UNIVERSIDADE DE SÃO PAULO ESCOLA POLITÉCNICA
DEPEARTAMENTO DE ENGENHARIA DE PRODUÇÃO

GUSTAVO RIBEIRO AGUERA

DESENVOLVIMENTO DE UM MODELO DE SELEÇÃO DE CLIENTES PARA CONTATO POR WHATSAPP

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Diploma de Engenheiro de Produção

São Paulo
2020

UNIVERSIDADE DE SÃO PAULO ESCOLA POLITÉCNICA
DEPEARTAMENTO DE ENGENHARIA DE PRODUÇÃO

GUSTAVO RIBEIRO AGUERA

DESENVOLVIMENTO DE UM MODELO DE SELEÇÃO DE CLIENTES PARA CONTATO POR WHATSAPP

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Diploma de Engenheiro de Produção

Orientador:
Prof. Alberto Wunderler Ramos

São Paulo
2020

FICHA CATALOGRÁFICA

Aguera, Gustavo Ribeiro

Desenvolvimento de um modelo de seleção de clientes para contato por whatsapp, por G. R. Aguera. São Paulo : EPUSP, 2020, 110 p.

Trabalho de formatura – Escola Politécnica da Universidade de São Paulo. Departamemto de Engenharia de Produção

1. Aprendizagem estatística. 2. Aprendizagem de máquina. CRM. 3. Business Intelligence. Data warehouse. Marketing digitalI.Universidade de São Paulo. Escola Politécnica. Departamente de Engenharia de Produção II.t.

DEDICATÓRIA

Dedico este trabalho à minha família, à minha universidade e à ciência.

AGRADECIMENTOS

Agradeço principalmente à minha família, que durante todos os anos de preparação para o vestibular e de graduação me deram todo o apoio necessário. Agradeço a meus pais José Tadeu e Mara Helena por sempre me apoiarem em minhas decisões e a minhas irmãs Natália e Luiza, por sempre serem além de irmãs, grandes amigas em que pude confiar. Especialmente, agradeço à Luiza cuja ajuda na revisão deste trabalho foi essencial para sua qualidade.

Agradeço também a meus amigos Beatriz, Kaíque, Naetê e Raphael que sempre foram fundamentais na minha vida e me ajudaram muito em todo o período de minha graduação. Agradeço também a todos amigos que fiz durante estes anos na Escola que, além de me transformarem em uma pessoa muito melhor do que eu era antes, fizeram desses anos os melhores de minha vida.

Agradeço também ao meu orientador Alberto Ramos pelo apoio durante a elaboração do trabalho.

Agradeço também à Elementar, e todas as pessoas que lá trabalham, por me dar todo apoio e liberdade que precisei na elaboração deste projeto.

Por fim, agradeço à Escola Politécnica pela formação como engenheiro de produção.

“There is no shortcut to truth, no way to gain a knowledge of the universe except through the gateway of scientific method.”

(Karl Pearson)

RESUMO

O objetivo deste trabalho é documentar o processo de criação de um sistema de seleção de clientes em uma base de compras *offline* para contato através de Whatsapp. Este trabalho detalha todo processo de criação de um *data warehouse* em SQL e a criação de um modelo de seleção de clientes através de técnicas de *machine learning*.

O trabalho teve como resultado, além da venda do sistema de contatos, um *conversion lift* de 32% nas vendas de cada pessoa contatada.

Palavras-chave: Aprendizagem estatística. Aprendizagem de máquina. CRM. Business Intelligence. Data warehouse. Marketing digital.

ABSTRACT

The objective of this work is to document the process of creating a customer selection system for contact through Whatsapp. This work details the entire process of creating a data warehouse in SQL and the creation of a customer selection model using machine learning techniques.

The work resulted in, in addition to the sale of the contact system, a 32% conversion lift in the sales of each person contacted.

Keywords: Statistical learning. Machine learning. CRM. Business Intelligence. Data warehouse. Digital marketing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API - *Application programming interface* (Interface de programação de aplicativos)

AWS - *Amazon Web Services* (Serviços Web da Amazon)

B2B - *Business to Business* (De empresa para empresa)

B2C - *Business to Consumer* (Negócio para consumidor)

BI - *Business Intelligence* (Inteligência de Negócio)

COVID-19 - *Coronavirus Disease* 2019

CRM - *Customer Relationship Management* (Gerenciamento de Relacionamento com o Cliente)

DA - *Discriminant Analysis* (Análise Discriminante)

ID (id) - Identificação

KNN - *K nearest neighbors* (K vizinhos mais próximos)

KPI - *Key Performance Indicator* (Indicador Chave de Desempenho)

LDA - *Linear Discriminant Analysis* (Análise Discriminante Linear)

QDA - *Quadratic Discriminant Analysis* (Análise Discriminante Quadrática)

RDBMS - *Relational Database Management System* (Sistema de gerenciamento de banco de dados relacional)

SI - Sistemas de Informação

SQL - *Structured Query Language* (Linguagem de Consulta Estruturada)

TI - Tecnologia da Informação

UTM - *Urchin Tracking Module* (Módulo de rastreamento de Urchin)

SUMÁRIO

1. INTRODUÇÃO

Primeiramente, é essencial para o desenvolvimento deste trabalho entender o contexto em que ele se passa. Esse capítulo é responsável por mostrar e contextualizar as duas empresas envolvidas no projeto e introduzir o porquê de sua realização.

A *Elementar Digital* é uma empresa que desenvolve tecnologia para otimização de marketing digital. A principal atividade da empresa é gerir as campanhas de marketing em diversas plataformas (*Google*, *Facebook* entre outros) e analisar a performance de vendas do cliente.

A empresa criada no início de 2018 conta hoje com 14 pessoas, que se dividem em atividades de *Business Intelligence*, análise de dados, design e TI.

A empresa conta hoje com 9 clientes, entre eles:

- Construtora Tenda: empresa com mais de 49 anos, já construiu mais de 100 mil habitações. A Tenda se encontra hoje como uma das maiores construtoras e incorporadoras do país focada em empreendimentos econômicos. A empresa já está presente em mais de 100 cidades, espalhadas por 11 estados, além do Distrito Federal, e conta atualmente com mais de 40 lojas próprias.
- *Lendico*: Uma *Startup* de empréstimo para pessoa física. A empresa nasceu na Alemanha com o propósito de oferecer solução adequada para quem busca empréstimo pessoal com taxas de juros justas.
- *Grand Cru*: A *Grand Cru* é a maior importadora e distribuidora especializada em vinhos de qualidade da América Latina. Com sólida atuação *omni-channel*, oferece mais de 2000 rótulos do mundo todo em mais de 60 pontos-de-venda de Manaus a Porto Alegre. Além das franquias e operações próprias, tem *e-commerce*, clube de vinhos e distribuição aos melhores restaurantes, hotéis e empórios. Com 15 anos de tradição, é reconhecida como sinônimo de qualidade pela experiência única que proporciona aos apaixonados por vinhos, aos experts e aos iniciantes.

A empresa vende seu serviço a preço fixo, cada venda é negociada individualmente, podendo ter valores distintos para o mesmo serviço em razão da extensão diferente, geralmente o valor cobrado mensalmente pelo serviço aumenta conforme o volume de investimento do cliente aumenta. Além disso, têm projetos extras de *data warehouse* vendidos para alguns clientes à preço fixo, que envolve processamento e integração de dados e a disposição deles através da plataforma do *Google* de BI chamada *DataStudio*.

1.1. CONTEXTO

Nesse trabalho, é necessário entender o contexto próprio da agência, que envolve o mercado de marketing digital, e também o contexto do cliente selecionado, que é uma empresa do ramo de varejo têxtil que atua no setor de cama, mesa e banho, sua peculiaridade de clientes possibilita a análise do tema a ser desenvolvido. A empresa será chamada de "Empresa de Varejo" neste trabalho com objetivo de proteger dados dados estratégicos da empresa.

1.1.1. Contexto da empresa

Segundo pesquisa realizada em 2013 pela Adobe 76% dos profissionais de marketing acreditam que o mesmo mudou mais nos últimos 2 anos (com relação ao ano de realização da pesquisa) do que nos últimos 50 (ADOBE, 2013).

A veiculação de publicidade em massa começou em 1941 com a veiculação de uma propaganda da *Bulova* em televisões americanas. Antes da internet, o problema era mais simplificado, tínhamos diferentes horários de veiculação em diferentes emissoras, a resolução do problema era feita com ferramentas simples de pesquisa operacional, estimando qual seria o número de vendas advindas da veiculação em cada posicionamento, otimizava-se o custo por venda.

Desde o surgimento do *Google* em 1998 e do *Blogger* em 1999, o mercado vem sofrendo drásticas mudanças. Hoje trabalha-se com uma quantidade enorme de informação e empresas hoje utilizam, para a otimização de seus lucros, modelos estatísticos, *Data Science*, modelos estocásticos, entre outros.

87% dos usuários de internet agora têm um *smartphone* (MCGRATH, 2016). Atualmente estamos passando por uma grande mudança para o formato *mobile*, o *Google*, por exemplo, desde 2015 favorece anúncios que levam para páginas *mobile-friendly*.

A empresa se encaixa hoje em um contexto onde todas empresas B2C (*Business-to-consumer*) são extremamente dependentes do marketing digital (isso também acontece em menor intensidade em modelos de negócio B2B), 66% dos profissionais acreditam que as empresas não serão bem-sucedidas ao menos que tenham um *approach* de marketing digital (ADOBE, 2013).

Outro fator que compõe esse contexto é a grande deficiência das empresas na área, 82% dos profissionais da área não possuem treinamento formal e apenas 9% dos entrevistados concordam fortemente com a declaração "Eu sei que nosso marketing digital está funcionando" (ADOBE,

2013). Ainda é importante salientar que a pesquisa foi feita nos Estados Unidos, que é o berço das maiores empresas do setor, podemos projetar um cenário muito mais precário no Brasil.

O mercado de publicidade digital vem a cada ano crescendo e mostra que é o futuro modal de propaganda. A publicidade digital no Brasil cresceu 25,4% entre 2016 e 2017, saltando de R$ 11,8 bilhões (2016) para R$ 14,8 bilhões. Os dados fazem parte da pesquisa "Digital AdSpend 2018", realizada pelo IAB Brasil (Interactive Advertising Bureau), entidade que congrega mais de 250 empresas, entre anunciantes, agências, veículos e empresas de tecnologia. Além de confirmar as previsões anteriores, os números mostram que o segmento digital já representa um terço do total investido em publicidade no País e revelam a chegada de um dinheiro totalmente novo (E-COMMERCE NEWS, 2018).

1.1.1.1. Mail Marketing

Um dos tipos de mídia que compõe o contexto do marketing atual são os disparos de e-mail. O *mail marketing* é feito através de disparos de e-mail para uma base de e-mail, que, normalmente, é composta por clientes que já compraram ou que alguma vez demonstraram intenção de compra. Outro método, muito menos eficiente, é através de compra de base de e-mail. Atualmente trabalhamos apenas com a primeira opção que ganha o nome de *remarketing*.

> "Coletar e-mails de visitantes do seu *website* é uma alternativa muito eficaz para aumento da receita, isso se dá pelo fato do anúncio atingir uma pessoa que já conhece seus produtos e sua marca, além disso, o formato é muito bem aceito pelo cliente, 79% das pessoas gostam de receber e-mails" (ROGGIO, 2011).

> "*A successful ecommerce marketing strategy is an ever-changing blend of techniques and tactics aimed at achieving specific, predetermined goals. If executed well, email marketing is often one of the most effective means of achieving those goals"* (ROGGIO, 2011).

A comunicação de e-amil marketing deve ser o mais eficiente possível, uma vez que o quão mais eficiente ela é, maior vai ser a capacidade da empresa de reter clientes, aumentando o *lifetime value* e assim aumentando a receita da empresa a lomgo prazo e também viabilizando seu negócio.

1.1.1.2. Facebook Ads

Facebook, lançado em 2004, conta hoje com mais de 2,3 bilhões de usuários onde cerca de 130 milhões são brasileiros (TECMUNDO, 2019).

A *Facebook Inc.* conta hoje não apenas com a plataforma do *Facebook*, mas também com o *WhatsApp* (este ainda não possui nenhum tipo de publicidade), *Instagram* e *Facebook Messenger*, a compra de todas essas mídias é feita através da plataforma unificada: *Facebook Business*.

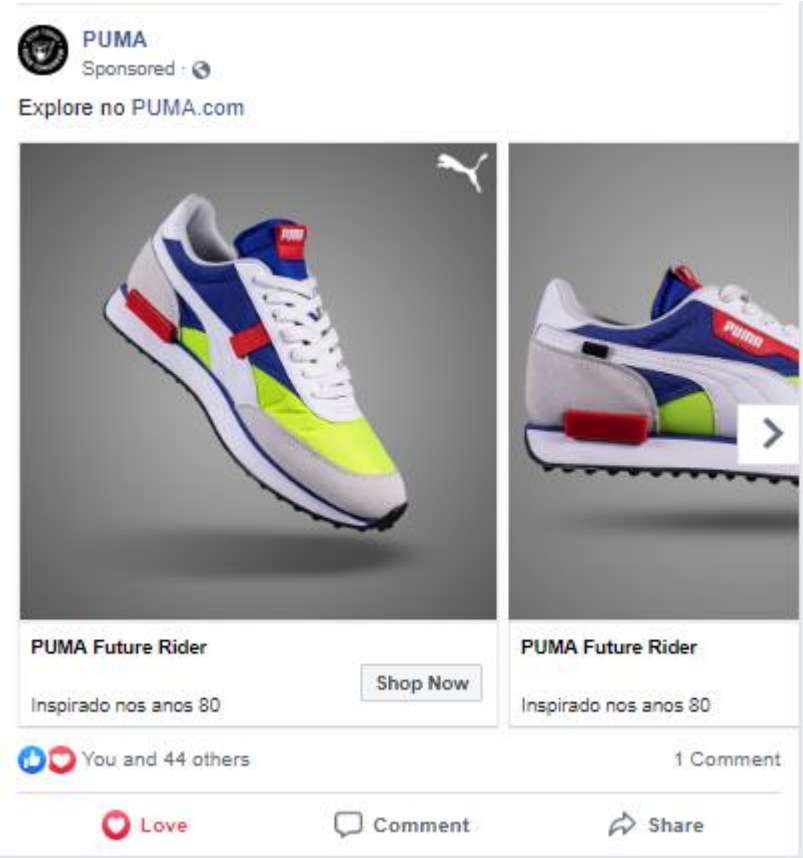
A maioria das pessoas trata o Facebook como sinônimo da rede social de mesmo nome. Entretanto, a Facebook Inc. é muito mais do que isso. Junto com o WhatsApp e o Messenger, a companhia é dona de três das maiores mídias sociais e serviços de mensagens do mundo. Só o Facebook alcança mais de 2 bilhões de pessoas por mês e tanto o WhatsApp quanto o Messenger também ultrapassaram o marco de 1 bilhão recentemente. A Tencent, empresa chinesa por trás do WeChat e do Qzone, está perto de 1 bilhão de usuários no total, mas ainda não chega perto das marcas da Facebook Inc. (FORBES, 2017)

O *Facebook* possui hoje um *market share* de 22,1% da receita de anúncios digitais nos Estados Unidos (STATISTA, 2019).

O diferencial da plataforma é a veiculação de anúncios nativos que se confundem com posts deixando-os mais atrativos para potenciais clientes. A plataforma também favorece criativos de vídeo. Em 2019, 80% do tráfego de internet será de vídeo. Nos EUA, o vídeo vai ser responsável por 85% do total (CISCO, 2019).

O *Facebook* se destaca também porque, uma vez que é uma rede social, possui uma enorme quantidade de informações sobre o comportamento de seus usuários. A plataforma disponibiliza a anunciantes a segmentação de audiência por interesses pré-estabelecidos, segmentações que utilizam também técnicas de clusterização, tema que será discutido mais a fundo nesse trabalho.

Figura 1: Anúncio de Facebook



Fonte: Captura feita pelo autor na plataforma do Facebook

1.1.1.3. Google Ads

Em 23 de Outubro de 2000 lançou o programa do AdWords (Hoje conhecido como Google Ads), no momento, a empresa já era a ferramenta de busca mais famosa e potente do mundo.

O programa AdWords se trata de um *Self-Service Advertising Program* como definido pela própria empresa, sendo, portanto, uma plataforma que possibilita a compra de palavras chaves de pesquisas de usuários por parte de pessoas físicas e jurídicas.

A ferramenta funciona a base de um leilão por palavras chave, o vencedor do leilão aparece na primeira posição do Google, é possível que mais de um anunciantes apareçam. A empresa é cobrada pelo anúncio quando um usuário do Google clica no anúncio.

Existem três fatores que determinam quais anunciantes aparecerão na busca e em qual ordem (GOOGLE ADS HELP):

- *Bid*: é definido pela estratégia (dentre as disponibilizadas pelo Google) selecionada pelo anunciante e quanto ele está disposto a pagar em um *click*;

- Qualidade dos anúncios: O Google possui um indicador de desempenho próprio chamado *quality score* que é uma nota de 1 a 10 baseada na esperança da taxa de clique, relevância do anúncio e experiência na página destino (GOOGLE ADS HELP);
- O possível impacto das extensões de anúncio e de outros formatos de anúncio.

O Google Ads também é composto pela *Google Display Network*, que funciona como um *marketplace* de banners de anúncios em sites. É composta por anunciantes que criam suas campanhas pelo Google Ads e por proprietários de sites que disponibilizam esse espaço.

O domínio do Google no mercado de anúncios ainda é brutal, em 2019 a empresa obteve um *market share* de 32,3% para todo mercado de anúncios digitais (STATISTA, 2019) e 73,1% para anúncios de busca (CNBC, 2019).

Figura 2: Anúncios de Google gerados ao buscar “lençol de seda” na plataforma

Fonte: Pesquisa realizada pelo autor na plataforma do Google

Figura 3: Anúncios da Google Display Network mostrado no site do Estadão

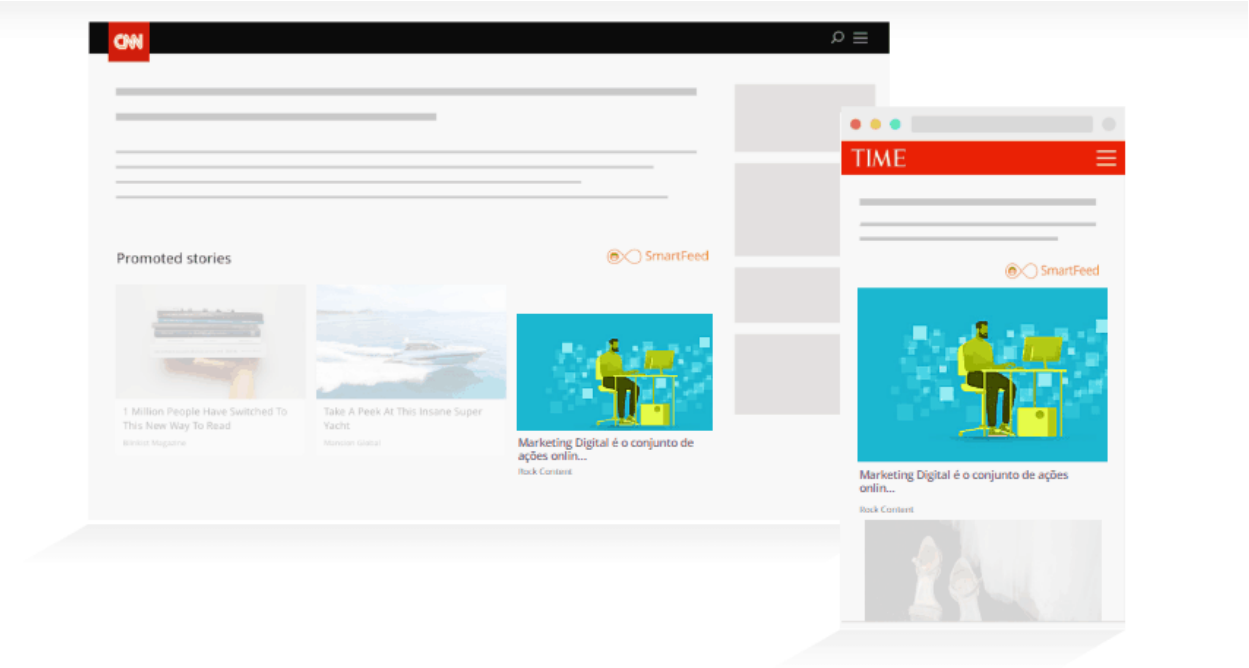
Fonte: Captura realizada pelo autor no site https://www.estadao.com.br/

1.1.1.4. Outras plataformas de marketing

Existem ainda outros players no mercado de anúncios, ferramentas de pesquisa menores como o Bing, DuckDuckGo, entre outros, são alternativas para o investimento em mídias pagas, porém o tamanho desses players não se compara ao do Google, como visto na seção anterior.

Existem também ferramentas de anúncios nativos como o Outbrain ou Taboola que são anúncios inseridos normalmente em páginas de notícias, cujo formato lembra o formato de notícias, dessa forma o anúncio se camufla no site e a taxa de clique aumenta.

Figura 4: Formato de anúcio nativo



Fonte: MORAES, 2019

1.1.2. Contexto do cliente

A Empresa de Varejo é uma empresa do ramo têxtil com décadas de atuação.

A partir de de 2009, a empresa passou a atuar no ramo de varejo de utensílios de cama, mesa e banho através das diversas marcas. As duas que aparecerão neste trabalho serão chamadas de “Marca A” e “Marca B”, para manter o anonimato da empresa. A Elementar digital faz a gestão do marketing digital do varejo da Empresa de Varejo, portanto toda publicidade é feita em torno destas marcas e não em cima do próprio nome da Empresa de Varejo.

A indústria têxtil brasileira é a quinta maior do mundo e existe a quase 200 anos no país. O faturamento da cadeia têxtil em 2018 foi de US$ 48,3 bilhões, apresentando queda quando comparada ao período de 2017, que teve faturamento de US$ 52,2 bilhões (ABIT, 2019).

A indústria é composta por 25,2 mil empresas que geram 1,5 milhão de empregos, sendo que 75% são de mão de obra feminina (ABIT, 2019).

A Empresa de Varejo, no período de 2018, apresentou faturamento de R$ 1,78 bilhões e lucro líquido de R$ 231.59 milhões (ADVFN, 2019), mostrando que é um dos grandes *players* da

indústria, principalmente se formos considerar somente a categoria de cama, mesa e banho, que é a única em que a empresa participa.

Hoje, grande parte da venda ainda é proveniente das lojas físicas, possuindo uma participação das vendas *online* mais reduzida. Porém essa venda *offline* consegue captar dados dos clientes, criando uma base de clientes e de pedidos disponíveis para utilização.

No início desse projeto a empresa ainda não possuía nenhuma iniciativa de canal de comunicação com esses clientes *offline*.

As lojas físicas possuem de 5 a 10 vendedores, esses vendedores compõem suas rendas principalmente à base de comissão. É fácil imaginar que esses vendedores possuam altas taxas de ócio. Os mesmos, por iniciativa própria, contatam um número reduzido de clientes conforme coleções novas vão saindo. Porém isso é feito sem controle ou mensuração nenhuma.

Vê-se uma oportunidade clara de negócio ao se criar um sistema de seleção de clientes, atribuição a lojas e vendedores e mensuração de resultados.

1.2. ESTRUTURA DO TRABALHO

Estre trabalho é divido em 15 sessões, que podem ser grupados nas seguintes partes:

- A primeira parte abrange os capítulos 1 a 4 que são de caráter introdutório, com o objetivo de descrever o contexto do trabalho, a metodologia adotada, seu objetivo e também descrever toda bibliografia utilizada no seu desenvolvimento;
- A segunda parte, capítulos de 5 a 7, descreve o planejamento e a consolidação de uma amostra para o modelo estatístico de previsão, essa metodologia é utilizada tanto no piloto quanto no projeto completo;
- A terceira, capítulos de 8 a 10, descreve o desenvolvimento e os resultados do processo de cração do modelo de previsão, esta metodologia também foi utilizada nas duas frentes;
- A quarta parte, capítulos 11 e 12 , descreve o desenvolvimento de um piloto, tratando-se de um teste prático do modelo de previsão desenvolvido.
- A quinta parte, capítulos 13 e 14, descreve como se deu o desenvolvimento da estrutura que possibilitou a alimentação de contatos e a mensuração de desempenho da plataforma de contatos;
- A última parte, capítulos 15 e 16, descreve resultados e conclusões alcançadas através do desenvolvimento deste trabalho.

2. METODOLOGIA

Se trata de um trabalho de caráter de desenvolvimento, portanto este trabalho é composto por descrições de desenvolvimento de cada etapa do projeto, experimentos de validação e análises finais de resultado.

O desenvolvimento descrito no trabalho foi composto por duas frentes distintas: uma em SQL, outra em R. A frente desenvolvida em SQL seguiu uma metodologia prória da Elementar, desenvolvida por seus funcionários nos últimos anos, que permite melhor organização de códigos e dados.

O desenvolvimento em R abrange a parte estatística de desenvolvimento de modelos de classificação, a metodologia utilizada foi a de aprendizagem estatística no qual são testados diferentes modelos e seleciona-se o modelo ganhador com menor erro de teste.

Foram adotadas três metodologias de validação distintas neste trabalho, o que torna seu resultado extremamente confiável. São elas:

- Validação através da metodologia de aprendizageme estatística no qual o calculo do erro de teste é uma validação eficiente;
- Validação através da criação de um piloto que se trata de um experimento em menor escala condizido pelo próprio autor com objetivo de obter regulariade entre dados reais e dados do modelo;
- Validação através da análise dos resultados finais.

Na descrição de resultados foi utilizada a metodologia de *conversion lift* que hoje é amplamente utilizada no setor de *marketing* digital. Também foram utilizados os conceitos de regressão linear e correlação.

3. DEFINIÇÃO DO OBJETIVO

Antes do planejamento específico das atividades e de sua execução é necessário traçar os objetivos do trabalho e assim fazer que que o objetivo final seja alcançado da melhor maneira possível.

Este capítulo descreve o objetivo geral, que é uma visão do projeto como um todo e define o que deve ser entregue ao cliente. Para que o objetivo final seja alcançado, é necessário traçar objetivos intermediários que organizarão o desenvolvimento do projeto.

3.1. OBJETIVO GERAL

Nosso objetivo neste projeto será a comunicação com clientes através do WhatsApp, tal atividade se enquadra como CRM. Nosso objetivo final é o desenvolvimento de uma plataforma onde vendedores de loja física possam contatar clientes enquanto estão ociosos. A atribuição de clientes para vendedores será feita pela própria plataforma.

Para que a atribuição ocorra, planejamos a construção de um modelo capaz de prever potenciais compradores, com o intuito de otimizar o máximo possível o tempo do vendedor e conseguir contatar clientes com alta propensão de compra. Para isso será selecionado o modelo estatístico de classificação que possui menor erro.

Como seria inviável mandar mensagem para todos clientes de todas as lojas físicas, optamos por selecionar uma loja específica da Marca A. Tal seleção se deu pois é a loja do que possui maior número de clientes, o que aumenta a amostra de nosso teste, e porque o gerente desta loja é um funcionário de longa data, portanto justificar o projeto e alinhar os detalhes será muito mais fácil. Durante o período de teste o autor se passou pelo gerente da loja e o avisou sobre clientes que possuíam intenção de compra.

No período inicial de teste nosso plano é enviar 10 mensagens ao dia, posteriormente aumentaremos para 30 mensagens.

3.2. OBJETIVOS INTERMEDIÁRIOS

Para que esse objetivo final seja alcançado, será necessário o desenvolvimento das seguintes etapas:

I. Planejamento do piloto (teste);
II. Planejamento do modelo: levantamento de variáveis dependente e independente;
III. Criação de uma estrutura em SQL capaz de criar uma amostra para ser utilizada no desenvolvimento do modelo;
IV. Seleção de do melhor modelo estatístico através de técnicas de aprendizagem estatística;
V. Utilização dos coeficientes do melhor modelo para seleção de contatos do piloto;
VI. Análise dos resultados do piloto e validação do modelo e do projeto;
VII. Expansão do modelo estatístico com dados de todos clientes da base;
VIII. Integração da seleção de clientes com a plataforma de contatos;
IX. *Rollout* da plataforma de contato para os vendedores da Empresa de Varejo;

X. Criação de uma estrutura de mensuração;

XI. Análise de resultados.

4. REVISÃO BIBLIOGRÁFICA

Este capítulo descreve toda base teórica necessário para a elaboração deste trabalho. A descrição foi feita de maneira mais objetiva possível, para aprofundamento o leitor deve ler o material presente nas referências.

4.1. CUSTOMER RELATIONSHIP MANAGEMENT (CRM)

O CRM (Customer Relationship Management) é uma abordagem de gerenciamento da interação de uma empresa com clientes atuais e potenciais. Ele usa análise de dados sobre o histórico dos clientes com uma empresa para melhorar as relações comerciais, concentrando-se especificamente na retenção de clientes e, com isso, impulsionando o crescimento das vendas.

> "*CRM initiatives have resulted in increased competitiveness for many companies as witnessed by higher revenues and lower operational costs. Managing customer relationships effectively and efficiently boosts customer satisfaction and retention rates*" (Chen & Popovich, 2003)

4.2. SQL PRESTO

SQL é uma linguagem específica de bancos de dados usada na programação e criada para gerenciar dados mantidos em um sistema de gerenciamento de banco de dados relacional (RDBMS) ou para processamento de fluxo em um sistema de gerenciamento de fluxo de dados relacional. É particularmente útil no tratamento de dados estruturados, isto é, dados que incorporam relações entre entidades e variáveis.

O Presto uma linguagem de SQL de alto desempenho para big data. Sua arquitetura permite que os usuários consultem uma variedade de fontes de dados, como Hadoop, AWS S3, Alluxio, MySQL, Cassandra, Kafka e MongoDB. Pode-se até consultar dados de várias fontes em uma única consulta. O Presto é um software de código aberto voltado para a comunidade, lançado sob a Licença Apache.

O Presto será utilizado neste trabalho pois a Elementar utiliza como estrutura de seu servidor o AWS (*Amazon Web Services*) e seu consultor de dados, chamado "Athena", utiliza a linguagem Presto para consultar dados do AWS S3.

4.3. APRENDIZAGEM ESTATÍSTICA

O conceito de aprendizagem estatística entrou muito em voga nos últimos anos com o nome de *machine learning*. Essa área ganhou força na última década uma vez que foram desenvolvidos

computadores com alta capacidade de processamento, possibilitando o cálculo de modelos mais complexos e o processamento de uma grande quantidade de dados. Aprendizagem estatística é, portanto, uma união entre a área de estatística e de computação.

4.3.1. Local de execução

Para executar as técnicas de aprendizagem estatística é necessária a utilização de linguagens de programação, uma vez que ainda não existem *softwares* que são capazes de executar todo o processo.

As linguagens de programação mais utilizadas para esse tipo de projeto são R e python. Enquanto o R tem uma interface mais organizada (RStudio) e um foco maior em aplicações estatísticas, o python tem como vantagem ser uma linguagem mais versátil, sendo possível a utilização de APIs, e ter um tempo de processamento um pouco menor. Porém, ambas as linguagens são eficientes e são utilizadas na indústria, acabando por ser selecionada a linguagem que o programador prefere.

Neste projeto será utilizada a linguagem R, através do *software* RStudio, uma vez que o autor tem maior conhecimento desta linguagem e o livro base selecionado (GARETH, J. et al.) possui exemplos em R.

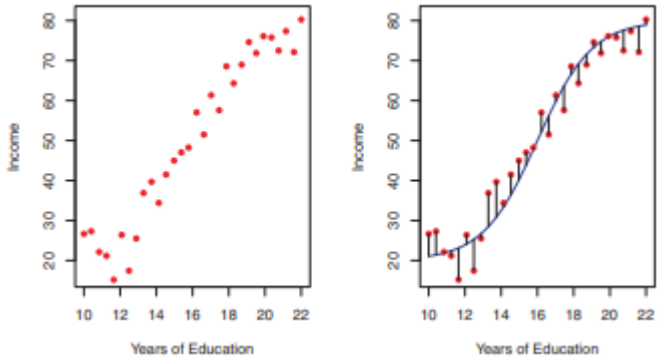
4.3.2. Seleção do melhor modelo

A aprendizagem estatística utiliza de modelos clássicos da estatística, como regressão linear ou LDA, e modelos modernos, como *random forest* e *neural network*, que agora são viáveis computacionalmente.

Cada modelo é responsável por calcular uma f que é uma função de previsão de uma variável dependente em função de p variáveis dependentes.

Assim, o processo de seleção de modelos consiste em calcular diferentes f's usando diferentes modelos e técnicas, com isso é possível calcular o erro de cada modelo, comparando dados reais com os valores previstos pela f. Seleciona-se então o modelo com menores erros de previsão.

Figura 5: Pontos experimentais, *f* e erros de previsão

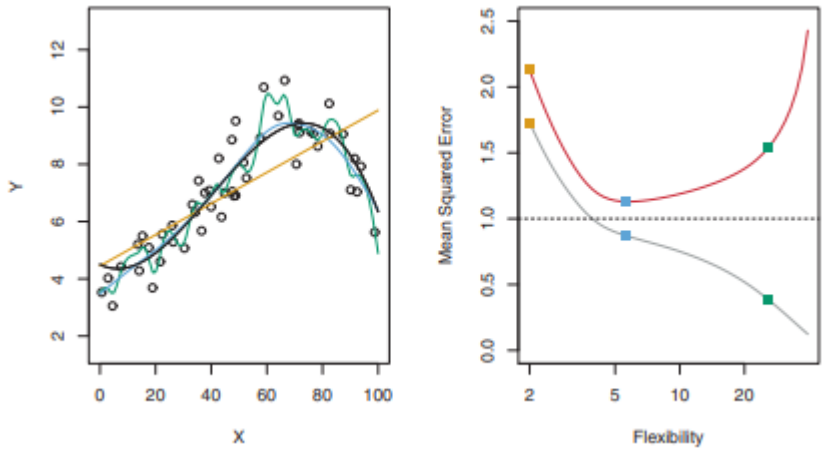


Fonte: An Introduction to Statistical Learning

4.3.3. Por que separar entre treino e teste?

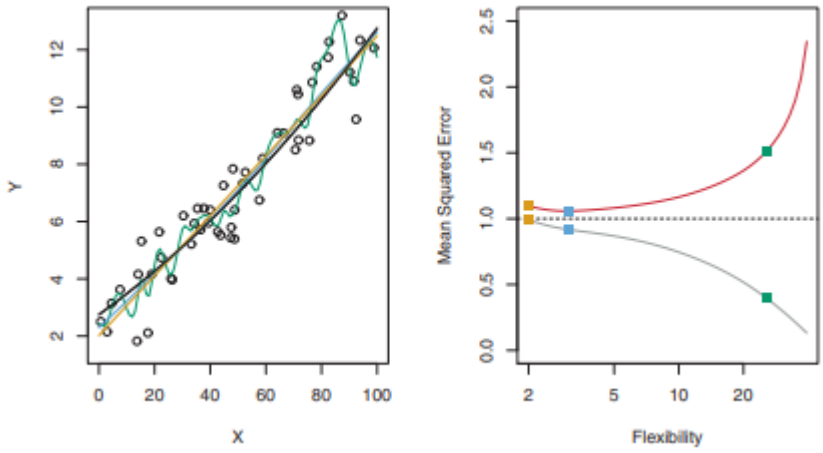
Para o cálculo dos erros, caso seja utilizado o mesmo conjunto de dados para o cálculo de coeficientes dos modelos e cálculo dos erros, isso acarretaria em um viés maior na seleção de modelos mais flexíveis, que não necessariamente são modelos de previsão melhores. Esse caso pode ser observado nas figuras 6, 7 e 8, onde a curva preta representa a distribuição real dos dados, a curva amarela, uma *f* de menor flexibilidade, a curva azul, uma *f* de flexibilidade intermediária e a verde, uma *f* de maior flexibilidade.

Figura 6: Exemplo 1 de *overfitting*



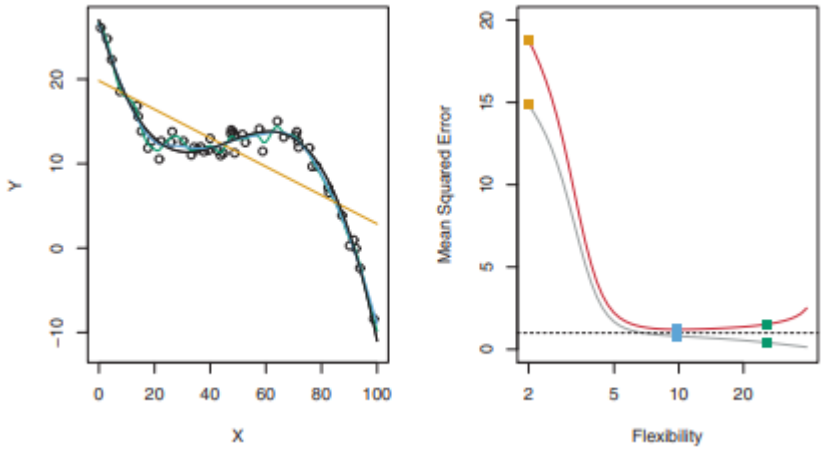
Fonte: An Introduction to Statistical Learning

Figura 7: Exemplo 2 de *overfitting*



Fonte: An Introduction to Statistical Learning

Figura 8: Exemplo 3 de *overfitting*



Fonte: An Introduction to Statistical Learning

Para contornar esse problema a técnica sugerida pelo livro "An Introduction to Statistical Learning" (GARETH, J. et al., 2013) é de separar a amostra em treino e teste, a amostra de treino é usada para o cálculo dos coeficientes dos modelos e a de teste, para o cálculo dos erros de teste. Desta forma, além de selecionar o melhor modelo, os resultados dos modelos são validados.

4.3.4. Modelos de regressão versus modelos de classificação

O modelo de regressão mais clássico é a regressão linear, nele, através de uma ou mais variáveis independentes (ou preditoras), é calculada uma *f* (pelo método dos mínimos quadrados) que estima um valor de uma variável dependente (ou variável resposta) contínua através de valores das variáveis independentes. Modelos de **regressão** trabalham, portanto, com variáveis dependentes **contínuas** (ou quantitativas).

Modelos de **classificação**, como a regressão logística ou LDA, trabalham com variáveis dependentes **categóricas** (ou qualitativas).

4.4. REGRESSÃO LINEAR

Como descrito anteriormente, a regressão linear é um modelo que, através do cálculo de uma f, estima o valor de uma variável dependente contínua através de uma ou mais variáveis independentes, podendo estas serem categóricas ou contínuas. O caso da utilização de mais de uma variável independentes também é conhecido como "Regressão Múltipla".

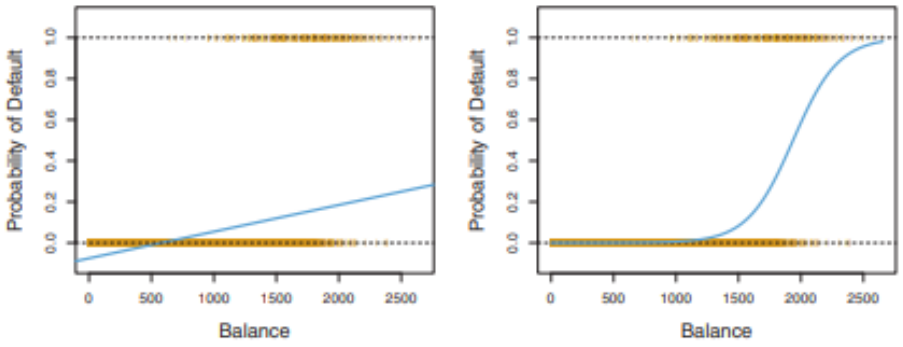
A regressão linear tem o seguinte formato: $f = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \ldots + \beta_n * X_n$, onde X_n representa a enésima variável independente e β_n representa o coeficiente calculado através do método de mínimos quadrados referente à enésima variável independente.

> "*Though it may seem somewhat dull compared to some of the more modern statistical learning approaches described in later chapters of this book, linear regression is still a useful and widely used statistical learning method.*" (GARETH, J. et al., 2013)

4.4.1. Utilização da regressão linear em casos de classificação

Mesmo a regressão linear sendo um modelo de regressão, ainda é possível utilizá-la para classificação. Basta modelar a variável resposta como sendo igual a 1 caso pertença a certo nível e 0 caso não pertença. Dessa forma teremos uma reta de regressão exemplificada na figura 9. A classificação é feita ao se determinar um valor de corte, como por exemplo 0,5, desta forma valores de previsão acima de 0,5 são classificados como o nível cujo valor 1 foi atribuído.

Figura 9: Comparação entre uso de Regressão Linear (direita) e Regressão Logística (esquerda)



Fonte: An Introduction to Statistical Learning

A regressão logística (será vista na seção 3.5) tende a gerar erros menores nesse caso por ter um melhor ajuste, como visto na exemplo da figura 9. Porém, como a regressão linear é um algoritmo de fácil execução (tanto ao escrever o código quanto na parte de processamento de dados), também a utilizaremos como método classificação e poderemos comparar na prática se esse melhor ajuste realmente ocorre.

4.5. REGRESSÃO LOGÍSTICA

O a regressão logística é um modelo que usado para prever a probabilidade de pertencimento a uma determinada classe de uma variável dependente categórica, tendo, portanto, um *output* entre 0 e 1.

A análise é aplicada normalmente em variáveis de duas categorias, porém pode-se também utilizar essa técnica em variáveis de mais de duas categorias, transformando a variável em binária, definindo uma categoria como 1 e o não pertencimento a ela como 0.

A regressão logística tem o seguinte formato:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}.$$

Após manipular a fórmula matematicamente chegamos em:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X.$$

Com isso, percebemos que a regressão logística é uma derivação da regressão linear e que seus coeficientes β são calculados utilizando a mesma técnica dos mínimos quadrados.

4.6. MULTICOLINEARIDADE

A multicolinearidade é uma propriedade definida pela correlação entre variáveis independentes. Um dos pré-requisitos da regressão logística e da regressão linear é a não multicolinearidade, uma vez que isso piora o poder de previsão do modelo.

> "O impacto da multicolinearidade é reduzir o poder preditivo de qualquer variável independente na medida em que ela é associada com as outras variáveis independentes. Quando a colinearidade aumenta, a variância única explicada por conta de cada variável independente diminui e o percentual da previsão compartilhada aumenta. Como essa previsão compartilhada pode ser considerada apenas uma vez, a previsão geral aumenta muito mais vagarosamente quando variáveis independentes com multicolinearidade elevada são acrescentadas." (HAIR, 2009)

Para contornar esse problema, um modelo com uma análise fatorial prévia será testado, como sugerido por Hair como alternativa de solução da multicolinearidade.

Uma vez que a multicolinearidade afeta o poder preditivo, seus efeitos adversos serão mensurados no erro de teste. Segundo Hair, uma outra solução para a multicolinearidade é usar o modelo somente para previsão, que será a utilização dos modelos multivariados de regressão neste trabalho.

4.7. ANÁLISE DISCRIMINANTE

A Análise Discriminante (DA, do inglês "*Discriminant Analysis*") é um modelo classificador de aprendizagem supervisionada. Trabalharemos neste projeto tanto com a análise discriminante linear (LDA), que possui superfície de decisão linear, quanto com a análise discriminante quadrática (QDA), que possui superfície de decisão não linear.

LDA é uma generalização do discriminante linear de Fisher, este método é usado em estatística e aprendizagem estatística para encontrar uma combinação linear de variáveis que caracteriza ou separa duas ou mais classes de objetos ou eventos.

QDA é uma variante de LDA na qual uma matriz de covariância individual é estimada para cada classe de observação. Por possuir superfície de decisão não linear, é um modelo de maior flexibilidade do que a LDA.

4.8. KNN

KNN (*K-nearest neighbor*) é um método de classificação não paramétrico tão fundamental quanto simples: para um determinado ponto, seleciona-se os k vizinhos (pontos experimentais) mais próximos e realiza-se uma contagem das classes destes, a classe com o maior número de ocorrência é usada na classificação do ponto.

Quanto maior o K, maior a flexibilidade do modelo. Por isso, no projeto de aprendizagem estatística, serão usados diferentes K's e o erro de teste será calculado para cada caso, sendo possível a comparação entre diferentes K's após a execução do modelo.

4.9. CONVERSION LIFT

Conversion lift é uma técnica desenvolvida pelo Facebook Ads para entender qual é o valor incremental gerado pelas suas campanhas de Facebook e Instagram (FACEBOOK).

A tecnica consiste em separar os usuários da plataforma em dois grupos, um grupo de controle, que não recebe nenhum tipo de conteúdo patrocinado, e um grupo de tratamento, que recebe conteúdo patrocinado.

O *conversion lift* é calculado através de uma comparação entre a receita por usuário dos dois grupos, podendo ter valores absolutos ou realtivos.

5. DADOS PRELIMINARES

Esta capítulo descreve os dados disponíveis anteriormente ao desenvolvimento deste trabalho. Analisar os dados disponíveis é essencial para o planejamento de variáveis que estarão presentes no modelo.

A Empresa Elementar Digital desenvolveu ao longo dos últimos anos uma grande estrutura de *Business Intelligence*, dessa estrutura demos o passo inicial para montar a tabela que representará nossa amostra.

Essa estrutura é capaz de extrair diariamente relatórios de diversas plataformas e assim controlar desde custos em anúncios online até fluxo no site e vendas.

Nossa fonte de dados principal utilizada será a tabela de vendas offline, nela temos todas as compras offline de clientes em todas as lojas de marcas da Empresa de Varejo.

Essa tabela é constituída pelos seguintes campos (somente os campos utilizados na construção da *query*):

- "*date*": *time stamp* com a data e horário da compra do cliente;
- "*distributorid*": id da loja física onde a compra ocorreu;
- "*amount*": total (em centavos) gasto pelo cliente;
- "*discount*": disconto (em centavos) de valor descontado do preço original;
- "site": marca em questão (Marca A, Marca B, etc);
- "*userid*": id do cliente;
- "*name*": nome da loja onde a compra ocorreu;

- "*State*": unidade federal da loja onde a compra ocorreu
- "*city*": cidade da loja onde a compra ocorreu
- "dt": data em que o dado foi inserido na tabela, ou seja, *load date*.

Cada linha desta tabela representa uma compra de um cliente, com isso pode se perceber que está longe de parecer uma amostra de um modelo de classificação.

6. VARIÁVEIS DEPENDENTES E INDEPENDENTES

Este capítulo descreve todas variáveis presentes no modelo. Embasado na sessão anterior, são levantadas e descritas a variável resposta e as variáveis utilizadas para previsão. Esta etapa é essencial para o planejamento do desenvolvimento da amostra, uma vez levantadas as variáveis onde se quer chegar, fica muito mais fácil o desenvolvimento de uma estrutura responsável por obtê-las.

A variável dependente de nosso estudo será uma variável categórica que representa se um determinado cliente comprou ou não comprou em um determinado mês. Desta forma o *output* de um algoritmo de classificação me indicaria potenciais compradores do mês. Foi utilizada a dimensão mês pois a frequência de um contato ao mês nos pareceu muito melhor. Caso o contato fosse bimestral, nosso impacto seria muito menor, caso fosse diário, acabaríamos por incomodar demais nosso cliente, também, como são enviadas 30 mensagens ao dia, acabaríamos conseguindo impactar somente os mesmos clientes várias vezes e não exploraríamos nossa vasta base de clientes.

Para facilitar o tratamento dos dados, a variável dependente será igual a 1 quando ocorreu compra e igual a 0 quando não ocorreu nenhuma compra no mês.

Nossas variáveis auxiliares da amostra, que não serão utilizadas em nenhum algoritmo, serão "*userid*" e "*month*" (*string* que representa um determinado mês). No código de SQL, por facilidade, a dimensão "*month*" será representada por uma variável de data do primeiro dia do mês.

Como variáveis independentes, foram levantadas uma série de variáveis. Nosso modelo será de dimensão 29: terá 29 variáveis preditoras.

O primeiro conjunto de variáveis são categóricas e representam se o cliente comprou ou não em meses ou trimestres (*quarters*), seguirá a mesma lógica da variável dependente, onde 1 significa que ocorreu uma compra e 0 significa que não ocorreu nenhuma compra. A nomenclatura utilizada para definir esses estados passados será a palavra do dicionário inglês "*lag*", que em

português significa “defasagem”. Dessa forma, o seguinte conjunto de variáveis categóricas foram escolhidas:

- D_LagQ1HadBought;
- D_LagQ2HadBought;
- D_LagQ3HadBought;
- D_LagQ4HadBought;
- D_LagM1HadBought;
- D_LagM2HadBought;
- D_LagM3HadBought;
- D_LagM4HadBought.

Utilizando “D_” para denotar que a variável é uma dimensão (ou seja, categórica), D_LagQ1HadBought representa se o cliente realizou alguma compra no trimestre anterior, D_LagQ2HadBought representa se o cliente realizou alguma compra no trimestre retrasado, D_LagM1HadBough representa se o cliente realizou alguma compra no mês anterior, e assim por diante.

Todas as seguintes variáveis são contínuas, foi utilizada “M_” para indicar que se trata de uma métrica:

- M_MonthsSinceLastPurchase: Quantidade de meses desde a última compra realizada pelo cliente;
- M_MonthsAsACustomer: Meses desde a primeira compra do cliente (observação: foram utilizados dados a partir de 01/01/2017, portanto pode ser que o cliente seja mais antigo, mas para o nosso modelo isso não fará muita diferença);
- M_HistoryCompletedPurchases: Número total de compras realizadas pelo cliente (novamente, desde 01/01/2017);
- M_HistoryRevenue: Receita histórica do cliente, ou seja, o total investido por ele em compras da empresa;
- M_HistoryFrequence: Frequência histórica de compras, ou seja: $\frac{M\_HistoryCompletedPurchases}{M\_MonthsAsACustomer}$.

O último grupo de variáveis são também contínuas e também utilizam a lógica do *lag*. São variáveis que representam o número de compras completas no dado período e a receita gerada nesse período. Desta forma temos:

- M_LagQ1CompletedPurchases;
- M_LagQ1Revenue;
- M_LagQ2CompletedPurchases;
- M_LagQ2Revenue;
- M_LagQ3CompletedPurchases;
- M_LagQ3Revenue;
- M_LagQ4CompletedPurchases;
- M_LagQ4Revenue;
- M_LagM1CompletedPurchases;
- M_LagM1Revenue;
- M_LagM2CompletedPurchases;
- M_LagM2Revenue;
- M_LagM3CompletedPurchases;
- M_LagM3Revenue;
- M_LagM4CompletedPurchases;
- M_LagM4Revenue.

7. CONTRUÇÃO DA TABELA DE AMOSTRA

Este capítulo tem o objetivo de documentar como foi realizado o desenvolvimento da estrutura de *queries* (em SQL) responsável por desenvolver amostra que será usada no processo de aprendizagem estatística.

Logo conseguimos perceber a dificuldade desse trabalho quando notamos que a tabela inicial possui um formato onde cada linha é uma compra e a tabela final deve possuir um formato onde cada linha demonstra o estado que um cliente estava em determinado mês.

Para ilustrar como estamos longe do *output* desejado, se considerarmos apenas a tabela inicial, a variável resposta teria o valor de 1 em cada linha, pois se trata de uma compra. Não poderíamos, portanto, utilizar um algoritmo de classificação, pois se trata de uma variável categórica de somente um nível.

Será descrito na sequência os passos realizados pelo autor para obtenção da tabela de amostra. Em parênteses em cada item está a nomenclatura utilizada no código para definir a tabela auxiliar.

I. Tratamento inicial dos dados e seleção das variáveis importantes (“store_orders”): A data original vem no formato de *string*, portanto foi utilizada a função “date_parse” para transformar o texto em uma variável de data. Também foi realizado a divisão por 100 nos valores de receita e disconto, uma vez que os dados vinham em centavos.

II. Criação de uma tabela auxiliar contendo todos os meses da base (“year_months”): desta forma, ao cruzar essa tabela com a de compras, acharemos os dias onde o cliente não realizou compra e assim conseguimos montar de forma correta nossa variável resposta.

III. Criação de uma tabela auxiliar contendo clientes da loja selecionada (“selected_store”): seleciona-se apenas clientes que realizaram compras na loja selecionada para o teste, dessa forma conseguimos filtrar apenas clientes de nosso interesse desde o início e diminuir a quantidade de dados processados.

IV. Criação de uma tabela auxiliar que contenha os empregados da empresa (“employees”): como acreditamos que essas pessoas não devem ser contatadas, elas foram retiradas de nossa amostra desde o início a fim de reduzir distorções. Essa tabela é composta pela união de duas tabelas: uma gerada em código que seleciona e-mails com o domínio da empresa, outra subida manualmente contendo outros IDs de empregados.

V. Criação de uma tabela auxiliar contendo todos os meses e todos os clientes da empresa (“clients_ym”): essa tabela é um cruzamento da tabela de meses com a de clientes, possui tamanho igual ao produto entre número total de clientes e o número de meses.

VI. Criação de uma tabela auxiliar de *cohort* (“cohort”): definindo *cohort* como um agrupamento de clientes através da data de sua primeira compra, esse dado será usado para o cálculo da variável independente “M_MonthsAsACustomer”.

VII. Criação de uma tabela auxiliar com as dimensões clientes e mês contendo compras realizadas como métricas (“transactions_by_ym” e “transactions_by_ym2”): nessa tabela, agrupa-se dados pelo *userid* e mês e conseguimos obter o número de compras realizadas no mês e a receita gerada por cliente. Nessa tabela já cruzaremos com a tabela auxiliar “clients_ym”, desta forma, até meses sem compras aparecerão na tabela, inclusive meses anteriores a primeira compra do cliente, porém esse dado será futuramente tratado. Aqui também é calculado o último mês com compra, que será utilizado no cálculo do campo “M_MonthsSinceLastPurchase”.

VIII. Criação de uma tabela auxiliar que organiza os dados da tabela anterior, calculando as variáveis que estão presentes no *output* (“client_data_by_ym”): essa tabela utiliza os dados da “transactions_by_ym2” para calcular as variáveis: D_HadBought, M_CompletedPurchases, M_Revenue, M_MonthsSinceLastPurchase, M_MonthsAsACustomer, M_HistoryCompletedPurchases, M_HistoryRevenue. É também aqui onde se aplica o filtro que exclui meses antes da primeira compra do cliente.

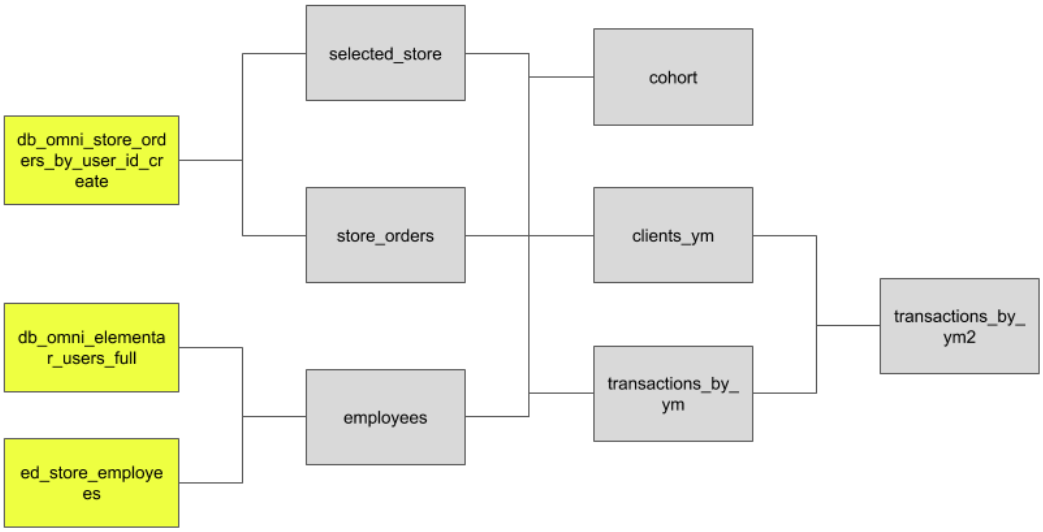
IX. Agrupamento trimestral dos dados de compra (“client_data_by_quarters”): aqui são agrupados os dados por trimestre e são calculados os *lags* trimestrais de um a quatro.

X. *Query* principal compilando todos os dados anteriores: Neste “*Select*” final todos os dados anteriormente calculados são agrupados. Os únicos cálculos que precisam ser realizados é o da frequência de compra e os *lags* mensais. É realizado um filtro final usando a função “*Having*” que exclui o mês da primeira compra da análise, uma vez que quando se dá a primeira compra, não existe nenhum dado preliminar, isso poderia criar um vies na solução de classificar clientes novos como potenciais compradores, uma vez que sempre que a variável “M_MonthsAsACustomer” fosse igual a 0, a variável resposta seria sempre igual a 1.

Feito isso, temos a *query* nomeada dwh_int_buy_prediction. A *query* que exporta os dados para a amostra realiza um filtro final que retira o mês vigente da análise, os dados do mês vigente serão utilizados na etapa prática para previsão. O código utilizado nesta etapa pode ser encontrado no Apêndice C deste trabalho.

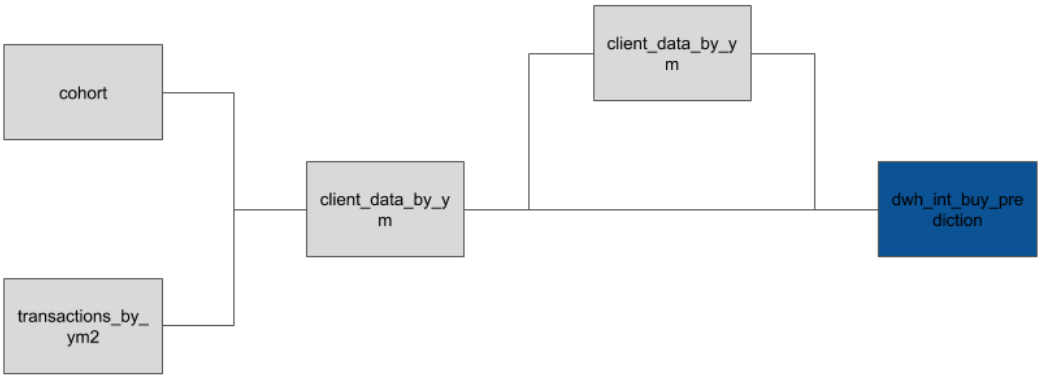
Desta forma, os seguintes fluxogramas foram criados para ilustrar a estrutura criada para a execução das tabelas descritas. Em amarelo estão as tabelas de dados crus, em cinza as tabelas auxiliares (todas dentro do mesmo código), em azul está a tabela final.

Figura 10: Fluxograma da estrutura de BI 1



Fonte: Autor

Figura 11: Fluxograma da estrutura de BI 2



Fonte: Autor

8. EXECUÇÃO DO MODELO DE CLASSIFICAÇÃO

Este capítulo descreve o desenvolvimento e a execução dos modelos de aprendizagem estatística introduzidos na revisão bibliográfica, descrevendo assim cada modelo utilizado e o cálculo dos erros de teste.

Com a amostra obtida, o próximo será estatístico. Para isso, foi utilizado o *software* RStudio para a execução e seleção do modelo de previsão e para a previsão de compra que será abordada mais para frente.

Foi então desenvolvido pelo autor um *script* em R que executa as seguintes funções, o *script* pode ser observado no Apêndice B deste trabalho.

8.1. TRATAMENTO INICIAL DOS DADOS

Inicialmente, o *output* da *query* descrita na última seção foi salvo em formato csv, então, a tabela foi carregada no R.

Mesmo que o código em SQL trate qualquer possibilidade de células nulas, para prevenir ainda mais a possibilidade de casos nulos, foi utilizado um filtro que retira linhas que contêm pelo menos uma célula nula. Tal cuidado foi tomado pois alguns modelos resultam em erro quando deparado com o problema descrito.

Nessa seção também foi criado o *data frame* da tabela de *output* dos erros de cada modelo. Esta tabela será exposta na seção 7 e será vital para a seleção do modelo ganhador. Na tabela também está presente uma coluna que conta o número de clientes selecionados, uma vez que a

probabilidade de compra é reduzida, modelos que selecionam poucos clientes possuem erros menores.

8.2. SEPARAÇÃO ENTRE TREINO E TESTE

Como descrito na revisão bibliográfica, separar a amostra entre teste e treino é vital para evitar erros de *overfitting*. Neste trabalho, foram realizados dois métodos de separação: seleção de um mês como teste (Janeiro de 2020) e uma divisão aleatória, utilizando 70% da amostra como treino e 30% como teste.

Como as duas técnicas produziram os mesmo resultado (mesmos modelos ganhadores), adotarei a técnica da utilização do mês de janeiro pois ela será mais ilustrativa na validação do modelo com dados do experimento prático.

8.3. EXECUÇÃO DOS MODELOS

O principal modelo utilizado neste trabalho foi o de regressão logística. Primeiramente foi testado o modelo simples de regressão logística e armazenado seu erro. Em seguida, foram retiradas as variáveis não significativas do modelo, teoricamente, isto reduz o erro por variância, uma vez que uma variável não significativa apenas aumenta a variabilidade dos dados sem oferecer uma previsão efetiva.

Uma vez que o modelo completo não respeita o pré-requisito da multicolinearidade, a solução para esse problema foi realizar uma análise fatorial, que agrupa variáveis correlacionadas em fatores e resolve o problema de multicolinearidade.

O último tratamento aplicado a clássica regressão logística foi a regressão polinomial. A variável selecionada foi M_HistoryFrequence, uma vez que seu coeficiente, dentro das variáveis contínuas foi o maior (1.109). Foram testados os modelos de grau 2 a grau 5.

Por fim, um modelo que combina dois anteriores também foi construído: Análise fatorial seguida de regressão polinomial. Uma vez que o número de fatores é reduzido, todos eles foram elevados a potência referente ao grau do modelo.

Para todos modelos de Regressão Logística, a classificação foi feita selecionando os 200 clientes cuja probabilidade de compra é maior. A classificação foi feita dessa forma pois ela ilustra muito melhor como o projeto funcionará na prática, seleciona-se, no dia, os clientes com maior probabilidade de compra e faz-se contato. O número 200 foi escolhido pois representa um pouco menos de 10 contatos por dia, o que parece um número bom para um teste prático.

Os modelos de análise discriminante foram feitos em seguida, sendo eles o linear (LDA) e quadrático (QDA).

O algoritmo k-vizinhos mais próximos (KNN) foi realizado. O programa contempla um *loop* que nos permite rodar o algoritmo 10 vezes, com K's progredindo de 1 a 10.

Por fim, foi realizado um modelo de regressão linear múltipla, a possibilidade de utilização desse modelo de regressão em uma situação de classificação é discutido na revisão bibliográfica e o porquê de ele ser incluídos será melhor explicado na seção de seleção do modelo.

9. RESULTADOS DO MODELO

Com a execução dos modelos realizada na sessão anterior, este capítulo expõe os erros de teste obtidos para cada modelo. Estes resultados serão essenciais para a seleção do modelo ganhador.

Com isso, rodamos o *script* e chegamos na seguinte tabela, nela pode se ver qual modelo possui menores erros de teste.

Tabela 1: Erros de Modelos Testados

Método	Taxa de Acerto	# Clientes Selecionados
KNN k = 7	0.9924	0
KNN k = 8	0.9924	0
KNN k = 9	0.9924	0
KNN k = 10	0.9924	0
KNN k = 5	0.9922	1
KNN k = 6	0.9922	1
KNN k = 4	0.9913	6
KNN k = 3	0.9907	9
KNN k = 1	0.9868	34
KNN k = 2	0.9856	36
LDA	0.9828	55
Logistic Regression Clean	0.9713	200
Logistic Regression	0.9572	200
Linear Regression	0.9572	200

Método	Taxa de Acerto	# Clientes Selecionados
Logistic Regression Plm(2)	0.9572	200
Logistic Regression Plm(3)	0.9572	200
Logistic Regression Plm(4)	0.9572	200
Logistic Regression Plm(5)	0.9572	200
Logistic Regression (Factors)	0.9557	200
Factor Logistic Regression Pln 2	0.9557	200
QDA	0.9474	250

Fonte: Autor

A tabela foi ordenada pela coluna "Taxa de Acerto" com o intuito de evidenciar quais modelos possuem menores erros.

10. SELEÇÃO DO MODELO

Com os resultados obtidos, este capítulo descreve o processo de seleção do modelo ganhador que será usado na classificação de clientes. Dada a seleção do modelo, esse capítulo também traz os coeficientes do modelo ganhador.

Dessa forma, percebemos que o método KNN possui menores erros, porém, ele acaba selecionando muito poucos clientes, portanto inútil. Esse resultado ocorre, pois a incidência de compra é muito baixa, dessa forma, selecionar nenhum cliente gera um erro baixo.

Portanto o primeiro método viável para utilização é o de LDA, uma vez que seleciona um número substancial de clientes.

Será utilizado, portanto, o **LDA** como forma primária de seleção. Entretanto, o número de clientes que serão contatados no mês pode ser maior que o número de clientes selecionados pelo algoritmo, portanto será utilizado um segundo critério de seleção. Um método muito interessante, nesse caso, seria a regressão logística ou até a linear, uma vez que esses modelos possuem um *output* contínuo, sendo possível ordenar os clientes.

Temos então duas possibilidades: a regressão logística *clean* e a regressão logística com análise fatorial. A primeira é uma opção porque possui menor erro de teste, e a segunda porque respeita o pré-requisito de multicolinearidade. Consequentimente será utilizado o modelo de ***Logistic Regression Clean***, pois ele possui um erro de teste muito menor e, como citado na revisão bibliográfica, o principal problema da multicolinearidade é reduzir o poder de previsão do modelo, uma vez que mesmo com essa redução de poder de previsão, o erro de teste ainda é melhor.

Notamos também, pelo análise de resíduos, que o modelo de regressão logística é heterocedástico, isto é outro fator que reduzirá o poder de previsão, mas isto também está contido no erro de teste.

As tabelas 2 e 3 expõem o valor dos coeficientes de cada análise. As variável que não aparecem na regressão logística limpa são não significativos.

Tabela 2: Coeficientes do Modelo LDA

Variável	Valor do Coeficiente
D_LagQ1HadBought	0.2667
D_LagQ2HadBought	-0.0788
D_LagQ3HadBought	0.2016
D_LagQ4HadBought	0.6145
D_LagM1HadBought	1.3311
D_LagM2HadBought	-0.2242
D_LagM3HadBought	1.4128
D_LagM4HadBought	-1.0648
M_MonthsSinceLastPurchase	0.0118
M_MonthsAsACustomer	0.0106
M_HistoryCompletedPurchases	0.1765
M_HistoryRevenue	0.0000
M_HistoryFrequence	2.0713
M_LagQ1CompletedPurchases	-0.1674
M_LagQ1Revenue	-0.0001
M_LagQ2CompletedPurchases	0.4139
M_LagQ2Revenue	-0.0002
M_LagQ3CompletedPurchases	0.2132
M_LagQ3Revenue	-0.0001
M_LagQ4CompletedPurchases	-0.1311
M_LagQ4Revenue	-0.0001
M_LagM1CompletedPurchases	1.4984
M_LagM1Revenue	-0.0002

Variável	Valor do Coeficiente
M_LagM2CompletedPurchases	2.3312
M_LagM2Revenue	-0.0008
M_LagM3CompletedPurchases	-0.2137
M_LagM3Revenue	0.0004
M_LagM4CompletedPurchases	3.4822
M_LagM4Revenue	-0.0018

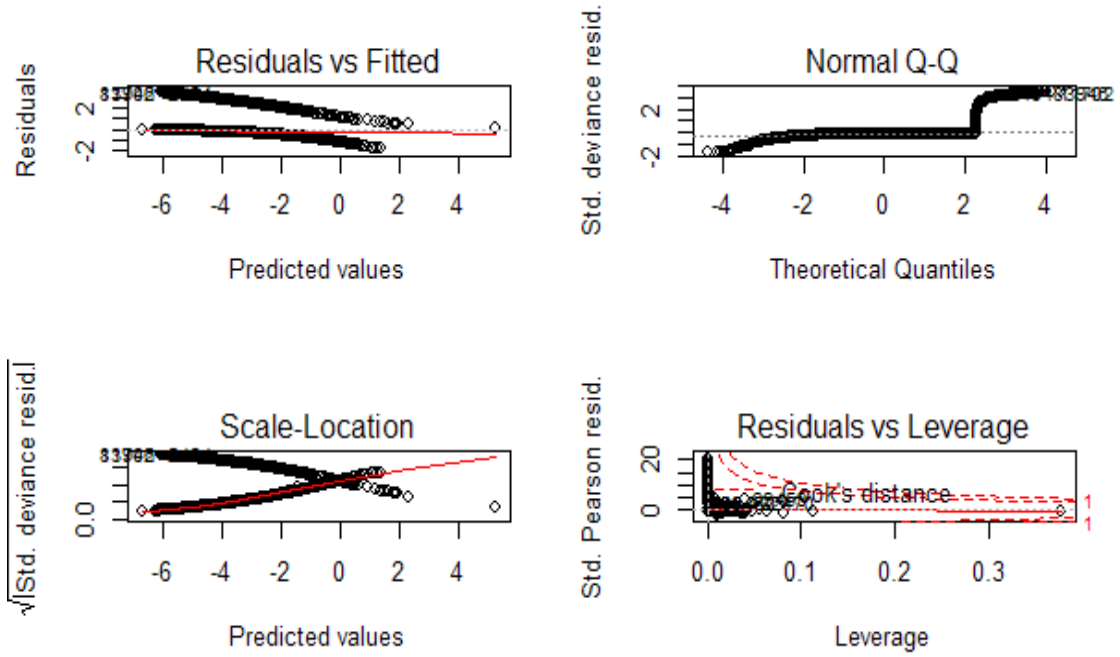
Fonte: Autor

Tabela 3: Coeficientes do Modelo *Logistic Regression Clean*

Variável	Valor do Coeficiente
(Intercept)	-4.7781
D_LagQ4HadBought	0.3460
D_LagM1HadBought	0.9449
D_LagM3HadBought	0.5662
M_MonthsSinceLastPurchase	-0.0985
M_MonthsAsACustomer	0.0563
M_HistoryRevenue	0.0001
M_HistoryFrequence	1.1535
M_LagQ1Revenue	-0.0001
M_LagQ2CompletedPurchases	0.1972
M_LagQ2Revenue	-0.0001
M_LagM4CompletedPurchases	0.7105
M_LagM4Revenue	-0.0005

Fonte: Autor

Figura 12: Análise de Resíduos Regressão Logística *Clean*



Fonte: Autor

11. IMPLEMENTAÇÃO PRÁTICA DA SAÍDA DO MODELO

Este capítulo descreve a utilização do modelo estatístico, responsável pela seleção de clientes de forma prática, desde o tratamento inicial dos dados para possibilitar o contato até a execução prática do envio de mensagens de WhatsApp.

11.1. EXPORTAÇÃO DOS DADOS

Com isso, um novo *data frame* é carregado no R com os dados dos clientes referentes ao mês atual e os coeficientes desses modelos são utilização para a previsão de compra do cliente no mês.

Dessa forma, ao se executar a parte final do *script*, adiciona-se ao *data frame* as previsões do modelo LDA e do modelo de Regressão Logística.

Por fim, o *output* é salvo em formato csv e de forma manual, se faz o *upload* dessa tabela no banco de dados da Elementar.

11.2. TRATAMENTO DOS DADOS

Com a tabela criada no banco de dados, uma *query* foi desenvolvida com o intuito de puxar as informações de cada cliente com o objetivo de facilitar o contato. As variáveis obtidas no banco de dados do cliente foram as seguintes:

- D_Email: para possibilitar um contato alternativo;
- D_Name: para possibilitar personalização da mensagem;
- D_Phone: essa é a principal variável, pois nosso contato será feito principalmente por *WhatsApp*;
- M_DiscountRate: Taxa de desconto em compras antigas, possibilita entender melhor o perfil do cliente;
- D_TopCateg: Categoria mais comprada pelo cliente;
- D_TopWeekday: dia da semana em que o cliente gastou mais dinheiro em uma loja;
- M_ShareLoja: Porcentagem de dinheiro que o cliente gastou na loja testada no piloto;
- M_AvgTicket: Ticket médio das compras do cliente, na Empresa de Varejo temos perfis de clientes muito distintos, temos com alto ticket e outros com tickets menores.

Também foram utilizadas variáveis presentes no output do *script* do R, que foram interessantes para um estudo de perfil do cliente. As variáveis são: M_MonthsSinceLastPurchase; M_MonthsAsaCustomer; M_HistoryRevenue; M_HistoryFrequence.

11.3. EXECUÇÃO

Os dados foram exportados para um planilha de Excel onde foi feito o controle de atribuição e controle de contatos, nesta planilha também foi gerada uma mensagem padrão, sugerida pelo departamento de comunicação da Empresa de Varejo.

O contato foi feito de forma manual pelo autor com o intuito de estudar o ciclo do início ao fim.

Figura 13: *Print* de uma conversa com o cliente onde o autor se passa pelo gerente da loja

Fonte: Autor

Contato por contato foram mensuradas as seguintes atividades:

- Se o número de celular possui *WhatsApp*, isso exclui clientes com número de telefone fixo ou que não possuem conta no aplicativo;
- Se a mensagem chegou ao cliente;
- Se o cliente respondeu;
- Se o cliente demonstrou interesse.

Dessa forma foram geradas variáveis binárias para cada atividade.

12. RESULTADOS DO PILOTO

Este capítulo descreve os resultados do teste prático realizado pelo autor, trazendo tanto dados operacionais (dados de contato) quanto dados de venda. Com isso é possível analisar o impacto do projeto até esse ponto e validar algoritmos estatísticos utilizados.

Durante o decorrer do mês de Março de 2020 foram então contatados pelo autor um total de 590, que gerou um total de 136 compras e R$76,190 de receita, se considerarmos que foi apenas um operador realizamos os contatos, notamos que o teste foi um sucesso.

Essas compras e receita incluem tanto compras realizadas em lojas físicas quanto compras online, foram consideradas apenas compras cuja data é maior ou igual a data de contato. Toda essa estrutura de mensuração foi também desenvolvida pelo autor.

As tabelas 4 e 5 mostram todos os indicadores de desempenho do realização do teste. “Contatos/dia” é definido pelo número de mensagens recebidas dividido pelos 22 dias úteis do mês de Março; “Taxa de Celulares com WhatsApp” é obtida pela divisão de “Celular com WhatsApp” por “Mensagens Recebidas”, esse indicador demonstra a qualidade da base de clientes da Empresa de Varejo; “Entregabilidade” é a taxa de entrega de mensagens no WhatsApp; “Taxa de Respostas” é obtida pela divisão das respostas pelas mensagens entreges.

Tabela 4: Dados Operacionais

Indicador de Desempenho	Valor
Clientes em potencial	965
Celular com WhatsApp	608
Mensagens Recebidas	590
Respostas	71
Contatos/Dia	26.8
Taxa de Celulares com WhatsApp	63%
Entregabilidade	97%
Taxa de Respostas	12%

Fonte: Autor

Tabela 5: Resultados do Piloto

Indicador de Desempenho	Valor
Número de Pessoas Contatadas	590
Número de Compradores	66
Taxa de Acerto	11.2%
Número de Compras	136
Receita Total	R$76,190

Fonte: Autor

Podemos então estimar a taxa de acerto com 95% de confiança como sendo o intervalo de 8.8% a 14.0%.

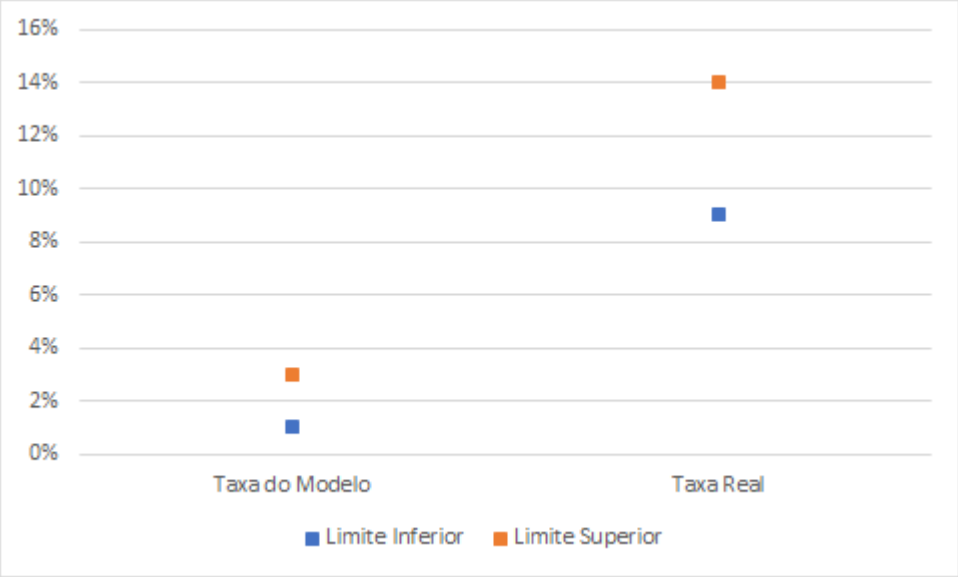
Tabela 6: Tabela de Contraste da Saída do modelo no R

Previsão do Algoritmo	Não Compra Real	Compra Real
Não Compra	4668	28
Compra	578	12

Fonte: Autor

Voltando a validação do R, quando selecionamos 590 clientes, a taxa de acerto foi de 2.1% (o número parece baixo, porém, se escolhermos aleatoriamente, a taxa de acerto será 1.1%). Podemos, portanto validar de forma prática o modelo utilizado no R e, além disso, inferir que o ato de contatar as pessoas aumenta sua taxa de conversão, uma vez que a taxa de acerto prática é estatisticamente maior que a taxa de acerto esperada caso nenhuma ação fosse tomada.

Figura 14: Intervalo de Confiança para taxas de sucesso (nível de confiança = 95%)



Fonte: Autor

Com isso conseguimos concluir que o piloto foi um sucesso, não só mostramos a eficiência do modelo como conseguimos demonstrar que mandar mensagem para os clientes é uma forma eficiente de fazê-los comprar. Essa segunda parte é essencial para aplicação prática de modelos teóricos, um algoritmo que apenas selecionasse quem já realizaria uma compra de qualquer forma, pouca utilidade teria. Podemos concluir que, além de selecionar clientes que já comprariam, melhorar nossa gestão de relacionamentos com o cliente, oferecer um canal de comunicação que torna esse processo de compra mais fácil e confiável, nosso modelo ainda seleciona pessoas cujas características se aproximam muito de alguém que realizaria uma compra, mas não iria comprar sem nenhuma ação.

13. IMPLEMENTAÇÃO DO SISTEMA

Este capítulo trata da expansão do projeto desenvolvido no piloto para a utilização diária do cliente. Toda adaptação do piloto é descrita e também a plataforma que possibilita diversos vendedores contatarem diversos clientes.

Devido a epidemia de COVID-19, durante o mês de Abril, a Empresa de Varejo foi obrigada a fechar a maioria de suas lojas físicas. Isso gerou uma drástica redução na receita da empresa. Como todos vendedores ficassem em casa ociosos, a empresa decidiu então implementar cupons para esses vendedores para que eles obtivessem vendas online e dessa forma ganhassem

comissão fazendo com que um novo canal de vendas fosse criado para o marketing da empresa, com o intuito de reduzir esta grande perda gerada.

Com a experiência positiva do piloto desenvolvida, o projeto de implementação de um sistema para possibilitar e mensurar a comunicação vendedor/cliente da Elementar foi comprado pela empresa.

13.1. A PLATAFORMA

A Elementar, durante o segundo semestre de 2019, desenvolveu uma plataforma chamada "Clerk", ela tem o objetivo de facilitar o contato de *leads* via WhatsApp e já vinha sendo usada em uma iniciativa da empresa no ramo imobiliário.

Nela é possível agendar contatos de clientes e atribuí-los a lojas e a usuários (vendedores). Clicando em apenas um botão, o usuário abre, no WhastApp dele, a conversa com o cliente com uma mensagem padrão já escrita, que pode ser personalizada com o nome do cliente, nome do vendedor, cupom do vendedor, etc.

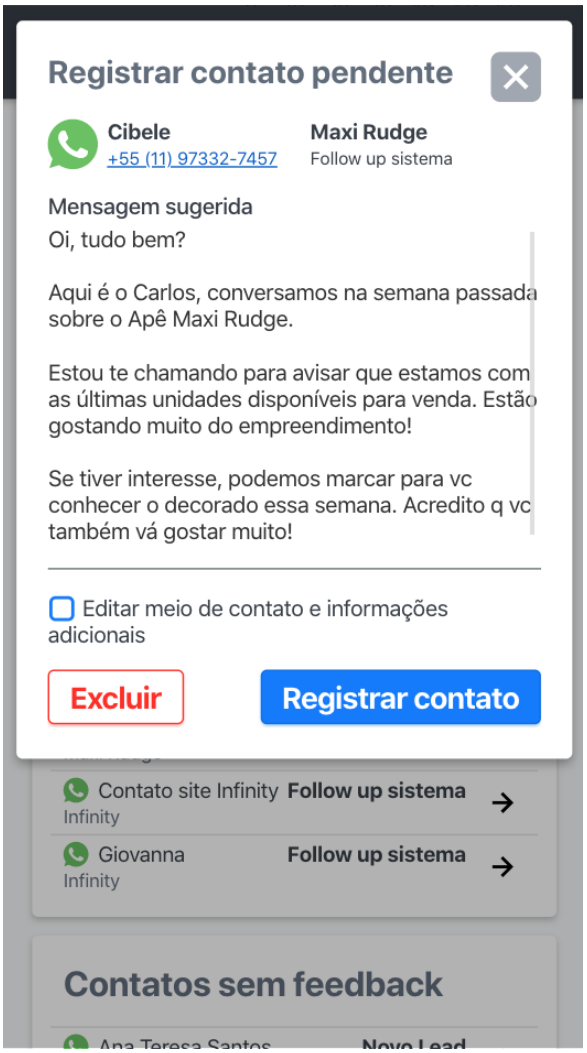
Após isso, ao retornar à plataforma, o sistema automaticamente abre uma mensagem para o usuário, nela o usuário deve confirmar se enviou a mensagem ou se teve algum problema. Dessa forma mensura-se o número de contatos.

A plataforma foi desenvolvida em um domínio da internet e é possível acessá-la através de um navegar. Mesmo ainda não tendo um aplicativo, a plataforma é 100% online (*mobile first*) com usabilidade intuitiva e com integração com WhatsApp

Figura 15: Página de Tarefas (Contatos Pendentes) do Clerk

Fonte: Clerk

Figura 16: Mensagem da plataforma aberta ao clicar em um contato



Fonte: Clerk

Figura 17: Redirecionamento para o WhatsApp com a mensagem sugerida preenchida

Fonte: Clerk

13.2. ADAPTAÇÃO DA ESTRUTURA CRIADA NO PILOTO

Do modo que a estrutura inicial foi construída, poucas alterações tiveram que ser feita nas *queries* que geram os dados da amostra e dados do mês atual. A única alteração feita foi retirar o filtro que selecionavam apenas clientes da loja selecionada da Marca A.

Feito isso, o mesmo *script* de R foi utilizado para primeiro selecionar um modelo ganhador, onde obtivemos como resultado os mesmos modelos ganhadores: LDA e Regressão Logística. Isso é um ótimo sinal e valida nossa análise em menor escala ao se reproduzir o resultado em uma amostra muito maior. Os novos coeficientes foram calculados e utilizados para previsão.

Tabela 7: Coeficientes LDA

Variável	Valor do Coeficiente
D_LagQ1HadBought	0.2137
D_LagQ2HadBought	0.3772
D_LagQ3HadBought	0.4045
D_LagQ4HadBought	0.7794
D_LagM1HadBought	0.7089
D_LagM2HadBought	0.6845
D_LagM3HadBought	0.3590

Variável	Valor do Coeficiente
D_LagM4HadBought	0.3672
M_MonthsSinceLastPurchase	-0.0234
M_MonthsAsACustomer	0.0392
M_HistoryCompletedPurchases	0.0389
M_HistoryRevenue	No0.0000
M_HistoryFrequence	1.4419
M_LagQ1CompletedPurchases	-0.0307
M_LagQ1Revenue	0.0001
M_LagQ2CompletedPurchases	0.0790
M_LagQ2Revenue	0.0001
M_LagQ3CompletedPurchases	0.0142
M_LagQ3Revenue	0.0001
M_LagQ4CompletedPurchases	-0.1905
M_LagQ4Revenue	0.0000
M_LagM1CompletedPurchases	0.9365
M_LagM1Revenue	0.0009
M_LagM2CompletedPurchases	0.3482
M_LagM2Revenue	-0.0005
M_LagM3CompletedPurchases	0.3755
M_LagM3Revenue	-0.0004
M_LagM4CompletedPurchases	0.5245
M_LagM4Revenue	-0.0006

Fonte: Autor

Tabela 8: Coeficientes *Logistic Regression Clean*

Variável	Valor do Coeficiente
(Intercept)	-3.9861
D_LagQ1HadBought	0.1063

D_LagQ2HadBought	0.2915
D_LagQ3HadBought	0.2965
D_LagQ4HadBought	0.4693
D_LagM1HadBought	0.6885
D_LagM2HadBought	0.3487
D_LagM3HadBought	0.2661
M_MonthsSinceLastPurchase	-0.0537
M_MonthsAsACustomer	0.0330
M_HistoryFrequence	0.6908
M_LagQ4CompletedPurchases	-0.0813
M_LagM4CompletedPurchases	0.1826

Fonte: Autor

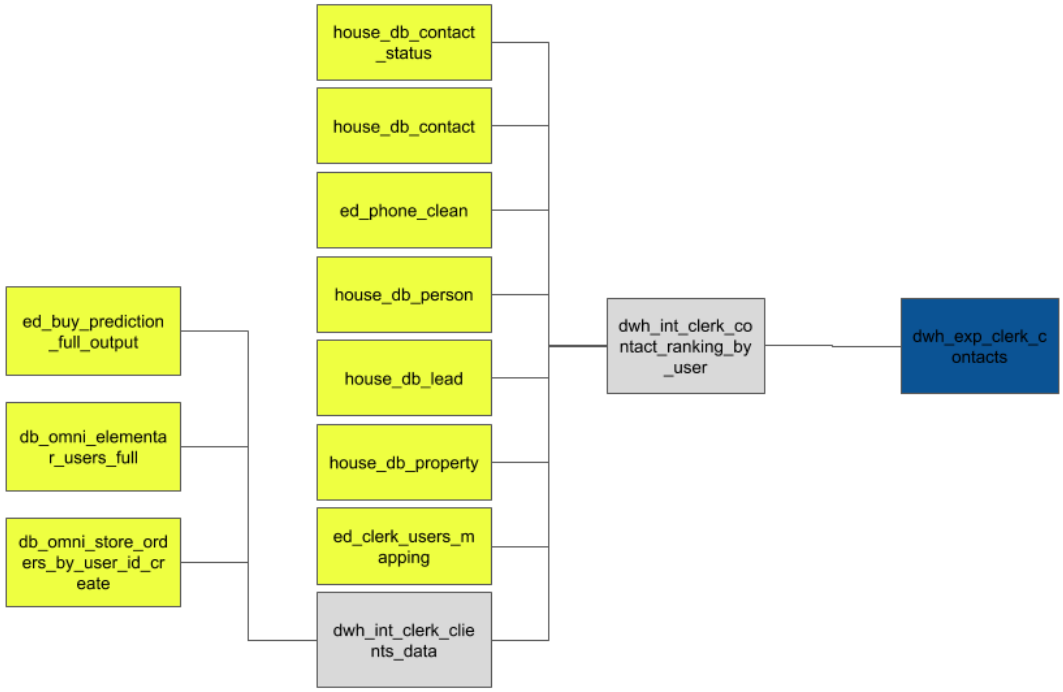
Com o uso desses novos coeficientes, as variáveis de previsão são adicionadas ao *data frame* e o *output* é salvo no banco de dados da Elementar.

13.3. TRATAMENTO DO *OUTPUT* E *UPLOAD* NO CLERK

Com isso, foi construída a seleção de *leads* e atribuição a lojas e usuários através, novamente, de SQL.

Todo código produzido pelo autor está presente no Apêndice D e sua lógica será em seguida descrita.

Figura 18: Fluxograma da Estrutura de SQL de seleção de contatos



Fonte: Autor

13.3.1. Atribuição de Cliente a Lojas

Uma vez que um cliente pode ter realizado compras em mais de uma loja, é necessário criar uma regra de atribuição. Se não, poderia ocorrer o problema de mais de uma loja contatar o mesmo cliente, gerando até um efeito negativo, irritando o cliente por excesso de contato. Também teríamos um problema depois ao atribuir vendas a lojas, uma vez que duas lojas diferentes contataram o cliente não fica claro a quem atribuir a venda.

A regra escolhida foi a de atribuir o cliente à loja onde ele gerou maior receita em todo período (desde Janeiro de 2017). As outras opções consideradas foram: loja no qual o cliente realizou o maior número de compras, mas atribuindo através da receita é a melhor estratégia, pois poderíamos selecionar um loja onde o cliente tende a ter um *ticket* médio menor, o que reduziria a receita total (principal indicador de desempenho para nosso caso); loja no qual o cliente realizou a última compra, mas isso poderia atribuir clientes fiéis a loja A, que por acaso realizaram uma compra na loja B, a uma loja onde ele não costuma fazer suas compras.

A *query* utiliza uma *window function* (*first value*) para selecionar a loja cuja receita do cliente foi maior.

13.3.2. Rankeando Clientes por usuário

A estratégia escolhida foi a de atribuir cerca de 20 clientes a cada vendedor por dia. Esse número foi escolhido junto com o entendimento com o cliente, a fim de evitar sobrecarregar os vendedores com excessivas tarefas em um único dia.

Para evitar acúmulo de contatos, o que poderia fazer com que um usuário contatasse mais de 20 clientes em um mesmo dia, somente usuários sem pendências receberão contatos. Por isso, uma tabela com usuários com pendências é criada, para que esses usuários sejam retirados mais a frente.

Com isso, cada usuário recebe um número de 1 até a soma de usuários disponíveis na loja, essa variável foi nomeada como “user_index”.

Para evitar que um mesmo contato seja selecionado todos os dias, foi criada uma tabela de quarentena de telefones, que armazena todos os telefones contatados nos últimos 30 dias, para que esses telefones não sejam selecionados. O número de 30 dias foi usado uma vez que nosso algoritmo funciona em uma base mensal, em 30 dias a resposta do algoritmo estatístico estará atualizada, também julgamos que uma frequência de um contato a cada 30 dias evita que o cliente se irrite com as ações de marketing realizadas.

Como ainda existe uma preocupação com relação a segurança de dados, muitos clientes ainda colocam telefones nulos no cadastro, como por exemplo ‘999999999’. Como o nosso contato é feito exclusivamente por WhatsApp, seria ineficiente incluir na plataforma esse tipo de número de telefone. Por isso, foi criado uma tabela (“ed_phone_clean”) com todos números encontrados na base que não são telefones válidos. Esses números foram então filtrados no *output* de contatos.

Caso o *lead* seja novo ele pode ser atribuído para qualquer vendedor. que não fará tanta diferença, entretanto, se o lead já estiver na plataforma é vital que ele seja atribuído ao mesmo vendedor. Atribuir o *lead* ao mesmo vendedor é importante por dois fatores: evita atrito entre vendedores, uma vez que existe comissão em jogo; e melhora a comunicação com o cliente, caso o *lead* vá para um vendedor diferente, ele achará que o *lead* é novo e a comunicação será menos eficiente. Para que essa regra seja respeitada é criada uma tabela index de *user/lead*.

Com esse index *user/lead* podemos atribuir um *lead* a um vendedor, caso o *lead* seja novo, a atribuição se dará de forma aleatória, eliminando qualquer possibilidade de viés e que um vendedor acabe sempre recebendo melhores clientes. Portanto utiliza-se o “user_index” da tabela

de usuários e caso ele seja nulo (*lead* novo), é gerado um número aleatório de 1 ao total de usuários disponíveis na loja (igual foi feito com os usuários, mas de forma aleatória).

Cruzando o "user_index" da tabela de usuários com o da tabela de clientes é feita a atribuição de todos clientes da base. Através de uma *window function* (*row_number*) é feito um *ranking* de clientes por usuário, sendo 1 o melhor cliente que pode ser atribuído a um determinado vendedor.

13.3.3. Tratamento Final dos Dados

Nosso planejamento é executar uma rotina todo dias às 21h que agenda contatos para o dia seguinte às 18h (o horário funciona como sugestão, o usuário pode realizá-lo antes), para isso é criado um campo usando a função "current_date", que retorna a data de hoje, e a função "date_add", que adiciona um dia a data, gerando como *output* a data de agendamento desejada.

Para gerar a mensagem sugerida foram criadas duas tabelas de *mapping*: de usuários, que mapeia id do usuário, loja em que ele trabalha, cupom, link e nome; e de mensagens, uma vez que cada marca da Empresa de Varejo tem uma mensagem sugerida e um *lead* novo tem uma mensagem diferente de um *lead* antigo. Com isso, utilizando de forma repetida a função "replace", substitui-se o nome do usuário, cupom e nome do cliente para os valores referentes a cada usuário e cliente. Gerando assim, uma mensagem completamente personalizada para cada cliente.

Por fim, com um filtro que seleciona *rankings* de clientes menores ou iguais a 20, seleciona-se os melhores 20 clientes para cada usuário.

14. MENSURAÇÃO

Este capítulo descreve todo desenvolvimento de uma estrutura de mensuração do Clerk, desde a estrutura feita em SQL até o desenvolvimento de um *Dashboard.* Esta estrutura possibilita tanto o levantamento de dados para a validação deste trabalho como também uma plataforma de acompanhamento de resultados que será utilizada pela Elementar e pela Empresa de Varejo.

De mensuração é fornecer à Empresa de Varejo um *dashboard* atualizado diariamente no formato D-1, ou seja, possui dados até a data anterior à data atual. Dessa forma o cliente pode monitorar o desempenho geral de vendas, o desempenho por marca, o desempenho por loja e até o desempenho por vendedor.

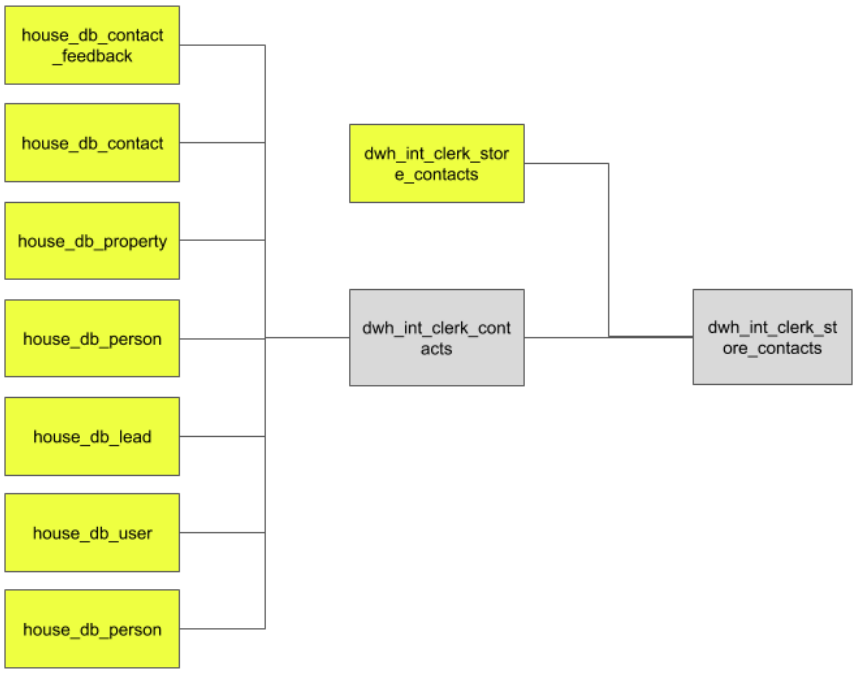
Para possibilitar a automação a solução será novamente, atravésde SQL e o código desenvolvido pelo autor se encontra no Apêndice E. Os dados serão disponibilizados no Google DataStudio, alimentado por uma planilha no Google Sheets. O autor foi o responsável pelo

processo de programação de SQL e de criação do relatório no DataStudio, o departamento de TI da empresa realizou a automação da rotina diária que importa os dados da *query* para o Sheets diariamente.

14.1. MENSURAÇÃO DE CONTATOS

A primeira parte da estrutura refere-se a mensuração dos contatos. Todos os dados são retirados da plataforma do Clerk (“house_db” indica que se trata de uma tabela de dados extraída da plataforma). Seu objetivo é analisar dados operacionais: quantos contatos cada loja está fazendo, quantos contatos atrasados temos, quantas respostas de clientes temos etc.

Figura 19: Fluxograma da Estrutura de SQL de mensuração de contatos



Fonte: Autor

Através de operações lógicas utilizando os valores do *status* dos contatos, *status* dos *leads*, e *feedbacks* de contatos, são calculadas as seguintes variáveis:

- m_NoWpp: número de contatos sem WhatsApp;
- m_YesWpp: número de contatos com WhatsApp;
- m_ExcludedNoWpp: número de contatos excluídos que não possuem WhatsApp;
- m_ExcludedYesWpp: número de contatos excluídos que possuem WhatsApp;
- m_Pending: número de contatos pendentes;
- m_PendingNoWpp: número de contatos pendentes que não possuem WhatsApp;
- m_LatePending: número de contatos atrasados;
- m_LatePendingNoWpp: número de contatos atrasados que não possuem WhatsApp;
- m_ContactsRealized: número de contatos realizados;
- m_NoFeedback: número de contatos sem *feedback*;
- m_LateNoFeedback: número de contatos atrasados sem *feedback*;
- m_YesFeedback: número de contatos com *feedback*;
- m_AnswersClients: número de contatos com respostas;

- m_NoAnswersClients: número de contatos realizados sem respostas;
- m_Optout: número de contatos que pediram para não receber mais mensagens.

Essa quebra realizada entre contatos com e sem WhatsApp foi feita pois dessa forma conseguimos analisar se os contatos não estão sendo feitos porque o vendedor não está usando a plataforma ou porque o cliente não tem WhatsApp.

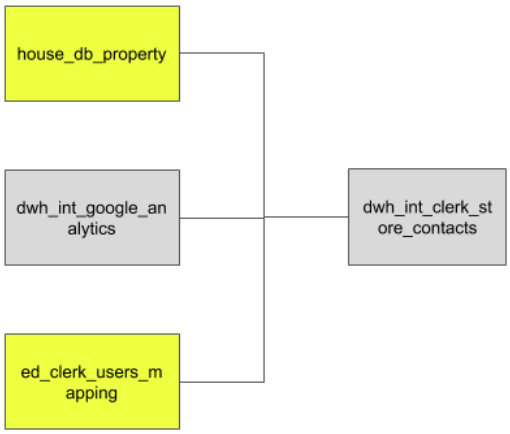
Esse conjunto de variáveis nos permite analisar toda utilização do Clerk e é vital para o cliente conseguir analisar o uso do projeto que por ele foi comprado.

14.2. MENSURAÇÃO DO TRÁFEGO

Para saber o quanto de tráfego é gerado no site provindo do Clerk, fizemos *links* encurtados (bit.ly) personalizados para cada usuário, o *link* da home da marca da loja do vendedor, seguido de um *traking* por UTM (*Urchin Tracking Module*) que mapeia como fonte "CRM", como canal, "WhatsApp" e como campanha, o id da loja e o id do usuário.

Dessa forma, podemos associar os dados presentes no Google Analytics com vendedores e lojas, analisando quantos clientes clicam efetivamente no *link*.

Figura 20: Fluxograma da Estrutura de SQL de mensuração de tráfego



Fonte: Autor

Esta a parte mais simples da estrutura, se trata apenas de um tratamento dos dados vindos da *view* "dwh_int_google_analytics" que já está presente na estrutura de BI padrão da empresa. Aplica-se um filtro para selecionar apenas o tráfego vindo de CRM/WhatsApp e cruza-se as tabelas de *property* e *users mapping* para obter as informações necessárias no *output*.

Desta forma, conseguimos mensurar as seguintes variáveis:

- M_NewUsers: Novos usuários;

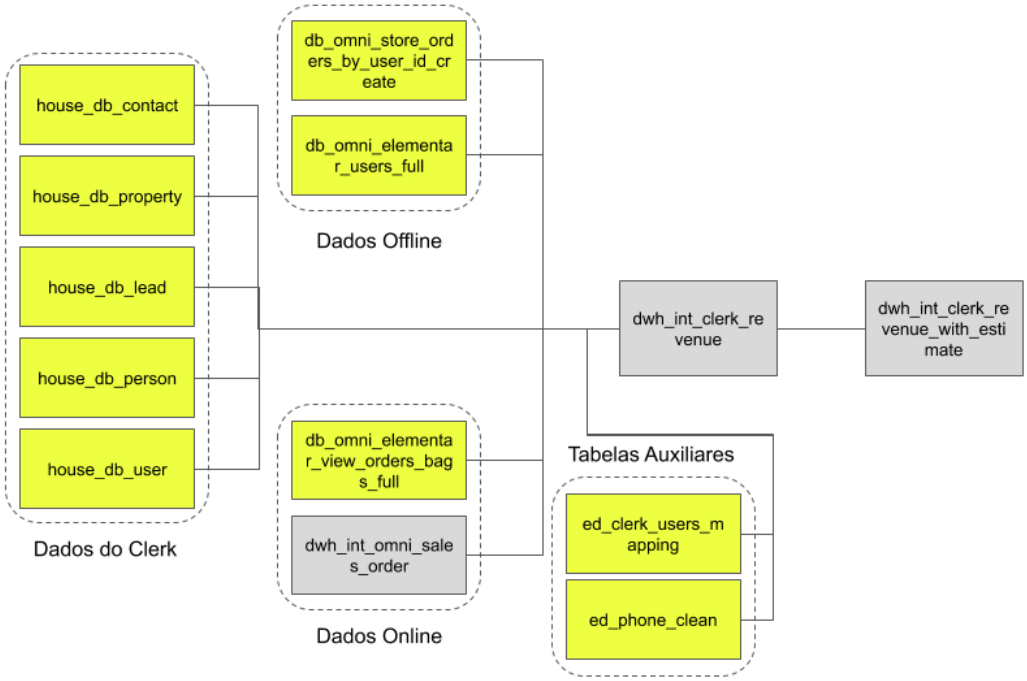
- M_Users: Usuários;
- M_Sessions: Número de sessões, difere de usuários, pois um mesmo usuário pode ter mais de uma sessão;
- M_PageViews: Número de visualizações de página;
- M_UniquePageViews: Número de visualizações únicas de página, um mesmo usuário pode visualizar mais de uma vez uma página em uma sessão;
- M_Transactions: Número de transações (mensuradas pelo Google Analytics);
- M_TransactionRevenue: Valor de receita gerado pelas sessões (mensurado pelo Google Analytics).

Entretanto, esses números são subestimados, uma vez que nem todos usuários clicam no *link*, muitos usuários acessam o site de forma direta ou através de pesquisas orgânicas do Google. Portanto, mesmo que tenhamos levantado o número para realizar acompanhamentos, nenhuma análise concreta pode ser feita.

14.3. MENSURAÇÃO DE VENDAS

Para realizar a mensuração de vendas, utilizamos tanto as vendas da plataforma *online* do cliente quanto as vendas da plataforma *offline*.

Figura 21: Fluxograma da Estrutura de SQL de mensuração de vendas



Fonte: Autor

Para o caso das compras *offline*, a chave utilizada para encontrar vendas foi o número de telefone do usuário do clerk. Em teoria o cliente Empresa de Varejo tem um mesmo cadastro tanto para compras online quanto para offline, entretanto, ao analisarmos esses dados mais a fundo, notamos uma quantidade significativo de usuários que possuíam dois cadastros, então a melhor forma de cruzar vendas de usuários foi pelo número de telefone. Também incluímos para essas vendas o conceito de janela de conversão de 30 dias. Isto é, consideramos compras de um determinado telefone apenas em um intervalo definido da data de contato dele no Clerk mais 30 dias. Desta forma, não consideramos compras feitas antes do contato nem compras realizadas a mais de 30 dias, neste último caso, a probabilidade do cliente ter comprado por um motivo que não seja o contato realizado no Clerk é muito maior do que a probabilidade do cliente ter comprado devido o contato do Clerk.

Para vendas *online*, consideramos dois de chaves distintas: vendas através de telefone e vendas através de cupom. O primeiro caso é semelhante à utilização de telefone como chave nas compras *offline*; o segundo caso, de vendas através de cupom, foi criado pois cada vendedor recebe um cupom único para ser usado no Clerk, este cupom dá 5% de desconto a compradores e a condição de frete grátis. Este método duplo de identificação de compras foi criado pois nem sempre

o cliente possui o mesmo telefone cadastrado *online* ou pode ocorrer de outro membro da família realizar compras pela pessoa contatada, em ambos os casos, queremos considerar essas vendas. O método duplo minimiza o caso de vendas que seriam do Clerk, mas que não conseguimos identificar.

Para o caso de *match* por telefone para cupom, foi utilizada a metodologia de janela de conversão de 30 dias novamente, uma vez que não temos telefone, não temos data de contato, portanto foi utilizada a data em que o vendedor entrou no Clerk como período válido de atribuição de seu cupom a compras.

Foi tomado o devido cuidado para que, uma vez que muitas compras possuem identificação de telefone e cupom, nenhuma venda seja duplicada. Ao juntar vendas por cupom com vendas por telefone, os dados foram agrupados através do id da compra, logo as compras são únicas.

Desta forma, as seguintes métricas são calculadas:

- M_Buyers: Número de compradores (um comprador pode realizar mais de uma compra);
- M_NumberOfOrdersCaptured: Número de transações captadas, ou seja, o total de transações realizadas, podendo ser pagas, pendentes ou canceladas;
- M_NumberOfOrdersInvoiced: Número de transações faturadas, ou seja, que já foram pagas pelo cliente;
- M_SalesCaptured: Receita total captada;
- M_SalesInvoiced: Receita total faturada.

Por fim, em um *query* diferente, calcula-se os dados de transações faturadas estimadas (M_NumberOfOrdersInvoicedEstimate) e receita faturada estimada (M_SalesInvoicedEstimate). Essas variáveis são muito úteis uma vez que os dados de vendas faturadas são subestimados se nos dias mais recentes, uma vez que os clientes podem ainda pagar esses pedidos. Para solucionar isso o autor desenvolveu a metodologia de venda faturada estimada, que é calculada com a seguinte regra:

- Caso a compra foi feita há mais de 14 dias: utiliza-se o valor de venda faturado;
- Caso a compra tenha sido realizada em 14 dias ou menos: multiplica-se o valor de venda captado por um estimador de faturamento.

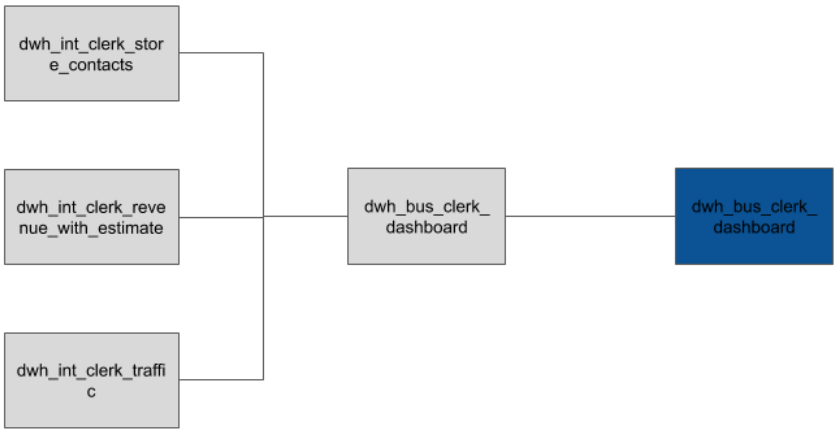
O estimador de faturamento é a esperança de faturamento de uma determinada loja, utilizando como base de cálculo toda as transações dessa loja em um período maior que 14 dias

(passado) e menor que 104 dias, o estimador é calculado pelo produto entre a receita faturada e a receita captada.

14.4. TRATAMENTO FINAL DOS DADOS

A etapa final deste processo de mensuração é unir todos os dados de contato, tráfego e vendas obtidos.

Figura 22: Fluxograma da Estrutura de SQL final de mensuração



Fonte: Autor

Os dados das três fontes são unidos através da função "*UNION ALL*" e a *query* final dwh_exp_clerk_dashboard apenas realiza um filtro de data. Os dados desta *query* final são exportados para uma planilha do Google Sheets.

14.5. CRIAÇÃO DO *DASHBOARD*

Para a visualização dos dados de maneira fácil pelo cliente, foi criado, com o auxílio da ferramenta DataStudio do Google, um *dashboard* de acompanhamento.

No relatório foram criadas quatro páginas: "*Overview*"; "KPI Operacional / Loja"; "KPI Operacional / Dia"; e "KPI Operacional / Usuário". Todas as páginas podem ser encontradas no Apêndice A.

Na primeira página, foram criadas tabelas de acompanhamentos dos dados de contatos por loja, pois é o dado que o cliente comunicou ser mais importante.

Também foi criada uma ilustração de funil de vendas, que mostra visualmente as perdas de contatos em cada etapa.

Na segunda página do *report* foram desenvolvidas quatro tabelas, agrupadas pela dimensão "loja", contendo as seguintes métricas:

- Contatos Agendados;
- Com Whatsapp;
- Sem WhastApp;
- Com Whatsapp - excluídos;
- Sem Whatsapp - excluídos;
- Contatos Realizados;
- Contatos Pendentes;
- Contatos Realizados;
- Contatos Pendentes;
- *Feedback* Realizado;
- *Feedback* Pendente;
- *Feedback* Atrasado;
- Respostas;
- *Optout*;
- Números com Whatsapp;
- Taxa de contato;
- Contato Realizado / Contato com whatsapp;
- Taxa de Resposta;
- Taxa de *Optout*;
- Contatos Realizados;
- Pedidos Faturados Estimados;
- Receita Faturada Estimada.

Dessa forma é possível analisar quais lojas estão melhor utilizando o Clerk e ver quanto de receita provém das mesmas.

A terceira página realiza o cálculo das mesmas métricas listadas na segunda página, mas agregadas pela dimensão “dia”, desta forma, a evolução diária das métricas de desempenho pode ser analisada.

Na quarta e última página, temos as mesmas variáveis listadas na segunda página, agora agregadas pela dimensão “Usuário” (ou seja, vendedor). Nesta página é possível analisar quais vendedores estão realizando o maior número de contatos e também podemos analisar quais clientes estão gerando mais receita a Empresa de Varejo.

15. RESULTADOS

Para preservar dados sensíveis, todos valores de receita foram multiplicados por uma constante C1, todos valores de contatos foram multiplicados por uma constante C2 e todos valores de transações foram multiplicados por uma constante C3. Foi necessário utilizar 3 constantes diferentes pois o cliente se negou a divulgar, além de valores absolutos, valores reais de ticket médio e de receita por contato. Dados como taxa de resposta e valores de *conversion lift* são os mesmos dos resultados originais.

Este capítulo expõem os resultados do Clerk entre os dias 23/03/2020 e 08/05/2020, desta forma podemos avaliar o impacto causado por este projeto nas vendas do cliente.

O início dos contatos se deu no dia 23/03/2020. Primeiramente, como teste da plataforma, apenas 14 lojas foram selecionadas. Desta forma, 14 usuários foram inseridos na plataforma, cada um responsável por uma loja. Os 14 primeiros funcionários da Empresa de Varejo a entrarem na plataforma foram gerentes regionais, que não participam direto das vendas, mas nessa ocasião testaram a plataforma pelos primeiros dias. Essa estratégia foi tomada uma vez que a comunicação com esses funcionários era muito mais fácil e eventuais *bugs* poderiam ser reportados de forma direta, outro fator da escolha, por parte da empresa, foi colocar pessoas com mais experiência para avaliarem a plataforma e assim decidir se o projeto completo seria comprado pela empresa.

O teste inicial foi um sucesso e o *rollout* completo foi requerido pela empresa. No dia 09/04/2020 foram adicionados mais 1.443 usuários, representando gerentes e vendedores de um total de 229 lojas.

Com isso, 20 contatos foram atribuídos por dia a cada usuário sem tarefas pendentes. Como descrito, atribuímos contatos apenas a usuários que não tinham nenhum contato atrasado para evitar que um usuário faça mais de 20 contatos por dia.

Tabela 9: Funil de Contatos

KPI	Valor
Contatos Agendados	536,607
Contatos com Whatsapp	402,455
Mensagens Enviadas	374,284
Respostas	74,857
Pedidos Faturados	26,200
% Contatos com WhatsApp	75%

% Mensagens Enviadas	93%
Taxa de Resposta	20%
Pedidos / Respostas	35%

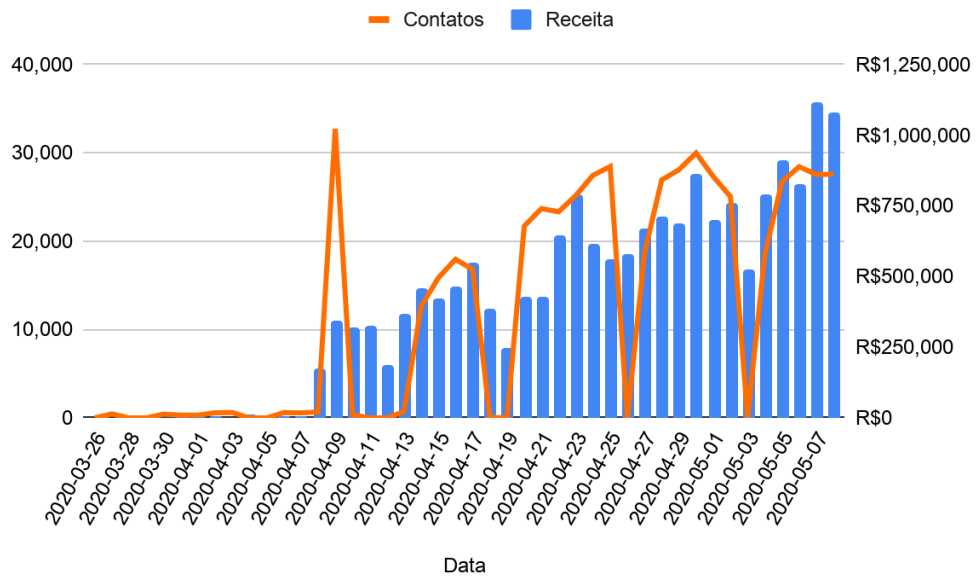
Fonte: Autor. Dados de 23/03/2020 a 08/05/2020.

Ao analisarmos os dados de contato através do funil, notamos que 75% dos números têm WhastApp, uma taxa acima de nossas expectativas, que mostra que existe uma tendência cada vez maior de utilização do WhatsApp, o que escancara o potencial de nossa iniciativa de realização de CRM através dessa plataforma.

Outro número importante de se destacar é que um a cada cinco clientes respondem as mensagens do vendedor, o que mostra uma altíssima taxa de interação. Para compararmos esse número, para o caso de buscas no Google contendo categorias da Marca A, excluindo buscas que contêm o nome da marca, temos um *click* a cada 20 impressões. O que mostra que a interação é pelo menos quatro vezes maior pelo WhatsApp, podendo ser até maior que isso, uma vez que um cliente que preferiu não responder ainda pode clicar no *link* ou até entrar no site por outro canal de marketing, tendo sido impactado originalmente no WhatsApp.

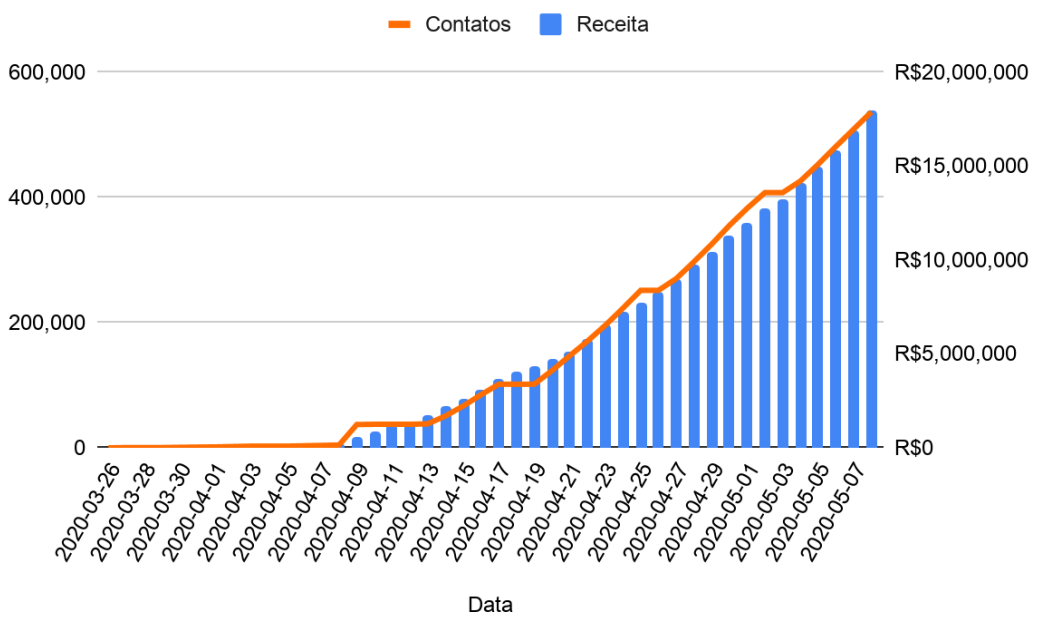
Utilizando a estrutura de mensuração criada foi possível analisar o impacto da utilização da plataforma na geração de receita da empresa. Até a data de 08/05/2020, apenas 29 dias após o *rollout* total do sistema, foi mensurada uma receita total de R$17,987,218 geradas por um total de 536,607 contatos. Com uma média de R$591,969 faturados ao dia pós *rollout* completo, chegando a um pico de R$1,114,759 em um dia.

Figura 23: Gráfico de Receita por Dia



Fonte: Autor. Dados de 23/03/2020 a 08/05/2020.

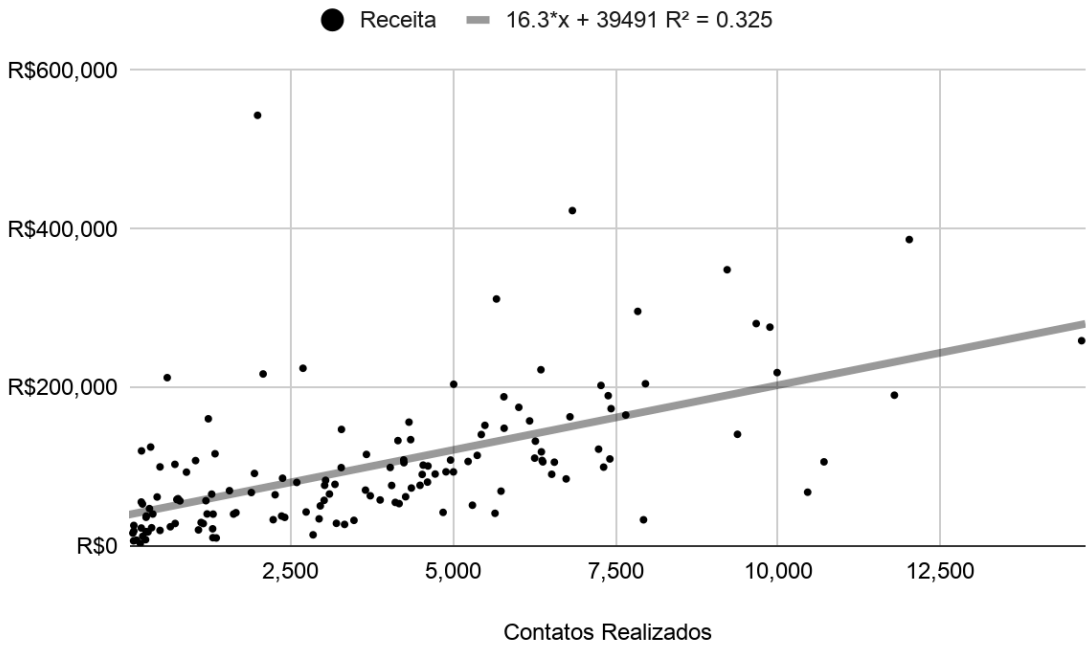
Figura 24: Gráfico de Receita por Dia acumulado



Fonte: Autor. Dados de 23/03/2020 a 08/05/2020.

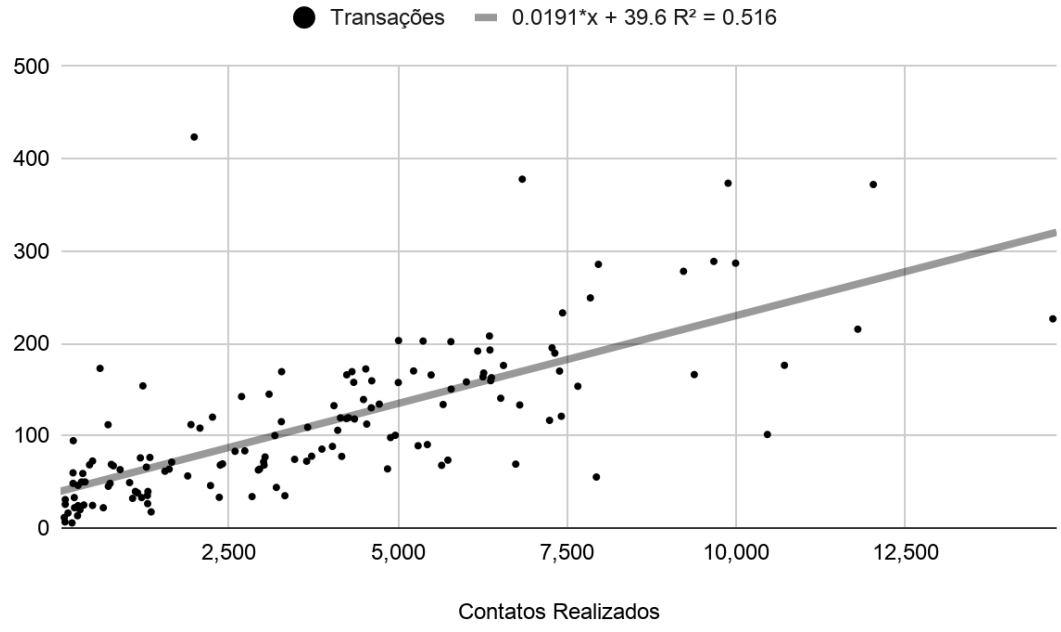
Durante o período analizado, as vendas através do Clerk representaram, cerca de 9% das vendas totais. Colocando o Clerk como o quarto maior canal de vendas, ficando atrás somente de Facebook e Google, cujo investimento é muito maior daquele realizado pela empresa no Clerk, e Google orgânico, que é um canal que gera bastante receita para a empresa, mas também envolve gastos maiores indiretos como SEO ou até campanhas publicitárias *offline* (televisão, *outdoors* etc).

Figura 25: Gráfico de Receita da loja em função dos Contatos realizados por ela



Fonte: Autor. Dados de 23/03/2020 a 08/05/2020.

Figura 26: Gráfico de Transações da loja em função dos Contatos realizados por ela



Fonte: Autor. Dados de 23/03/2020 a 08/05/2020.

Figura 27: Matriz de Correlação entre Contatos, Receita e Transações

Correlation Matrix

	Contatos	Receita	Transações
Contatos	—		
Receita	0.669 ***	—	
Transações	0.792 ***	0.888 ***	—

Note. * p < .05, ** p < .01, *** p < .001

Fonte: Autor utilizando o *Software* Jamovi. Dados de 23/03/2020 a 08/05/2020.

Analisando os dados podemos concluir que existe uma correlação forte entre contatos realizados por uma loja e o número de transações, ao analisarmos a receita, concluímos existir uma correlação moderada com o número de contatos realizados.

Mostrar essa correlação é um ponto importantíssimo da validação de nosso modelo de mensuração. Uma vez que os dados de contato são retirados da plataforma do Clerk e as transações são retiradas da plataforma de venda, provar que os dois são correlacionados mostra que o modelo funciona. Uma vez que os dados de contato são mais simples de obter e vêm de uma plataforma já validada, podemos considerar seus erros praticamente inexistentes. Uma vez que os dados de receita estão correlacionados com dados confiáveis, podemos admitir como sendo mais provável a hipótese de nosso modelo de mensuração estar correto.

Outra análise que podemos fazer é a de *conversion lift*, para isso separamos dois grupos de clientes: clientes contactos e clientes que o contato não foi realizado. Desta forma, comparando a receita por contato entres estes dois grupos podemos mensurar precisamente o impacto que o Clerk tem em gerar receita adicional, uma vez que o cliente poderia realizar uma compra de qualquer forma, com ou sem contato.

Tabela 10: Análise de *Conversion Lift*

	Marca B	Marca A	Total
Clientes não receberam contato (controle)	84,142	103,278	187,420
Clientes que receberam contato (experimento)	273,441	355,555	628,996
Total de clientes	357,583	458,833	816,416
Vendas controle	R$780,118	R$2,322,952	R$3,103,070

	Marca B	Marca A	Total
Vendas experimento	R$4,282,596	R$9,418,403	R$13,700,999
Vendas por cliente controle	R$9.3	R$22.5	R$16.6
Vendas por cliente experimento	R$15.7	R$26.5	R$21.8
Aumento de vendas	R$6.4 *69%*	R$4.0 *18%*	R$5.2 *32%*

Fonte: Autor. Dados de 25/04/2020 a 24/05/2020

Com isso, percebemos que temos um *conversion lift* substancial, principalmente na Marca B. Mostrando mais uma vez o grande impacto positivo do Clerk na receita final do cliente.

16. CONCLUSÃO

Para a Elementar existia a vontade de ampliação do Clerk, que antes do início do projeto era construído apenas para o uso de incorporadoras no setor imobiliário, para o varejo. Esse projeto não só possibilitou isso, como gerou uma nova fonte de renda a Elementar, uma vez que o projeto foi vendido à Empresa de Varejo. Hoje, o Clerk é uma plataforma com aplicações bem sucedidas tanto no setor imobiliário quanto no setor de varejo. Com isso, cria-se uma confiança para ampliar as vendas B2B do Clerk e talvez torná-lo a principal fonte de renda da Elementar à médio prazo.

Para a Empresa de Varejo, inicialmente existia a necessidade de utilizar melhor seus vendedores, este projeto possibilitou uma atribuição de clientes a vendedores e a mensuração de contatos em um patamar incomparável com o anterior. Anteriormente, os vendedores pegavam individualmente o contato de clientes com alto ticket médio, não existia nenhum tipo de mensuração, não existia nenhum tipo de otimização e a empresa acabava fazendo CRM apenas com uma pequena parcela de clientes, nosso projeto escalou a comunicação da empresa para um patamar de centenas de milhares de clientes contatados por mês.

Após a pandemia de COVID-19, surgiu uma necessidade urgente da Empresa de Varejo e aumentar suas vendas online e tentar recuperar o impacto que o fechamento de todas suas lojas físicas teriam no faturamento da empresa. Com a utilização do Clerk, somente no primeiro mês foi possível estabelecer esse novo canal com o quarto maior canal de vendas da empresa, se tornando uma importante fonte de renda para e empresa em um momento de crise. Hoje existe total confiança

da Empresa de Varejo na plataforma e isso fornece uma segurança para Elementar desenvolver ainda mais esse projeto.

Este projeto começa com uma iniciativa do autor de trazer uma abordagem estatística para a Elementar que a empresa ainda não tinha. A possibilidade de selecionar clientes para contato foi tomada como uma possibilidade tanto de aprendizado do autor quanto como de estruturação de um novo tipo de análise da empresa. Estes dois objetivos do autor foram extremamente bem sucedidos.

Hoje a Elementar conta com a aplicação da aprendizagem estatística em seu portfólio de análises e existe o plano de utilizar modelos de *machine learning* para solucionar problemas de clientes e também existe a possibilidade de implementação do Clerk em outros clientes de varejo. Também iniciou-se um projeto de aplicação destes modelos estatísticos aqui descritos nas aplicações do Clerk no setor imobiliário.

Porém, o maior sucesso deste trabalho foi a aprendizagem do autor, uma vez que o projeto foi, quase que completamente, desenvolvido individualmente. Desde a comunicação da possibilidade de se realizar um projeto desse tipo por parte de um dos sócios da empresa, todo planejamento e execução foi realizado pelo autor, a única etapa deste projeto que não foi desenvolvida individualmente pelo autor foi o desenvolvimento da plataforma do Clerk, pois não é de sua competência. A multidisciplinaridade do trabalho abrangeu CRM, ciência de dados, tratamento de dados através de SQL avançado, estatística, aprendizagem estatística em R, relacionamento com o cliente, uma vez que no piloto o próprio autor contatou os clientes, e *business intelligence* através da criação de um *dashboard* para acompanhamento de desempenho. Este projeto envolveu a aplicação tanto de conceitos aprendidos na graduação, como CRM, estatística e SI, quanto de conceitos aprendidos de forma extracurricular pelo autor para realizar esse projeto, como SQL avançado e aprendizagem estatística.

Fica evidente, portanto a importância da aplicação prática de conceitos aprendidos na Escola, tanto para geração de valor para a empresa e clientes quanto para o aprendizado do aluno.

REFERÊNCIAS

GARETH, J. et al. An Introduction to Statistical Learning: with Applications in R. Springer. 2013.

HAIR, J. Análise Multivariada de Dados. 6ª Edição. Porto Alegre : Bookman, 2009.

CHEN, J. Popovich, K. Understanding customer relationship management (CRM). Business process management journal, 2003.

THARWAT, A. Linear vs. quadratic discriminant analysis classifier: a tutorial. International Journal of Applied Pattern Recognition, 2016.

PETERSON, L. K-nearest neighbor. Scholarpedia, 2009. Disponível em: <http://scholarpedia.org/article/K-nearest_neighbor>. Acesso em: 18 Maio 2020.

RSTUDIO. Interface para utilização da linguagem R. Versão 1.1.456. Disponível em: <https://rstudio.com/>. Acesso em: 20 Maio 2020.

DBEAVER: Universal Database Tool. Versão 6.1.4. Disponível em: <https://dbeaver.io/>. Acesso em: 20 Maio 2020.

JAMOVI. Versão 1.1.9.0. Disponível em: <https://www.jamovi.org/>. Acesso em: 20 Maio 2020.

PRESTO. Presto 0.234.2 Documentation. Disponível em: <https://prestodb.io/docs/current/>. Acesso em: 09 Maio 2020.

ADOBE. Digital Distress Infographic. ADOBE, 2013. Disponível em: <https://www.adobe.com/aboutadobe/pressroom/pdfs/Adobe_Digital_Distress_Infographic.pdf>. Acesso em: 05 Maio 2019.

MCGRATH, F. 87% of internet users now have a smartphone. Globalwebindex.com, 2016. Disponível em: <https://blog.globalwebindex.com/chart-of-the-day/87-of-internet-users-now-have-a-smartphone/>. Acesso em: 07 Maio 2019.

ROGGIO, A. 7 Steps to Improve Your Email Marketing. Practical Ecommerce, 2011. Disponível em: <https://www.practicalecommerce.com/7-Steps-to-Improve-Your-Email-Marketing>. Acesso em: 07 Maio 2019.

TECMUNDO. Brasil é o terceiro país com mais usuários no Facebook. Tecmundo, 2019. Disponível em: <https://www.tecmundo.com.br/redes-sociais/139130-brasil-terceiro-pais-usuarios-facebook.htm>. Acesso em: 07 Maio 2019.

STATISTA. Amazon Challenges Ad Duopoly. Statista, 2019. Disponível em: <https://www.statista.com/chart/17109/us-digital-advertising-market-share/>. Acesso em: 07 Maio 2019.

FORBES. Facebook Inc. domina o cenário das mídias sociais. Forbes, 2017. Disponível em: <https://forbes.uol.com.br/negocios/2017/07/facebook-inc-domina-o-cenario-das-midias-sociais/>. Acesso em: 07 Maio 2019.

CISCO. Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper. Cisco, Trend 4: Applications traffic growth. 2019.

E-COMMERCE NEWS. Digital representa R$ 14,8 bilhões de investimento publicitário no Brasil em 2017. E-Commerce News, 2018. Disponível em: <https://ecommercenews.com.br/noticias/pesquisas-noticias/digital-representa-r-148-bilhoes-de-investimento-publicitario-no-brasil-em-2017/>. Acesso em: 07 Maio 2019

CNBC. Amazon is eating into Google's most important business: Search advertising. CNBC, 2019. Disponível em: <https://www.cnbc.com/2019/10/15/amazon-is-eating-into-googles-dominance-in-search-ads.html>. Acesso em: 16 Maio 2020

NEWS FROM GOOGLE. Google Launches Self-Service Advertising Program. Google Press, 2000. Disponível em: <http://googlepress.blogspot.com/2000/10/google-launches-self-service.html>. Acesso em: 21 Abril 2020.

GOOGLE ADS HELP. How the Google Ads auction works. Google Support. Disponível em: <https://support.google.com/google-ads/answer/6366577?hl=en>. Acesso em: 21 Abril 2020.

GOOGLE ADS HELP. Quality Score: Definition. Google Support. Disponível em: <https://support.google.com/google-ads/answer/140351>. Acesso em: 21 Abril 2020.

ABIT. Dados gerais do setor referentes a 2018. Associação Brasileira da Indústria Têxtil e de Confecção, 2019. Disponível em: <https://www.abit.org.br/cont/perfil-do-setor>. Acesso em: 21 Abril 2020.

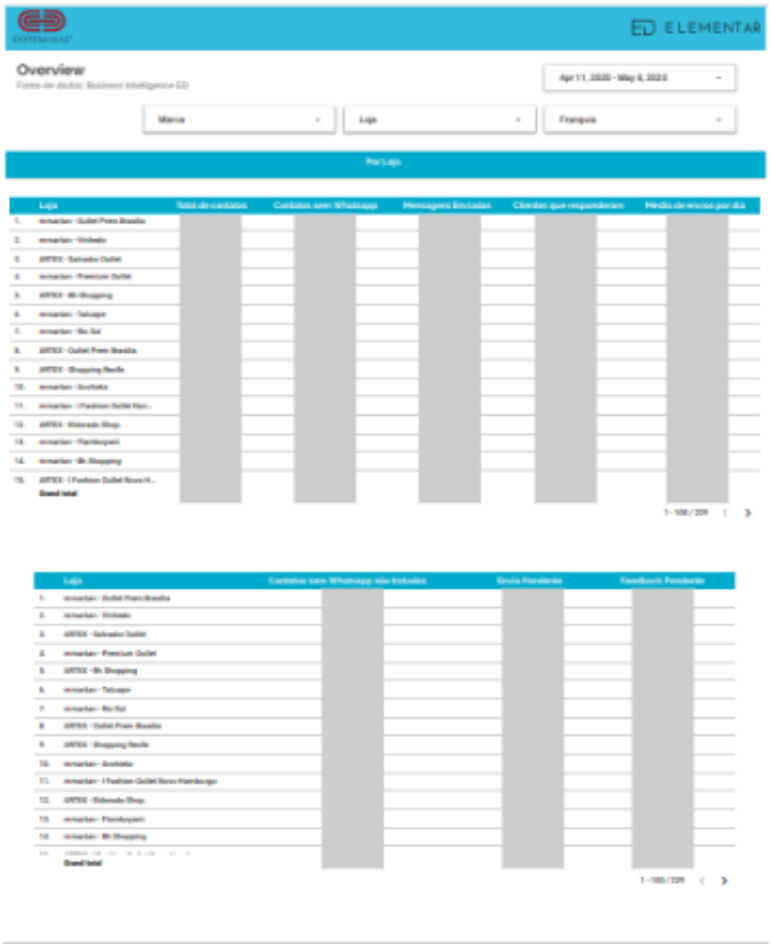
ADVFN. Empresa de Varejo (CTNM3 e CTNM4) teve lucro de R$ 231.59 milhões em 2018. ADVFN, 2019. Disponível em: <https://br.advfn.com/jornal/2019/04/Empresa de Varejo-ctnm3-e-ctnm4-teve-lucro-de-r-231-59-milhoes-em-2018>. Acesso em: 21 Abril 2020.

MORAES, D. Outbrain: como e por que usar publicidade nativa em sua estratégia digital. Rockcontent, 2019. Disponível em: <https://rockcontent.com/blog/outbrain/>. Acesso em: 23 Maio 2020.

FACEBOOK. Conversion Lift: Learn how your ads drive sales and conversions. Facebook for Business. Disponível em: < https://www.facebook.com/business/m/one-sheeters/conversion-lift>. Acesso em: 13 Junho 2020.

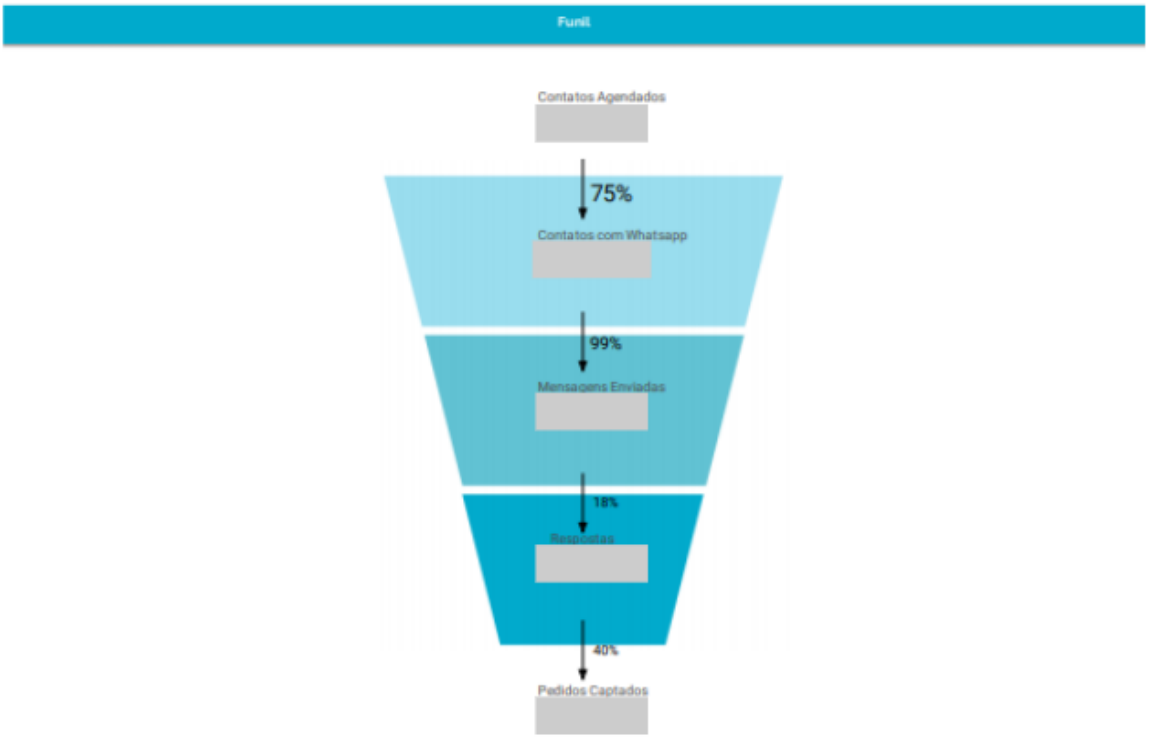
APÊNDICE A

Figura 28: Tabelas presentes na página "*Overview*"



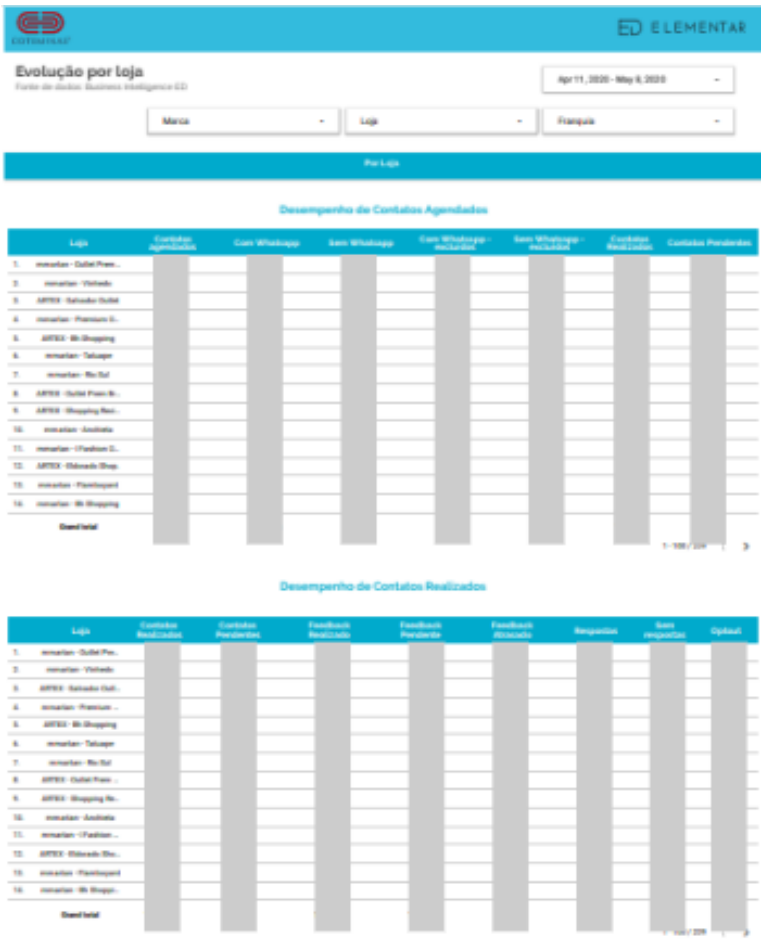
Fonte: Autor. Visualizado em 09 de Maio de 2020.

Figura 29: Representação de um funil de vendas na página "*Overview*"



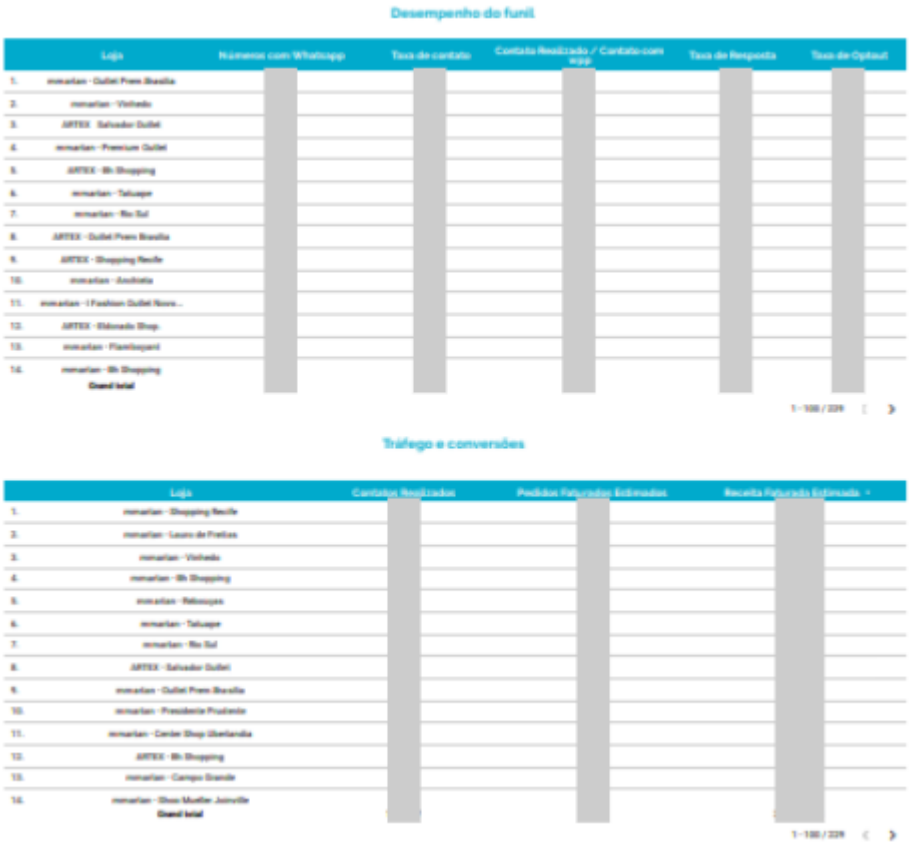
Fonte: Autor. Visualizado em 09 de Maio de 2020.

Figura 30: Primeira parte da página “KPI Operacional / Loja”



Fonte: Autor. Visualizado em 09 de Maio de 2020.

Figura 31: Segunda parte da página “KPI Operacional / Loja”

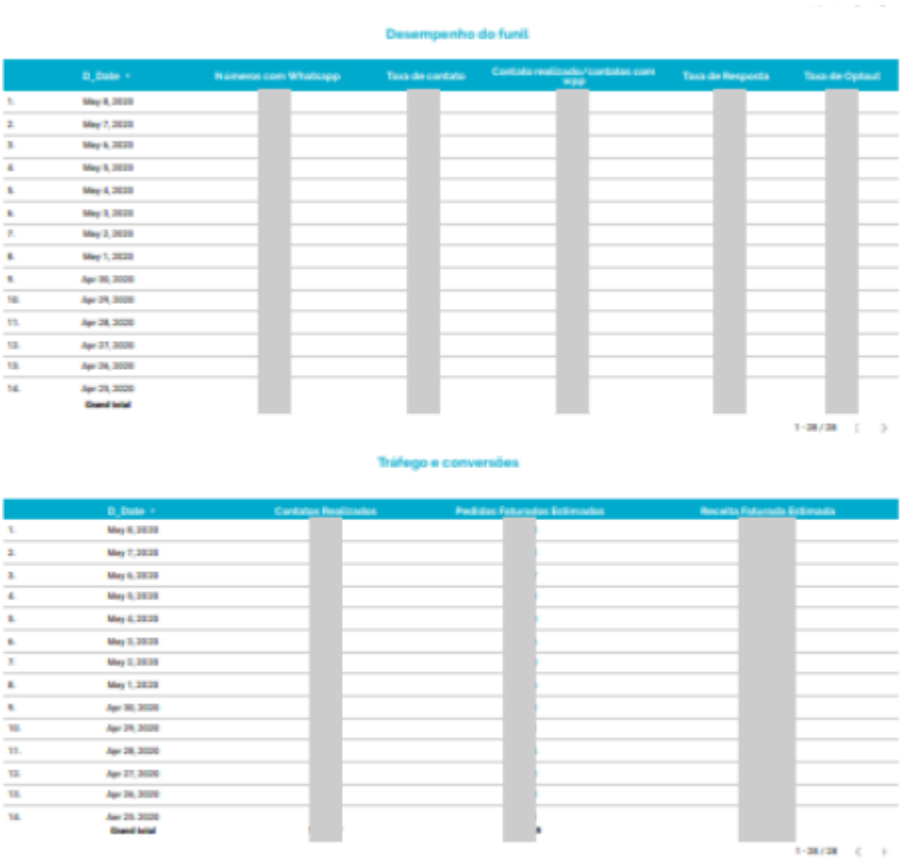


Fonte: Autor. Visualizado em 09 de Maio de 2020.

Figura 32: Primeira parte da página “KPI Operacional / Dia”

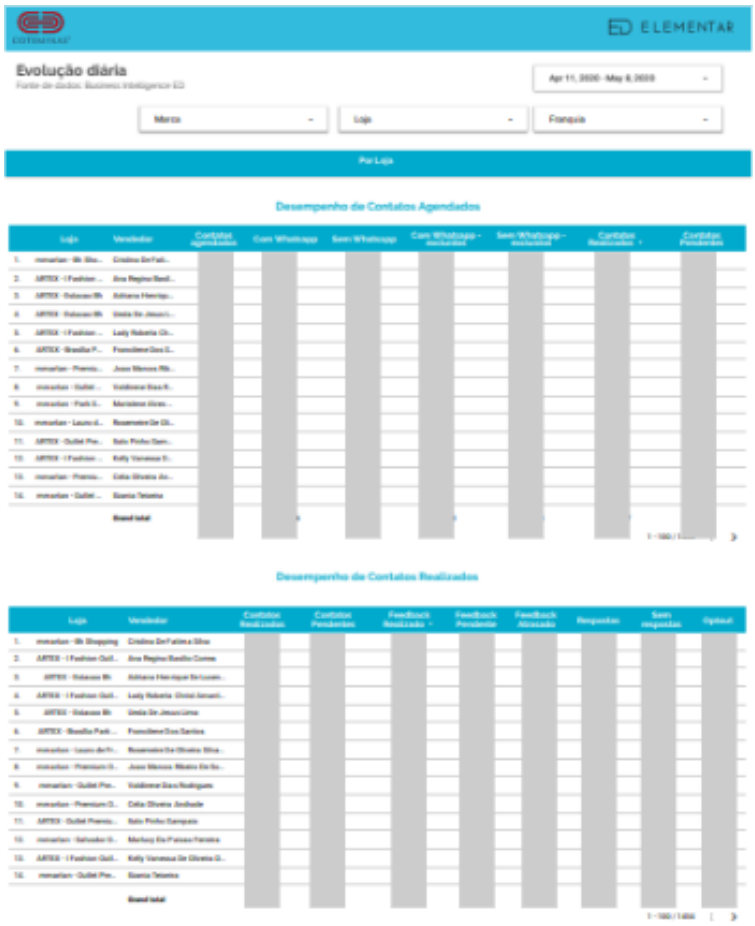
Fonte: Autor. Visualizado em 09 de Maio de 2020.

Figura 33: Segunda parte da página “KPI Operacional / Dia”



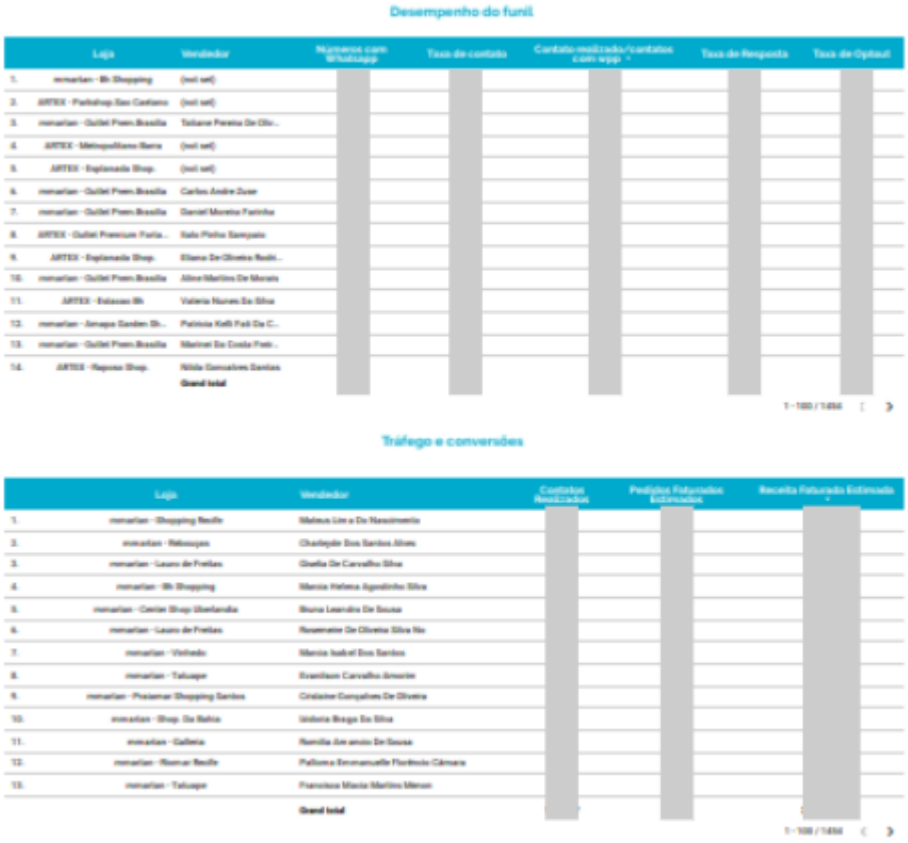
Fonte: Autor. Visualizado em 09 de Maio de 2020.

Figura 34: Primeira parte da página “KPI Operacional / Usuário”



Fonte: Autor. Visualizado em 09 de Maio de 2020.

Figura 35: Segunda parte da página “KPI Operacional / Usuário”



Fonte: Autor. Visualizado em 09 de Maio de 2020.

APÊNDICE B

Aprendizagem estatística em R:

```
# 0. Libraries ----------------------------------------------------------
library(readr)
library(caTools)
library(MASS)
library(class)
library(nFactors)
# 1. Data Loading and treatment -----------------------------------------------
# reading and attaching data set
set.seed(123) # setting a random number generator seed
clients_data <- read_csv("model_sample_data.csv")
attach(clients_data)
# removing NAs
clients_data <- clients_data[complete.cases(clients_data), ]
# correcting year month field
clients_data$d_yearmonth <- as.Date(d_yearmonth, "%Y-%m-%d")
# creating results table that will be used to compare models
resultados <- data.frame(metodo = character(),
                         taxa_acerto = numeric(),
                         num_selected_clients = numeric(),
                         stringsAsFactors = FALSE)
# 2. Train and Test separation ------------------------------------------------
# separating test and training sample
train_v <- c(d_yearmonth!=as.Date("2020-01-01")) # defining test as the month of january of 2020
#train_v <- sample.split( 1:nrow(clients_data), SplitRatio = 0.7) # defining test and train randomly
train <- clients_data[train_v,]
test <- clients_data[!train_v,]
# creating dependent variable vector
dv.test <- clients_data$d_hadbought[!train_v]
# removing id fields from train and test sample
train$d_userid <- NULL
train$d_yearmonth <- NULL
test$d_userid <- NULL
test$d_yearmonth <- NULL
# 3. Simple Logistic Regression -----------------------------------------------
# running the logistic regression
glm.fits <- glm(d_hadbought ~ .,
                data = train,
                family = binomial)
# calculating test error and saving results
logr.probs <- predict(glm.fits, test, type="response")
n.200 <- sort(logr.probs, decreasing = TRUE)[200]
logr.pred <- ifelse(logr.probs >= n.200, 1, 0)
#table(logr.pred, test$d_hadbought) # run this to see contrast table of the results
resultados[nrow(resultados)+1,] <- c("Logistic Regression",
                                     mean(dv.test == logr.pred),
                                     sum(as.integer(logr.pred)))
# 3.b. Simple Logistic Regression without non significant ------------------
# creating a vector that cleans non significant variables
glm.p_valeus <- summary(glm.fits)$coefficients[-1,4]
clean_v <- ifelse(glm.p_valeus < 0.05, TRUE, FALSE) # creating clean vector that filters non significant variables
clean_v <- c(TRUE,clean_v)
train.clean <- train[,clean_v]
```

```
test.clean <- test[,clean_v]
# running the logistic regression
glm.fits.clean <- glm(d_hadbought ~ .,
             data = train.clean,
             family = binomial)
# calculating test error and saving results
logr.probs.clean <- predict(glm.fits.clean, test.clean, type="response")
n.200.clean <- sort(logr.probs.clean, decreasing = TRUE)[200]
logr.pred.clean <- ifelse(logr.probs.clean >= n.200.clean, 1, 0)
#table(logr.pred.clean, test.clean$d_hadbought)
resultados[nrow(resultados)+1,] <- c("Logistic Regression Clean",
                              mean(test.clean$d_hadbought == logr.pred.clean),
                              sum(as.integer(logr.pred.clean)))
# 4. LDA ------------------------------------------------------------------
lda.fit <- lda(d_hadbought ~ .,
            data = train)
lda.pred <- predict(lda.fit, test)
#table(lda.pred$class, dv.test)
resultados[nrow(resultados)+1,] <- c("LDA",
                              mean(lda.pred$class == dv.test),
                              sum(as.integer(lda.pred$class)-1))
# 5. QDA ------------------------------------------------------------------
qda.fit <- qda(d_hadbought ~ .-d_hadbought,
            data = train)
qda.pred <- predict(qda.fit, test)
#table(qda.pred$class, dv.test)
resultados[nrow(resultados)+1,] <- c("QDA",
                              mean(qda.pred$class == dv.test),
                              sum(as.integer(qda.pred$class)-1))
# 6. KNN (Large running time) ---------------------------------------------------------------
# running knn with k from 1 to 10 and storing results
for (i in 1:10){
  knn.pred <- knn(train, test, train[,1, drop = TRUE], k=i, prob = TRUE)
  resultados[nrow(resultados)+1,] <- c(paste("KNN k =",i),
                                mean(as.double(knn.pred)-1 == test[,1]),
                                sum(as.integer(knn.pred)-1))
}
# 7. Linear Regression ----------------------------------------------------
# runnig the regression
lm <- lm(d_hadbought ~ .-d_hadbought,
         data = train)
# calculating test error and saving results
lprobs <- predict(lm, test)
n.200.lm <- sort(lprobs, decreasing = TRUE)[200]
lpred <- ifelse(lprobs >= n.200.lm, 1, 0)
#table(lpred, test$d_hadbought)
resultados[nrow(resultados)+1,] <- c("Linear Regression",
                              mean(test$d_hadbought == lpred),
                              sum(as.integer(lpred)))
# 8. Factor Analysis + LogReg ---------------------------------------------------
# Determine Number of Factors to Extract
ev <- eigen(cor(train[,-1])) # get eigenvalues
ap <- parallel(subject=nrow(train[,-1]),var=ncol(train[,-1]),
               rep=100,cent=.05)
nS <- nScree(x=ev$values, aparallel=ap$eigen$qevpea)
# Maximum Likelihood Factor Analysis
```

```
# entering raw data and extracting 4 factors (as found in late section),
# with varimax rotation
fit <- factanal(train[,-1], 4, rotation="varimax")
# criating train set dataframe with only factors
factorial.train <- train
factorial.train[,2:ncol(factorial.train)] <- NULL
factorial.train$Factor1 <- as.matrix(train[,-1]) %*% as.vector(fit$loadings[,1])
factorial.train$Factor2 <- as.matrix(train[,-1]) %*% as.vector(fit$loadings[,2])
factorial.train$Factor3 <- as.matrix(train[,-1]) %*% as.vector(fit$loadings[,3])
factorial.train$Factor4 <- as.matrix(train[,-1]) %*% as.vector(fit$loadings[,4])
# doing the same to the test sample
factorial.test <- test
factorial.test[,2:ncol(factorial.test)] <- NULL
factorial.test$Factor1 <- as.matrix(test[,-1]) %*% as.vector(fit$loadings[,1])
factorial.test$Factor2 <- as.matrix(test[,-1]) %*% as.vector(fit$loadings[,2])
factorial.test$Factor3 <- as.matrix(test[,-1]) %*% as.vector(fit$loadings[,3])
factorial.test$Factor4 <- as.matrix(test[,-1]) %*% as.vector(fit$loadings[,4])
# Running the log reg
factorial.glm.fits <- glm(d_hadbought ~ .,
            data = factorial.train,
            family = binomial)
factorial.logr.probs <- predict(factorial.glm.fits, factorial.test, type="response")
factorial.n.200 <- sort(factorial.logr.probs, decreasing = TRUE)[200]
factorial.logr.pred <- ifelse(factorial.logr.probs >= factorial.n.200, 1, 0)
# saving results
resultados[nrow(resultados)+1,] <- c("Logistic Regression (Factors)",
                             mean(dv.test == factorial.logr.pred),
                             sum(factorial.logr.pred))
# 9. Polynomial Logistic Regression ----------------------------------------------------
# plm 2
glm.fits.plm <- glm(d_hadbought ~ .+
                  I(M_HistoryFrequence^2),
                data = train,
                family = binomial)
# calculating test error and saving results
logr.probs.plm <- predict(glm.fits.plm, test, type="response")
n.200.plm <- sort(logr.probs.plm, decreasing = TRUE)[200]
logr.pred.plm <- ifelse(logr.probs.plm >= n.200.plm, 1, 0)
#table(logr.pred.plm, test$d_hadbought)
resultados[nrow(resultados)+1,] <- c("Logistic Regression Plm(2)",
                             mean(dv.test == logr.pred),
                             sum(logr.pred))
#
# plm 3
glm.fits.plm <- glm(d_hadbought ~ .+
                  I(M_HistoryFrequence^2) +
                  I(M_HistoryFrequence^3),
                data = train,
                family = binomial)

# calculating test error and saving results
logr.probs.plm <- predict(glm.fits.plm, test, type="response")
n.200.plm <- sort(logr.probs.plm, decreasing = TRUE)[200]
logr.pred.plm <- ifelse(logr.probs.plm >= n.200.plm, 1, 0)
#table(logr.pred.plm, test$d_hadbought)
resultados[nrow(resultados)+1,] <- c("Logistic Regression Plm(3)",
```

```
                            mean(dv.test == logr.pred),
                            sum(logr.pred))
#
# plm 4
glm.fits.plm <- glm(d_hadbought ~ .+
                     I(M_HistoryFrequence^2) +
                     I(M_HistoryFrequence^3) +
                     I(M_HistoryFrequence^4),
                   data = train,
                   family = binomial)
# calculating test error and saving results
logr.probs.plm <- predict(glm.fits.plm, test, type="response")
n.200.plm <- sort(logr.probs.plm, decreasing = TRUE)[200]
logr.pred.plm <- ifelse(logr.probs.plm >= n.200.plm, 1, 0)
#table(logr.pred.plm, test$d_hadbought)
resultados[nrow(resultados)+1,] <- c("Logistic Regression Plm(4)",
                            mean(dv.test == logr.pred),
                            sum(logr.pred))
#
# plm 5
glm.fits.plm <- glm(d_hadbought ~ .+
                     I(M_HistoryFrequence^2) +
                     I(M_HistoryFrequence^3) +
                     I(M_HistoryFrequence^4) +
                     I(M_HistoryFrequence^5),
                   data = train,
                   family = binomial)
# calculating test error and saving results
logr.probs.plm <- predict(glm.fits.plm, test, type="response")
n.200.plm <- sort(logr.probs.plm, decreasing = TRUE)[200]
logr.pred.plm <- ifelse(logr.probs.plm >= n.200.plm, 1, 0)
#table(logr.pred.plm, test$d_hadbought)
resultados[nrow(resultados)+1,] <- c("Logistic Regression Plm(5)",
                            mean(dv.test == logr.pred),
                            sum(logr.pred))
# 9.b. Polinomial LogReg with Factors ------------------------------------
pl.fc.glm.fits <- glm(d_hadbought ~ .
                     +I(Factor1)^2
                     +I(Factor2)^2
                     +I(Factor3)^2
                     +I(Factor4)^2,
                        data = factorial.train,
                        family = binomial)
pl.fc.logr.probs <- predict(pl.fc.glm.fits, factorial.test, type="response")
pl.fc.n.200 <- sort(pl.fc.logr.probs, decreasing = TRUE)[200]
pl.fc.logr.pred <- ifelse(pl.fc.logr.probs >= pl.fc.n.200, 1, 0)
#table(pl.fc.logr.pred, test$d_hadbought)
resultados[nrow(resultados)+1,] <- c(paste("Factor Logistic Regression Pln 2"),
                            mean(dv.test == pl.fc.logr.pred),
                            sum(pl.fc.logr.pred))
# 10. Current Month Prediction and Data Export ---------------------------------------------
# treating "resultados" data frame
resultados <- resultados[order(-resultados$taxa_acerto),] # ordering by lowest test error
resultados$taxa_acerto <- round(as.double(resultados$taxa_acerto), digits = 4) # rounding values
resultados # printing results table to choose the better model
write.csv(resultados,"resultados.csv", row.names = FALSE) # saving data frame to csv
```

```
# loading data from current month
current_month_data <- read_csv("current_month_data.csv")
# reg log clean
clean_v_final <- c(FALSE, FALSE, clean_v)
projection.clean <- predict(glm.fits.clean, current_month_data[,clean_v_final], type="response")
# reg log
projection <- predict(glm.fits, current_month_data[,3:ncol(current_month_data)], type="response")
# lda
lda.proj <- predict(lda.fit, current_month_data[,3:ncol(current_month_data)])
# saving log reg clean and lda outputs into the data frame
current_month_data$M_Score <- projection.clean
current_month_data$M_Class <- lda.proj$class
# saving output from the current month prediction
write.csv(current_month_data,"output.csv", row.names = FALSE)
# saving coefficients data frame into csv
coefficients <- as.data.frame(summary(glm.fits.clean)$coefficients[,1])
write.csv(coefficients, "output_coeficients_lr.csv", row.names = TRUE)
coefficients.lda <- as.data.frame(lda.fit$scaling)
write.csv(coefficients.lda, "output_coeficients_lda.csv", row.names = TRUE)
```

APÊNDICE C

SQL View: dwh_int_buy_prediction_full

```
-- creating a table that contains every year month in the data set
with store_orders as (
	select
	  date(date_parse(substr("date",1,10), '%Y-%m-%d')) d_ordercreationdate
	, userid as d_userid
	, amount/100 as m_total
	, discount/100 as m_discount
	from db_omni_store_orders_by_user_id_create
	where 1=1
		and date(date_parse(substr("date",1,10), '%Y-%m-%d')) >= date '2017-01-01'
)
, year_months0 as (
	select
	  date_trunc('month', d_ordercreationdate) D_YearMonth
	from store_orders
	union all select
	  date_trunc('month', current_date) D_YearMonth -- inputting current month, i was having problems to get
-- current month data when there's no sales yet
)
, year_months as (
	select distinct
	  *
	from year_months0
)
-- i've created this view to remove people that works in the company
, employees as (
	select distinct
	  userid as D_UserId
	, email
	from db_omni_elementar_users_full
	where 1=1
		and (false
			or email like '%@[nome_empresa].com%'
			or email like '%@[nome_grupo].com%'
			or email like '%@[nome_equipe].com%')
	union all select
	  d_userid
	, d_email
	from ed_store_employees
)
-- i needed to create a table here containing every ym and every client, this will be used in the window function
, clients_ym as (
	select distinct
	  store_orders.D_UserId AS D_UserId
	, year_months.D_YearMonth
	from store_orders
	left join year_months on 1=1
	left join employees on employees.d_userid = store_orders.d_userid
	where 1=1
		and d_ordercreationdate >= date '2017-01-01'
		and trim(store_orders.D_UserId) <> '' and store_orders.D_UserId is not null
		and employees.d_userid is null -- removing employees
```

```
)
-- this subquery calculates each client cohort
, cohort as (
	SELECT DISTINCT
	  store_orders.D_UserId
	, first_value(d_ordercreationdate) OVER (PARTITION BY store_orders.D_UserId ORDER BY
d_ordercreationdate ASC) D_FirstOrderDate
	, date_trunc('month', first_value(d_ordercreationdate) OVER (PARTITION BY store_orders.D_UserId
ORDER BY d_ordercreationdate ASC)) D_CohortYearMonth
	FROM store_orders
	left join employees on employees.d_userid = store_orders.d_userid
	WHERE 1=1
		and employees.d_userid is null -- removing employees
)
-- in this subquery i've listed all client transactions data grouped by client_id and d_ordercreationdate
, transactions_by_ym as (
	select
	  store_orders.D_UserId
   , date_trunc('month', d_ordercreationdate) as D_YearMonth
	, count(*) as M_CompletedPurchases
	, sum(M_total) as M_Revenue
	from store_orders
	left join employees on employees.d_userid = store_orders.d_userid
	where 1=1
		and trim(store_orders.D_UserId) <> '' and store_orders.D_UserId is not null
		and employees.d_userid is null -- removing employees
	group by 1, 2
)
-- this subquery calculates the last purchase month
, transactions_by_ym2 as (
	select
	  clients_ym.D_UserId
	, clients_ym.D_YearMonth
	, M_CompletedPurchases
	, M_Revenue
	, lag(transactions_by_ym.D_YearMonth, 1, transactions_by_ym.D_YearMonth) over (partition by
clients_ym.D_UserId order by clients_ym.D_YearMonth asc) D_LastPurchaseMonth
	from clients_ym
	left join transactions_by_ym on 1=1
		and clients_ym.D_UserId = transactions_by_ym.D_UserId
		and clients_ym.D_YearMonth = transactions_by_ym.D_YearMonth
)
select
  transactions_by_ym2.D_UserId
, cohort.D_CohortYearMonth
, case
	when M_Revenue > 0
	then 1
	else 0
  end as D_HadBought -- this var returns 1 if any purchase had occurred in the month and 0 otherwise, this will be
-- used as the response variable
, M_CompletedPurchases
, M_Revenue
, date_diff('month'
	, max(D_LastPurchaseMonth) over (partition by transactions_by_ym2.D_UserId order by D_YearMonth
asc)
```

```
        , D_YearMonth
  ) as M_MonthsSinceLastPurchase
, date_diff('month', cohort.D_CohortYearMonth, D_YearMonth) as M_MonthsAsACustomer
, sum(M_CompletedPurchases) over (partition by transactions_by_ym2.D_UserId order by D_YearMonth asc) -
coalesce(M_CompletedPurchases, 0)
        as M_HistoryCompletedPurchases
, sum(M_Revenue) over (partition by transactions_by_ym2.D_UserId order by D_YearMonth asc) -
coalesce(M_Revenue, 0)
        as M_HistoryRevenue
, D_YearMonth
from transactions_by_ym2
join cohort on 1=1
        and transactions_by_ym2.D_UserId = cohort.D_UserId
        and cohort.D_CohortYearMonth <= transactions_by_ym2.D_YearMonth
where cohort.D_CohortYearMonth is not null -- removing YMs prior to the client first purchase
```

SQL View: dwh_int_buy_prediction_full_quarters

```
select
  D_UserId
, date_trunc('quarter', d_yearmonth) D_Quarter
, max(D_HadBought) D_LagQ0HadBought
, sum(M_CompletedPurchases) M_LagQ0CompletedPurchases
, sum(M_Revenue) M_LagQ0Revenue
, lag(max(D_HadBought), 1) over (partition by D_UserId order by date_trunc('quarter', d_yearmonth) asc)
D_LagQ1HadBought
, lag(sum(M_CompletedPurchases), 1) over (partition by D_UserId order by date_trunc('quarter', d_yearmonth) asc)
M_LagQ1CompletedPurchases -- this var represents the sum of the metric in -1 quarters
, lag(sum(M_Revenue), 1) over (partition by D_UserId order by date_trunc('quarter', d_yearmonth) asc)
M_LagQ1Revenue
, lag(max(D_HadBought), 2) over (partition by D_UserId order by date_trunc('quarter', d_yearmonth) asc)
D_LagQ2HadBought
, lag(sum(M_CompletedPurchases), 2) over (partition by D_UserId order by date_trunc('quarter', d_yearmonth) asc)
M_LagQ2CompletedPurchases -- this var represents the sum of the metric in -2 quarters
, lag(sum(M_Revenue), 2) over (partition by D_UserId order by date_trunc('quarter', d_yearmonth) asc)
M_LagQ2Revenue
, lag(max(D_HadBought), 3) over (partition by D_UserId order by date_trunc('quarter', d_yearmonth) asc)
D_LagQ3HadBought
, lag(sum(M_CompletedPurchases), 3) over (partition by D_UserId order by date_trunc('quarter', d_yearmonth) asc)
M_LagQ3CompletedPurchases -- this var represents the sum of the metric in -3 quarters
, lag(sum(M_Revenue), 3) over (partition by D_UserId order by date_trunc('quarter', d_yearmonth) asc)
M_LagQ3Revenue
, lag(max(D_HadBought), 4) over (partition by D_UserId order by date_trunc('quarter', d_yearmonth) asc)
D_LagQ4HadBought
, lag(sum(M_CompletedPurchases), 4) over (partition by D_UserId order by date_trunc('quarter', d_yearmonth) asc)
M_LagQ4CompletedPurchases -- this var represents the sum of the metric in -4 quarters
, lag(sum(M_Revenue), 4) over (partition by D_UserId order by date_trunc('quarter', d_yearmonth) asc)
M_LagQ4Revenue
from dwh_int_buy_prediction_full_mv
group by 1, 2
```

SQL View: dwh_int_buy_prediction_full_all_months

```
select
  dwh_int_buy_prediction_full_mv.D_UserId -- id variable (will not be used in the model)
, dwh_int_buy_prediction_full_mv.D_YearMonth -- id variable
, dwh_int_buy_prediction_full_mv.D_HadBought -- Dependent variable
```

```
-- all the next variables are independent and qualitative
--, coalesce(D_LagQ0HadBought, 0) D_LagQ0HadBought
, coalesce(D_LagQ1HadBought, 0) D_LagQ1HadBought
, coalesce(D_LagQ2HadBought, 0) D_LagQ2HadBought
, coalesce(D_LagQ3HadBought, 0) D_LagQ3HadBought
, coalesce(D_LagQ4HadBought, 0) D_LagQ4HadBought
, lag(coalesce(D_HadBought, 0), 1, 0) over (partition by dwh_int_buy_prediction_full_mv.D_UserId order by
dwh_int_buy_prediction_full_mv.D_YearMonth asc) D_LagM1HadBought
, lag(coalesce(D_HadBought, 0), 2, 0) over (partition by dwh_int_buy_prediction_full_mv.D_UserId order by
dwh_int_buy_prediction_full_mv.D_YearMonth asc) D_LagM2HadBought
, lag(coalesce(D_HadBought, 0), 3, 0) over (partition by dwh_int_buy_prediction_full_mv.D_UserId order by
dwh_int_buy_prediction_full_mv.D_YearMonth asc) D_LagM3HadBought
, lag(coalesce(D_HadBought, 0), 4, 0) over (partition by dwh_int_buy_prediction_full_mv.D_UserId order by
dwh_int_buy_prediction_full_mv.D_YearMonth asc) D_LagM4HadBought

-- all the next variables are independent and quantitative
, coalesce(M_MonthsSinceLastPurchase, 0) M_MonthsSinceLastPurchase
, coalesce(M_MonthsAsACustomer, 0) M_MonthsAsACustomer
, coalesce(M_HistoryCompletedPurchases, 0) M_HistoryCompletedPurchases
, coalesce(M_HistoryRevenue, 0) M_HistoryRevenue
, coalesce(CAST(M_HistoryCompletedPurchases AS double), 0)/coalesce(M_MonthsAsACustomer,1) as
M_HistoryFrequence

-- here we have the lag quarter metrics
--, coalesce(M_LagQ0CompletedPurchases, 0) M_LagQ0CompletedPurchases
--, coalesce(M_LagQ0Revenue, 0) M_LagQ0Revenue
, coalesce(M_LagQ1CompletedPurchases, 0) M_LagQ1CompletedPurchases
, coalesce(M_LagQ1Revenue, 0) M_LagQ1Revenue
, coalesce(M_LagQ2CompletedPurchases, 0) M_LagQ2CompletedPurchases
, coalesce(M_LagQ2Revenue, 0) M_LagQ2Revenue
, coalesce(M_LagQ3CompletedPurchases, 0) M_LagQ3CompletedPurchases
, coalesce(M_LagQ3Revenue, 0) M_LagQ3Revenue
, coalesce(M_LagQ4CompletedPurchases, 0) M_LagQ4CompletedPurchases
, coalesce(M_LagQ4Revenue, 0) M_LagQ4Revenue

-- here i've calculated the month lags
, lag(coalesce(M_CompletedPurchases, 0), 1, 0) over (partition by dwh_int_buy_prediction_full_mv.D_UserId order
by dwh_int_buy_prediction_full_mv.D_YearMonth asc) M_LagM1CompletedPurchases
, lag(coalesce(M_Revenue, 0), 1, 0) over (partition by dwh_int_buy_prediction_full_mv.D_UserId order by
dwh_int_buy_prediction_full_mv.D_YearMonth asc) M_LagM1Revenue
, lag(coalesce(M_CompletedPurchases, 0), 2, 0) over (partition by dwh_int_buy_prediction_full_mv.D_UserId order
by dwh_int_buy_prediction_full_mv.D_YearMonth asc) M_LagM2CompletedPurchases
, lag(coalesce(M_Revenue, 0), 2, 0) over (partition by dwh_int_buy_prediction_full_mv.D_UserId order by
dwh_int_buy_prediction_full_mv.D_YearMonth asc) M_LagM2Revenue
, lag(coalesce(M_CompletedPurchases, 0), 3, 0) over (partition by dwh_int_buy_prediction_full_mv.D_UserId order
by dwh_int_buy_prediction_full_mv.D_YearMonth asc) M_LagM3CompletedPurchases
, lag(coalesce(M_Revenue, 0), 3, 0) over (partition by dwh_int_buy_prediction_full_mv.D_UserId order by
dwh_int_buy_prediction_full_mv.D_YearMonth asc) M_LagM3Revenue
, lag(coalesce(M_CompletedPurchases, 0), 4, 0) over (partition by dwh_int_buy_prediction_full_mv.D_UserId order
by dwh_int_buy_prediction_full_mv.D_YearMonth asc) M_LagM4CompletedPurchases
, lag(coalesce(M_Revenue, 0), 4, 0) over (partition by dwh_int_buy_prediction_full_mv.D_UserId order by
dwh_int_buy_prediction_full_mv.D_YearMonth asc) M_LagM4Revenue
from dwh_int_buy_prediction_full_mv
left join dwh_int_buy_prediction_full_quarters on 1=1
        and dwh_int_buy_prediction_full_quarters.D_Quarter = date_trunc('quarter',
dwh_int_buy_prediction_full_mv.D_YearMonth)
```

```
        and dwh_int_buy_prediction_full_quarters.D_UserId = dwh_int_buy_prediction_full_mv.D_UserId
having 1=1
        and dwh_int_buy_prediction_full_mv.D_CohortYearMonth <>
                dwh_int_buy_prediction_full_mv.D_YearMonth -- here i've excluded the users first buy, since it
                        -- doesn't have an history
--  use the following filter to extract the algorithm sample
--          and dwh_int_buy_prediction_full_mv.D_YearMonth <> date_trunc('month', current_date)
--  use the following filter to extract the current month data
--          and dwh_int_buy_prediction_full_mv.D_YearMonth = date_trunc('month', current_date)
```

APÊNDICE D

SQL View: dwh_int_clerk_clients_data

```
with q1 as (
	select distinct
	  db_omni_store_orders_by_user_id_create.userid as D_UserId
	, db_omni_elementar_users_full.name as D_Name
	, db_omni_elementar_users_full.phone as D_Phone
	, db_omni_elementar_users_full.email as D_Email
	, first_value(db_omni_store_orders_by_user_id_create.distributorid) over (partition by
db_omni_store_orders_by_user_id_create.userid order by
date(date_parse(substr(db_omni_store_orders_by_user_id_create."date",1,10), '%Y-%m-%d')) desc) as D_Store
	, ed_buy_prediction_full_output.m_logregscore as M_LogRegScore
	from db_omni_store_orders_by_user_id_create
	join ed_buy_prediction_full_output on 1=1
		and db_omni_store_orders_by_user_id_create.userid = ed_buy_prediction_full_output.d_userid
	join db_omni_elementar_users_full on 1=1
		and db_omni_store_orders_by_user_id_create.userid = db_omni_elementar_users_full.userid
	where 1=1
		and db_omni_elementar_users_full.userid is not NULL
		and date(date_parse(substr("date",1,10), '%Y-%m-%d')) >= date '2017-01-01'
)
select
  q1.D_UserId
, q1.D_Name
, q1.D_Phone
, q1.D_Email
, coalesce(ed_store_reatribution.new_store, q1.D_Store) D_Store
, q1.M_LogRegScore
from q1
left join ed_store_reatribution on 1=1
	and q1.D_Store = ed_store_reatribution.old_store
```

SQL View: dwh_int_clerk_contact_ranking_by_user

```
-- selecting stores that are in the system
with stores AS (
	select
	  cast(id as integer) as property_id
	, json_extract_scalar(json_parse(brand_assets), '$.store_id') Empresa de Varejo_property_id
	, name as property_name
	from elementar_imoveis.house_db_property
	where element_at(split(name,' - '),1) in ('Marca A', 'Marca B')
)
-- user with pending contacts
, users_pending as (
	select distinct
	  house_db_contact.assignee_id
	from elementar_imoveis.house_db_contact
	left join elementar_imoveis.house_db_contact_status on house_db_contact_status.id =
house_db_contact.contact_status_id
	where 1=1
		and house_db_contact_status.name = 'pending'
		and house_db_contact.assignee_id is not null
		and requester_id <> assignee_id
```

```
)
-- i'm numbering users from 1 to total user in store, so that i can match it with the clients
, indexing_users as (
select
    ed_clerk_users_mapping.*
  , row_number() over (partition by property_id_clerk) user_index
from ed_clerk_users_mapping
left join users_pending on 1=1
  and cast(user_id_clerk as integer) = cast(assignee_id as integer)
where 1=1
  and attribute_contacts = '1' -- filtering users that are supposed to receive messages
  and assignee_id is null -- removing users that have pending tasks
)
-- calculating number of users per store
, users_per_store as (
            select
              cast(property_id_clerk as integer) property_id_clerk
            , cast(max(user_index) as double) users
            from indexing_users
            group by 1
)
-- selecting creating a lead quarantine
, phone_quarantine as (
            select
              substr(house_db_person.phone, -8, 8) D_Phone
            , date(date_format(date_add('hour', -3, from_unixtime((house_db_contact.scheduled_date / 1000000))),
'%Y-%m-%d')) D_ContactDate
            from elementar_imoveis.house_db_lead
            join stores on 1=1
                        and house_db_lead.property_id = stores.property_id
            join elementar_imoveis.house_db_contact on 1=1
                        and house_db_lead.id = house_db_contact.lead_id
            join elementar_imoveis.house_db_person on 1=1
                        and house_db_lead.person_id = house_db_person.id
            where 1=1
                        and house_db_contact.scheduled_date is not null
                        and date_add('day',-30,current_date) < date(date_format(date_add('hour', -3,
from_unixtime((house_db_contact.scheduled_date / 1000000))), '%Y-%m-%d'))
)
, optouts AS (
            select
              substr(house_db_person.phone,-8,8) D_Phone
            from elementar_imoveis.house_db_person
            join elementar_imoveis.house_db_lead on 1=1
                        and house_db_person.id = house_db_lead.person_id
            where lead_status_id in (2, 3)
)
-- creating a user/lead index to assignee old leads to the same user
, user_lead_index as (
            select
              substr(house_db_person.phone, -8, 8) phone
            , cast(house_db_contact.assignee_id as integer) assignee_id
            , cast(house_db_contact.lead_id as integer) lead_id
            , coalesce(indexing_users.user_index, -1) user_index
            , cast(indexing_users.property_id_clerk as integer) property_id_clerk
            , 'old lead' lead_type
```

```
	from elementar_imoveis.house_db_contact
	join elementar_imoveis.house_db_lead on 1=1
		and house_db_contact.lead_id = house_db_lead.id
	join elementar_imoveis.house_db_person on 1=1
		and house_db_lead.person_id = house_db_person.id
	join stores on 1=1
		and house_db_lead.property_id = stores.property_id
	left join indexing_users on 1=1
		and house_db_contact.assignee_id = cast(indexing_users.user_id_clerk as integer)
)
-- this subquery assign clients to users that had already contacted them, if is a new lead assigned randomly
, indexing_clients as (
	select
	  dwh_int_clerk_clients_data.*
	, coalesce(user_lead_index.property_id_clerk, stores.property_id) as property_id
	, coalesce(user_lead_index.user_index, ceiling(users_per_store.users*rand())) user_index
--	, users_per_store.users
	, coalesce(user_lead_index.lead_type, 'new lead') lead_type
	, user_lead_index.lead_id
	from dwh_int_clerk_clients_data
	join stores on 1=1
		and trim(dwh_int_clerk_clients_data.D_Store) = trim(stores.Empresa de Varejo_property_id)
	left join user_lead_index on 1=1
		and substr(replace(replace(dwh_int_clerk_clients_data.D_Phone ,' '), '+'), -8,8) =
user_lead_index.phone
	join users_per_store on 1=1
		and stores.property_id = users_per_store.property_id_clerk
	left join phone_quarantine on 1=1
		and substr(replace(replace(dwh_int_clerk_clients_data.D_Phone ,' '), '+'), -8,8) =
phone_quarantine.D_Phone
	left join ed_phone_clean on 1=1
		and substr(replace(replace(dwh_int_clerk_clients_data.D_Phone ,' '), '+'), -8,8) =
substr(ed_phone_clean.original_phone, -8, 8)
	left join optouts on 1=1
		and substr(replace(replace(dwh_int_clerk_clients_data.D_Phone ,' '), '+'), -8,8) = optouts.D_Phone
	where 1=1
		and dwh_int_clerk_clients_data.D_Phone is not null
		and phone_quarantine.D_Phone is null
		and ed_phone_clean.original_phone is null
		and optouts.D_Phone is null
)
select
  indexing_clients.property_id
, stores.property_name
, indexing_clients.D_Name as person_name
, indexing_clients.D_Email as person_email
, replace(replace(indexing_clients.D_Phone,'+55'),' ') as person_phone
, indexing_users.user_id_clerk as assignee_id
, indexing_clients.lead_id
, row_number() over (partition by indexing_clients.property_id, indexing_clients.user_index order by
M_LogRegScore desc) as client_ranking
, M_LogRegScore
, indexing_users.marca as brand
, indexing_users.nome as user_name
, indexing_users.link_encurtado
, indexing_users.cupom
```

```
, indexing_clients.lead_type
from indexing_clients
join stores on 1=1
        and indexing_clients.property_id = stores.property_id
join indexing_users on 1=1
        and indexing_clients.property_id = cast(indexing_users.property_id_clerk as integer)
        and indexing_clients.user_index = cast(indexing_users.user_index as integer)
where 1=1
        and indexing_clients.user_index <> -1
        and lead_type = 'new lead'
```

SQL View: dwh_exp_clerk_contacts

```
select
   lead_id
  , property_id
  , person_phone
  , person_name
  , person_email
  , 'Empresa de Varejo' as lead_source
  , cast(null as varchar) lead_additional_info
  , assignee_id
  , date_format(
      date_add('day', 1, current_timestamp at time zone 'America/Sao_Paulo'),
    '%Y-%m-%d 18:00:00') as lead_contact_scheduled_date
  , 5 as lead_contact_reason_id
  , replace(replace(replace(replace(mensagem
          , '[user_short_name]', element_at(split(user_name,' '),1))
          , '[link]', link_encurtado)
          , '[cupom]', cupom)
          , '[client_short_name]',
element_at(split(upper(substr(coalesce(person_name,''),1,1))||lower(substr(coalesce(person_name,''),2,100)),' '),1))
  as lead_contact_suggested_message
from dwh_int_clerk_contact_ranking_by_user
left join ed_clerk_message_mapping on 1=1
  and trim(lower(dwh_int_clerk_contact_ranking_by_user.brand)) = trim(lower(ed_clerk_message_mapping.marca))
  and dwh_int_clerk_contact_ranking_by_user.lead_type = ed_clerk_message_mapping.lead_type
where 1=1
  and client_ranking <= 20 -- selecting 20 clients for each user
```

APÊNDICE E

SQL View: dwh_int_clerk_store_contacts

```
select
	D_StoreID
	, D_StoreName
	, D_ScheduledDate
	, coalesce(cast(D_AssigneeID as varchar), '(not set)') D_AssigneeID
   , coalesce(ed_clerk_users_mapping.nome, '(not set)') D_AssigneeName
	, sum (M_NumberOfScheduledContacts) m_ClientsQuantity
	, sum(case
			when D_TemWhatsApp = '0. Não tem WhatsApp'
			then M_NumberOfScheduledContacts
			end) m_NoWpp
	, sum(case
			when D_TemWhatsApp = '1. Tem WhatsApp'
			then M_NumberOfScheduledContacts
			end) m_YesWpp
	, sum(case
			when D_StatusDoContato = '0. Contato excluído' and D_TemWhatsApp = '0. Não tem
WhatsApp'
			then M_NumberOfScheduledContacts
			end) m_ExcludedNoWpp
	, sum(case
			when D_StatusDoContato = '0. Contato excluído' and D_TemWhatsApp = '1. Tem
WhatsApp'
			then M_NumberOfScheduledContacts
			end) m_ExcludedYesWpp
	, sum(case
			when D_StatusDoContato = '1. Envio pendente'
			then M_NumberOfScheduledContacts
			end) m_Pending
	, sum(case
			when D_StatusDoContato = '1. Envio pendente' and D_TemWhatsApp = '0. Não tem
WhatsApp'
			then M_NumberOfScheduledContacts
			end) m_PendingNoWpp
	, sum(case
			when D_StatusDoContato = '1. Envio pendente' and D_Atrasado = '0. Atrasado'
			then M_NumberOfScheduledContacts
			end) m_LatePending
	, sum(case
			when D_StatusDoContato = '1. Envio pendente' and D_Atrasado = '0. Atrasado' and
D_TemWhatsApp = '0. Não tem WhatsApp'
			then M_NumberOfScheduledContacts
			end) m_LatePendingNoWpp
	, sum(case
			when D_StatusDoContato = '2. Enviado, falta dar o feedback' or D_StatusDoContato = '3.
Finalizado, enviado e com fedback'
			then M_NumberOfScheduledContacts
			end) m_ContactsRealized
	, sum(case
			when D_StatusDoContato = '2. Enviado, falta dar o feedback'
			then M_NumberOfScheduledContacts
			end) m_NoFeedback
```

```
        , sum(case
                when D_StatusDoContato = '2. Enviado, falta dar o feedback' and D_Atrasado = '0.
Atrasado'
                then M_NumberOfScheduledContacts
                end) m_LateNoFeedback
        , sum(case
                when D_StatusDoContato = '3. Finalizado, enviado e com fedback'
                then M_NumberOfScheduledContacts
                end) m_YesFeedback
        , sum(case
                when D_FeedbackDoCliente = '3. Cliente respondeu'
                then M_NumberOfScheduledContacts
                end) m_AnswersClients
        , sum(case
                when D_FeedbackDoCliente = '2. Cliente não respondeu'
                then M_NumberOfScheduledContacts
                end) m_NoAnswersClients
        , sum(case
                when D_FeedbackDoCliente = '1. Cliente pediu para não ser mais contatado'
                then M_NumberOfScheduledContacts
                end) m_Optout
from dwh_int_clerk_contacts
left join ed_clerk_users_mapping on 1=1
        and cast(D_AssigneeID as varchar) = ed_clerk_users_mapping.user_id_clerk
group by 1, 2, 3, 4, 5
```

SQL View: dwh_int_clerk_traffic

```
select
  house_db_property.name as D_StoreName
, element_at(split(D_Campaign,'-'),1) D_StoreId
, coalesce(element_at(split(D_Campaign,'-'),2), '(not set)') D_AssigneeID
, coalesce(ed_clerk_users_mapping.nome, '(not set)') D_AssigneeName
, D_Date
, sum(M_NewUsers) M_NewUsers
, sum(M_Users) M_Users
, sum(M_Sessions) M_Sessions
, sum(M_PageViews) M_PageViews
, sum(M_UniquePageViews) M_UniquePageViews
, sum(M_Transactions) M_Transactions
, sum(M_TransactionRevenue) M_TransactionRevenue
from dwh_int_google_analytics
left join elementar_imoveis.house_db_property on 1=1
        and cast(house_db_property.id as varchar) = element_at(split(D_Campaign,'-'),1)
left join ed_clerk_users_mapping on 1=1
        and element_at(split(D_Campaign,'-'),2) = ed_clerk_users_mapping.user_id_clerk
where 1=1
        and d_source = 'crm'
        and d_medium = 'whatsapp'
group by 1, 2, 3, 4, 5
```

SQL View: dwh_int_clerk_revenue

```
-- calculating leads first contact
with selected_stores as (
        select
          id as property_id
```

```
        , name as property_name
        from elementar_imoveis.house_db_property
        where element_at(split(name,' - '),1) in ('Marca A', 'Marca B')
)
, selected_leads as (
        select
          house_db_lead.id as lead_id
        , house_db_lead.person_id
        , selected_stores.property_id
        , selected_stores.property_name
        from elementar_imoveis.house_db_lead
        join selected_stores on 1=1
                and house_db_lead.property_id = selected_stores.property_id
)
, lead_contact_dates as (
        SELECT distinct
          selected_leads.lead_id
        , house_db_person.phone D_Phone
        , selected_leads.property_id
        , selected_leads.property_name
        , house_db_contact.assignee_id
        , date(date_format(date_add('hour', -3, from_unixtime((house_db_contact.scheduled_date / 1000000))),
'%Y-%m-%d')) D_ContactDate
        from selected_leads
        left join elementar_imoveis.house_db_contact on 1=1
                and selected_leads.lead_id = house_db_contact.lead_id
        left join elementar_imoveis.house_db_person on 1=1
                and selected_leads.person_id = house_db_person.id
        left join ed_phone_clean on 1=1
                and substr(house_db_person.phone,-8,8) = substr(ed_phone_clean.original_phone, -8, 8)
        where 1=1
                and house_db_contact.scheduled_date is not NULL
                and ed_phone_clean.original_phone is null
)

-- crossing phone numbers with Empresa de Varejo user id
, userid_contact_dates as (
        select distinct
          db_omni_elementar_users_full.userid as D_UserId
        , D_Phone
        , property_id as D_StoreId
        , property_name as D_StoresName
        , D_ContactDate
        , lead_contact_dates.assignee_id
        from lead_contact_dates
        join db_omni_elementar_users_full on 1=1
                and substr(replace(lead_contact_dates.D_Phone,' ',''),-8,8) =
substr(replace(db_omni_elementar_users_full.phone,' ',''),-8,8)
        where 1=1
                and db_omni_elementar_users_full.userid is not null
                and db_omni_elementar_users_full.phone is not null
)


-- calculating sales data
, offline_orders as (
```

```
        select distinct
          date(date_parse(substr("date",1,10), '%Y-%m-%d')) as D_Date
        , userid as d_userid
        , D_StoreId
        , D_StoresName
        , userid_contact_dates.assignee_id
        , D_Phone
        , 1 as m_offlinepurchases
        , amount/100 as m_offlinetotal
        from db_omni_store_orders_by_user_id_create
        join userid_contact_dates on 1=1
                and db_omni_store_orders_by_user_id_create.userid = userid_contact_dates.d_userid
                and date(date_parse(substr("date",1,10), '%Y-%m-%d')) >= D_ContactDate
                and date(date_parse(substr("date",1,10), '%Y-%m-%d')) < date_add('day',30,D_ContactDate)
        where 1=1
                and date(date_parse(substr("date",1,10), '%Y-%m-%d')) >= date '2020-02-01'
--      group by 1, 2, 3, 4, 5, 6
)

, cupom_online_sales as (
        select distinct
          D_OrderCreationDate as D_Date
        , D_DisplayCode
        , dwh_int_omni_sales_order.d_userid
        , property_id_clerk as D_StoreId
        , cupom
--      , date(date_format(date_add('hour', -3, from_unixtime((house_db_user.created_at / 1000000))), '%Y-%m-
%d')) as D_ContactDate
        , ed_clerk_users_mapping.user_id_clerk as assignee_id
        , m_total as m_salescaptured
        , case when D_Status = 'Faturado' then m_total else 0 end as m_salesinvoiced
        from dwh_int_omni_sales_order
        left join db_omni_elementar_view_orders_bags_full on 1=1
                and dwh_int_omni_sales_order.D_DisplayCode =
db_omni_elementar_view_orders_bags_full.displaycode
        left join ed_clerk_users_mapping on 1=1
                and coupon = cupom
        left join elementar_imoveis.house_db_user on 1=1
                and cast(house_db_user.id as varchar) = ed_clerk_users_mapping.user_id_clerk
        where 1=1
                and cupom is not null
                and D_OrderCreationDate >= date(date_format(date_add('hour', -3,
from_unixtime((house_db_user.created_at / 1000000))), '%Y-%m-%d'))
)
, phone_online_sales as (
        select distinct
          D_OrderCreationDate as D_Date
        , D_DisplayCode
        , dwh_int_omni_sales_order.d_userid
        , first_value(cast(D_StoreId as varchar)) over (partition by D_DisplayCode order by D_ContactDate asc)
D_StoreId
        , first_value(cast(userid_contact_dates.assignee_id as varchar)) over (partition by D_DisplayCode order by
D_ContactDate asc) assignee_id
--      , substr(D_Phone, -8, 8) as D_Phone
--      , D_ContactDate
        , m_total as m_salescaptured
```

```
        , case when D_Status = 'Faturado' then m_total else 0 end as m_salesinvoiced
        from dwh_int_omni_sales_order
        join userid_contact_dates on 1=1
                and userid_contact_dates.d_userid = dwh_int_omni_sales_order.d_userid
        where 1=1
                and userid_contact_dates.d_userid is not null
                and D_OrderCreationDate >= D_ContactDate
                and D_OrderCreationDate < date_add('day',30,D_ContactDate)
)

, online_orders as (
        select
          coalesce(cupom_online_sales.D_Date, phone_online_sales.D_Date) D_Date
        , coalesce(cupom_online_sales.D_DisplayCode, phone_online_sales.D_DisplayCode) D_DisplayCode
        , coalesce(cupom_online_sales.d_userid, phone_online_sales.d_userid) d_userid
        , coalesce(cupom_online_sales.D_StoreId, phone_online_sales.D_StoreId) D_StoreId
        , coalesce(cupom_online_sales.assignee_id, phone_online_sales.assignee_id) assignee_id
        , 1 as m_numberoforderscaptured
        , case when coalesce(cupom_online_sales.m_salesinvoiced, phone_online_sales.m_salesinvoiced)>0 then 1
else 0 end as m_numberofordersinvoiced
        , coalesce(cupom_online_sales.m_salescaptured, phone_online_sales.m_salescaptured) m_salescaptured
        , coalesce(cupom_online_sales.m_salesinvoiced, phone_online_sales.m_salesinvoiced) m_salesinvoiced
        from cupom_online_sales
        full join phone_online_sales on cupom_online_sales.D_DisplayCode = phone_online_sales.D_DisplayCode
)
, q1 as (
        select
          D_Date
        , cast(D_StoreId as varchar) D_StoreId
        , cast(assignee_id as varchar) D_AssigneeId
        , d_userid
        , coalesce(m_offlinepurchases,0) M_NumberOfOrdersCaptured
        , coalesce(m_offlinepurchases,0) M_NumberOfOrdersInvoiced
        , coalesce(m_offlinetotal,0) M_SalesCaptured
        , coalesce(m_offlinetotal,0) M_SalesInvoiced
        from offline_orders

        union all select
          D_Date
        , cast(D_StoreId as varchar) D_StoreId
        , cast(assignee_id as varchar) D_AssigneeId
        , d_userid
        , coalesce(m_numberoforderscaptured,0) M_NumberOfOrdersCaptured
        , coalesce(m_numberofordersinvoiced,0) M_NumberOfOrdersInvoiced
        , coalesce(m_salescaptured,0) M_SalesCaptured
        , coalesce(m_salesinvoiced,0) M_SalesInvoiced
        from online_orders
)

select
  q1.D_Date
, q1.D_StoreId
, selected_stores.property_name D_StoreName
, q1.D_AssigneeId
, coalesce(ed_clerk_users_mapping.nome, '(not set)') D_AssigneeName
, count(distinct q1.d_userid) M_Buyers
```

```
, sum(M_NumberOfOrdersCaptured) M_NumberOfOrdersCaptured
, sum(M_NumberOfOrdersInvoiced) M_NumberOfOrdersInvoiced
, sum(M_SalesCaptured) M_SalesCaptured
, sum(M_SalesInvoiced) M_SalesInvoiced
from q1
left join selected_stores on 1=1
        and cast(selected_stores.property_id as varchar) = q1.D_StoreId
left join ed_clerk_users_mapping on 1=1
        and q1.D_AssigneeId = cast(ed_clerk_users_mapping.user_id_clerk as varchar)
group by 1, 2, 3, 4, 5
```

SQL View: dwh_int_clerk_revenue_with_estimate

```
WITH estimate AS (
  SELECT
    D_StoreId
  , D_StoreName
  , SUM(M_SalesInvoiced)/SUM(M_SalesCaptured) M_InvoicedRate
  FROM dwh_int_clerk_revenue
  WHERE d_date BETWEEN
       date_add('day', -104, current_date)
    AND date_add('day', -14, current_date)
  GROUP BY 1, 2
  having SUM(M_SalesInvoiced)/SUM(M_SalesCaptured) > 0.05 -- removing estimate from stores that have a too lo
M_InvoicedRate
)
SELECT
  dwh_int_clerk_revenue.d_date D_Date
, dwh_int_clerk_revenue.D_StoreId D_StoreId
, dwh_int_clerk_revenue.D_StoreName D_StoreName
, dwh_int_clerk_revenue.D_AssigneeId D_AssigneeId
, dwh_int_clerk_revenue.D_AssigneeName D_AssigneeName
, dwh_int_clerk_revenue.M_Buyers M_Buyers
, dwh_int_clerk_revenue.M_NumberOfOrdersCaptured M_NumberOfOrdersCaptured
, dwh_int_clerk_revenue.M_NumberOfOrdersInvoiced M_NumberOfOrdersInvoiced
, CASE
    WHEN dwh_int_clerk_revenue.d_date < date_add('day', -14, current_date)
      OR estimate.m_invoicedrate IS NULL
        THEN dwh_int_clerk_revenue.M_NumberOfOrdersInvoiced
    ELSE dwh_int_clerk_revenue.M_NumberOfOrdersCaptured * estimate.m_invoicedrate
  END M_NumberOfOrdersInvoicedEstimate
, dwh_int_clerk_revenue.M_SalesCaptured M_SalesCaptured
, dwh_int_clerk_revenue.M_SalesInvoiced M_SalesInvoiced
, CASE
    WHEN dwh_int_clerk_revenue.d_date < date_add('day', -14, current_date)
      OR estimate.m_invoicedrate IS NULL
        THEN dwh_int_clerk_revenue.M_SalesInvoiced
    ELSE dwh_int_clerk_revenue.M_SalesCaptured * estimate.m_invoicedrate
  END M_SalesInvoicedEstimate
FROM dwh_int_clerk_revenue
LEFT JOIN estimate ON 1=1
  AND estimate.D_StoreId = dwh_int_clerk_revenue.D_StoreId
```

SQL View: dwh_bus_clerk_dashboard

```
-- contacts data
SELECT
```

```
  dwh_int_clerk_store_contacts.D_StoreName D_StoreName
, cast(dwh_int_clerk_store_contacts.D_StoreId as integer) D_StoreId
, element_at(split(D_StoreName, ' - '), 1) D_ClientBrand
, dwh_int_clerk_store_contacts.D_ScheduledDate D_Date
, date_format(D_ScheduledDate, '%W') D_WeekDay
, dwh_int_clerk_store_contacts.D_AssigneeID D_AssigneeID
, dwh_int_clerk_store_contacts.D_AssigneeName D_AssigneeName
, COALESCE(dwh_int_clerk_store_contacts.M_ClientsQuantity, 0) M_ClientsQuantity
, COALESCE(dwh_int_clerk_store_contacts.M_NoWpp, 0) M_NoWpp
, COALESCE(dwh_int_clerk_store_contacts.M_YesWpp, 0) M_YesWpp
, COALESCE(dwh_int_clerk_store_contacts.M_ExcludedNoWpp, 0) M_ExcludedNoWpp
, COALESCE(dwh_int_clerk_store_contacts.M_ExcludedYesWpp, 0) M_ExcludedYesWpp
, COALESCE(dwh_int_clerk_store_contacts.M_Pending, 0) M_Pending
, COALESCE(dwh_int_clerk_store_contacts.M_LatePending, 0) M_LatePending
, COALESCE(dwh_int_clerk_store_contacts.M_LatePendingNoWpp, 0) M_LatePendingNoWpp
, COALESCE(dwh_int_clerk_store_contacts.M_NoFeedback, 0) M_NoFeedback
, COALESCE(dwh_int_clerk_store_contacts.M_YesFeedback, 0) M_YesFeedback
, COALESCE(dwh_int_clerk_store_contacts.M_AnswersClients, 0) M_AnswersClients
, COALESCE(dwh_int_clerk_store_contacts.M_Optout, 0) M_Optout
, COALESCE(dwh_int_clerk_store_contacts.m_ContactsRealized, 0) M_ContactsRealized
, COALESCE(dwh_int_clerk_store_contacts.m_LateNoFeedback, 0) M_LateNoFeedback
, COALESCE(dwh_int_clerk_store_contacts.m_NoAnswersClients, 0) M_NoAnswersClients
, COALESCE(dwh_int_clerk_store_contacts.m_PendingNoWpp, 0) M_PendingNoWpp
, 0 M_NewUsers
, 0 M_Users
, 0 M_Sessions
, 0 M_PageViews
, 0 M_UniquePageViews
, 0 M_GATransactions
, 0 M_GATransactionRevenue
, 0 M_NumberOfOrdersCaptured
, 0 M_NumberOfOrdersInvoiced
, 0 M_NumberOfOrdersInvoicedEstimate
, 0 M_SalesCaptured
, 0 M_SalesInvoiced
, 0 M_SalesInvoicedEstimate
FROM dwh_int_clerk_store_contacts
-- traffic data
UNION ALL
SELECT
  dwh_int_clerk_traffic.D_StoreName D_StoreName
, cast(D_StoreId as integer) D_StoreId
, element_at(split(D_StoreName, ' - '), 1) D_ClientBrand
, dwh_int_clerk_traffic.D_Date D_Date
, date_format(D_Date, '%W') D_WeekDay
, dwh_int_clerk_traffic.D_AssigneeID D_AssigneeID
, dwh_int_clerk_traffic.D_AssigneeName D_AssigneeName
, 0 M_ClientsQuantity
, 0 M_NoWpp
, 0 M_YesWpp
, 0 M_ExcludedNoWpp
, 0 M_ExcludedYesWpp
, 0 M_Pending
, 0 M_LatePending
, 0 M_LatePendingNoWpp
, 0 M_NoFeedback
```

```
, 0 M_YesFeedback
, 0 M_AnswersClients
, 0 M_Optout
, 0 M_ContactsRealized
, 0 M_LateNoFeedback
, 0 m_NoAnswersClients
, 0 M_PendingNoWpp
, COALESCE(dwh_int_clerk_traffic.M_NewUsers, 0) M_NewUsers
, COALESCE(dwh_int_clerk_traffic.M_Users, 0) M_Users
, COALESCE(dwh_int_clerk_traffic.M_Sessions, 0) M_Sessions
, COALESCE(dwh_int_clerk_traffic.M_PageViews, 0) M_PageViews
, COALESCE(dwh_int_clerk_traffic.M_UniquePageViews, 0) M_UniquePageViews
, COALESCE(dwh_int_clerk_traffic.M_Transactions, 0) M_GATransactions
, COALESCE(dwh_int_clerk_traffic.M_TransactionRevenue, 0) M_GATransactionRevenue
, 0 M_NumberOfOrdersCaptured
, 0 M_NumberOfOrdersInvoiced
, 0 M_NumberOfOrdersInvoicedEstimate
, 0 M_SalesCaptured
, 0 M_SalesInvoiced
, 0 M_SalesInvoicedEstimate
FROM dwh_int_clerk_traffic
-- revenue data
UNION ALL
SELECT
  dwh_int_clerk_revenue_with_estimate.D_StoreName D_StoreName
, cast(D_StoreId as integer) D_StoreId
, element_at(split(D_StoreName,' - '), 1) D_ClientBrand
, dwh_int_clerk_revenue_with_estimate.D_Date D_Date
, date_format(D_Date, '%W') D_WeekDay
, dwh_int_clerk_revenue_with_estimate.D_AssigneeID D_AssigneeID
, dwh_int_clerk_revenue_with_estimate.D_AssigneeName D_AssigneeName
, 0 M_ClientsQuantity
, 0 M_NoWpp
, 0 M_YesWpp
, 0 M_ExcludedNoWpp
, 0 M_ExcludedYesWpp
, 0 M_Pending
, 0 M_LatePending
, 0 M_LatePendingNoWpp
, 0 M_NoFeedback
, 0 M_YesFeedback
, 0 M_AnswersClients
, 0 M_Optout
, 0 M_ContactsRealized
, 0 M_LateNoFeedback
, 0 m_NoAnswersClients
, 0 M_PendingNoWpp
, 0 M_NewUsers
, 0 M_Users
, 0 M_Sessions
, 0 M_PageViews
, 0 M_UniquePageViews
, 0 M_GATransactions
, 0 M_GATransactionRevenue
, COALESCE(M_NumberOfOrdersCaptured, 0) M_NumberOfOrdersCaptured
, COALESCE(M_NumberOfOrdersInvoiced, 0) M_NumberOfOrdersInvoiced
```

```
, COALESCE(M_NumberOfOrdersInvoicedEstimate, 0) M_NumberOfOrdersInvoicedEstimate
, COALESCE(dwh_int_clerk_revenue_with_estimate.M_SalesCaptured, 0) M_SalesCaptured
, COALESCE(dwh_int_clerk_revenue_with_estimate.M_SalesInvoiced, 0) M_SalesInvoiced
, COALESCE(dwh_int_clerk_revenue_with_estimate.M_SalesInvoicedEstimate, 0) M_SalesInvoicedEstimate
FROM dwh_int_clerk_revenue_with_estimate
```

SQL View: dwh_exp_clerk_dashboard

```
SELECT
  *
FROM dwh_bus_clerk_dashboard
WHERE 1=1
        AND D_Date IS NOT NULL
        AND D_Date >= date '2020-03-01'
```