraudes dos Abastecimentos de Postos de Gasolina**

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Recife, 18 de Novembro de 2020.

BANCA EXAMINADORA

_____

Prof. Cícero Garrozi
Departamento de Computação
Universidade Federal Rural de Pernambuco

_____

Prof. Gabriel Alves
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco

*A Deus…*

# Agradecimentos

Agradeço aos meus pais, José Luiz Arruda e Sinmônia Arruda, minha irmã Gabriela Arruda, por sempre me forçarem ao máximo do meu potêncial.

À minha esposa, Isabelle Lins, que me incentiva a melhorar cada dia mais.

Ao meu orientador Gabriel Alves, por ter me ajudado em vários momentos ao longo da minha caminhada de graduação.

Agradeço à Universidade Federal Rural de Pernambuco, por ter sido um ambiente que me proporcionou a estrutura necessária para construir minha base profissional.

Agradeço à faculdade Dublin Business School, que foi minha faculdade por quase dois anos, e que tive o prazer de evoluir como cidadão.

Agradeço às empresas WG2, Microsffer, Orpak e Gilbarco, por terem me acolhido como funcionário e me deram condições de crescer no mercado de trabalho e de me tornar um grande profissional da área de TI.

E em especial aos meus padrinhos, Mauro e Tereza, que foram os primeiros a celebrarem a jornada de BSI comigo.

*“O que sabemos é uma gota; o que ignoramos é um oceano.”*
*(Isaac Newton)*

# Resumo

Nos dias de hoje, se faz necessário como uma prática essencial do empreendedorismo, um estudo descritivo das vendas realizadas pelas empresas levando em consideração fatores como localização, horário, fidelização do cliente e outros. Tal estudo, é necessário para compreender os padrões envolvidos nos volumes de vendas, bem como, em alguns casos, mensurar a produtividade de vendedores e criar uma métrica de produtividade baseado nos dados obtidos. Assim como toda empresa, os postos de gasolinas também gerenciam as suas vendas fazendo projeções, melhorias e estratégia de vendas. De acordo com a lei federal nº 9.9562 , os postos de gasolinas são obrigados a terem técnicos responsáveis para realizar os abastecimentos, comumente chamados de frentistas. Os frentistas são tratados nos postos como vendedores, e então existe todo um controle de vendas para eles. Também é regulamentado em lei, para postos de combustíveis, o uso de automação comercial, antes pelo regulamento da SEFAZ PAF-ECF e agora pela NFC-e. A automação comercial usada nos postos de combustíveis, trabalha em conjunto com as bombas de combustíveis e os frentistas. As bombas de combustíveis, nos postos, são os geradores de dados para o posto. É através das bombas que os consumidores recebem o que compram, e é o papel da automação registrar os dados oriundos dos abastecimentos e associar o frentista para o seu respectivo abastecimento. Esses dados são enviados para sistemas comerciais que fazem a tratativa do gerenciamento de venda. Este estudo tem por finalidade usar os conceitos da ciência de dados e *machine learning*, para identificar falhas, possíveis fraudes, automatizar a análise dos logs e extrair dados relevantes para a análise de abastecimentos através de logs. Por isso, foram criados algoritmos de identificação de falhas e fraudes, que alimentam uma tabela de dados, e posteriormente, é criada uma *machine learning*, alimentada por ess tabela, para que seja possível prever futuros abastecimentos com erro. Após a aplicação dos treinos e testes, a máquina teve uma precisão (precision) de 96% de acerto das previsões de falhas nos abastecimentos.

**Palavras-chave**: Python, ciência de dados, Análise de dados, Machine Learning, Automação, Posto de Gasolina.

# Abstract

Nowadays, it is necessary as an essential practice of entrepreneurship, a descriptive study of the sales made by the companies taking into consideration factors such as location, time, customer loyalty and others. Such a study is necessary to understand the patterns involved in sales volumes, as well as, in some cases, to measure salesman productivity and to create a productivity metric based on the data obtained. Like every company, gas stations also manage their sales by making projections, improvements and sales strategy. According to federal law No. 9,956, gas stations are required to have technicians responsible for making supplies, commonly called gas station attendants. Gas station attendants are treated as salespeople, so there is a complete sales control for them. It is also regulated for gas stations, the use of commercial automation, before by the regulation of SEFAZ PAF-ECF and now by NFC-e. Commercial automation used at gas stations works in conjunction with fuel pumps and gas station attendants. The fuel pumps at the stations are the data generators for the station. It is through the pumps that consumers receive what they buy, and it is the role of automation to record the data from the supplies and to associate the attendant with their supply. This data is sent to commercial ERPs that deal with sales management. The purpose of this study is to use emachine learning data science concepts to identify failures, possible failures, automate log analysis and extract relevant data for log analysis. For this reason, fault and fraud identification algorithms were created, which feed a data table, and later, a machine learning, fed by this table, is created, so that future supplies with error can be predicted. After the drills and tests were applied, the machine had a 96% accuracy of prediction

**Keywords**: Python, Data Mining, Data Analysis, Machine Learning, Automation, Gas Station.

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| SEFAZ | Secretária da Fazenda |
| API | *Aplication Programming Interface* |
| SigPosto | Sistema Integrado de Postos |
| ANP | Agência Nacional de Petróleo, Biocombustíveis e Gás Natural |
| NFC-e | Nota Fiscal do Consumidor Eletrônica |
| RFID | Identificação por radiofrequência |
| DLL | *Dynamic Link Library* |
| SI | Sistema de Informação |
| BI | *Business Intelligence* |
| LMC | Livro de Movimentação de Combustível |
| DLL | *Dynamic Link Library* |
| TCP | Transmission Control Protocol |
| IP | Protocolo da Internet |
| TXT | Arquivo de texto do windows |
| TCP | *Transmission Control Protocol* |
| KPI | *Key Performance Indicator KPI* |
| SVM | *Support Vector Machine* |

# Sumário

# 1 Introdução

No segundo semestre do ano de 2014, o Brasil teve uma forte crise econômica que afetaram diretamente vários setores da indústria e prestação de serviço, somadas instabilidades políticas (CARLEIAL, 2015). Um dos setores mais afetados foi o de postos de revenda de combustíveis, comumente chamado de postos de gasolina ou postos de combustíveis, que teve uma queda de venda de 1,9% de gasolina, produto mais comercializado no Brasil (CAPELETTO, 2015).

Os postos de combustíveis são microempresas que revendem os combustíveis das refinarias para a população. Os postos de combustíveis, em geral, estão espalhados, estrategicamente, ao longo de todo território nacional. Com isso, a competitividade do setor de revenda de combustível tem levado os donos dos postos de combustíveis cada vez mais investirem em elementos gerenciais e ferramentas para análises e diagnósticos de produção.

No Brasil, em 2016, foram contabilizados 40.802 postos de revenda de combustível (**??**), espalhados em todo território nacional. Em Pernambuco, mais especificamente, agrega-se um total de 1.382 postos de combustível. É pertinente frisar a importância desde setor de negócio para o país, visto que em 2012, o país contava com 76 milhões de veículos (INCT, 2013), contando automóveis, motos e demais veículos, havendo um incremento de 138,6% se comparado com os números em 2001 que era de 34,9 milhões. Tendo aumento da frota nacional de veículos, tem-se um a aumento no consumo de combustível, e logo, um aumento na busca e revenda de combustível.

Situados neste cenário, os postos de combustíveis do estado de Pernambuco competem entre si para oferecer os seus melhores serviços, maximizar seus lucros e inovar seus serviços. Para isso, a Agência Nacional de Petróleo (ANP, 1997) regulamenta além da parte tributária dos produtos derivados do petróleo e etanol, também regulamenta o uso de sistema de informação e as tecnologias associadas para os postos de revenda de combustível, o que obriga os postos a utilizarem de automação comercial e sistemas de emissão de nota fiscal, atualmente sendo a Nota Fiscal Eletrônica - NFC-e (SEFAZ, 2017).

A automação comercial para postos de combustíveis é o dispositivo eletrônico que faz o controle das bombas de combustíveis, sendo ela responsável por autorizar o abastecimento e registrar os dados do abastecimento segundo descrito no manual de orientação do contribuinte da Secretaria da Fazenda – SEFAZ (SEFAZ, 2015). A automação dos postos de combustíveis coleta os dados que são enviados para sistemas

de informação que gerenciam esses dados como vendas de produtos, e acabam por emitir a nota fiscal.

## 1.1 Objetivo

Este trabalho tem como objetivo estudar os dados gerados pelas automações comerciais de alguns postos de combustível da região metropolitana do Recife (FNEM-BRASIL, 2018), para automatizar a análise de logs de abastecimentos, com a finalidade de encontrar falhas e/ou fraudes. Atualmente, este trabalho é feito de modo manual, ou seja sem ferramentas de análises, por técnicos de suporte remotos, chamados de *Help-Desks*. Com este projeto pronto, vai ser possível automatizar este trabalho, além de enriquecer a análise criando gráficos das análises, algoritmos que identificam falhas nos abastecimentos, algoritmos que identificam possíveis fraudes nos abastecimentos e agilidado no tempo de resposta da análise.

Como objetivos específicos, se destacam:

- Criar métodos para extrair os dados, organizá-los e visualizá-los.
- Criar métodos para análise descritiva dos dados.
- Criar métodos que identificam abastecimentos com falhas e fraudes, baseados nos manuais técnicos da automação e das médias de volumes de abastecimentos estudados com os dados obtidos.
- Aplicar as técnicas de *machine learning* que cria previsões para indicar falhas e possíveis fraudes em abastecimentos.

## 1.2 Estrutura do Trabalho

Este trabalho foi estruturado em 6 capítulos. O primeiro é a introdução do trabalho, que justifica as motivações e objetivos. A partir daí, o trabalho conterá no segundo capítulo, a fundamentação teórica e o conjunto de definições que são essenciais para o entendimento do projeto.

No terceiro capítulo, serão apresentados alguns trabalhos relacionados ao ambiente de posto de gasolina, análise de dados, python e frameworks de análise de dados. No quarto capitulo, os métodos utilizados e as ferramentas usados na construção do projeto serão apresentados.

No quinto capítulo, é apresentado o desenvolvimento do projeto, com a separação dos dados que vão ser usados para os testes e treinos da máquina de aprendizado,

e resultado do treinamento das máquinas e todos os dados relevantes conquistados. Já no sexto e último capítulo, encontraremos a conclusão e as considerações finais do trabalho.

# 2 Trabalhos Relacionados

É dedicado neste capítulo apresentar alguns trabalhos relacionados a postos de combustiveis e o uso da análise de dados em conjunto ao machine learning, que serviram para formular o conhecimento teórico do projeto. Todos os trabalhos mostrados a seguir tem pontos relevantes que foram usados neste projeto, mesmo que o tema em si não seja totalmente de acordo com o tema do projeto. Mas mesmo assim, foram importantes para definir métodos, comparações e conhecimento técnico.

O ”Manual de Orientação do Contribuinte 6.0”(SEFAZ, 2015), serve como manual técnico de como todas as empresas que emitem cupom fiscal deve se comportar e se comunicar com os servidores da Secretaria da Fazenda, SEFAZ. É obrigatório seguir esse manual técnico, uma vez que estão descritos nele os layouts dos xmls que são gerados para emissão da nota fiscal do consumidor eletrônica, NFC-e. Portanto, o entendimento desse manual nos faz ter segurança para saber quais são os dados que devemos extrair dos postos de combustíveis para comercializar os abastecimentos. No trabalho, ”A Análise de Dados na Pesquisa Cientifica”(TEIXEIRA, 2003), podemos percerber as benéfices de ter os conceitos de análise de dados para o uso da pesquisa científica. Além de fortalecer o entendimento de que o matérial a ser estudado, a análise de dados fortelece o entendimento dos dados, através das ferramentas de visualização e a otimização de navegar entre os dados, ou objetos de estudo, melhorando a qualidade da pesquisa. Um dos pontos mais importantes mostrado, são os processos de organização dos dados, e como o conceito da análise de dados ajuda na categorização dos dados, conceito que será usado neste projeto, onde será a visualização e acesso dos dados extraídos a partir das ferramentas de ciência de dados

Outro trabalho que reforça a importância da produção cientifica usando analise de dados e técnologia da informação é ”Tecnologias digitais na produção e análise de dados qualitativos”(JAVARONE, 2011). Nesta produção acadêmica, são experimentadas duas pesquisas, de cunho qualitativo, mas usando diferentes formas de coletar os dados. Com isso, é possível entender que dependendo de como os dados são coletados e visualizados, os objetivos do pesquisador e a visão sobre o projeto podem ser mudados graças ao uso de softwares, no caso do artigo do autor supracitado, o software foi o Camtasia Studio.

Ao ler o trabalho ”O Uso do Big Data na inteligência competitiva e na percepção do produto pelo cliente”(CASALINHO, 2017), firmamos o entendimento da importância de se ter a relação entre Big Data e os ambientes de marketing e TI. Os conceitos básicos sobre o que é Big Data são demostrados com um olhar mais voltado ao mercado,

fazendo pontes entre a relação de mercado com o produto e a obtenção de informações sobre o produto/serviço oferecido. A partir do Big Data é possível criar cenários de expansão de vendas e inovação de produto/serviço. Podemos juntar o Big Data com a análise de dados para melhorar o entendimento e o cenário competitivo que as empresas vivem.

”Uma Introdução as Support Vector Machines”(LORENA, 2007) temos a teorização introdutório do que é o Support Vector Machines, SVM. O SVM é uma técnica de aprendizado de máquinas, do inglês *Machine Learning*, na qual inserimos entradas de dados na máquina, onde terá valores para alguns tipos de campos, categorizada conforme o tipo de dado, e para cada entrada de dados é sinalizado, de forma binária, a classificação daquela entrada, ou seja, cada entrada de dado vai ter uma classificação já citada. A máquina do SVM vai treinar uma série de dados, conforme inserido, para entender o comportamento dos dados. Após os treinos, ele vai testar se o treino foi satisfatório, ou seja, se a máquina aprendeu o comportamento dos dados. A partir daí, o algoritmo vai aprender quais são os padrões possíveis para se chegar à uma determinada característica. O SVM, portanto, é basicamente o classificador linear binário não probabilístico, que aprende padrões de dados, e a partir de uma entrada, ele consegue determinar qual é a provável característica do conjunto de dados que foi introduzido.

No trabalho de (VELLOSO, 2017) há um levantamento bibliométrico sobre trabalhos feitos acerca do uso de *machine learning* para idenficar falhas em máquinas de gás e óleo. Ele faz algumas comparações entre algumas técnicas, e traz alguns conceitos sobre as técnicas encontradas nos artigos, como a SVM.

Outro trabalho sobre o uso de *machine learning* para detectar falhas ”Sobre aplicação de Sistemas Inteligentes para Diagnósticos de Falhas em Máquinas de Indução”(SUETAKE, 2011), aprofunda mais um pouco neste assunto. O trabalho além de fazer um comparativo entre a SVM e outras técnicas de *machine learning*, mostra como podemos usar técnicas de *machine learning* e ciência de dados, para prever possíveis falhas mecânicas e elétricas em máquinas elétricas, através de um estudo do que ocasiona as falhas nas máquinas, e como a *machine learning* e ciência de dados, conseguiriam avisar as máquinas que iriam falhar.

O trabalho ”Redes Bayesianas e Aprendizagem Aplicadas À Detecção de Falhas em Sistemas Dinâmicos”(MATSUURA, 2016) propõe uma nova solução para detecção de falhas em sistemas dinâmicos. O trabalho faz um estudo sobre a viabilização de introduzir a metodologia de aprendizado de máquina Bayesiana, que é uma avaliação de hipóteses, em decorrência da fórmula de Bayes. Usa-se o conceito das redes Bayesianas para fazer um estudo da representação do conhecimento e inferência sob a incerteza. Como resultado do trabalho, o conceito de *machine learning*, através da

metodologia Bayesiana, foi aplicado satisfatóriamente para detecção de falhas em sistemas dinâmicos.

”A Utilização dos Métodos de Data Mining e Machine Learning no Processo de Prevenção À Fraudes no Mercado Segurador”(MACHADO, 2017) é um trabalho que faz a união da ciência de dados (*data mining*) e da *machine learning* para prever fraudes no mercado de seguros. O projeto usa um aplicativo, chamado WEKA, que faz a mineração dos dados e a partir daí, é usado os conceitos da ciência de dados para entender padrões e descobrir fraudes através de comportamentos que fogem da normalidade.

O trabalho de (VELLOSO, 2017) foi usado como referênica neste projeto para o entendimento dos conceitos sobre *machine learning* e onde pode ser encontrada técnicas que podem ser usada para aprendizado de máquina. O trabalho de (SUETAKE, 2011) conclui, no capítulo 4, página 558, que a SVM teve o melhor resultado das técnicas comparadas para detectar falhas: *”Na segunda fase, diferentes algoritmos de diagnóstico (rede MLP, SVM, AG, AIS - Artificial Immune System, e FAIS - Fuzzy Artificial Immune System) são combinados ao choquet fuzzy integral para diagnosticar com exatidão as falhas do motor, sendo o SVM o algoritmo de diagnóstico que obteve melhores resultados dentre os demais”*. E esta conclusão tem relevância neste projeto, visto que o trabalho de Suetake visa a identificação de falhas. O trabalho de (LORENA, 2007) foi usado como base de conhecimento para o entendimento do que é SVM e como pode ser usado para este projeto. Partindo do conceito estatístico, mostrado capítulo 3 e demostrado a aplicação como máquina de aprendizado no capítulo 4. Portanto, estes trabalhos ajudaram a definir que o conceito da SVM vai ser usado neste projeto.

Todos os trabalhos relacionados contribuíram em partes conceituais e para melhorar as tomadas de decisões do desenvolvimento do projeto, visto que não foi encontrado algum projeto similar à proposta deste projeto.

# 3 Fundamentação Teórica

Neste capítulo, é abordado o funcionamento de um posto de combutível, que além de um ponto comercial, o posto de gasolina faz parte do projeto, pois é a partir dele que coletaremos os dados da pesquisa.

A extração de dados se dará a partir de um dispositivo eletrônico do posto de gasolina, a automação comercial. Com os dados obtidos, vai ser usado os conceitos da ciência de dados, para inicar as análises dos abastecimentos. O fluxo do projeto parte da extração de dados, identificação de falhas e possíveis fraudes de abastecimentos e termina no uso desses dados, para treinamento de uma *machine learning*, SVM, com previsões de identificação de falhas e fraudes dos abastecimentos.

## 3.1 Posto de Distribuição de Combustível

Postos de distribuição de combustível fazem parte de uma arquitetura de distribuição de combustível que começa nas refinarias e termina no consumidor final, no caso, a população (BNDES, 2018).

O combustível é gerado nas refinarias ou usinas, onde podem ser transformados em gasolina, diesel ou álcool. Após serem gerados, como mostra na Figura 1, os combustíveis ficam armazenados em centros de distribuição, ou bases de distribuição. As transportadoras realizam o trabalho de coletar os combustíveis nas bases de distribuição, e levam para os postos de distribuição, comumente chamados de postos de gasolina. Esse transporte é realizado por caminhões específicos para transportar combustível.

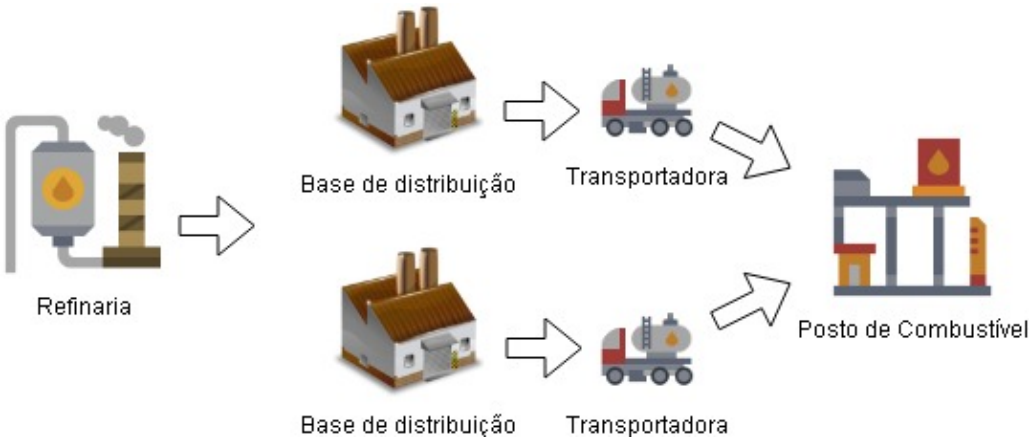


Figura 1 – Esquema de distribuição de combustível

Uma vez que o combustível esteja no posto, ele pode ser comercializado, obede-

cendo várias normas tributárias e de qualidade, conforme regulamentadas pela Agência Nacional do Petróleo, Gás Natural e Biocombustíveis, de sigla ANP (ANP, 1997).

Os postos de combustíveis operam com uma arquitetura tecnológica que une o hardware e o software, para gerenciar vendas e controlar estoques. Para este trabalho, será desconsiderado qualquer outro tipo de produto comercializado que não seja combustível.

Nos postos de gasolina (Figura 2), os combustíveis ficam armazenados em tanques específicos para armazenar cada tipo de combustível, variando o tipo do material que é feito e o tamanho. Esses tanques podem receber dispositivos de medição eletrônica de combustível, que fazem a medição de quanto de combustível tem em cada tanque, e se comunica com sistemas de informação para facilitar a leitura de cada tanque. É possível fazer essa leitura manualmente através de réguas de medição de tanque, que são dispositivos específicos para efetuar esta atividade.

As bombas de combustíveis são os dispositivos que transportam os combustíveis dos tanques para os veículos. Dutos de transporte fazem a comunicação entre tanques e bombas. As bombas de combustíveis registram todo o volume de combustível que sai dela, esse registro é chamado de encerrante. O encerrante é o valor acumulado de todo combustível que já foi trafegado por cada bico. Os bicos são os terminais das bombas. Cada bomba pode ter mais de um bico, e em cada bico trafega um combustível específico. Então, cada bomba pode se comunicar com mais de um tanque.

Os postos devem utilizar automação comercial para efetuar os registros de todas as vendas e recebimentos de combustível (SEFAZ, 2015; SEFAZ, 2017). Também é função da automação se comunicar com as bombas de combustível e liberar abastecimento.

Os postos de combustível são pontos comerciais, ou seja, efetuam vendas, e por conta disso emitem notas ficais, através de computadores que fazem o papel de caixa, chamados de Pontos de Venda ou PDV. Os PDV's são os computadores que a automação envia os dados dos abastecimentos, e esses dados viram vendas. Todas as vendas são enviadas para a secretaria da fazenda que faz a validação das informações contidas na venda, como informações tributárias. Outra função dos sistema de gerenciamento de postos de combustíveis, é a criação do livro de movimentação de combustível, ou LMC.

O LMC é uma obrigação fiscal na qual os postos de combustíveis devem sempre sinalizar toda a movimentação de cada combustível comercializado no posto, onde é registrado mensalmente, toda a entrada de combustível e toda a saída. Os encerrantes das bombas ficam responsáveis por calcular essas saídas, e as compras de combustíveis do posto com as distribuidoras de combustível são as entradas. Este livro é

apresentado nas auditorias da secretaria da fazenda.

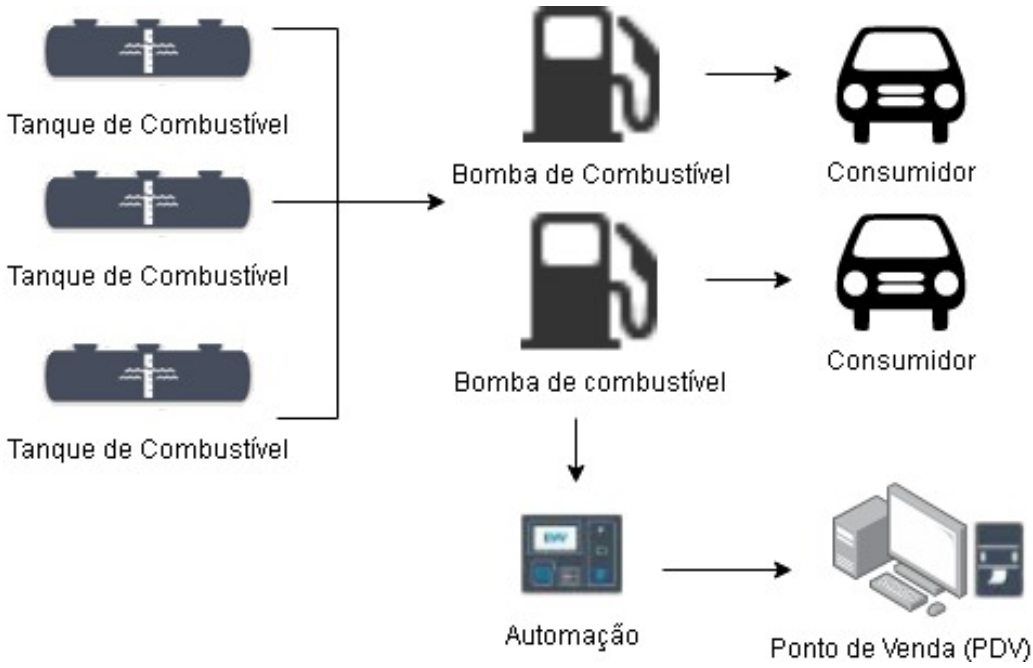


Figura 2 – Arquitetura Posto de Combustível

Para este trabalho, usaremos os dados de abastecimento gerados pela automação comercial Companytec, de modelo CBC06 (Figura 3). A CBC 06, fica responsável por se comunicar com as bombas e jogar os dados dos abastecimentos para o PDV. Em cada bomba, existe um dispositivo chamado de Identifid. Este dispositivo serve como leitor de cartão RFID com criptografia própria da automação CBC 06. Cada frentista, fica responsável por cartão RFID de identificação, ao iniciar o processo de abastecimento, os frentistas usam seus cartões RFID nos dispositivos Identifid, com isso, os frentistas são identificados e liberam as bombas para iniciarem os abastecimentos. Cada abastecimento é salvo na memória da automação, indicando qual foi o frentista que autorizou o abastecimento, a quantidade de combustível vendido, o preço unitário, qual foi a bomba autorizada, qual foi o bico, a data e hora do abastecimento, o encerrante final do bico e o valor total da venda (COMPANYTEC, 2005).

Figura 3 – Automação comercial CBC-06

## 3.2 Análise de Dados

A ciência sempre buscou entender fenônemos que acontecem na natureza através de observações, medições, induções e outros métodos (DIAS, 2004). Ela também tenta encontrar respostas do funcionamento orgânico e não-orgânico do nosso planeta (LEDERMANN, 1987). Hoje em dia existe um compartilhamento diário de dados entre as pessoas. Essas dados acabam virando informações, e estão sendo armazenados em vários ambientes e computadores. A quantidade de dados gerados diariamente está sendo cada vez maior, acumulando grande volume de dados, e são chamados de Big Data (FAGUNDES, 2017).

Manipular o Big Data é um dos objetos de estudo da atualidade, além de fazer o gerencimento de armazenamento desses dados. Extrair dados de Big Data está sendo essencial para o mundo dos negócios hoje em dia, (CASALINHO, 2017). As empresas estudam qualitativamente e estatisticamente dados extraídos de Big Data para entender a aceitação de seus produtos e/ou serviços. Usam também esses dados para entender o mercado que sua empresa está inserida, bem como, estudar os seus concorrentes.

Uma tendência que vem sendo muito bem executada, é a utilização do Big Data para a inovação de produtos e serviços (NASCIMENTO, 2017b). É correto afirmar que hoje em dia, graças à técnologia da informação, nossos produtos e serviços estão tendo ganhos reais em qualidade, gerenciamento de dados, armazenamento e entendimento de mercado.

Através da organização dos dados obtidos em Big Data, podemos criar ferramentas de diagnósticos para entender fatores que desejamos estudar. Os dados obtidos se transformam em variáveis para a criação de indicadores de performance, ou KPI, da sigla em inglês para Key Performance Indicator (NASCIMENTO, 2017a).

É também possível, através do estudo de Big Data, entender razões pelas quais serviços e produtos falham (ALMEIDA, 2006). Estudar a falha é algo apenas pontual, porém, quando se tem um grande volume de dados sobre todas as falhas, e o que ocasiona as falhas, é possível criar mecanismos que evitem ao máximo ocasionar falhas. A tecnologia da informação tem ferramentas que criam cenários para este tipo de atividade, como por exemplo, as árvores de falhas (SAKURADA, 2001).

## 3.3 Machine Learning e Python

Python é uma linguagem de programação considerada de alto nível, como Java e C#. Python é bastante usado em ambientes acadêmicos, entre alguns motivos, o

fato de ser uma liguagem cuja curva de aprendizado é menor que a de seus principais concorrentes, java e C++. Além de ter sido muito bem avaliada pelos programadores (Tabela 1), segundo a revista IEEE Spectrum (SPECTRUM, 2019).

| Ranking Linguagem de Programação - 2019 | | |
|---|---|---|
| Ranking | Linguagem | Score |
| 1 | Python | 100 |
| 2 | Java | 96,3 |
| 3 | C | 94,4 |
| 4 | C++ | 87,5 |
| 5 | R | 81,5 |
| 6 | JavaScript | 74,4 |
| 7 | C# | 74,5 |

Tabela 1 – Ranking de linguagem de programação em 2019.
fonte: IEEE Spectrum

Existem várias bibliotecas para python que auxiliam na resolução de várias atividades matemáticas, estatísticas e, o principal para este projeto, a análise de dados (GRUS, 2016).

O *Machine Learning* é uma área da computação, mais específica da inteligencia artificial, que reconhece padrões e comportamentos. Através do estudo automático do algoritmo, sobre os dados imputados nele, uma *machine learning* pode aprender de seus erros e faz previsões sobre os dados (KOHAVI, 1998).

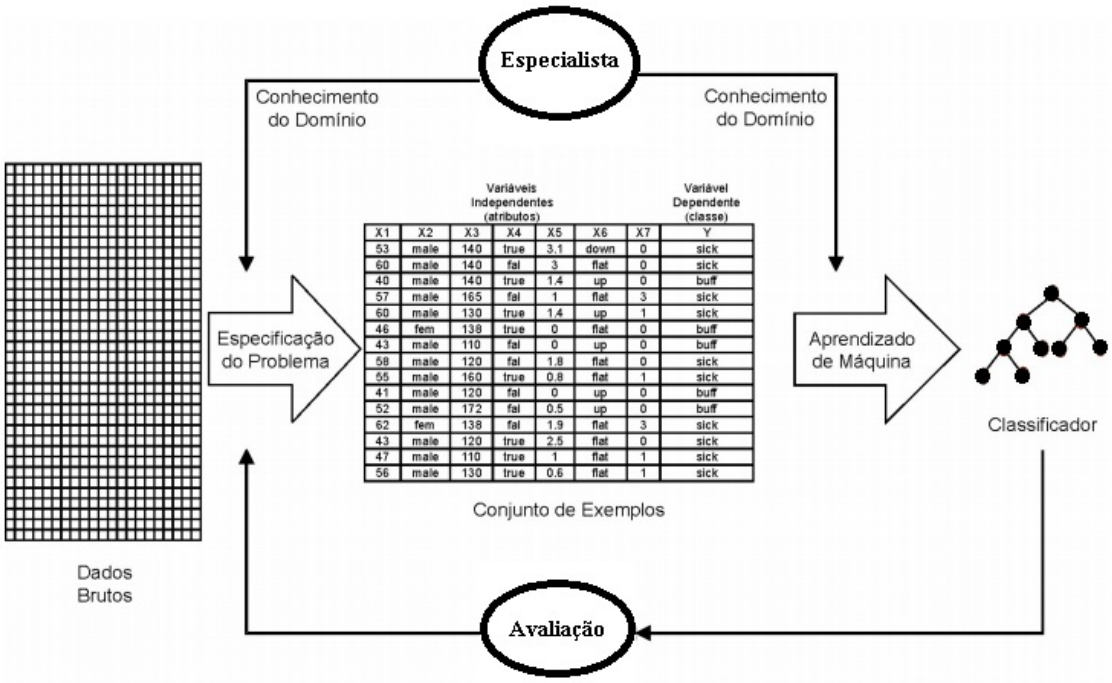


Figura 4 – Classificador do aprendizado da máquina.

Alguns dados vão ser passados como parâmetro na máquina, indicando para ela quais sãos variáveis dos dados, e quais são os indicativos que temos a partir de um conjunto de dados, geralmente uma flag que indica um boleano sobre a amostra de dados.

Enquanto é inserido entrada de dados, a máquina vai interpretando os muitos exemplos de dados e quais são os comportamentos estatísticos destes. Este tipo de aprendizado, é chamado de aprendizado estatístico (MONARD, 2017). O resultado da máquina vai ser chamado de classificador (Figura 4), que é uma estrutura que monta as possíveis classificações dos dados passados, baseadas no que foi aprendido pelo algoritmo. Quanto maior for o número de dados para testes, bem como maior número de exemplos de padrões, melhor será a tendência de sesr ter um bom aprendizado da máquina. A precisão na qual a máquina consegue acertar as previsões é chamada de *Accuracy*, ou acurácia.

# 4 Análise Exploratória dos Dados

Este capítulo tem como objetivo mostrar a metodologia usada para obter os dados dos abastecimentos vindo da automação, bem como, o tratamento deles para a construção da tabela de dados.

Todas as tratativas realizadas nos dados obtidos, bem como a análise descritiva dos dados são descritas neste capítulo. Também é explicada a preparação dos dados que são usados para treinamento.

## 4.1 Extração dos dados

A automação Companytec CBC-06 se comunica com os sistemas de informação de dois modos: via TCP/IP ou via biblioteca de arquivos .dll (COMPANYTEC, 2005). Com essa comunicação, os sistemas de informação podem obter dados em tempo real, desde que criem rotinas de extração de dados com esse intuito.

Em todo caso, a automação também guarda em memória os registros de abastecimentos que ocorreram na automação. Esses registros podem ser exportados em um arquivo de texto, formato .txt. Esse arquivo é chamado de log. O log é um processo de registro de dados, que armazena dados relevantes para efetuar auditorias ou registros de atividades.

Para extrair o log de abastecimentos da CBC-06, é necessário ter um software específico, que a própria Companytec disponibiliza (Tabela 5), chamado CBC Manager 2k9 (https://www.companytec.com.br/index.php/servicos/software).


Figura 5 – Tela Inicial do CBC Manager 2k9

Uma vez que o CBC Manager é acessado, o log de abastecimentos pode ser extraído. Um arquivo .txt vai ser gerado, e nele constará o limite de dez mil abastecimentos. Este é o limite de abastecimentos salvo na memória do CBC-06. O arquivo pode ser consultado através do notepad do Windows.

Os dados são disponibilizados através de uma estrutura definida, conforme descrito no DT435 - Protocolo de Comunicação Companytec (COMPANYTEC, 2005). Os dados ficam registrados em uma string, onde cada abastecimento está representada com a sua respectiva string.

A string vem em um formato conforme exemplificado abaixo na Tabela 2:

*ATTTTTTLLLLLLPPPPVVCCCCBBDDHHMMNNRRRREEEEEEEEEESSIIIIIIIIIIIIIIII MMMMPPKK*

| Detalhamento da string do log CBC-06 | | |
|---|---|---|
| Estrutura | Descrição | Tipo do Dado |
| A | Cabeçalho | CAR |
| TTTTTT | Total A Pagar | DEC |
| LLLLLL | Volume Abastecido | DEC |
| PPPP | Preço Unitário | DEC |
| VV | Código de Vírgula (aplicável em T, L e P) | HEX |
| CCCC | Tempo de Abastecimento | HEX |
| BB | Código de Bico | HEX |
| DD | Dia | DEC |
| HH | Hora | DEC |
| MM | Minuto | DEC |
| RRRR | Número do Abastecimento | DEC |
| EEEEEEEEEE | Encerrante do Abastecimento | DEC |
| SS | Status da Memória de Abastecimentos - 00 ok | CAR |
| IIIIIIIIIIIIIIII | Cartão Identifid Autorizador | HEX |
| MMMM | Número da Leitura de Identificação | DEC |
| PP | Status da Memória de Identificadores - 00 ok | CAR |
| KK | Checksum | HEX |

Tabela 2 – Legenda dos Campos da String do Log. Fonte: DT435- Companytec

Devido à ajuda de empresa de TI do ramo de gerenciamento de postos, foram coletados os logs de abastecimentos de 5 postos de combustível da região metropolitana do Recife. Todas as empresas usam a automação CBC-06. Para este trabalho, os postos não vão ser identificados com os nomes fantasias ou CNPJ. Vão ser identificados apenas como Posto 1, Posto 2, Posto 3, Posto 4 e Posto 5. Os frentistas também não terão seus dados pessoais expostos, serão identificados apenas com o código do cartão identifid.

O Sistema de Gerenciamento de posto se comunica com a automação CBC-06, recebe os abastecimentos em tempo real da automação e efetua a emissão de NFC-e

das vendas de abastecimentos. Também é feito todo o gerenciamento fiscal e tributário das vendas, fechamento de caixas e faturamento de clientes fidelizados ao posto.

## 4.2 Análise dos Dados

Após obter o log de abastecimentos, um algoritmo em python vai efetuar a leitura do arquivo e guardar os dados, de forma organizada, conforme descrita na estrutura acima.

Os dados são armazenados em um DataGrid, usando o framework Pandas. O Pandas é uma biblioteca do python, muito utilizada para análise de dados. A estrutura do Datagrid do Pandas é usada para montar a tabela de manipulação dos dados e através da tabela, é possível efetuar a análise descritiva dos dados.

Cada arquivo de log lido é diferenciado de outro adicionando o campo Posto, na estrutura do datagrid, podendo trabalhar melhor com os dados, visto que estão separados.

| Data | Hora | Dia Semana | Codigo | Cod. Bico | Identificação | Bico |
|---|---|---|---|---|---|---|
| 2019-04-04 | 17:18 | Thursday | 0000 | 3A | 0000 | 0C |
| 2019-04-04 | 17:19 | Thursday | 0001 | 3A | 0001 | 0C |
| 2019-04-04 | 17:20 | Thursday | 0002 | 3A | 0002 | 0C |
| 2019-04-04 | 17:21 | Thursday | 0003 | 3A | 0003 | 0C |
| 2019-04-04 | 17:22 | Thursday | 0004 | 3A | 0004 | 0C |

Tabela 3 – Exemplo Estrutura DataGrid Pandas - Primeira parte

| Encerrante | Quantidade | Valor | PPU | Identifid | Posto |
|---|---|---|---|---|---|
| 173106.13 | 5.27 | 20.02 | 3.799 | B3CF6C2EC1D36AE7 | Posto 1 |
| 173111.40 | 5.27 | 20.02 | 3.799 | B3CF6C2EC1D36AE7 | Posto 1 |
| 173116.67 | 5.27 | 20.02 | 3.799 | B3CF6C2EC1D36AE7 | Posto 1 |
| 173121.94 | 5.27 | 20.02 | 3.799 | B3CF6C2EC1D36AE7 | Posto 1 |
| 173127.21 | 5.27 | 20.02 | 3.799 | B3CF6C2EC1D36AE7 | Posto 1 |

Tabela 4 – Exemplo Estrutura DataGrid Pandas - Segunda Parte

O Pandas consegue armazenar grandes quantidades de dados, e para este projeto ele vai armazenar a leitura dos cinco arquivos de logs, dos cinco postos, totalizando 50 mil abastecimentos salvos no datagrid. Com esses dados, estruturados como o exemplo acima (Tabelas 3 e 4), podemos iniciar as análises descritivas.

Com o datagrid montado, podemos começar explorando os dados obtidos. Graças ao framework Pandas, podemos entender o comportamento estatístico do posto de combustível, baseando-se apenas nos abastecimentos.

Tomando como exemplo o Posto 1, conseguimos visualizar a distribuição descritiva do volume vendido no posto, na leitura de 10 mil abastecimentos. Para esta análise, vale salientar que iremos pegar todos os combustíveis comercializados pelos postos: Etanol, Gasolina e Diesel. Também é importante frisar que todos os postos operam entre 05:00 e 00:00 todos os dias.

| Análise Descritiva para Abastecimentos - Posto 1 | | | | | |
|---|---|---|---|---|---|
| Média | Mediana | Moda | Max | Min | Amplitude |
| 21,289 | 7,89 | 5,26 | 950,73 | 0,0 | 950,73 |

Tabela 5 – Análise Descritiva para Abastecimentos em litros - Diesel, Gasolina e Etanol

| Análise Descritiva para Abastecimentos - Posto 1 | | |
|---|---|---|
| Variancia | Desvio Padrão | Desvio Absoluto |
| 2774,333 | 52,671 | 22,144 |

Tabela 6 – Análise Descritiva para Abastecimentos em litros - Diesel, Gasolina e Etanol

Com esses dados mais básicos, conseguimos entender a quantidade média de abastecimento que foi lida no arquivo de log (Tabelas 5 e 6). Empiricamente, os funcionários dos postos acabam sabendo quais são as quantidades de litros mais comuns entre os abastecimentos, então essa primeira análise nos faz ter argumentos estatísticos para confrontá-los, caso seja necessário.

Aprofundando mais um pouco, conseguimos encontrar os valores que fogem das margens de abastecimentos, e a partir daí, é possível identificar possíveis fraudes, visto que valores muito discrepantes do que ocorre naturalmente, vão ser indicativos para possíveis fraudes, tanto para mais, quanto para menos.

| Análise Descritiva para Abastecimentos - Posto 1 | | | | | |
|---|---|---|---|---|---|
| Q25 | Mediana | Q75 | IQR | Limite Minímo | Limite Máximo |
| 5,0 | 7,89 | 15,79 | 10,779 | 11,184 | 31,974 |

Tabela 7 – Análise Descritiva para Abastecimentos em litros - Diesel, Gasolina e Etanol

Com os valores de Q25 e Q75 (Tabela 7), conseguimos visualizar quais são as médias maiores e menores, para a amostra, nas colunas Limite Máximo e Limite Mínimo. Com isso, temos dados para criar mecanismos que irá informar-nos de todos os abastecimentos extremos.

Podemos criar o limite superior (Tabela 7), para o volume de abastecimento efetuado, e verificar se ele ultrapassa o limite estipulado. Graficamente, os limites de volume vendido e valor, em reais, arrecadado.

O valor arrecadado é a soma da quantidade de combustível vendido multiplicado pelo valor unitário do produto. Dito isso, conseguimos confirmar, através dos

gráficos, que a distribuição de combustível versus o valor arrecadado se equivalem (Figuras 6 e 7). Uma notória fraude que pode ocorrer seria se o limite do valor arrecadado estivesse bem abaixo da quantidade vendida. Isso ocorreria caso, os frentistas, tivessem trocado o preço do valor unitário dos combustíveis em alguns abastecimentos pontuais.

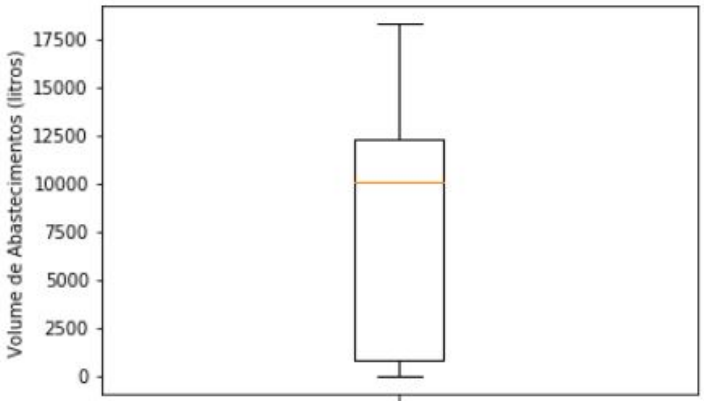


Figura 6 – Distribuição abastecimento - Posto 1

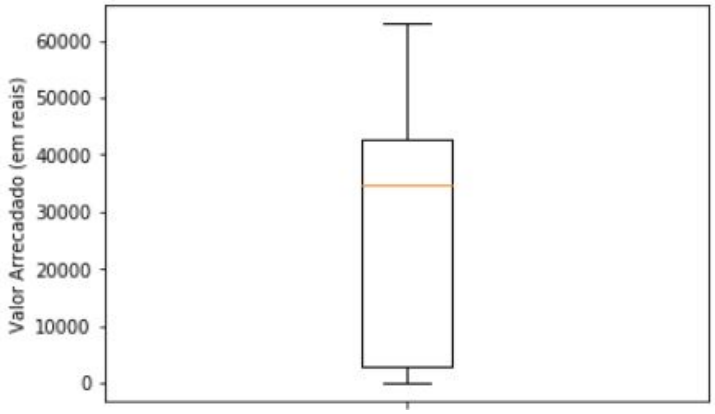


Figura 7 – Arrecadação abastecimento - Posto 1

Através do gráfico de barra de distribuição de abastecimento (Figura 8), conseguimos observar quais são os valores, em litros, que mais se repetem no posto de gasolina. Esse gráfico nos dá o poder de confrontar possíveis fraudes partindo de abastecimentos exageradamente vendidos.

O gráfico de dispersão (Figura 9) trabalharia em conjunto com o de distribuição e facilitaria o entendimento de abastecimentos discrepantes.

Usando o dataframe também é possível acessar informações que são importantes para os donos dos postos de combustíveis e que os sistemas de gerenciamento de postos, ainda não implementaram, que é a arrecadação de abastecimentos por dia da semana (Figura 10).

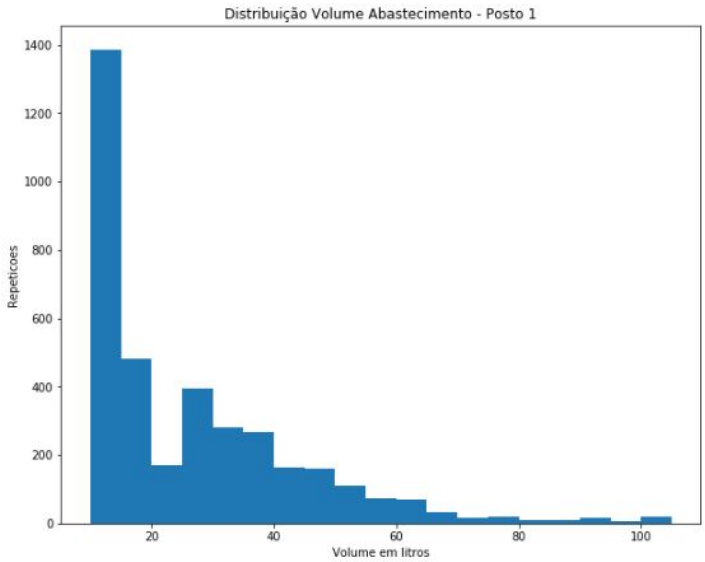


Figura 8 – Distribuição de abastecimento - Posto 1

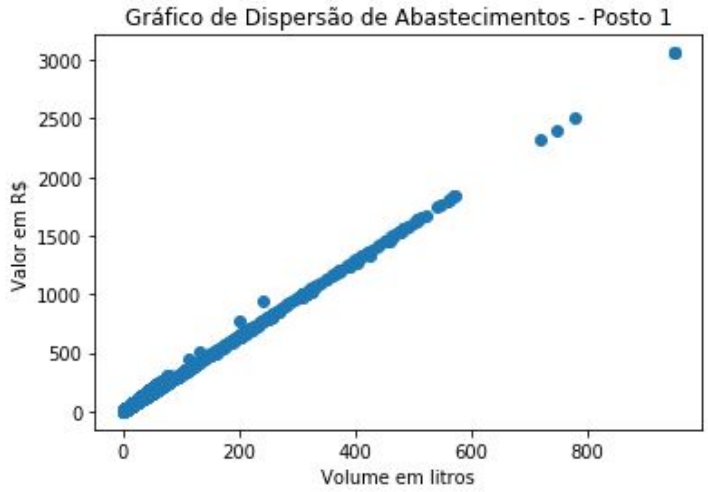


Figura 9 – Dispersão de abastecimento - Posto 1

Para criar este gráficos, tivemos que adicionar a nomeclatura dos dias da semana no dataframe original, conforme o trecho de código abaixo:

*x = datetime.datetime (int(ano-criacao), int(line[38:40]),int(line[32:34]))*

Neste trecho de código, utilizamos a biblioteca datetime do python para converter uma data em um valor de timestamp códificado. Ao utilizar a função *strftime*, conseguimos traduzir o código do timestamp para dias da semana, passando o parâmetro *'A'*, e assim adicionamos o dia da semana no nosso dataframe.

Com a visualização desses valores, os donos de postos de combustíveis conseguem melhorar o gerenciamento de compras de combustível, pois eles vão ter um melhor entendimento dos dias que mais se comercializa combustível (Figura 10). Outro ponto positivo, é a possibilidade da criação de campanhas de fidelidade do cliente,

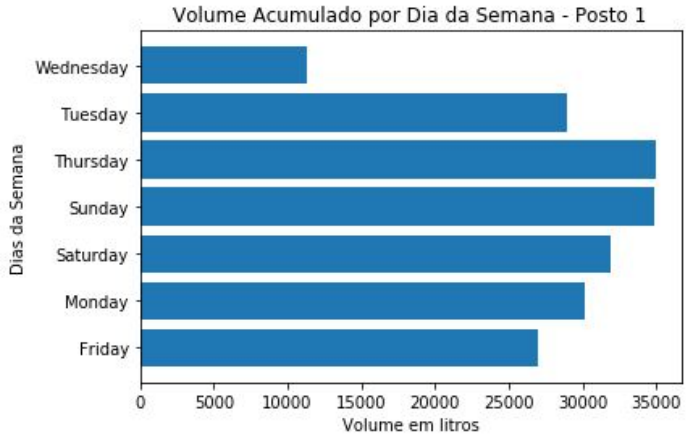


Figura 10 – Arrecadação de abastecimento por dia da semana - Posto 1

pois o dono do posto pode explorar os dias que menos vende, e criar promoções para chamar mais clientes para esses dias deficitários, e premiar os melhores clientes dos dias que mais se arrecada.

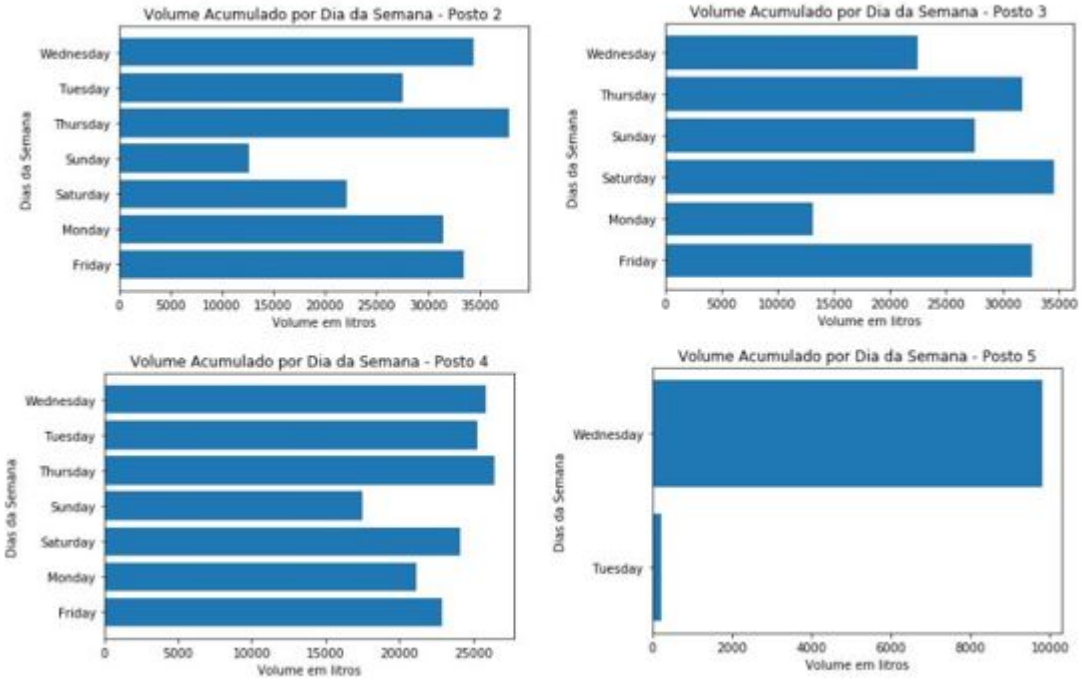


Figura 11 – Arrecadação dos postos 2, 3, 4 e 5 por dia da semana

Ao comparar os valores arrecadados dos postos 2, 3, 4 e 5 (Figura 11) observamos que não existe um padrão em si, para qual dia da semana que mais arrecada. Com esses gráficos, conseguimos visualizar quais dias da semana os postos operaram, bem como quais dias da semana cada um obteve um melhor resultado de vendas. O posto 5 nos mostra que, de acordo com o log, ele operou apenas dois dias da semana, a quarta-feira (Wednesday) e na terça-feira (Tuesday).

O log do posto 5 foi recolhido para a analisar as vendas da quarta-feira, pois os valores, em reais, que os frentistas entregaram para o dono do posto estava divergindo com o que o sistema de gerenciamento de posto estava calculando. Com esse gráfico, rapidamente o dono do posto poderia observar que o sistema estava calculando vendas da terça-feira e da quarta-feria juntos. Esta análise, atualmente, é feita por help-desks que fazem esta análise lendo o arquivo de log, sem nenhuma ferramenta para auxílio. Com a análise de dados, temos esse diagnóstico automatizado e mais rápido.

Para uma análise mais profunda do padrão de abastecimento por dias da semana, deveríamos considerar algumas variáveis importantes e que não entram no mérito deste projeto, como por exemplo, localização dos postos. A localização em si, interfere na arrecadação de cada posto, variando pela quantidade de concorrência que os postos tem, por estar localizado em alguma região específica e também pelo poder aquisitivo da região.

Usando o dataframe, podemos criar outro gráfico que automatiza uma parte importante da análise, que é a arrecadação total por cada frentista. Com já foi informado, os frentistas são indentificados pela automação através de um cartão RFID, chamado de Identifid. Os relatórios de Identifid são extremamente úteis para os donos e gerentes dos postos, pois é a partir deste tipo de relatório, que eles confrontam com o dinheiro que é entregue pelos frentistas. Por exemplo, ao final de um dia de trabalho, o frentista entrega R$100 para o gerente. Com o relatório de Identifid, o gerente consegue comprovar se o valor que o frentista deveria ter entregue era R$100 ou algum outro valor.

Os sistemas de gerenciamento de postos geram este relatório, mas sem nenhuma tratativa dos dados que vem da automação. Todos os dados são tratados durante a camada de negócio, e se houver algum tipo de falha transferência de dados da automação para o sistema, o relatório estará comprometido. Portanto, sempre que existe um problema, ou questionamento de arrecadação, o log da automação é lido, e seus abastecimentos são contabilizados manualmente pelos helps-desks, onde eles são livres para usar qualquer recurso que eles preferirem. Geralmente, eles usam o excel para separar os dados e também fazem os cálculos mais básicos. Normalmente, demoram algumas horas para terminar este levantamento de dados.

Com as duas linhas abaixo, conseguimos acessar os dados que precisamos e geraremos o gráfico de arrecadação agregado para cada cartão Identifid.

```
cartao = df.groupby(df['Identifid']).sum()

plt.barh(cartao.index, cartao.Valor)
```

A primeira linha, definimos uma variável que vai receber um dataframe que terá

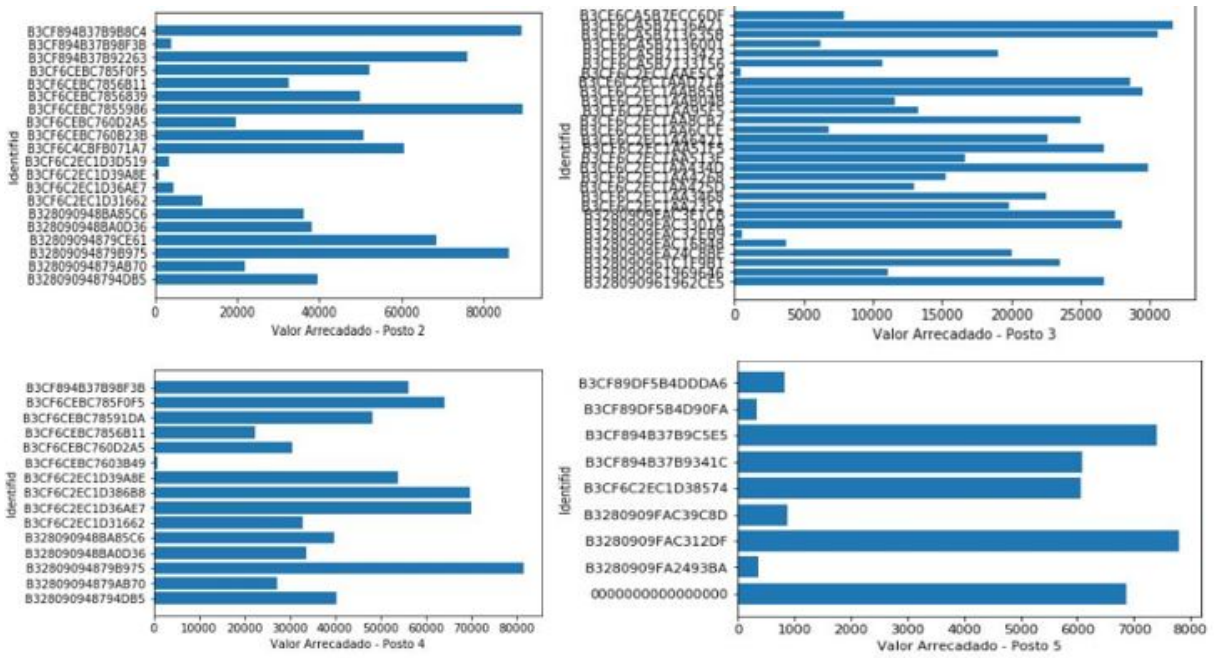


Figura 12 – Gráficos de Identifid dos postos 2, 3, 4 e 5

todos os abastecimentos agregados por Identifid, além de ter a soma de tudo que eles venderam em reais e litros de combustível. Na segunda linha, montamos a estrutura do gráfico, para gerar com os parâmetros com o que precisamos. Seguramente, é uma operação mais confiável e mais eficaz do que a análise manual que é feita hoje em dia.

E com isso, conseguimos gerar o gráfico de Identifid pronto.

Do mesmo jeito que conseguimos gerar o gráfico, também conseguimos gerar relatórios de Identifid. O relatório de Identifid gerado a partir do log da automação é o mais confiável disponível. Pois estamos tratando diretamente com o gerador dos dados, e a partir daí, conseguimos ter os valores de referência para consulta.

# 5 Identificação Automatizada de Falhas e Fraudes

Com os dados extraídos, podemos partir para a tratativa de identificação de fraudes e falhas. Como já foi citado anteriormente, toda esta análise é feita de forma manual. Com isso, a qualidade da análise varia muito de acordo com a experiência do help-desk que vai efetuar a análise, o que acaba comprometendo mais tempo de análise e, em algumas vezes, uma re-análise por outro help-desk mais experiente.

Para identificar falhas, temos que nos basear pelo manual de desenvolvedor da Company (COMPANYTEC, 2005) e para as fraudes vamos nos basear pelos dados que extraímos e verificar os abastecimentos discrepantes.

Uma vez que conseguirmos mapear as falhas e fraudes dos abastecimentos, teremos a possibilidade de preparar os dados para treinar nossa máquina SVM, e principalmente, criar mecanismos automatizados de identificação de falhas e fraudes.

Para este projeto, vamos tratar fraude e falha com o mesmo nome no DataFrame, pois ambos vão ter o mesmo efeito nas tratativas do que fazer com eles quando identificados. O que acontece é que quando um abastecimento é identificado, sendo por falha ou fraude, o frentista é comunicado de que algum abastecimento efetuado por ele não está coerente, seja por falta, seja por acréscimo fora da realidade ou por algum outro motivo. Geralmente, processos internos (de cada posto) são tomados para identificar a causa.

Um dos objetivos deste trabalho é auxiliar nesta identificação e criar uma SVM que será capaz de identificar alguma falha ou fraude, e com isso, evitar que algum frentista registre alguma venda de abastecimento fraudado ou com falha.

## 5.1 Alteração no DataFrame

Para não alterar a estrutura do dataframe que contém os dados, será criado um novo dataframe para começar a identificação dos abastecimentos que contem falha. Uma nova coluna vai ser incluída, que vai ser chamada de Falha. Esta coluna vai receber um inteiro, 0 ou 1. Onde zero será o valor padrão, e significa que o abastecimento não tem falha. Ao encontrar algum abastecimento com falha, a coluna Falha vai ter o valor atualizado para 1, que significa que o abastecimento contém uma falha.

Esta identificação vai servir para:

- Criar a classificação para treinar a machine learning
- Identificar os abastecimentos com falhas e/ou fraudes, com a possibilidade de criar relatórios de falhas.

No trecho de código abaixo, é mostrado como vamos criar um novo data frame, copiando toda a estrutura do dataframena qual os dados foram extraídos, chamado de df. Em seguida, alteraremos a estrutura do novo dataframe adicionando a coluna de falha, e vamos padronizar todas as linhas da nova coluna, preenchendo com o valor 0 (zero).

```
df_posto1_falha = pd.DataFrame(df)

df_posto1_falha['Falha'] = 0
```

E mais abaixo, o exemplo (Figura 13) de como ficará o novo dataframe.

```
In [460]: df_posto1_falha
Out[460]:
```

| | Data | Hora | Dia Semana | Codigo | Cod. Bico | Identificacao | Bico | Encerrante | Quantidade | Valor | PPU | Identifid | Posto | Falha |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2019-05-22 | 10:19 | Wednesday | 0000 | 3A | 0000 | 10 | 642511.53 | 3.08 | 12.01 | 3.899 | B3CF894B37B92263 | Posto 2 | 0 |
| **1** | 2019-05-22 | 10:19 | Wednesday | 0001 | 3A | 0001 | 0D | 722894.63 | 3.26 | 15.00 | 4.599 | B328090948BA0D36 | Posto 2 | 0 |
| **2** | 2019-05-22 | 10:20 | Wednesday | 0002 | 3A | 0002 | 0C | 886065.82 | 2.17 | 10.00 | 4.599 | B32809094879B975 | Posto 2 | 0 |
| **3** | 2019-05-22 | 10:21 | Wednesday | 0003 | 3A | 0003 | 51 | 160371.42 | 31.42 | 152.04 | 4.839 | B32809094879CE61 | Posto 2 | 0 |
| **4** | 2019-05-22 | 10:23 | Wednesday | 0004 | 3A | 0004 | 10 | 642515.38 | 3.84 | 15.00 | 3.899 | B3CF894B37B92263 | Posto 2 | 0 |

Figura 13 – Exemplo da adição da coluna Falha

Esta alteração será nosso pontapé de partida para inicarmos as identificações de falhas e fraudes, e posteriormente, treino da SVM.

## 5.2 Falhas

As falhas são consideradas todos os abastecimentos gerados através de um mau funcionamento da automação, que são catalogados no manual da fabricante da automação (COMPANYTEC, 2005).

### 5.2.1 Falha de Encerrante

A primeira falha que ocorre na automação, na verdade, é cometida em razão da bomba de abastecimento. A falha de encerrante é causada quando a numeração de encerrante da bomba vem zerada ou com ecrito 'EEEEEEEEE', conforme descrito no manual de desenvolvedor da Companytech (COMPANYTEC, 2005), para a CBC-06.

Este é um erro de hardware que pode ocorrer por alguns motivos, como falha na comunicação entre bomba e automação ou erro na placa da bomba.

O algoritmo vai percorrer todos os dados do dataframe usando um for, e usando o método iterrows(), nativo do framework do Pandas. O método iterrows retorna o iterador que gera cada valor de índice junto com uma série que contém os dados em cada linha. E a partir daí, conseguimos manipular cada linha lida.

Também é criado uma variável que irá contabilizar quantos abastecimentos com erro de encerrante ocorreram durante a leitura.

```
#Falha de encerrante
aux_encerrante = 0

for i, row in df_posto1_falha.iterrows():
    if row['Encerrante'] == 0 or row['Encerrante'] == 'EEEEEEEEE':
        df_posto1_falha.loc[i, 'Falha'] = 1
        aux_encerrante = aux_encerrante + 1

if (aux_encerrante > 0):
    print('Houveram ' + str(aux_encerrante) + ' erros de encerrantes.')
else:
    print ('Nao houveram falhas de encerrantes')
```

Figura 14 – Código de identificação de falha de encerrante

Uma vez que a linha de abastecimento é lida, e o valor da coluna de encerrante é igual a 0 ou a 'EEEEEEEEE', o valor da coluna falha, para aquele abastecimento, se tornará 1. E assim, identificamos um abastecimento com falha de encerrante.

O contador de encerrante, a variável chamada de aux_encerrante, também será incrementada. Ao fim, o valor desta variável vai ser checada, e caso seja maior que zero, uma mensagem irá ser retornada para o usuário, informando quantas falhas houve.

### 5.2.2 Falha de Cálculo da Automação

Para calcular o valor a ser pago de um abastecimento é necessário multiplicar o preço unitário com a quantidade vendida. O problema acontece quando se 'pre-seta' um valor. O pre-set é um recurso que as bombas de abastecimentos utilizam para pré-definir um valor tanto para dinheiro, quanto para volume.

Quando um cliente chega ao posto, ele tem a opção de pedir para abastecer apenas 45 reais. Então, o frentista 'pré-seta' o valor de 45 reais na bomba, e a bomba irá controlar a quantidade a ser vendida, e se encerrará quando bater os 45 reais. Acontece que neste momento, o preço unitário pode estar cadastrado com até 3 casas decimais, exemplo, 3,999. E se dividir os 45 reais por 3,999, teríamos a quantidade de 11,25281 registrada na bomba.

Acontece que quando a automação recebe este valor de 11,25281, ela precisa transformá-lo em um decimal com 2 ou 3 casas após a vírgula, dependendo de como a automação está configurada. Por isso ele arredonda a quantidade. E esse valor vai se transformar em 11,26 litros, por exemplo.

Neste momento que acontece esta falha, nos arredondamentos dos abastecimentos gerados pelo 'pré-set'. Dependendo deste arredondamento, o valor do abastecimento pode superar o valor pré-setado, e o valor registrado da venda de 45 reais pode conflitar se multiplicarmos 11,26 por 3,999, que daria 45,02 reais. Uma diferença de dois centavos.

Por isso, a SEFAZ (SEFAZ, 2017) limita essa diferença da venda em até 5 centavos por venda. Acima disso, existe um crime contra o consumidor. Por conta disso, foi criado o código que faz a validação se este limite está sendo respeitado.

```
#Falha de calculo da automacao
aux_falhacalculo = 0

for i, row in df_posto1_falha.iterrows():
    if (row['Valor'] - row['PPU'] * row['Quantidade']) > 0.05:
        df_posto1_falha.loc[i, 'Falha'] = 1
        aux_falhacalculo = aux_falhacalculo + 1

if (aux_falhacalculo > 0):
    print('Houveram ' + str(aux_falhacalculo) + ' erros de calculo da automacao.')
else:
    print ('Nao houveram falhas de calculo da automacao')
```

Figura 15 – Código de identificação de falha de cálculo da automação

O dataframe vai ser percorrido, e abastecimento por abastecimento vai ser lido. Uma checagem no calculo vai ser feita, pegaremos o valor da venda que está registrado e iremos subtrair a multiplicação entre o preço unitário e quantidade. Essa subtração não pode ultrapassar dos cinco centavos.

Se o valor for maior que 5 centavos, o campo de falha vai ser marcado com 1, e a variável que contabiliza a quantidade de abastecimentos com erro de cálculo vai ser incrementada.

Este algoritmo será extremamente útil para os postos de gasolina, pois os problemas de arredondamento ocorrem com frequência, e causa problema nas contabilização dos caixas dos frentistas. Pois com o acúmulo dos arredondamentos, os frentistas serão cobrados por valores maiores do que eles abasteceram de fato.

Este nível de checagem, a nível de log, é importantíssimo, pois em alguns sistemas de gerenciamento de posto, um outro arredondamento também pode ser feito, o que irá divergir ainda mais com o valor abastecido.

### 5.2.3 Falha de Identificação do Identifid

Assim como ocorre com os encerrantes, as leituras de identificações dos frentistas também podem vir corrompidas. Segundo o manual da Company (COMPANY-TEC, 2005), quando existe erro de comunicação entre os leitores de RFID Identifid e a automação, o abastecimento pode ser registrado por um código de erro, marcado como 'EEEEEEEEEEEEEEE' ou 'FFFFFFFFFFFFFFF', dependendo da versão do software da automação.

Outros erros de comunicação comprometem a autorização do abastecimento, e com isso, a automação não libera a bomba para abastecer. E por isso, os leitores de Identifid são desligados, e a automação não registra nenhum frentista. Para este caso, o código 'EEEEEEEEEEEEEEE', será apresentado.

```
#Falha de Identifid
aux_identifid = 0

for i, row in df_posto1_falha.iterrows():
    if row['Identifid'] == 'FFFFFFFFFFFFFFF' or row['Identifid'] == 'EEEEEEEEEEEEEEE':
        df_posto1_falha.loc[i, 'Falha'] = 1
        aux_identifid = aux_identifid + 1

if (aux_identifid > 0):
    print('Houveram ' + str(aux_identifid) + ' erros de leitura de Identifid.')
else:
    print ('Nao houveram falhas de leitura de Identifid')
```

Figura 16 – Código de identificação de falha de leitura de Identifid

O algoritmo faz uma varredura no dataframe e lê abastecimento por abastecimento. Na coluna Identifid, ele faz uma checagem e analisa se o código do identifid está sendo igual a 'EEEEEEEEEEEEEEE' ou 'FFFFFFFFFFFFFFF', caso seja positivo, a coluna falha vai ser alterada para 1.

Uma variável vai ser incrementada toda vez que um abastecimento com falha de leitura de Identifid for identificada, e no final, uma mensagem será exibida para o usuário, informando quantos abastecimentos tiveram este tipo de problema.

### 5.2.4 Falha de Abastecimentos Duplicados

Uma das falhas que mais preocupa quando ocorre é abastecimento duplicado, ou seja, quando um abastecimento é registrado duas vezes pela automação.

Quando ocorre esta falha, a automação envia para os sistemas de gerenciamento de postos um abastecimento por duas vezes. E este tipo de problema acaba gerando prejuízo para o posto e para o frentista, pois a venda foi contabilizada por duas vezes. Para o posto, se a venda for efetuada, ela deverá ser cancelada na SE-

FAZ usando os webservices (SEFAZ, 2017) de cancelamento. Caso contrário, o posto deverá pagar imposto por uma venda que foi emitida incorretamente.

Atualmente, esta checagem é feita manualmente pelos help-desks, quando acessam o arquivo de log da CBC-06. O que dificulta bastante a maioria deles, é o fato deles não saberem manipular os dados dos abastecimentos no log, mesmo utilizando programas como Notepad++ e Office Excel.

Os abastecimentos quando são duplicados repetem todos os campos de um outro abastecimento. Portanto, é mais seguro verificarmos se o campo Código é único, pois este campo é o identificador de cada abastecimento.

Esta falha pode ocorrer por conta de falha elétrica de uma bomba, que duplica os abastecimentos, ou algum problema no banco de dados da automação.

```
#Falha Registros Duplicados
from collections import defaultdict

lista_codigo = []
keys = defaultdict(list);
aux_duplicado = 0
valoresduplicados = []


for i, row in df_posto1_falha.iterrows():
    lista_codigo.append(row['Codigo'])

for key, value in enumerate(lista_codigo):
    keys[value].append(key)


for i, row in df_posto1_falha.iterrows():
    for value in keys:
        if (row['Codigo'] == value ):
            df_posto1_falha.loc[i, 'Falha'] = 1
            aux_duplicado = aux_duplicado + 1

if (aux_duplicado >= 1):
    print('Houve ' + str(aux_duplicado) + ' abastecimentos duplicados!')
else:
    print('Nao houve abastecimento duplicado!')
```

Figura 17 – Código de identificação Abastecimentos Duplicados

O algoritmo de identificação de abastecimentos duplicados funciona fazendo uma primeira varredura no dataframe e salvando todos os códigos de abastecimentos em uma lista, chamada lista_codigo.

Depois, vai haver uma varredura na lista_codigo, usando o método ’enumerate’.

O método enumerate adiciona um contador a um iterável e o retorna em uma forma de objeto enumerado. Esse objeto enumerado pode então ser usado diretamente em loops ou ser convertido em uma lista de tuplas usando outro método, chamado list. Sendo assim, cada código vai receber uma chave, e será adicionada em uma lista, chamada de keys, mas que é passado um método chamado defaultdict.

Esse método percorre uma lista e elimina todos os valores que não se repetem dentro da lista, ou como diz no manual do python, ele encontra valores que estão faltando dentro de uma lista.

Depois disso, é feito uma checagem se o tamanho da lista keys é maior que 1, ou seja, se existe abastecimentos duplicados dentro da lista, identificado pelo código do abastecimento. Caso seja positivo, a lista keys vai ser varrida e cada elemento da lista vai percorrer o dataframe dos dados de abastecimentos e vai procurar qual o abastecimento que corresponde com o código de abastecimento. Ao encontrar, o abastecimento vai ser marcado com 1 na coluna de falha.

E ao fim, uma variável que contabiliza quantos abastecimentos duplicados houve e vai enviar uma mensagem informando a contabilidade disto. A lista key também fica disponível para o usuário, caso ele queira saber quais abastecimentos foram duplicados.

## 5.3 Fraude de Quantidade do Abastecimento

Como foi visto anteriormente, através da análise de dados, conseguimos encontrar médias, padrões, desvio padrão e os limites de dispersão dos abastecimentos. Com isso, nós conseguimos calcular quais são os abastecimentos que podem ser considerados uma possível fraude.

Para isto, vamos considerar que um possível abastecimento fraudado é um abastecimento que equivale a 3 vezes a média dos abastecimentos do posto. Com esse mecanismo, cada vez que a média de abastecimento do posto se refina, o limite para abastecimento também irá aumentar.

Outro valor que pode ser considerado fraude, são abastecimentos menores que 1 litro. Pois como podemos ver nos gráficos de distribuição de abastecimentos anteriormente (Figura 8), não conseguimos ver abastecimentos que fossem menor que 1 litro.

O algoritmo faz uma varredura no dataframe e verifica, em cada abastecimento, se o valor abastecido foi maior que 3 vezes a média de abastecimentos efetuados ou se o valor é inferior a 1 litro. Caso seja positivo, em alguma dessas medidas, o abastecimento será identificado como falha e além disso, uma lista, chamada de lista-

```
#Fraude de Quantidade
aux_Quantidade = 0
lista_funcionario = []

for i, row in df_posto1_falha.iterrows():
    if row['Quantidade'] > 3*df['Quantidade'].mean() or row['Quantidade'] < 1:
        df_posto1_falha.loc[i, 'Falha'] = 1
        aux_Quantidade = aux_Quantidade + 1
        lista_funcionario.append([str(row['Identifid']),row['Valor']])

if (aux_Quantidade > 0):
    print('Houveram ' + str(aux_Quantidade) + ' abastecimentos fora da media.')
else:
    print ('Nao houve possivel fraude nos abastecimentos')
```

Figura 18 – Código de identificação possível fraude de abastecimento

funcionario, vai armazenar todos os abastecimentos que tiveram este problema.

Uma variável também é incrementada, cada vez que se identifica um abastecimento possivelmente fraudado.

```
funcionarios_abastecimentos_fraudes = pd.DataFrame(lista_funcionario)
Valor_Funcionario_Falha = funcionarios_abastecimentos_fraudes.groupby(funcionarios_abastecimentos_fraudes[0]).sum()
```

Figura 19 – Código de arrecadação de abastecimentos possivelmente fraudados

Através da lista, podemos criar um novo dataframe, que vai fazer o somatório de todos os abastecimentos identificados como possíveis fraudes, e vai agrupá-los por Identifid, assim facilitará a análise de quanto cada frentista teve de abastecimentos identificados como possíveis fraudes.

Ao chamar o dataframe valor_funcionario_falha, podemos visualizar os dados em uma tabela, e enviar para a análise interna do posto, onde o dono do posto ou gerente, vai analisar o paradeiro deste motante.

## 5.4 Fraude de Horário

Todos postos de gasolina operam através de turnos e esses turnos controlam as horas de trabalho, de funcionamento do posto e também os caixas dos frentistas, ou seja, o quanto os frentistas arrecadaram no turno específico de trabalho.

Por padrão, os postos operam das 05:00h até as 00:00h, quando não são postos 24h. Para este projeto, todos os postos estudados, operam com o turno das 05:00 até a 00:00, fora deste intervalo de horário, os abastecimentos não devem ocorrer.

| | 0 |
|---|---|
| **0000000000000000** | 2.47 |
| **B3280909FAC312DF** | 5617.38 |
| **B3280909FAC39C8D** | 200.00 |
| **B3CF6C2EC1D38574** | 2008.98 |
| **B3CF894B37B9341C** | 4139.25 |
| **B3CF894B37B9C5E5** | 4082.81 |
| **B3CF89DF5B4DDDA6** | 800.00 |

Figura 20 – Visualização da arrecadação dos abastecimentos possivelmente fraudados

```
#Fraude de Horario
aux_Horario = 0

for i, row in df_posto1_falha.iterrows():
    if row['Hora'] > '00:00' and row['Hora'] < '04:59':
        df_posto1_falha.loc[i, 'Falha'] = 1
        aux_Horario = aux_Horario + 1


if (aux_Horario > 0):
    print('Houveram ' + str(aux_Horario) + ' abastecimentos fora do Horario.')
else:
    print ('Nao houve abastecimento fora do horario')
```

Figura 21 – Código possível fraude de horário

O algoritmo faz uma varredura no dataframe e identifica os abastecimentos que ocorreram fora do turno estipulado. Caso o abastecimento seja identificado, eles serão marcados como falha.

Uma variável também é incrementada, cada vez que se identifica um abastecimento possívelmente fraudado.

## 5.5 Machine Learning SVM

Com os abastecimentos devidamente identificados como falha ou não, a machine learning pode ser criada. Para isso, será usado os conceitos de Máquina de Vetores de Suporte, ou SVM, pois foi tomado como base o bom desempenho do SVM frente a identificação de falhas (SUETAKE, 2011).

A biblioteca sklearning será usada para a criação da máquina, bem como treinos e testes.

A primeira parte para a criação da SVM vai ser a preparação dos dados. Nesta fase, vai ser selecionado a amostra de 100 abastecimentos identificados com falha ou fraude e outros 100 abastecimentos que não contêm fraudes ou falhas. Esses dados vão ser extraídos do dataframe que consta todas os abastecimentos dos 5 postos.

```
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn import svm
```

Figura 22 – Biblioteca da SVM

O primeiro passo é importar as bibliotecas para a SVM, usando o comando 'from sklearn import svm' adicionamos a biblioteca do SVM para o projeto. Já o comando 'from sklearn.model-selection import train-test-split' importa a biblioteca que vai ser usada para treinar nossa SVM e gerar o classificador dos abastecimentos.

```
for i, row in df_posto1_falha[df_posto1_falha['Falha']==0].sample(100).iterrows():
    a = row['Quantidade'], row['PPU'], row['Valor']
    lista_dadosfalha.append(a)
    lista_target.append(row['Falha'])

for i, row in df_posto1_falha[df_posto1_falha['Falha']==1].sample(100).iterrows():
    a = row['Quantidade'], row['PPU'], row['Valor']
    lista_dadosfalha.append(a)
    lista_target.append(row['Falha'])

df_treino_iris = pd.DataFrame(lista_dadosfalha, columns = ['Quantidade','PPU' ,'Valor'])
df_target = pd.DataFrame(lista_target, columns = ['Falha'])
```

Figura 23 – Coleta de dados para treino e teste da máquina

A Figura 23 mostra o código que vai separar os dados para o teste e treinamento da SVM. Os dados vão ser coletados aleatoriamente, mas tendo a mesma quantidade de dados com falhas e dados sem falhas.

No primeiro laço do código, é selecionado 100 abastecimentos sem falhas, onde o filtro será dado na coluna Falha, nos abastecimentos marcados com 0. Já o segundo laço, segue a mesma lógica, porém filtrando os abastecimetnos que foram identificados com falha.

A partir destes dados, duas listas vão ser criadas, a primeira, chamada de lista_dadosfalha vai armazenar os dados dos abastecimentos, como quantidade, PPU (preço unitário) e valor. A SVM vai encontrar padrões de abastecimentos a partir destas três variáveis.

A segunda lista, chamada de lista_target, vai ser a lista que guardará os indicadores de falha dos abastecimentos. Essas duas listas vão ser usadas para os testes e treinos da SVM.

```
df_treino_iris = pd.DataFrame(lista_dadosfalha, columns = ['Data','Hora','Dia Semana', 'Quantidade','PPU' ,'Valor'])
df_target = pd.DataFrame(lista_target, columns = ['Falha'])
```

Figura 24 – Código para a criação de dataframes para treino e teste

Com as listas preenchidas, devemos salientar que a lista_target vai ser usada como classificador de cada abastecimento, pois a *machine learning* vai gerar o classificador se baseando nas classificações passadas pela lista-target.

Um novo dataframe vai ser gerado, chamado de df_treino_iris, onde serão criadas as colunas dia (que se refere ao dia do abastecimento), hora (que se refere a hora do abastecimento), dia semana, quantidade, ppu e valor. Os dados deste dataframe vai ser a lista_dadosfalha, alimentado anteriormente com o algoritmo de extração.

O dataframe df_target, vai ser criado apenas com os inteiros relacionados as falhas dos abastecimentos das amostras, que agora pertencem ao dataframe df-treino-iris.

```
X_train, X_test, y_train, y_test = train_test_split(
    df_treino_iris, np.ravel(df_target), test_size=0.30, random_state=0)
```

Figura 25 – Divisão de matrizes de dados para treinos e testes

Próximo passo vai ser dividir os dados em matrizes randômicas para usá-los em treinos e testes. O método 'train-test-split' faz esta divisão e alimenta as variáveis X_train, X_test, Y_train e Y_test. Que vão ser as variáveis usadas para os testes e treino.

Foi configurado para que 70% dos dados sejam usados para treino, o restante vai ser usado para os testes.

Com os dados divididos, vamos iniciar os testes. O método *'fit'* faz o treino dos dados, conforme a figura 26. É passado para este método os dados X de treino, que são os dados de quantidade, ppu e valor. O y de treino são os seus classificadores correspondentes.

Com os dados treinandos, podemos testar se a SVM consegue prever abastecimentos com falhas, para isso outros dados vão ser usados, no caso os X-teste.

Passado o X_test, a máquina vai retornar um array contendo os predictions, que são as análises que a SVM efetuou. A partir daqui com a SVM treinada, podemos ana-

```
In [444]: model.fit(X_train,y_train)

Out[444]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
            decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
            max_iter=-1, probability=False, random_state=None, shrinking=True,
            tol=0.001, verbose=False)
```

Figura 26 – Treino dos dados - SVM

```
predictions = model.predict(X_test)
```

Figura 27 – Teste dos dados - SVM

lisar se a acurácia dela está satisfatória. Acurácia significa a pontuação de confiança do classificador, quanto mais perto de 100% melhor.

Para avaliar se nossa máquina está tendo um bom desempenho, devemos importar um novo pacote do sklearn, conforme figura abaixo.

```
from sklearn.metrics import classification_report,confusion_matrix
```

Figura 28 – Import Confusion_Matrix - SVM

O confusion vai gerar uma matriz de confronto, para saber se os dados que a máquina previu, equivale com o que de fato já foi classificado. E a partir daí, vamos ter o nosso nível de acurácia. Para este caso, iremos passar o array do Y-test.

Para visualizarmos o resultado da acurácia vamos usar o método classification_report passando o Y-test e o array do prediction. E assim teremos:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.90 | 0.95 | 29 |
| 1 | 0.91 | 1.00 | 0.95 | 31 |
| avg / total | 0.95 | 0.95 | 0.95 | 60 |

Figura 29 – Resultado da SVM

Sendo assim, nossa SVM obteve 95% de acurácia, o que é um resultado de precisão elevado, e que vai crescendo conforme, o dataframe é alimentado com dados de abastecimentos futuros.

```
from pandas_ml import ConfusionMatrix
Confusion_Matrix = ConfusionMatrix(y_test, predictions)
Confusion_Matrix.print_stats()
```

```
population: 60
P: 25
N: 35
PositiveTest: 33
NegativeTest: 27
TP: 25
TN: 27
FP: 8
FN: 0
TPR: 1.0
```

Figura 30 – Matriz de Confusão da SVM

Porém devemos observar na Figura 29 que na coluna *precision* na linha 0 e na coluna *recall* na linha 1, os valores desses campos ficaram em 1.00, ou 100%. Isto é considerado *overfitting*. *Overfitting* é um termo usado em estatística para descrever quando um modelo estatístico se ajusta muito bem ao conjunto de dados anteriormente observado, mas se mostra ineficaz para prever novos resultados.

Com isso, este resultado não é satisfatório, pois existe um possível *overfitting* no resultado. Por conta disso é preciso acrescentar um código para parametrizar a SVM. No primeiro teste e treino, o próprio SVM escolheu que a parametrização em 'rbf' era a melhor escolha para o treino dos dados, porém foi encontrado um overfit. Pois a categorização dos dados escolhidos foi muito simples para o 'rbf' aprender. Sendo assim, foi parametrizado por linear à SVM. Os parâmetros do kernel selecionam o tipo de hiperplano usado para separar os dados. O uso de 'linear' usará um hiperplano linear (uma linha para dois planos, x e y).

```
model = svm.SVC(kernel='linear', C=1, gamma='auto')
```

Figura 31 – Parâmetro Kernel Linear

A Figura 31 mostra o código que deve ser adicionado antes de treinar a SVM. Este código mudará o treinamento da máquina para que ela se comporte de forma linear. O Kernel linear é usado quando os dados são separáveis linearmente, ou seja, podem ser separados usando uma única linha. É um dos kernels mais comuns a serem usados. É usado principalmente quando há um grande número de recursos em um

conjunto de dados específico.

Feito isso, treinaremos a SVM novamente, dessa vez coletando o valor de 1000 abastecimentos identificados com falhas e mais 1000 abastecimentos identificados sem falhas, bastando alterar para *"sample(1000)"* o parâmetro *sample* do código da Figura 23. Foi mantido o índice de uso de 70% dos dados para treino e o restante para testes. O que dará 1400 abastecimentos para treino, e 600 abastecimentos para testes.

O resultado obtido é diferente (Figura 32), com a média de precisão menor, porém mais real, pois a SVM contém margem para novos aprendizados. Conforme ela for treinando com mais dados, a sua acurácia vai, possivelmente, melhorar.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.97 | 0.86 | 296 |
| 1 | 0.96 | 0.71 | 0.82 | 304 |
| avg / total | 0.87 | 0.84 | 0.84 | 600 |

Figura 32 – Resultado da SVM Kernel Linear

Agora a SVM consegue prever falhas com 96% de acurácia, e consegue identificar um abastecimento sem nenhuma falha com 76% (Figura 32). Observando as colunas *precision*, *recall* e *f1-score* não encontramos mais nenhum *overfitting*, o que faz a SVM ter um resultado satisfatório. A média de acurácia da SVM ficou em 87%.

Podemos verificar a matriz de prediction da nossa SVM chamando a variável, que foi criada, chamada de 'predictions'.

A matriz de previsão (Figura 33) são os resultados que a SVM previu depois dos testes e treinos, e é usada para o confronto dos dados reais para descobrir o nível de acerto, ou acurácia.

Podemos conferir a coerência da previsão através da matriz de confusão. A matriz de confusão é o recurso da SVM que vai indicar quais são os abastecimentos identificados com falhas que de fato ocorreram, que vai ser classificado como positivo verdadeiro, ou 'TP'. Também vai indicar os abastecimentos erroneamente identificados com falha, classificado com 'FP'. E o mesmo vai acontecer para os abastecimentos não marcados com falhas e possíveis fraudes.

Na matriz de confusão, Figura 34, podemos perceber que foram selecionados 304 abastecimentos sem falhas, e 296 com falhas. Desses abastecimentos, houve 8

```
array([1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1,
       0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1,
       1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 1, 0, 1, 0, 1], dtype=int64)
```

Figura 33 – Matriz de previsão - SVM

```
from pandas_ml import ConfusionMatrix
Confusion_Matrix = ConfusionMatrix(y_test, predictions)
Confusion_Matrix.print_stats()
```

```
population: 600
P: 304
N: 296
PositiveTest: 223
NegativeTest: 377
TP: 215
TN: 288
FP: 8
FN: 89
```

Figura 34 – Matriz de Confusão

falsos positivos e 89 falsos negativos. Ao passo que tiveram 215 positivos verdadeiros e 288 positivos negativos.

Após estes resultados, será acrescentado mais três parametros como entrada de dados da SVM. Vão ser selecionado 1000 abastecimentos com falhas e mais 1000 abastecimentos normais. Vão ser incluídos como parametros, o dia do abastecimento, a hora do abastecimento e o dia da semana.

Conforme a Figura 35, novas amostras são coletadas e a máquina vai ser treinada de usando o parâmetro linear.

Após terminar o treino e testes, é feito um teste com outra amostra de dados para verificar a acurácia da SVM.

```
lista_dadosfalha = []
lista_target = []


for i, row in df_posto1_falha[df_posto1_falha['Falha']==0].sample(100).iterrows():

    dia_semana=0

    if(row['Dia Semana']=='Sunday'):
        dia_semana = 1
    elif (row['Dia Semana']=='Monday'):
        dia_semana = 2
    elif (row['Dia Semana']=='Tuesday'):
        dia_semana = 3
    elif (row['Dia Semana']=='Wednesday'):
        dia_semana = 4
    elif (row['Dia Semana']=='Thursday'):
        dia_semana = 5
    elif (row['Dia Semana']=='Friday'):
        dia_semana = 6
    elif (row['Dia Semana']=='Saturday'):
        dia_semana = 7

    data = row['Data'].strftime("%Y/%m/%d")[8:]
    hora = (row['Hora'])[0:2]
    a = data, hora, dia_semana, row['Quantidade'], row['PPU'], row['Valor']
    lista_dadosfalha.append(a)
    lista_target.append(row['Falha'])
```

Figura 35 – Nova Coleta de Dados

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.95 | 0.84 | 482 |
| 1 | 0.94 | 0.71 | 0.81 | 518 |
| avg / total | 0.85 | 0.83 | 0.83 | 1000 |

Figura 36 – Nova Parametrização da Coleta

Na Figura 35 temos o resultado da acurácia da SVM com 76% para abastecimentos identificados como normais, e 94% para abastecimentos identificados com falha. Na Figura 36, temos a matriz de confusão do teste.

A partir daí, será feito uma nova coleta de dados e será feito um novo treinamento da SVM, onde será parametrizado como ’rbf’ novamente. Será coletado 1000 amostras de abastecimentos com falha e mais 1000 amostras de abastecimentos sem falhas, para treinamento e testes.

Depois de treinada, a SVM vai testar uma nova amostra de 1000 abastecimentos para analisar a acurácia (Figura 38) e verificar se ocorrerá o possível *overfitting*.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.95 | 0.84 | 482 |
| 1 | 0.94 | 0.71 | 0.81 | 518 |
| avg / total | 0.85 | 0.83 | 0.83 | 1000 |

Figura 37 – Matriz Confusão Figura 35

Para analisar se ocorrerá *overfitting*, a matriz de confusão (Figura 39) será usada para entender quantos abastecimentos identificados como falsos positivos e falsos negativos existem, e se se a SVM conseguiu aprender a configuração dos dados.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.88 | 0.93 | 482 |
| 1 | 0.90 | 1.00 | 0.95 | 518 |
| avg / total | 0.95 | 0.94 | 0.94 | 1000 |

Figura 38 – SVM RBF

```
from pandas_ml import ConfusionMatrix
Confusion_Matrix = ConfusionMatrix(y_test, predictions)
Confusion_Matrix.print_stats()
```

```
population: 1000
P: 518
N: 482
PositiveTest: 577
NegativeTest: 423
TP: 518
TN: 423
FP: 59
FN: 0
TPR: 1.0
```

Figura 39 – Matriz de Confusão da Figura 38

Novamente a SVM parametrizada com ’rbf’ teve 100% reconhecimento dos

abastecimentos sem falhas, porém, ao analisar a matriz de confusão é notado que a SVM conseguiu prever, de fato, todos os abastecimentos sem falhas

Para entender se não há *overfitting* vão ser feitos 30 rodadas de testes com amostras diferentes para cada rodada, ao final, uma tabela com os resultados da matriz de confusão vai ser criada para visualização dos resultados. Na tabela vai estar contido o número da rodada, TP, FP, TN e FN.

| Rodada | TP | TN | FP | FN |
|---|---|---|---|---|
| 1 | 518 | 359 | 123 | 0 |
| 2 | 518 | 364 | 118 | 0 |
| 3 | 518 | 349 | 133 | 0 |
| 4 | 518 | 383 | 99 | 0 |
| 5 | 518 | 356 | 126 | 0 |
| 6 | 518 | 356 | 126 | 0 |
| 7 | 518 | 371 | 111 | 0 |
| 8 | 518 | 362 | 120 | 0 |
| 9 | 518 | 381 | 101 | 0 |
| 10 | 518 | 375 | 107 | 0 |
| 11 | 518 | 363 | 119 | 0 |
| 12 | 518 | 373 | 109 | 0 |
| 13 | 518 | 360 | 122 | 0 |
| 14 | 518 | 371 | 111 | 0 |
| 15 | 518 | 373 | 109 | 0 |
| 16 | 518 | 362 | 120 | 0 |
| 17 | 518 | 378 | 104 | 0 |
| 18 | 518 | 353 | 129 | 0 |
| 19 | 518 | 378 | 104 | 0 |
| 20 | 518 | 364 | 118 | 0 |
| 21 | 518 | 368 | 114 | 0 |
| 22 | 518 | 374 | 108 | 0 |
| 23 | 518 | 364 | 108 | 0 |
| 24 | 518 | 375 | 107 | 0 |
| 25 | 518 | 370 | 112 | 0 |
| 26 | 518 | 375 | 107 | 0 |
| 27 | 518 | 353 | 129 | 0 |
| 28 | 518 | 367 | 115 | 0 |
| 29 | 518 | 368 | 114 | 0 |
| 30 | 518 | 365 | 117 | 0 |
| Média | 518 | 367,43 | 114,56 | 0 |
| Mediana | 518 | 368 | 114 | 0 |

Tabela 8 – Tabela de Resultado dos 30 Testes SVM ’rbf’

Depois das 30 rodadas de testes, usando amostras de dados diferentes , é observado na tabela que a SVM configurada como ’rbf’ conseguiu aprender a identificar os abastecimentos marcados sem falha ou possível fraude. Mantendo a acurácia de

100% para abastecimentos normais, e 90% para abastecimentos identificados com falha.

Com a matriz de confusão (Figura 39) conseguimos perceber que não existe *overfitting* na SVM parametrizada com 'rbf', as rodadas de testes servem para identificar se a SVM está sendo estável.

## 5.6 Inconsistências

Algumas considerações sobre os dados foram citadas durante este projeto, mas é sempre bom lembrar que os dados que estamos usando são extraídos por técnicos de informática diretamente de um hardware industrial. Os valores recebidos nos logs são limitados, o que pode ser diferente quando este projeto evoluir e se comunicar diretamente com a automação, para se extrair os dados.

Os dados são extraídos dos logs, que são arquivos .txt. A garantia da integridade dos dados é passada pela empresa de suporte. Com isso, não se pode garantir a integridade dos dados.

## 5.7 Resultados e Conclusões

Após extrair os dados, criar algoritmos de identificação de falhas e fraudes e por fim criar uma máquina SVM para estudar os abastecimentos, tivemos um aprendizado de máquina com 95% de acurácia, mas foi reconhecido com um possível *overfitting*, dado que a SVM foi parametrizada automaticamente para 'rbf' que é tem uma caracteristisca de refinar melhor dados não lineares, podendo fazer "curva"na separação dos dados. Visto isso, a SVM foi reconfigurada para uma estrutura mais simples de separação dos dados, a linear. Após re-parametrizar a SVM, a máquina teve 87% de acurácia, onde foi 76% para abastecimento normal, e 96% para abastecimento com falhas e possíveis fraudes. O uso da matriz de confusão e da matriz de previsão ajuda a entender melhor a qualidade e nível de acerto da SVM, além de saber o quanto a máquina indica de previsões erradas.

Em seguida foi feito uma comparação entre a SVM configurada como linear e outra SVM configurada como rbf. Para essa nova etapa de treinamento e testes, foram adicionados mais parâmetros à SVM, como dia, hora e dia da semana do abastecimento. Depois das SVMs treinadas, foram efetuados testes com novas amostras de dados, e conferindo com a tabela de confusão, foi visto que a SVM parametrizada com 'rbf' obteve um resultado melhor. Logo após foram feitos 30 rodadas de testes com 1000 diferentes amostras de abastecimentos a cada rodada, para comprovar se houve, ou não, o *overfitting*. Após todo o processo, foi visto que não houve *overfitting*,

e que a SVM parametrizada como ’rbf’ foi a melhor *machine learning* para o objetivo do projeto, onde a SVM ’rbf’ consegue identificar os abastecimentos sem falhas, e tem uma pequena margem de erro na previsão de abastecimentos identificados com falha.

A geração de gráficos de distribuição e dispersão ajuda o usuário a entender possíveis fraudes, o que torna o projeto, minimamente, projeto de B.I., visto que o usuário vai ter os melhores dados para tomar as melhores decisões.

Um grande diferencial como ferramenta automatizada é a possibilidade de analisar vários postos ao mesmo tempo. É possível integrar este projeto, a qualquer sistema de gerenciamento de posto, e alimentá-lo com gráficos, dados descritivos e previsões com a SVM.

Também é totalmente viável integrar este projeto com algum outro serviço ou sistema, que carregará os arquivos de log de abastecimentos, automatizando mais o serviço de análise.

# 6 Conclusões Finais

Tendo em vista que este projeto vai ser aplicado comercialmente, o entendimento dos conceitos e a automatização dos serviços foram cruciais para afirmar que o resultado obtido pelo projeto é satisfatório. Além dos algoritmos de identificação, o uso do framework Pandas, nos traz a autonomia de, em análises, analisar mais dados em menos tempo do que a análise humana.

A criação da SVM traz um novo conceito para as análises de dados no ambiente de posto de gasolina. A partir daqui, pode-se usar SVM para entender os padrões encontrados nos abastecimentos, prever falhas e fraudes, e com isso nos dá mais recursos para mitigar dúvidas e sanar problemas em potencial.

Também podem ser criados outras técnicas de aprendizado de máquinas, como a Random Forest, e poderia haver uma comparação entre os dois métodos para saber quais dos dois se aplicariam melhor para o cenário de identificação de falhas e fraudes em postos de gasolina.

O aprendizado da análise de dados para postos de combustíveis vai ser explorado cada vez mais, visto que assim como todo comércio, estratégias de vendas e melhoramento de serviços vão ser aplicados. Mas este projeto começou com um olhar voltado para o suporte. Na medida que automatizamos extração de dados, análises de dados e podemos automatizar as decisões das análises.

## 6.1 Limitações

Como limitação do trabalho, foram selecionados postos que trabalham com turnos definidos, entre 05:00 e 00:00. Postos com 24h de funcionamento não foram selecionados para a pesquisa.

Por se tratar de dados reais de produção e cliente, e pelo fato de envolver uma empresa que tem políticas de segurança de dados, este projeto não explorou as possíveis validações de abastecimentos no banco de dados do sistema de gerenciamento de posto.

Uma vez autorizado o acesso ao banco de dados do sistema de gerenciamento de postos, conseguiríamos sinalizar, e até mesmo alterar abastecimentos sinalizados com falha ou fraude. Com por exemplo, os abastecimentos duplicados, uma vez com acesso ao banco de dados do sistema de gerenciamento de postos, esse projeto consegue identificar (via SVM) um abastecimento duplicado, e removeria algum abastecimento duplicado que caísse no banco de dados do sistema. O que iria evitar erro no

caixa dos frentistas, e evitaria o prejuízo do posto ter que pagar imposto sobre uma venda que não ocorreu.

## 6.2 Trabalhos Futuros

Este projeto vai ser usado pela Gilbarco, empresa que subsidia a Orpak, para que sejam aplicados os conceitos aprendidos, bem como a SVM e a automatização das análises junto com o sistema de gerenciamento de postos.

Uma vez que é possível criar métodos para analisar grande volumes de dados, este projeto vai ser melhorado continuamente adicionando outros modelos de automação, por exemplo. Com isso também vai ser possível criar mecanismos para que se escolha qual é a melhor automação para ser oferecida aos clientes, quando for preciso trocar.

O uso da SVM para indicar falhas e fraudes vai ser usado em um breve intervalo de tempo e ao que tudo indica, também vão ser criadas outras SVM que vão auxiliar no time de suporte da Orpak. Como por exemplo, uma nova SVM que vai avaliar o desempenho dos técnicos da empresa. Ou, outro exemplo, uma SVM que vai auxiliar na padronização de atendimento e melhor solução para novos técnicos de suporte que entrarem na empresa.

# Referências


ALMEIDA, D. *Gestão do Conhecimento na análise de falhas: mapeamento de falhas através de sistema de informação.* [S.l.]: SCIELO, 2006. Citado na página 21.

ANP. *LEI Nº 9.478, DE 6.8.1997*. Brasil, 1997. Disponível em: <http://www.planalto.gov.br/ccivil_03/LEIS/L9478.htm>. Citado 2 vezes nas páginas 12 e 19.

BNDES. *Mercado de Refino de Petróleo no Brasil*. [S.l.]: BNDES, 2018. Citado na página 18.

CAPELETTO, G. J. *Balanço energético do Rio Grande do Sul 2015*. [S.l.]: Secretaria de Infra-Estrutura e Logística do Rio Grande do Sul, 2015. Citado na página 12.

CARLEIAL, L. F. *Política econômica, mercado de trabalho e democracia: o segundo governo Dilma Rousseff*. [S.l.]: Revista Estudos Avançados, 2015. Citado na página 12.

CASALINHO, G. *O Impacto do Uso do Big Data na Inteligência Competitiva e na Percepção do Produto Pelo Cliente*. UNIVATES, 2017. Disponível em: <http://www.univates.br/revistas/index.php/estudoedebate/article/view/660>. Citado 2 vezes nas páginas 15 e 21.

COMPANYTEC. *Manual do Desenvolvedor - CBC 06*. [S.l.]: Companytec, 2005. Citado 6 vezes nas páginas 20, 24, 25, 33, 34 e 37.

DIAS, A. *Reflexiones sobre las finalidades de la enseñanza de las ciencias: educación científica para la ciudadanía*. Eureka, 2004. Disponível em: <https://revistas.uca.es/index.php/eureka/article/view/3968>. Citado na página 21.

FAGUNDES, M. *A Produção Científica Sobre Qualidade de Dados em Big Data: Um Estudo na Base de Dados Web Of Science*. UNICAMP, 2017. Disponível em: <https://periodicos.sbu.unicamp.br/ojs/index.php/rdbci/article/view/8650412>. Citado na página 21.

FNEMBRASIL. *Lei complementar Nº382*. Brasil, 2018. Disponível em: <http://fnembrasil.org/pe/>. Citado na página 13.

GRUS, J. *Data Science do Zero*. [S.l.]: Alta Books, 2016. Citado na página 22.

INCT. *Evolução da frota de automóveis e motos no Brasil 2001-2012*. Instituto Nacional de Ciência e Tecnologia., 2013. Disponível em: <http://www.observatoriodasmetropoles.net/download/auto_motos2013.pdf>. Citado na página 12.

JAVARONE, S. *Tecnologias digitais na produção e análise de dados qualitativos*. [S.l.]: UNESP, 2011. Citado na página 15.

KOHAVI, V. *Machine Learning 30*. [S.l.]: Scientific Research, 1998. Citado na página 22.

LEDERMANN, N. G. *Science teacher's conceptions of the nature of science: do they really influence teaching behavior*. [S.l.]: Science Education, 1987. Citado na página 21.

LORENA, A. *Uma Introdução às Support Vector Machines*. UFRGS, 2007. Disponível em: <http://www.seer.ufrgs.br/rita/article/download/rita_v14_n2_p43-67/3543>. Citado 2 vezes nas páginas 16 e 17.

MACHADO, F. *A utilização dos Métodos de Data Minig e Machine Learning no Processo de Prevenção À Fraudes no Mercado Segurador*. [S.l.]: IGTI - Instituto de Gestão da Tecnologia da Informação, 2017. Citado na página 17.

MATSUURA, J. *Redes Bayesianas e Aprendizagem Aplicada À Detecção de Falhas em Sistemas Dinâmicos*. [S.l.]: ITA, 2016. Citado na página 16.

MONARD, M. *Conceitos sobre Aprendizado de Máquina*. [S.l.]: USP, 2017. Citado na página 23.

NASCIMENTO, C. *Analytics: critical success factors on implementation in organizations.* USP, 2017. Disponível em: <https://www.teses.usp.br/teses/disponiveis/3/3136/tde-22062017-112809/pt-br.php>. Citado na página 21.

NASCIMENTO, C. *Inovação nos negócios por meio da Análise de Big Data*. SINGEP, 2017. Disponível em: <https://singep.org.br/6singep/resultado/333.pdf>. Citado na página 21.

SAKURADA, E. *Técnicas de Análise dos Modos de Falhas e seus Efeitos e Análise da Arvore de Falhas no desenvolvimento e na avaliação de produtor*. [S.l.]: UFSC, 2001. Citado na página 21.

SEFAZ. *Manual de orientação do contribuinte*. Brasil, 2015. Disponível em: <http://www.nfe.fazenda.gov.br/portal/exibirArquivo.aspx?conteudo=9hd38oni4Nc=>. Citado 3 vezes nas páginas 12, 15 e 19.

SEFAZ. *decreto no 44.650*. Brasil, 2017. Disponível em: <https://www.sefaz.pe.gov.br/Legislacao/Tributaria/Documents/Legislacao/44650/Texto/Dec44650_2017.htm#art147>. Citado 4 vezes nas páginas 12, 19, 36 e 38.

SPECTRUM, I. *Ranking programming languages*. UFSC, 2019. Disponível em: <https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2019>. Citado na página 22.

SUETAKE, M. *Sobre a Aplicação de Sistemas Inteligentes para Diagnóstico de Falhas em Máquinas de Indução - Uma Visão Geral*. [S.l.]: USP, 2011. Citado 3 vezes nas páginas 16, 17 e 41.

TEIXEIRA, E. *A Análise de Dados na Pesquisa Científica.* Editora Unijuí, 2003. Disponível em: <https://www.revistas.unijui.edu.br/index.php/desenvolvimentoemquestao/article/view/84/41>. Citado na página 15.

VELLOSO, H. *Inteligência Artificial para Detecção de Falhas em Equipamentos: Um Estudo Bibliométrico no Setor de Óleo e Gás*. [S.l.]: IFF Fluminense, 2017. Citado 2 vezes nas páginas 16 e 17.