hefEStos
Rede Gazeta

inteli
instituto
de tecnologia
e liderança

Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
08/08/2022	Marcos, Priscila, Maria Luísa, Matheus e Henrique	1.1	Criação do documento Edição do tópico 4.1.1 Edição do tópico 4.1.4 Edição do tópico 4.1.5
09/08/2022	Pedro Priscila Marcos	1.2	Edição do tópico 4.1.2 Edição do tópico 4.1.3 Inserção de dados nos subtópicos do tópico 4
10/08/2022	Maria Luísa Marcos Pedro	1.3	Revisão e conclusão dos tópicos do artefato 1 (4.1.1, 4.1.2, 4.1.3, 4.1.4 e 4.1.5)
15/08/2022	Maria Luisa Pedro Rafael Henrique Marcos Matheus	2.1	Fazer 4.2 - análise de dados
17/08/2022	Pedro Priscila Matheus	2.2	Persona Jornada de usuário
20/08/2022	Priscila	2.3	Formatação da documentação 4.1.2 - Descrição da matriz SWOT
24/08/2022	Maria Luisa	2.4	Revisão dos tópicos dos artefatos da sprint 1
25/08/2022	Maria Luisa Henrique Matheus	2.5	Formatação + passar para outro documento Realização do tópico 4.3
26/08/2022	Maria Luisa Priscila Falcão	2.6	Revisão de formatação, ortografia e conteúdo Tópicos 4.3.2 e 4.3.3

29/08/2022	Priscila Falcão	3.1	Texto de personas Texto de jornada de usuário Texto de matriz de risco
30/08/2022	Maria Luisa Priscila Falcão	3.2	Revisão análise SWOT CRISP-DM
07/09/2022	Priscila Falcão	3.3	CRISP-DM Formatação código
08/09/2022	Matheus Marcos	3.4	Tópicos 4.4 e 4.5
09/09/2022	Rafael Matheus Marcos	3.5	Revisão da parte de Metodologia Tópicos 4.4 e 4.5
10/09/2022	Matheus Marcos Pedro Henrique	3.6	Tópico 4.3 - Anonimização Tópicos 4.4 e 4.5 Revisão
12/09/2022	Maria Luisa Priscila Falcão Henrique Rafael	4.1	Remoção dos termos “data warehouse” Alteração da explicação da escolha de features Detalhamento de persona Formatação Revisão Tópico 3.3
13/09/2022	Maria Luísa Priscila	4.2	Revisão e formatação
20/09/2022	Priscila Pedro Marcos Matheus	4.3	Texto - proposta de valor Tópico 4.2.7 Tópico 4.3.1 e 4.3.3
21/09/2022	Priscila Henrique Matheus Marcos	4.4	Tópicos: 4.4 4.5
23/09/2022	Maria Luísa	4.5	Revisão e formatação

03/10/2022	Marcos Pedro	5.1	Primeira revisão final Tópico 4.5
05/10/2022	Maria Luísa	5.2	Segunda revisão final Formatação final Tópico 5 e 6
06/10/2022	Priscila Falcão Marcos Moura	5.3	Ajustes finais

Sumário

1. Introdução

A **Rede Gazeta de Comunicações**, ou **Rede Gazeta**, é um conjunto de mídia brasileiro localizado no estado do Espírito Santo. Possuindo mais de 500 funcionários, a emissora em questão é o maior grupo de comunicação do estado, que foi fundada em 1928, com o jornal A Gazeta, porém apenas no ano de 1976 a TV Gazeta surgiu, aproveitando o grande crescimento dos meios de comunicação em massa. Atualmente, eles contam com a presença de 16 veículos de comunicação, abrangendo a TV, o rádio e a internet. Contudo, nos últimos anos, viu-se necessária a criação de um meio para melhorar a média de audiência dessa emissora

Nos últimos anos a empresa cresceu bastante pelas suas produções próprias, porém com a grande competitividade das plataformas de streaming e das redes sociais, é de plena importância a constante pesquisa e análise de dados para suas próximas empreitadas. Por isso, o grupo **hefEStos** realizou a criação de um software que através da análise de dados e a partir da inteligência artificial realiza um modelo preditivo que ajudará na previsão da audiência de novos programas.

2. Objetivos e Justificativa

2.1. Objetivos

A Rede Gazeta procurou o Inteli para fazer esse projeto com os alunos, como o grupo **hefEStos**, para poder analisar e prever quais programas de TV são potenciais produtos para um investimento e maior expectativa. A empresa propôs a construção de um software que conta com machine learning para se obter uma previsão de audiência para programas que já existem e que venham a ser lançados. O modelo preditivo entregue, deve receber alguns dados de entrada, sendo eles: data, confirmação se é feriado ou não, intervalo de horário e segmento do programa, com isso, o software entregará uma previsão para o tipo de audiência, e tamanho da audiência medido em Rat% e Shr%.

2.2. Justificativa

O modelo preditivo em questão, entrega previsões se um programa específico terá ou não uma audiência adequada para tal (número de telespectadores), e também qual será o gênero, faixa etária e/ou a idade e o público alvo daquele programa piloto. Fazendo com que o cliente consiga otimizar seus gastos com programas, melhorar seu modelo preditivo e o modo como analisa seus dados .

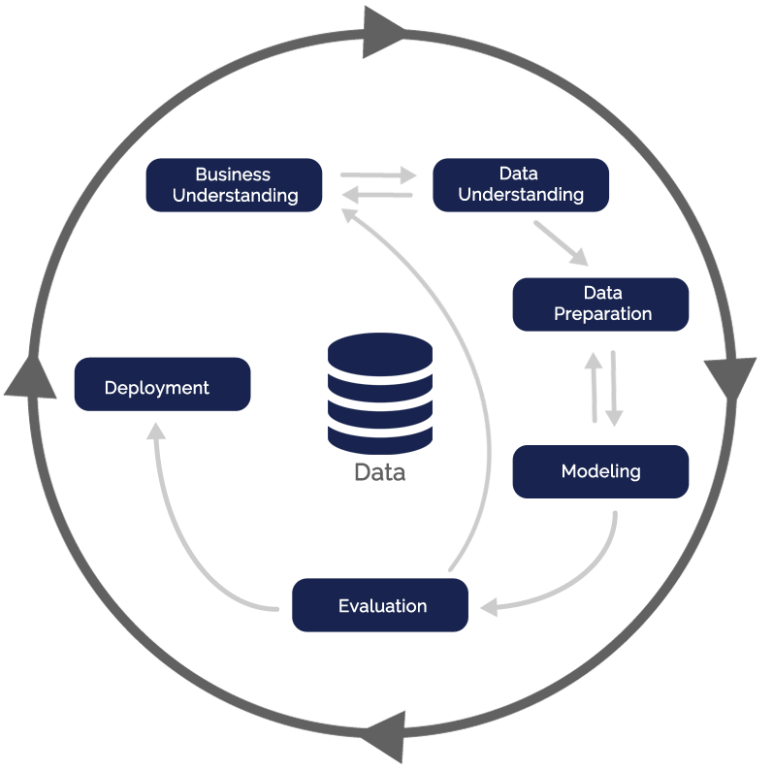
3. Metodologia

3.1. CRISP-DM

A **metodologia CRISP-DM** se trata de um método de mineração de dados, o qual é descrito em termos de um modelo de processo hierárquico, consistindo em *sets* de tarefas, baseados em um conjunto de boas práticas para aplicação em Ciência de dados.

Dividida em alguns passos, a metodologia exige o entendimento do negócio (Business Understanding), entendimento dos dados (Data Understanding), a preparação (Data Preparation) e modelagem dos dados (Modeling), além da avaliação do modelo (Evaluation) e o deployment, conforme o **Fluxograma 1.** Tendo em vista que esses se distribuem nos 4 níveis de abstração: fases, tarefas genéricas, tarefas especializadas e instância de processo.

Fluxograma 1 - Metodologia CRISP-DM.



Fonte: MAGRATHEA (2018).

Tomando a primeira parte da metodologia, para o entendimento do negócio (Business Understanding) houve entrevista com o parceiro, além da apresentação do onboarding, na qual foi possível captar informações sobre a esfera em que o projeto se desenvolveria. Dessa forma, obteve-se o entendimento do funcionamento da análise de dados dentro da emissora, somado a forma de como esses eram extraídos e utilizados, bem como, a identificação de prováveis vieses que poderiam existir nos dados.

A partir do primeiro contato, desenvolveu-se a etapa seguinte de entendimento dos dados (Data Understanding). Após o recebimento, os mesmos foram analisados, avaliando o que significavam, onde entravam durante o processo e como seriam explorados. Assim, foi observada a necessidade de mais informações para a predição, haja vista que em primeira análise, só eram disponibilizados os valores de audiência total, audiência de cada perfil do público, alcance e fidelidade, mostrando um hiato no quesito de programação e categorias que os programas poderiam entrar. Dado esse pedido, o parceiro providenciou o necessário para a continuação do projeto. Após algumas semanas, foi pedido que o *dataset* fosse aumentado, disponibilizando dados desde 2015, o que também foi realizado.

Subsequentemente, na fase de preparação dos dados (Data Preparation), onde eles são tratados, corrigidos e anulados em alguns casos de anomalias, de forma que seja mais benéfico para o projeto. Por isso, é possível verificar diversas tarefas como: a mescla de datasets e registros; a seleção de subconjunto de amostra de dados; a agregação de registros; a derivação de novos atributos; a ordenação de dados para o modelo; a remoção ou substituição de valores nulos ou inválidos; o treinamento do modelo, etc. O pré-processamento dos dados do projeto em questão contou com a agregação de features e a seleção das que seriam realmente necessárias para avaliação, não necessitando de limpeza ou exclusão de registros, uma vez que os *datasets* disponibilizados estavam já corretos e limpos.

Na fase da modelagem (Modeling), na qual o objetivo é determinar qual o modelo mais apropriado para o projeto. Baseando-se no tipo de dado disponível para a mineração, o alvo da mineração e alguns dos requisitos específicos do modelo, houve o treinamento e teste de diversos modelos diferentes, buscando pelo melhor índice de acurácia. Usando dos *datasets* tratados e anonimizados, esses foram expostos a cada um dos modelos, obtendo resultados diferenciados.

Para o diagnóstico desses resultados, a metodologia CRISP-DM conta com a fase de avaliação de modelo (Evaluation), que é quando se avalia a resposta obtida, usando de métricas de erro e outros artifícios, a fim de obter dimensões factuais da eficiência do modelo. Tratando do projeto do modelo preditivo para a audiência das emissoras, foram utilizadas especificamente métricas de avaliação de modelos de regressão, como o Erro Quadrático Médio (MSE), o qual penaliza mais erros maiores, já que os erros (diferença entre o valor previsto e o correto) são elevados ao quadrado; a Distância Absoluta Média (MAD), em que se faz a média do erro absoluto de cada previsão; e o R^2, que trata de uma métrica que varia entre -∞ e 1, sendo uma razão que indica o quão bom o modelo está em comparação com um modelo *naive*, que faz a predição com base no valor médio do *target*.

Após as avaliações se passa para o deployment, que trata-se da entrega e apresentação do produto para o stakeholder. É a partir dessa fase que a seguinte, feedback, é permitida. Na última fase, tem-se o retorno do cliente, na qual torna-se possível o aperfeiçoamento do produto final, dando caráter iterativo ao ciclo.

3.2. Ferramentas

- **Colab**: É uma plataforma em nuvem que simula um ambiente de programação com tudo pronto para o programador, no colab é colocado o *core* do projeto (o código do modelo preditivo).

- **Python**: É uma linguagem de programação de compreensão bastante acessível, com uma sintaxe simples e legibilidade clara, além de ter um vasto número de bibliotecas.

- **Pandas**: É uma biblioteca para uso em Python, *open-source* e de uso gratuito (sob uma licença BSD), que fornece ferramentas para análise e manipulação de dados.

- **Matplotlib**: É uma biblioteca para a visualização de dados em Python. Ele apresenta uma *API* orientada a objetos que permite a criação de gráficos em 2D de uma forma simples e com poucos comandos. A ferramenta disponibiliza diversos tipos de gráficos, como em barra, em linha, em pizza, histogramas, entre outras opções.

- **Sklearn (scikit-learn)**: O *scikit-learn* é uma biblioteca da linguagem Python desenvolvida especificamente para aplicação prática de *machine learning*. Esta biblioteca dispõe de ferramentas simples e eficientes para análise preditiva de dados, podendo ser reutilizável em diferentes situações, possui código aberto, ou seja é acessível a todos, e foi construída sobre os pacotes *NumPy*, *SciPy* e *Matplotlib*.

- **Numpy**: é uma biblioteca para a linguagem Python com funções para se trabalhar com computação numérica.

- **Sheets**: É um programa de planilhas incluído como parte do pacote gratuito de Editores de Documentos Google baseado na *Web* oferecido pelo Google, se tratando como uma planilha do *excel* online.

3.3. Principais técnicas empregadas

- No *one hot encoding*, foram transformadas cada categoria em diversas colunas contendo zeros (0) e uns (1), os quais representam os valores booleanos *true* e *false*. Nesse caso, é empregado o valor 0, quando aquela categoria não pertence a linha, e 1 quando pertence. Usamos essa técnica na coluna de categoria dos programas, transformando todas as linhas de categorias em colunas de 1 a 44.

- Foi utilizado o Pandas para retirar as colunas que não eram necessárias, para juntar a tabela com as informações Fid%, Rat% e Shr% com a que tinha os programas. Também foi utilizado para ordenar os valores da tabela pelas datas e tirar médias das colunas para se fazer análises de dados.

- Uma das técnicas empregadas foi a visualização por gráficos das informações, com o uso do *matplotlib*. Na criação de todos os gráficos foi feita a seleção das features que estariam presentes no eixo X e eixo Y, e para isso, é necessário saber como se quer que se retorne essa informação (gráfico de barras, frequência, distribuição normal...).

- Outra técnica utilizada foram as ferramentas do *sklearn* (*scikit-learn*), que foi utilizada para fazer algumas contas matemáticas, e principalmente para fazer e testar os modelos preditivos usados (KNN, Light GBM e Regressão linear).
    - Para a normalização dos dados também foi usado o *Sklearn*, onde todas as colunas de "Fid", "Shr" e "Rat" tiveram seus valores normalizados entre si para ser possível utilizar esses valores no modelo preditivo, atingindo uma maior acurácia.
    - Foi utilizado o *.score*, R^2 e MAE. A utilidade deles foi para avaliar a acuracidade e precisão do modelo, verificando se os dados procedem e se a predição foi efetiva.
    - O R^2 e o *score* é o erro quadrático médio, que representa a raiz quadrada média de todos os erros. Essa é considerada uma das melhores métricas de erro de propósito geral para previsões numéricas, sendo utilizado dentro do modelo preditivo. Esse, foi utilizado para verificar a acurácia de modelos e dá um maior peso aos maiores erros, já que, ao ser calculado, cada erro é elevado ao quadrado individualmente e, após isso, a média desses erros quadráticos é calculada.

- O MAE é o erro médio absoluto é calculado a partir da média dos erros absolutos, ou seja, deve se utilizar o módulo de cada erro para evitar a subestimação, isso porque, o valor é menos afetado por pontos especialmente extremos (*outliers*).

4. Desenvolvimento e Resultados

4.1. Compreensão do Problema

4.1.1. Contexto da indústria

Como o maior grupo de comunicação do Espírito Santo, a Rede Gazeta é muito tradicional no estado e está presente nos mais diversos veículos de comunicação. Assim, a Rede Gazeta chega ao lar da maior parte da população capixaba com programações diversas, que vão desde o jornalismo, até aos programas de auditório.

As principais concorrentes deste mercado de comunicação no estado do Espírito Santo são a TV Tribuna, afiliada do SBT, e a TV Vitória, afiliada da Record. Na grande maioria do tempo a TV Gazeta se apresenta mais forte que suas concorrentes, salvo algumas exceções ocasionadas por eventos específicos.

4.1.2. 5 forças de Porter

Ameaça de produtos substitutos - apesar de ser quase impossível a substituição total da TV aberta, por já ser um meio consolidado há várias décadas, é observada a transferência de certa parcela do seu público para os meios digitais.

Ameaça de entrada de novos concorrentes - o único tipo de ameaça sofrido pela TV são os meios digitais, que contam com conteúdos "*on demand*", ou seja, que o usuário consegue escolher o que vai assistir. Porém, agora comparando a Rede Gazeta com as outras emissoras, sua vantagem se estabelece pela confiança dos telespectadores, sendo ressaltada através dos dados disponibilizados.

Poder de negociação dos clientes - seus clientes, como os patrocinadores, antigamente não possuíam um grande poder de barganha, por não haver nenhum canal de comunicação tão abrangente quanto a TV aberta. Porém, com a evolução dos meios digitais, essa competição ficou mais acirrada, com outros locais para veiculação de anúncios publicitários, aumentando o poder de barganha a esses anunciantes, e assim, forçando os detentores do meio a baixarem o preço.

Poder de barganha dos fornecedores - os principais fornecedores de uma emissora são as produtoras, que montam alguns programas, e as empresas que trabalham com equipamentos de áudio e vídeo. Já que existem poucas emissoras no estado, que tem uma relevância grande no mercado, essas produtoras terceirizadas possuem baixo poder de barganha, uma vez que o poder econômico não é tão alto, por ser, na maioria das vezes, empresas da própria cidade ou estado. Porém, como a Rede Gazeta é filial de outra emissora, empresas de áudio e vídeo geralmente são padronizadas, por isso elas têm um poder de barganha maior. Dessa forma, há uma maior dependência dos fornecedores para com a emissora, do que o contrário.

Rivalidade entre os concorrentes - essa força se refere a necessidade de antecipar tendências e estar sempre atualizado com as novidades do mercado. A solução desenvolvida pelo grupo **hefEStos** atuará mais fortemente nesse aspecto de ficar a par das novidades do mercado, ajudando a empresa a prever a reação do público de um novo programa.

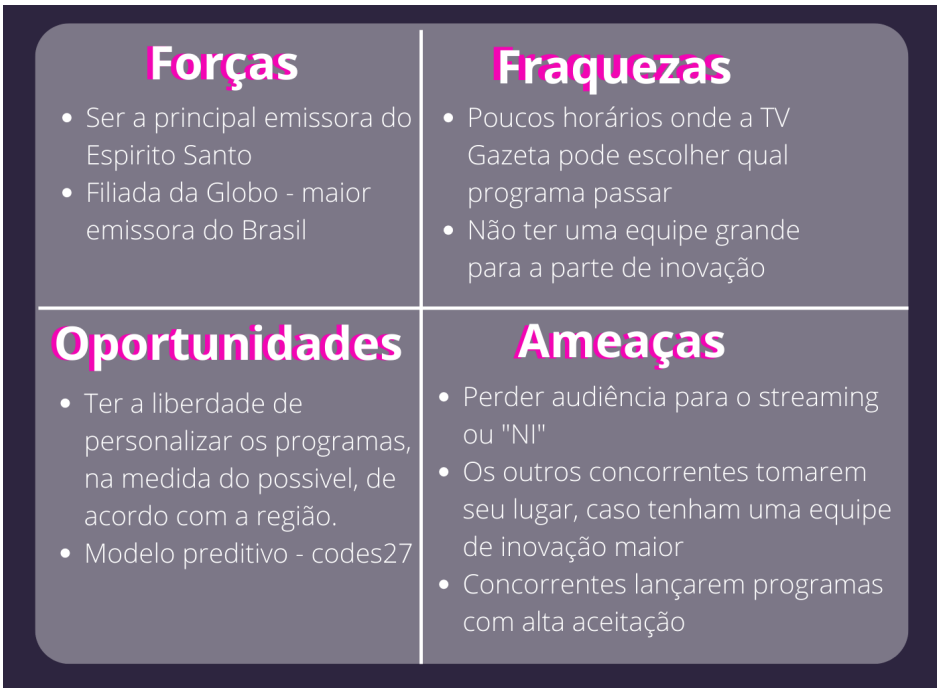
4.1.2. Análise SWOT

Com as análises feitas, foi possível verificar algumas forças, fraquezas, oportunidades e ameaças que o negócio apresenta. Dentro de **forças**, os principais pontos relacionados a essa parte da análise residem no fato de que a emissora parceira exerce um papel de líder no Espírito Santo, além de ser uma emissora afiliada da maior emissora do Brasil.

Já em **fraquezas**, fatores que se apresentam como ponto de melhoria frente ao negócio, pode-se inferir que os poucos horários nos quais a emissora parceira pode escolher qual programa exibir pode ser problemático, além disso, não ter uma equipe grande para a parte de inovação pode retardar o processo de renovação dos conteúdos programáticos para o público mais jovem e para o público ativo em mudança.

Considerando fatores externos, tratando de **oportunidades**, constata-se a possibilidade de conseguir personalizar os programas exibidos, na medida do possível, de acordo com a região, agradando mais a audiência e gerando mais engajamento. Além disso, o modelo preditivo, **codes27**, desenvolvido pelo grupo **hefEStos** é vista como uma oportunidade para a TVGazeta. Ainda sobre elementos externos que podem afetar o negócio, tem-se as **ameaças**, em que se vê um cenário em que a empresa pode ter uma tendência a perder cada vez mais audiência para o *Streaming* ou "NI", além disso, uma negligência com a equipe de inovação pode levar emissoras concorrentes, que investem muito nessa área, a se tornarem mais relevantes no mercado do que a emissora parceira. Nesse mesmo sentido, quando os concorrentes lançam programas novos eles podem ter um alto índice de aceitação, assim afetando diretamente a TV Gazeta.

Figura 1 - Matriz SWOT



Fonte: do próprio autor (2022).

4.1.3. Planejamento Geral da Solução

4.1.3.1. Dados disponíveis

Para o desenvolvimento da aplicação a Rede Gazeta disponibilizou alguns dados referentes à audiência, são esses:

- Datas;
- Hora de início;
- Emissoras;
- Dias da semana;
- Porcentagens utilizadas:
    - Rat%
    - Shr%
    - Rch%
    - Fid%

- Total de domicílios;
- Caracterização do público telespectador:
    - Classe:
        - AB;
        - C1;
        - C2;
        - DE;
    - Gênero:
        - Masculino;
        - Feminino;

        - Faixa etária:
            - 4-11 anos;
            - 12-17 anos;
            - 18-24 anos;
            - 25-34 anos;
            - 35-49 anos;
            - 50-59 anos;
            - 60+ anos.
- Grade de programação de cada emissora;

Divididos nas seguintes emissoras:

- Emissora 1;
- Emissora 2;
- Emissora 3;
- Canais pagos;
- Total Ligados Especial (TLE);
- Não identificado (NI);

Além disso, é dividido em dias da semana:

- Segunda a sexta;
- Sábado;
- Domingo.

4.1.3.2. Solução proposta

Sabendo da necessidade do parceiro por uma análise preditiva da audiência da Rede Gazeta, a solução proposta busca descrever, por meio de uma Inteligência Artificial, a pontuação que determinado programa terá, quando lançado, de acordo com o horário, dia da semana, confirmação se o dia em questão é feriado, eixo e público especificado.

4.1.3.3. Tipo de tarefa

O tipo de método empregado será o de Regressão, pois deve estimar dados de audiência de acordo com os valores de entrada que foram coletados anteriormente.

4.1.3.4. Como a solução deverá ser utilizada

O usuário deverá inserir os horários, a data, feriado (caso o dia se trata de um), o segmento do novo programa e escolher o modelo de acordo com as características do público desejada e a solução irá retornar uma previsão dos pontos de audiência.

4.1.3.5. Quais são os benefícios trazidos

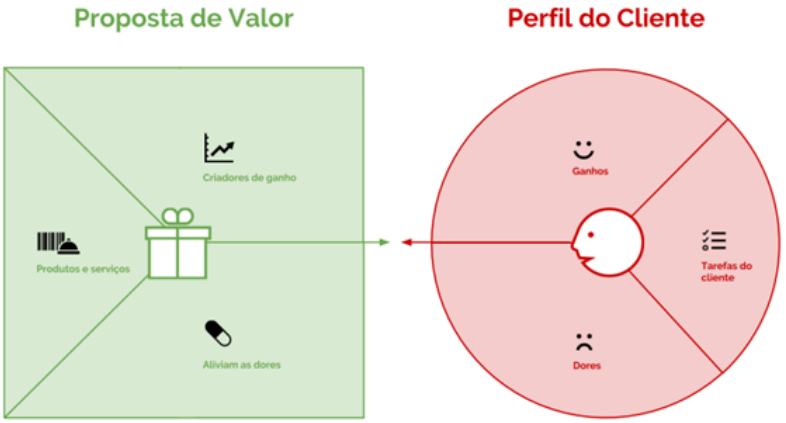
Os benefícios trazidos pela solução são: os de melhorar a capacidade na realização da análise dos dados, ajudar em previsões para lançamentos de programas futuros, e promover uma melhora no desempenho de programas já existentes.

4.1.3.6. Qual será o critério de sucesso e qual será a medida utilizada

O critério utilizado como parâmetro de sucesso será a comparação com os valores fornecidos pelo banco de dados. Nesse, utilizamos parte do dataset para o treinamento do modelo, e o restante foi utilizado para comprovar o quão próximo foi o retorno do modelo com relação à realidade.

4.1.4. Value Proposition Canvas

Figura 2 - Value Proposition Canvas



Fonte: do próprio autor (2022).

Um framework que visa a garantia da relação produto cliente, o **Canvas de Proposta de Valor** se apresenta como uma ferramenta detalhada da relação entre as duas partes do negócio: o cliente e a solução. Usado principalmente para refinar o serviço oferecido ou desenvolver da melhor forma um rascunho de um novo produto (B2B, 2022).

Neste sentido, buscando aproximação com o cliente e o modelo preditivo elaborado, foi feito um Canvas de proposta de valor para o empreendimento corrente, objetivando o perfil do cliente, sendo esse baseado em suas dores, ganhos e atividades realizada pelo mesmo e também objetivando o mapa de valor da solução, esse contando com o produto que seria oferecido, como ele funcionaria de alívio para as dores do cliente e o papel dele como criador de ganho.

4.1.4.1 Perfil do cliente

Tarefas do cliente:

- ➔ Criação de conteúdo para programas de TV;
- ➔ Analisar, de acordo com o histórico, se o programa iria se encaixar na programação.

Dores:

- ➔ Dificuldade em determinar o conteúdo adequado para agradar a audiência em um determinado horário;
- ➔ Alto investimento em programas que não repercutiram da forma esperada.

Ganhos do cliente:

- ➔ Melhora na acurácia da capacidade de prever a audiência de um determinado programa, baseado em algumas de suas informações como horário, data e tema. Adaptando, assim, o conteúdo, para aumentar o *score* de audiência.

4.1.4.2 Mapa de Valor

Produtos e serviços:

- ➔ Um modelo preditivo que recebe informações básicas de um possível novo programa, e retorna um score de audiência e as principais variáveis que pesaram nesse.

Analgésicos/alívio das dores:

- ➔ Previsão acurada do possível sucesso de conteúdos, antes da produção.

Criadores de ganhos:

- ➔ Evita gastos com programas de baixa audiência;
- ➔ Fornece informações de conteúdos capazes de maximizar a audiência.

4.1.5. Matriz de Riscos

Para o desenvolvimento do projeto foi observado alguns riscos que poderiam ocorrer, sendo eles classificados entre negativos (riscos) e positivos (oportunidades). Cada um desses foi posicionado de acordo com seu impacto e probabilidade de acontecer, sendo os de vermelho os de maior preocupação e os verdes os menos preocupantes.

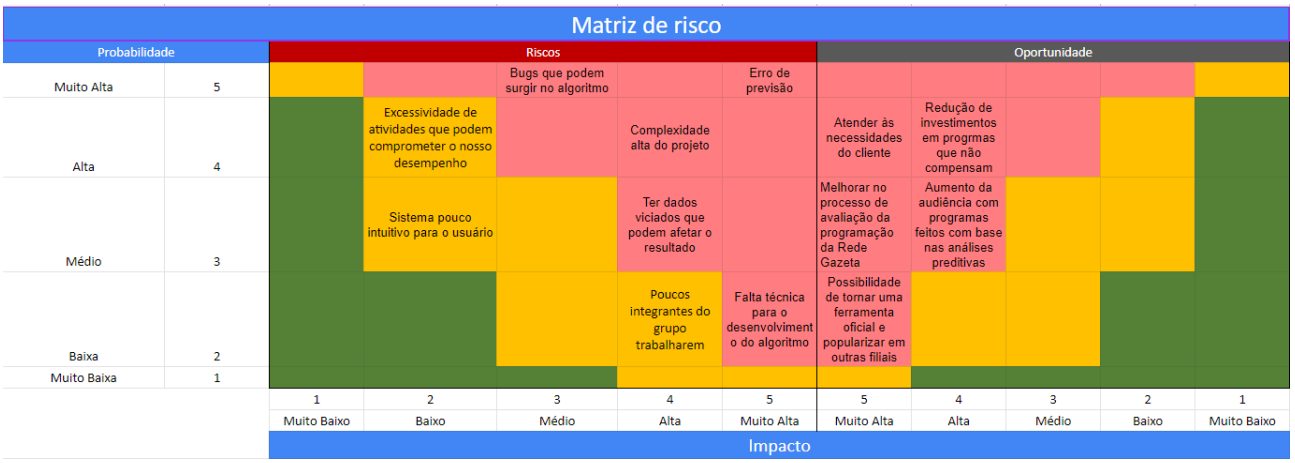
Reconheceu-se como riscos de alta probabilidade e alto impacto possíveis bugs que poderiam surgir no algoritmo, como consequência, erro de previsão, além do risco de complexidade alta do projeto, que pode afetar no sucesso do produto final.

Ademais, identificou-se outros riscos de menor gravidade, como excessividade de atividades que podem comprometer o desempenho da equipe, somado com a possibilidade de poucos integrantes trabalharem, havendo uma concentração de tarefas, e a criação de um sistema pouco intuitivo para o usuário.

Como oportunidades, determinou-se com alta possibilidade e impacto a redução de investimentos em programas que não compensam para a emissora, além da grande chance de atender às necessidades do cliente, melhorando o processo de avaliação da programação, podendo sofrer aumento da audiência com programas feitos baseados nas análises preditivas

e, a longo prazo, podendo se tornar uma ferramenta com escalas maiores que apenas a aplicação inicial do parceiro.

Figura 3 - Matriz de risco.



Fonte: do próprio autor (2022).

4.1.6. Personas

Pensando no público que se buscava atender, foram feitas duas personas, na qual uma representaria o público que usaria o produto para a produção de novos programas, pensando na audiência, público e horário. E a outra, tem como objetivo dinamizar a logística da análise de dados da emissora e aumentar o desempenho da grade de programação.

Ayumi Sato é uma produtora de programas de TV que tem dificuldades em criar novos conteúdos pois não consegue prever a audiência do programa a ser criado e quais características serão determinantes para obter-se uma boa audiência. Ela utilizará o modelo preditivo para conseguir trabalhar de forma mais assertiva, levando em consideração a previsão do público e do valor de audiência para o horário em que o programa será transmitido.

Figura 4 - Persona

Fonte: do próprio (2022).

Já Arthur Silva é engenheiro de computação e trabalha para uma filial de emissora de TV. Ele tem dificuldades em fazer uma boa previsão da audiência de futuros programas a serem lançados e consequentemente em passar informações objetivas para os produtores. Ele utilizará o modelo preditivo para obter informações mais úteis e precisas considerando múltiplas variáveis simultaneamente.

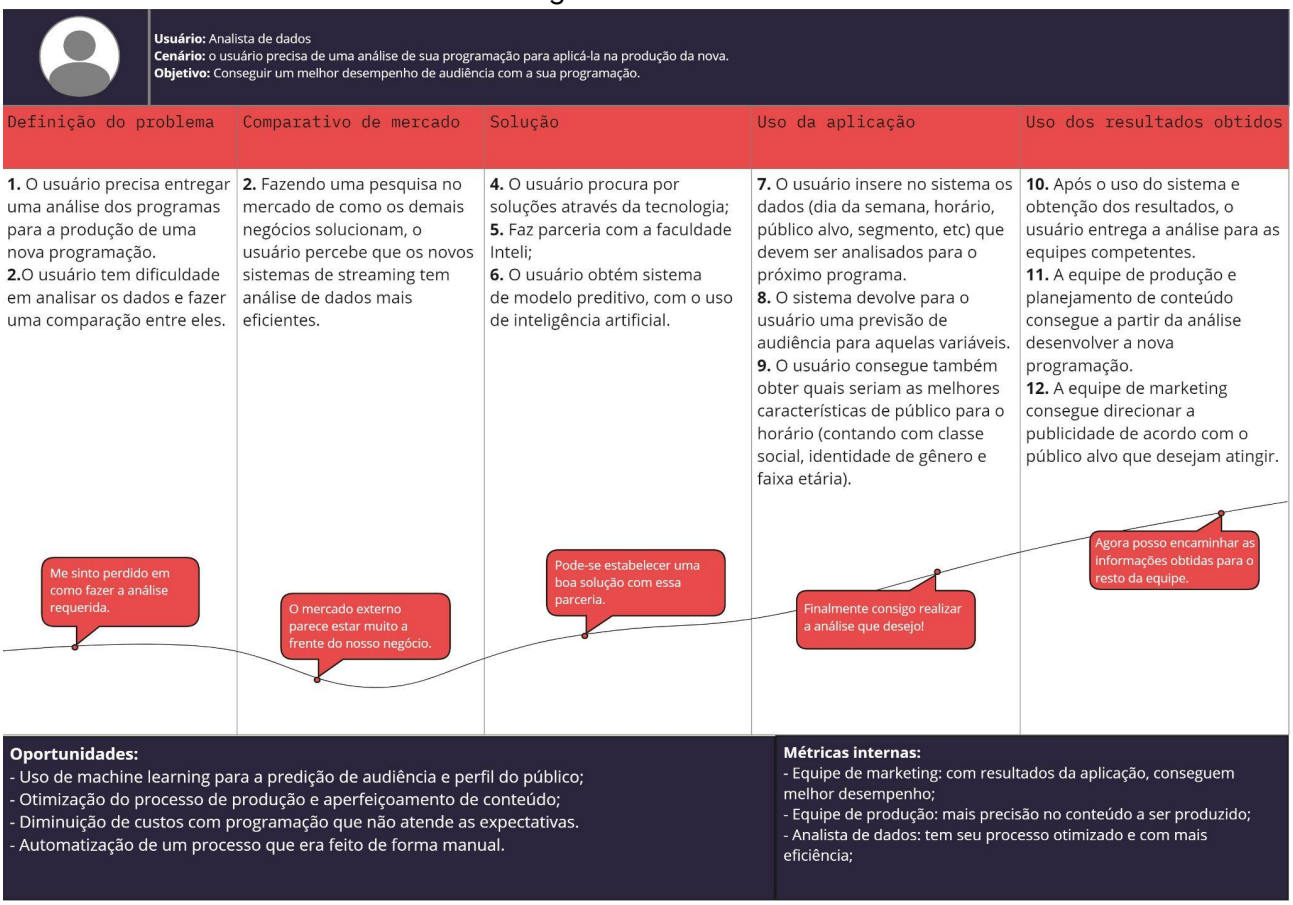
Figura 5 - Persona

Fonte: Elaborada pelos Autores (2022).

4.1.7. Jornadas do Usuário

A jornada de usuário é um mapa visual de todo o caminho traçado pelo cliente antes, durante e depois do produto em questão. O processo se inicia com um período de definição, onde o usuário identifica qual é o contratempo deve ser resolvido, e após isso, existe a fase de comparação de mercado, onde deve-se buscar quais são as opções existentes no mercado. Nesse momento a TV Gazeta encontrou o grupo **hefEStos**, com a sua solução **codes27**, seguindo assim para o seu uso. Após o uso da AI produzida, os resultados obtidos são, finalmente, aplicados em análises e produção de novos programas.

Figura 6 - Jornada do usuário.



Fonte: do próprio (2022).

4.2. Compreensão dos Dados

4.2.1. Descrição dos dados

Foi disponibilizados dados de diferentes emissoras no formato XLSX, posteriormente convertido para CSV, baseados em pesquisas do IBOPE, utilizando os seguintes parâmetros percentuais:

Rat%, medida que é calculada a partir da quantidade de indivíduos/domicílios ligados em determinado evento de TV, sendo que 1 ponto de audiência equivale a 1% do universo pesquisado. Esse é calculado da seguinte forma: Rat% = (Rat# / universo) x 100.

Shr% descreve a participação da audiência em um determinado evento, sobre o total de televisores ligados, em um determinado período. Esse é calculado da seguinte maneira: Shr% = (Rat% / TLE%) x 100.

Rch% é o total de indivíduos, ou domicílios, diferentes alcançados por pelo menos 1 minuto. Reforçando que o tempo total, nesse caso, não está sendo considerado, e sim o contato que houve com a programação, faixa horária, emissora, etc. Esse é calculado da seguinte forma: Rch% = número de telespectadores / universo x 100.

Fid% ilustra a permanência dos telespectadores no evento em questão, ou seja, quanto tempo daquele programa foi consumido pelo público. Esse é calculado da seguinte maneira: Fid% = Rat% / Rch% x 100.

Além disso, informações acerca do perfil da audiência, como gênero, faixa de idade e classe social foram enviadas.

4.2.1.1. Descrição de como os dados serão agregados/mesclados

Devido ao alto volume de dados, para melhor visualização, esses foram mesclados usando a média dos valores, e posteriormente agregados a partir de uma funcionalidade da plataforma Google Sheets. Ademais, na segunda semana do projeto, depois de múltiplos pedidos, foi adicionado uma nova planilha no conjunto de dados, que contempla a grade horária, que conta com as seguintes informações: praça, data, faixa horária, o nome do programa e segmento das três emissoras (Emissora 1, 2 e 3). Com essas novas informações, se torna mais fácil ter uma noção de qual programa está fazendo o maior sucesso, comparando com as outras emissoras, e assim analisar o porquê isso acontece, se é por conta do horário ou pela falta de concorrência, por exemplo.

4.2.1.2. Descrição dos riscos e contigências relacionados a esses dados

Nesse contexto, durante a análise dos dados, foi verificado que há dois riscos a serem considerados na análise de tais dados: o primeiro risco é os dados tenham viés, já o segundo se trata da concorrência injusta na coleta dos dados.

No primeiro ponto é necessário ponderar que, no aparelho utilizado na medição da audiência, quando colocado no domicílio de cada família, é criado um perfil para cada integrante, com informações de classe social, gênero e idade. Quando a TV é ligada, quem está assistindo deve selecionar o seu próprio perfil, é nesse momento que pode acontecer o viés, já que uma criança, por exemplo, pode selecionar errado ou mais de uma pessoa pode estar vendo TV. Adicionalmente, a TV pode continuar ligada em um perfil originalmente correto, mas que já não é válido, ou seja, outro integrante pode ter começado a ver TV. e não selecionou o seu perfil, ou até mesmo uma mãe/pai pode, na pressa, selecionar o perfil próprio para uma criança assistir.

O segundo risco é por conta da concorrência injusta, já que a emissora chamada de Não Identificado, abrange algo muito genérico, isto é, apesar de ser chamada de uma emissora, dentro dela há diversas plataformas de vídeos, tornando a comparação injusta para as emissoras 1,2 e 3, que estão separadas.

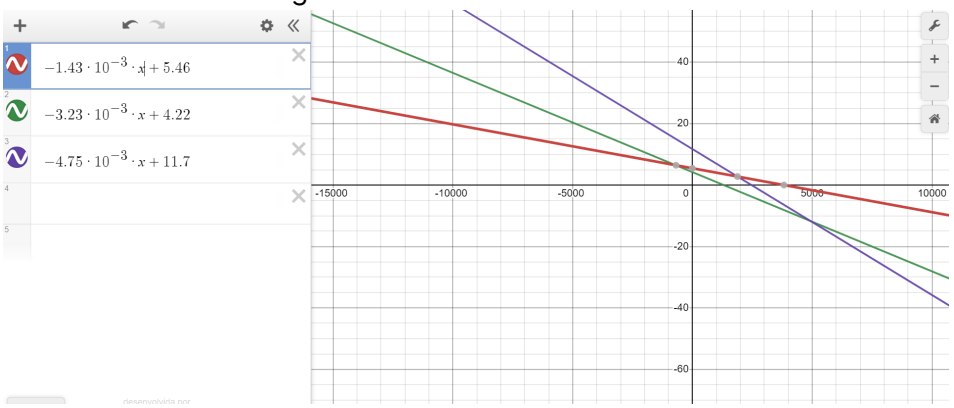
4.2.1.3. Descrição de como será selecionado o subconjunto para análise inicial

4.2.1.4. Descrição das restrições de segurança

Para a elaboração do modelo, embora tenha sido concedido os dados de programação e audiência de algumas emissoras de TV aberta do Espírito Santo para a criação do modelo, foi proibida a publicação dos nomes das emissoras.

4.2.2. Descrição estatística básica dos dados

Figura 7 - Gráfico de tendência



Fonte: do próprio (2022).

- ➔ Emissora 1 → azul
- ➔ Emissora 2 → verde
- ➔ Emissora 3 → vermelho

Este gráfico ilustra a tendência de audiência nas emissoras em um intervalo de 2 anos, pode ser observado uma queda forte de audiência em todas, se destacando a maior queda na emissora 1.

4.2.3. Dicionário dos dados

Alguns termos são específicos para a esfera televisiva, sendo necessário destacá-las e explicitar seus significados. Dessa forma, seguem abaixo descritos os termos usados no projeto.

Tabela 1 - Dicionário dos dados.

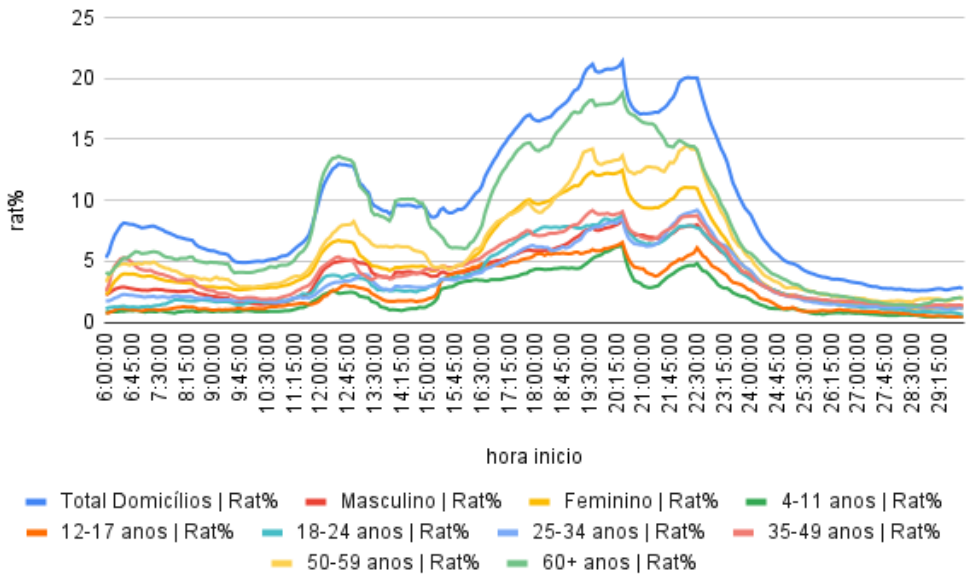
Parâmetros	Descrição	Como calcular
Rat%	Quantidade de indivíduos/domicílios ligados na TV (1 ponto = 1%)	Rat% = (Rat# / universo) x 100
Shr%	Participação da audiência em um evento, sobre o TLE de um período	Shr% = (Rat% / TLE%) x 100
Rch%	Total de domicílios ou indivíduos alcançados por 1 minuto ou mais	Rch% = número de telespectadores / universo x 100
Fid%	Permanência dos telespectadores naquele evento	Fid% = Rat% / Rch% x 100

Fonte: do próprio autor (2022).

4.2.4. Emissora 1

Segunda a sexta

Figura 8 - Audiência de segunda à sexta da emissora 1.



Fonte: do próprio autor (2022).

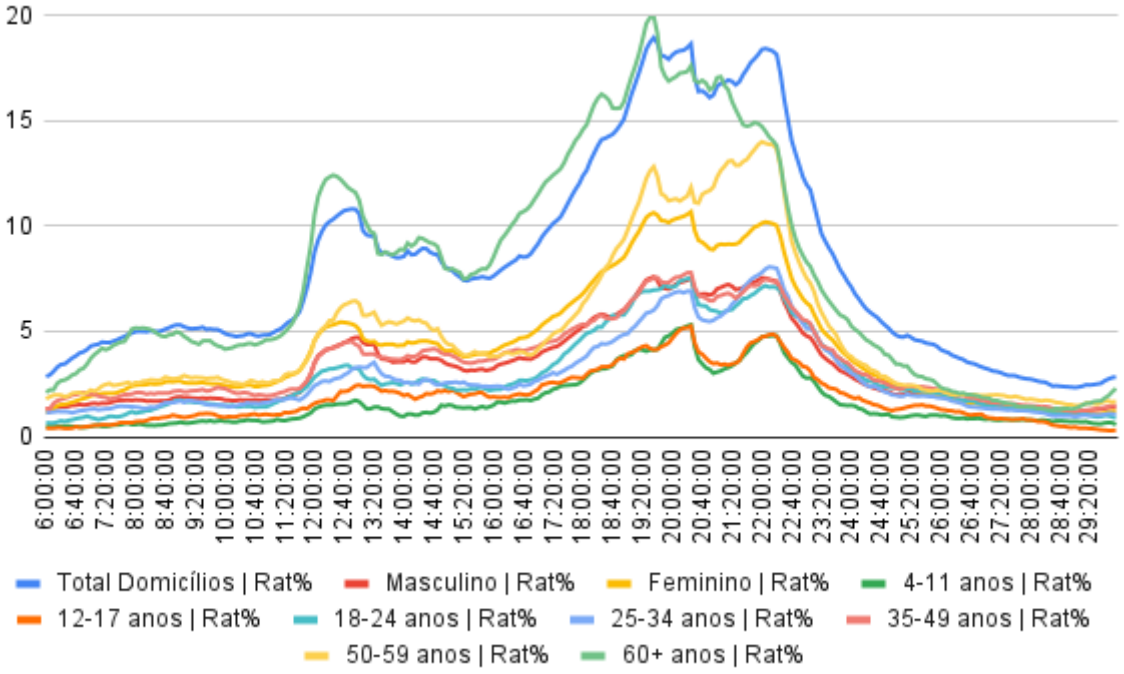
Para uma primeira análise dos dados, foi necessário fazer a junção dos dados de segunda a sexta, entre 2020 e 2022 da emissora 1. Sendo assim, o Rat% foi dividido em segmentos de idade, gênero e total de domicílios. Todos os dados são coletados a cada 5 minutos.

Na maioria dos horários o público feminino é predominante em comparação ao masculino. Seguindo a mesma ideia, o público com mais de 60 anos é predominante em comparação ao resto das idades, se mantendo estável durante toda a tarde, o único momento em que eles perdem a soberania é depois das 22h, onde o público de 50-59 anos ganha.

Existem dois picos claros nesse gráfico: 1. No horário do almoço (aproximadamente às 12h), no momento em que está passando programa jornalístico regional. 2. Durante a noite (das 18h até aproximadamente 00h), momento que são exibidos programas de entretenimento como novelas e reality shows e jornais. Pode-se notar também que no horário de exibição de um jornal à noite (aproximadamente das 20h30 às 21h30), há uma redução do público.

Sábado

Figura 9 - Audiência de sábado da emissora 1.



Fonte: do próprio autor (2022)

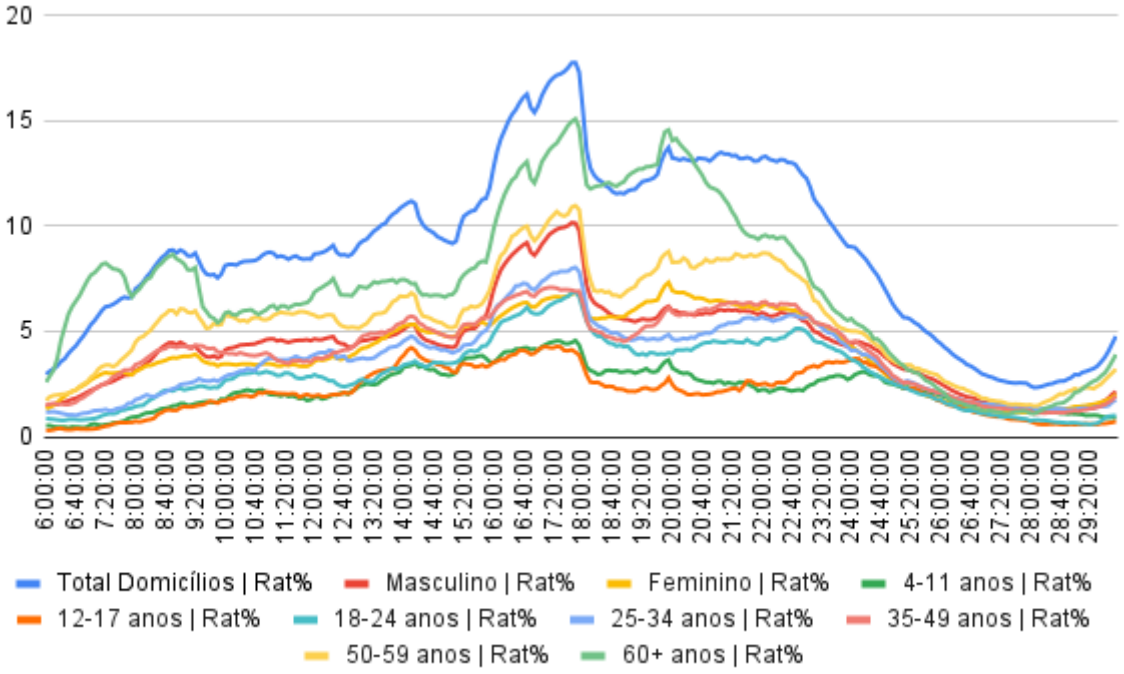
O gráfico acima reúne todos os sábados do mesmo espaço de tempo (junho de 2020 até junho de 2022), sendo dividido nos mesmos segmentos do gráfico anterior: todos os dados de Rat% com gêneros e idades e o total de domicílios.

Seguindo o mesmo padrão do gráfico anterior, as pessoas que têm mais de 60 anos continuam na soberania de audiência, porém essa diferença entre eles é bem menor quando comparado aos dias úteis. Além disso, em alguns horários essa faixa etária ganha do total de domicílios, como por exemplo entre 14h15 até 18h50.

As mulheres continuam com uma maior audiência do que homens, seguindo o mesmo padrão do gráfico anterior, porém em alguns horários, como 14h e na madrugada, essa diferença diminui bastante.

Domingo

Figura 10 - Audiência de domingo da emissora 1.



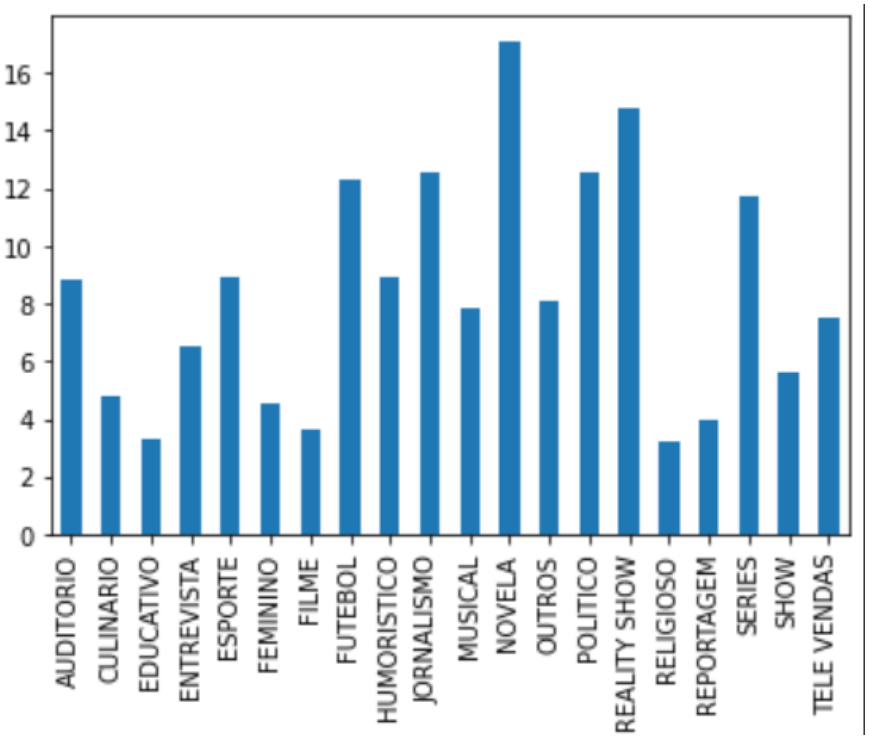
Fonte: do próprio autor (2022).

Fizemos um gráfico que reúne todos os domingos do mesmo espaço de tempo (junho de 2020 até junho de 2022), dividimos nos mesmo segmentos do gráfico anterior: todos os dados de Rat% com gêneros e idades e o total de domicílios.

Nesse gráfico, pode-se notar que a variação da audiência ocorre de outra forma, devido à diferença muito grande da grade de programação com relação aos dias úteis, sendo muito constante ao longo do dia com programações variadas de reality show, carros, rural, etc, exceto no horário do programa de futebol (das 15h50 às 18h05), em que há um pico em praticamente todos os nichos. Ainda nesse contexto, deve-se ponderar que logo após o término da programação de futebol há uma queda que leva a audiência de volta para o patamar anterior, e fica estável durante a noite com uma programação de auditório, show e jornalística.

Divisão por segmento

Figura 11 - Gráfico de audiência por segmento da emissora 1.



Fonte: do próprio autor (2022).

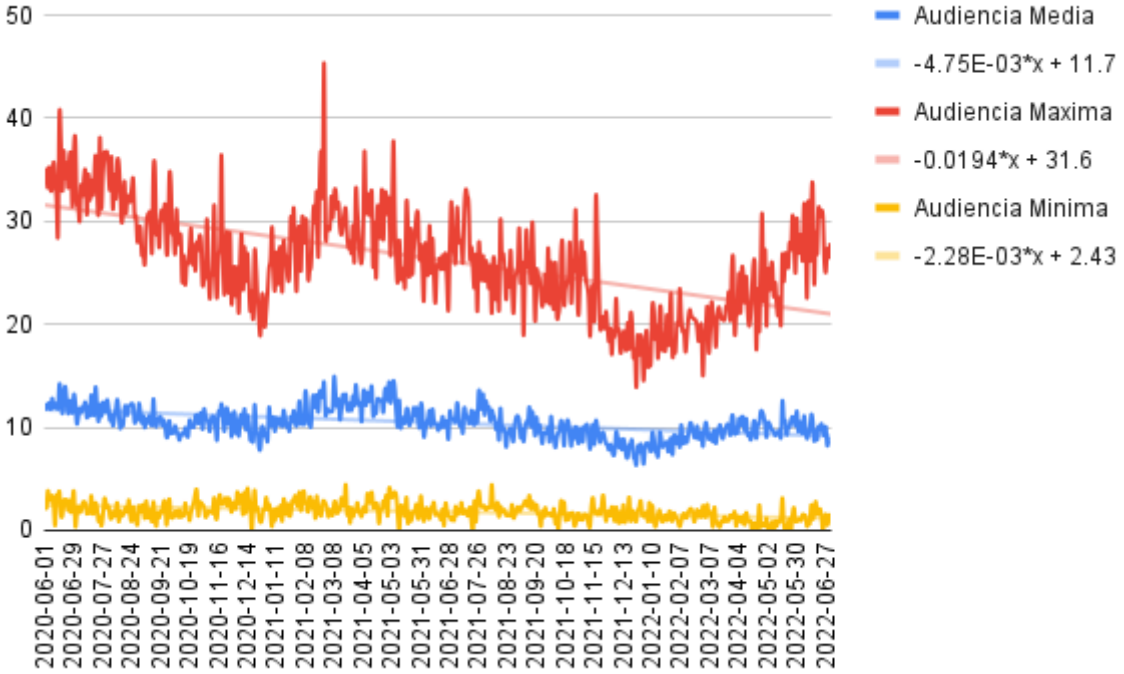
Cada programa tem a sua respectiva categoria, ou seja segmento, com isso foi realizado um gráfico com o Rat% com o total de domicílios, assim os número no eixo y representam as porcentagens e o eixo x os segmentos. Esse gráfico representa somente os segmentos da emissora 1.

Pode-se observar que os segmentos com maior audiência são: novela, reality show, jornalismo e políticos, que estão empatados, e futebol, em ordem decrescente. Já os com pior audiência são: educativo, religioso, filme e reportagem, em ordem crescente.

Porém, interpretando-se os dados foi possível perceber que essa alta de audiência no reality show pode acontecer por conta de um programa específico, que ocorre em alguns meses do ano, podendo maximizar os dados, sendo um ruído.

Segunda a Domingo - Audiência

Figura 12 - Gráfico de médias da audiência diária.



Fonte: do próprio autor (2022).

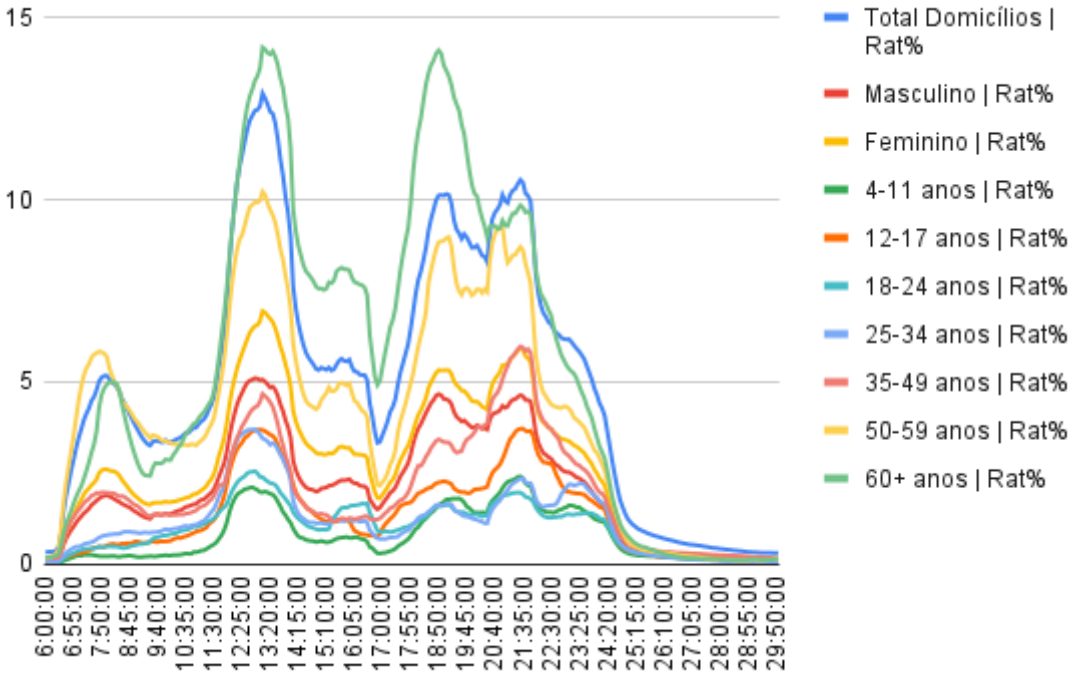
Esse gráfico apresenta uma média de 3 dados importantes: audiência máxima, média e mínima. A primeira mede qual foi a maior quantidade de pessoas que estavam com a TV ligada na emissora 1 naquele dia específico, ou seja, o pico do dia. Já a segunda é a média de audiência naquele dia, que se mantém bem estável, comparando com o primeiro dado citado. O terceiro dado é exatamente o oposto do primeiro, então ele mede quanto que foi a menor quantidade de pessoas ligadas na TV naquele dia, ou seja, o ponto baixo do dia.

Pode-se notar com clareza que, a audiência média e mínima se mantém estável quando comparado com a linha azul, que mostra a média daqueles valores. Já na audiência máxima isto não acontece, existem pontos específicos em que a audiência é visivelmente mais alta ou mais baixa. Esses picos mais altos são em momentos particulares, como final de algum reality show, final de campeonato de futebol, ou em feriados como o Natal.

4.2.4. Emissora 2

Segunda a sexta

Figura 13 - Gráfico de audiência de segunda à sexta emissora 2.



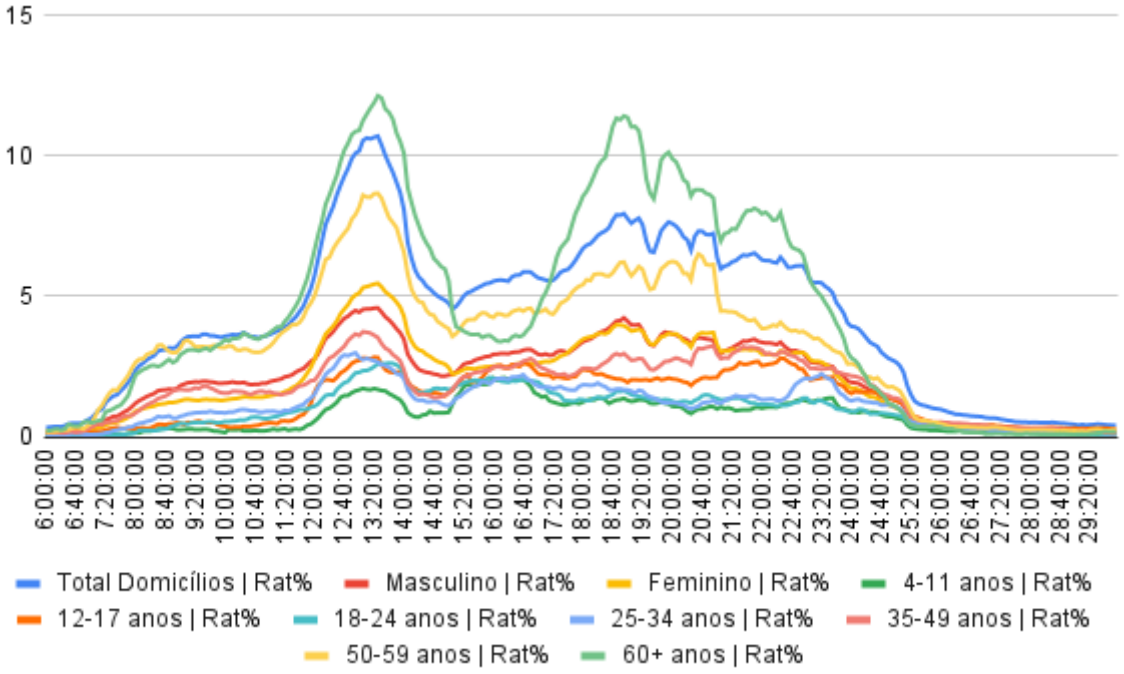
Fonte: do próprio autor (2022).

O gráfico acima representa a média de audiência dos dias de segunda a sexta dos últimos dois anos na emissora 2. Sendo separado por idade, gênero e total de domicílios. Como a maioria dos gráficos citados nesse documento, esse se mantém no mesmo padrão: o sexo feminino ganha do sexo masculino em todos os horários e o mesmo acontece com a idade 60+, porém algo se diferencia: as pessoas que tem entre 50 e 59 anos ganham de todas as outras idades, menos de 60+, os dois gêneros e em alguns momentos até mesmo do total.

Os picos principais acontecem durante o horário de almoço, 12h25 - 14h15, e no início da noite (18h50 - 20h40) e isso se dá em todas as linhas, em proporções diferentes. Por outro lado existem momentos que há uma queda muito grande, 14h15 - 15h10 e 17h, isso pode ser por dois motivos: um deles é que as concorrentes podem estar passando algo que é mais interessante para esse público ou o programa que passa na emissora 2, nesses horários, não agrada o público.

Sábado

Figura 14 - Gráfico da audiência de sábado da emissora 2.

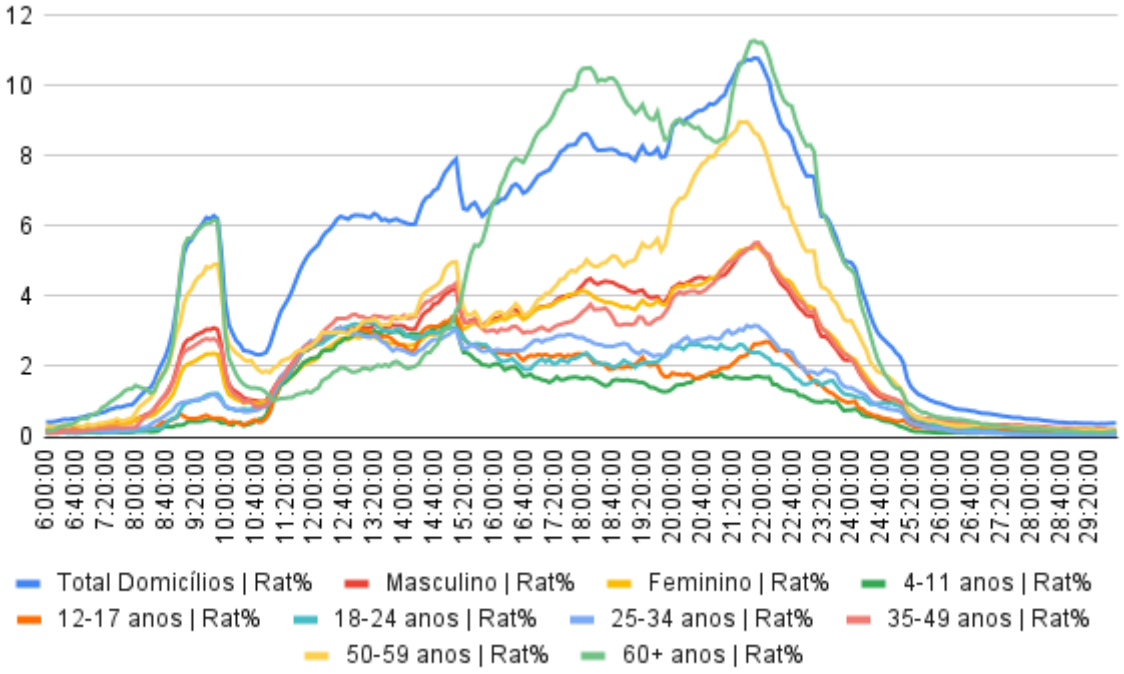


Fonte: do próprio autor (2022).

Este gráfico mostra a média de audiência por horário nos sábados da emissora 2 dividido por faixas etárias. Pode-se perceber que ao longo do dia inteiro a audiência predominante é de mais de 50 anos. Nota-se também dois picos de audiência, o primeiro às 13h20 e o segundo por volta das 19h, sendo o segundo mais dispersado. Os picos do gráfico são mais definidos nas faixas etárias mais velhas, ou seja, apesar de o público jovem ter menor audiência em todos os horários sua audiência é mais constante ao longo do dia.

Domingo

Figura 15 - Gráfico de audiência de domingo da emissora 2.



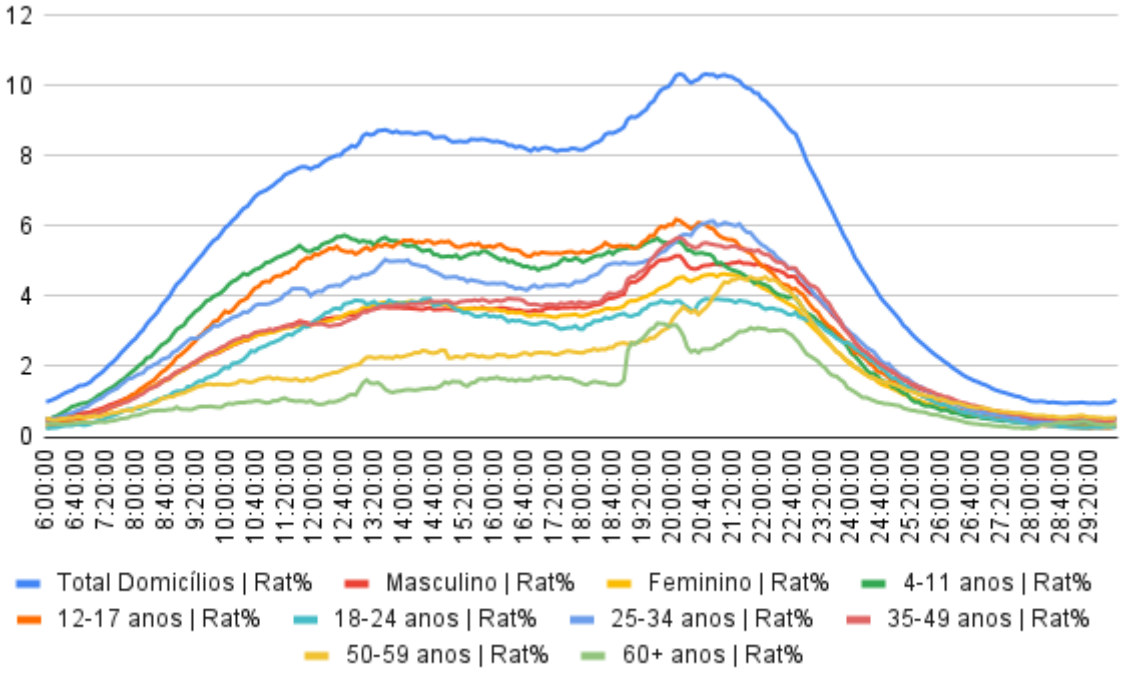
Fonte: do próprio autor (2022).

Analisando o gráfico, é possível perceber que em todos os horários o público feminino é predominante em comparação com o masculino. Seguindo a mesma ideia, o público com mais de 60 anos é predominante em comparação ao resto das idades, e se mantém estável desde meio-dia até o final da tarde. Porém o público de 50-59 anos ganha a soberania depois das 19h. Como em todos os gráficos já analisados, o horário da madrugada, 00h até 7h, a audiência se mantém baixa em todas as idades.

4.2.5. Conteúdo Não Identificado

Segunda a sexta

Figura 16 - Gráfico da audiência de segunda à sexta de conteúdo não identificado.



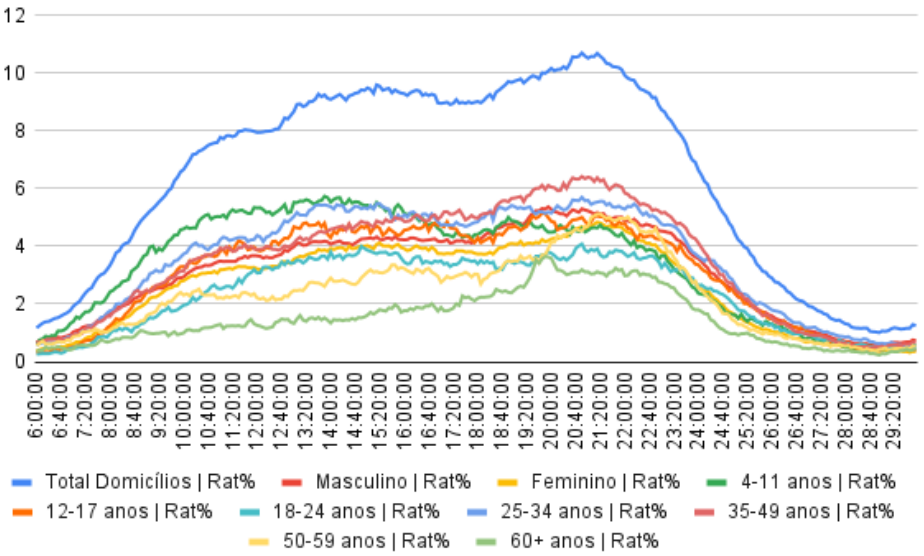
Fonte: do próprio autor (2022).

Se tratando de Conteúdo Não Identificado foi feito um gráfico que une todos os dados de segunda a sexta, entre junho de 2020 e junho de 2022. Dividindo o Rat% em segmentos de idade, gênero e total de domicílios, e todos os dados foram coletados com um intervalo de 5 minutos. Analisando o gráfico, é possível perceber que a partir das 19h até 00h o público masculino é predominante, e o público entre 12-17 anos é predominante em todos os horários.

Existem dois picos claros nesse gráfico: 1. A partir de 18h40, pessoas cuja idade é maior que 60 anos, tendem a ligar mais a TV nessa emissora, sendo a única idade onde há um pico relevante. 2. Nas outras idades esse pico ocorre a partir das 20h, porém há um impacto bem menor.

Sábado

Figura 17 - Gráfico de audiência de sábado do conteúdo não identificado.



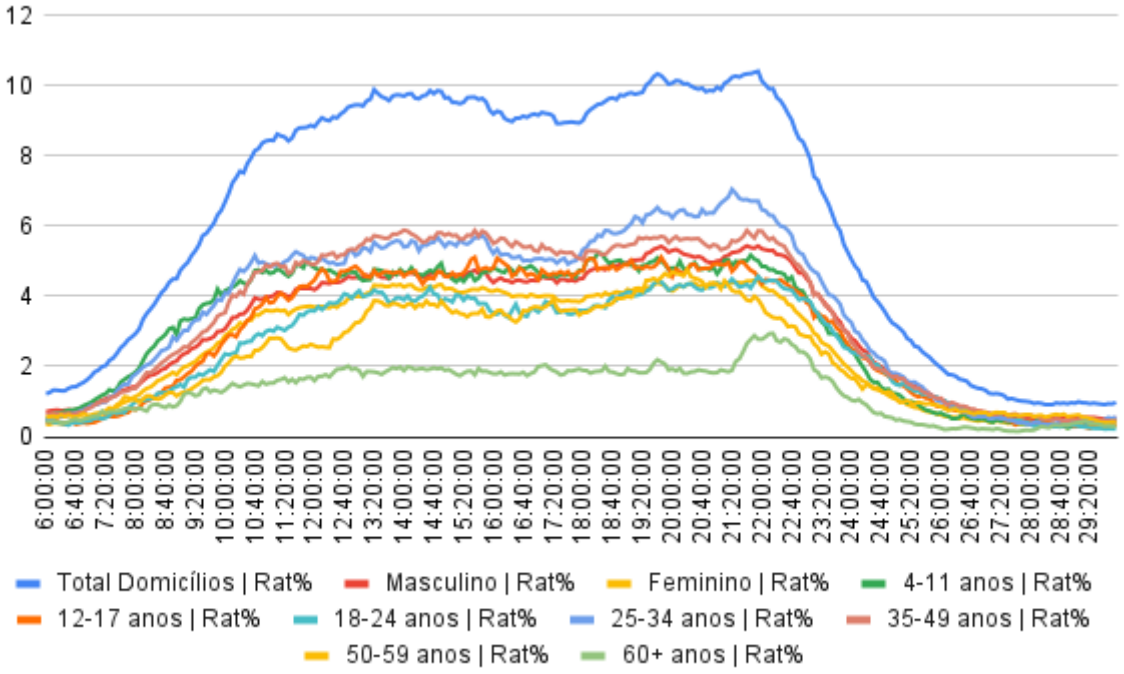
Fonte: do próprio autor (2022).

Diferente do gráfico anterior, o público masculino não se mantém predominante durante o dia inteiro aos sábados, por outro lado, o público de 4-11 anos se mantém predominante durante o período da manhã até a tarde, por volta de 16h40, que é quando o público de 35-39 anos vira o público predominante durante o resto do dia.

Também é possível perceber um aumento de audiência na maior parte dos públicos por volta de 19h45, menos no público infantil (4-11 anos) que tem um leve pico de audiência e depois começa a cair.

Domingo

Figura 18 - Gráfico de audiência de domingo do conteúdo não identificado.



Fonte: do próprio autor (2022).

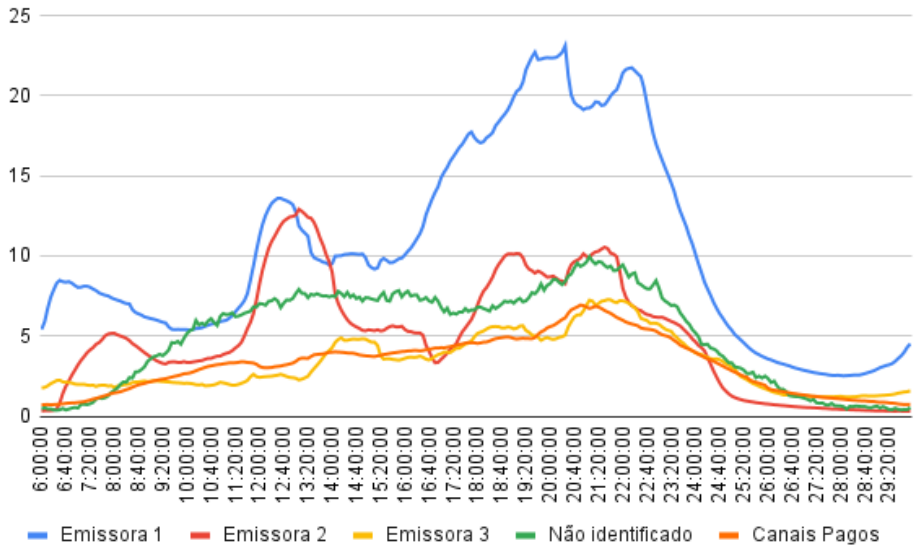
Foi feito também um gráfico que reúne todos os domingos do mesmo espaço de tempo (junho de 2020 até junho de 2022), e foi dividido nos mesmo segmentos do gráfico anterior: todos os dados de Rat% com gêneros, idades e o total de domicílios.

No gráfico de domingo, pode-se perceber que a audiência do público de 60+ anos é a menor de todas e se mantém assim durante o dia inteiro, tendo um leve pico no período das 21h45. Outro ponto interessante é que o público de 50-59 anos começa o dia com um audiência bem diferente das outras faixas etárias e por volta de 16h50 começa a se assemelhar com os outros públicos.

Já os públicos de outras faixas etárias se mantêm semelhantes durante o dia inteiro, tendo um aumento de audiência no final da tarde (por volta de 17h40), e começa a ter uma queda de audiência por volta das 22h40. Algo que pode ser percebido nesse intervalo de tempo é que o aumento de audiência no público de 24-35 anos é o mais relevante entre esses públicos.

4.2.6. Comparação entre emissoras

Figura 19 - Gráfico de comparação de audiência entre as emissoras.



Fonte: do próprio autor (2022).

É possível observar uma predominância de audiência da emissora 1 comparada a todas as outras. Porém a emissora “Não identificado" sobressai durante o período de almoço, tendo uma audiência equilibrada com a primeira.

A emissora 1 tem picos de audiência durante o período da noite, como por exemplo às 19h20, 20h40 e às 22h, algo que não se repete com tanta nitidez nas outras emissoras. Todas as emissoras seguem o mesmo padrão, durante 00h e 04h, já que a audiência cai muito e se mantém em decréscimo durante esse intervalo. Porém a emissora 1 continua tendo o seu diferencial: enquanto a partir das 5h ela começa a ter um aumento de audiência, o resto das emissoras continua decaindo.

As demais emissoras têm um constante crescimento de audiência ao longo do dia, até as 21h e depois disso sua audiência começa a cair e só volta a se recuperar no período da manhã por volta das 10h10.

4.2.7. Descrição da predição desejada, identificação da sua natureza

Para o desenvolvimento do projeto, após análises, concluiu-se que o modelo preditivo ideal é o de regressão. Esse quantifica a relação entre uma ou mais variáveis preditoras e uma representativa de *output*, significando a variável de resultado. Os dados nesse modelo têm a necessidade de serem independentes, ou seja, os valores de uma das *features* não pode depender diretamente da outra.

Uma vez que o *target* desejado é em formato numérico, a natureza da predição desejada se apresenta de forma contínua, haja vista que a saída dos dados será feita através da predição de um score de audiência. no aprendizado supervisionado, ou seja, construído a partir dos dados de entrada (*features* pré-selecionadas).

4.3. Preparação dos Dados

4.3.1. Descrição das manipulações necessárias nos registros e suas respectivas features

4.3.1.1. Etapa de Junção

Inicialmente foram disponibilizados dados de 2020 até 2022, porém depois de alguns pedidos, a emissora enviou mais dados e, por isso, foi necessário juntá-los.

Código 1 - Dados de 2015-2019 com os de 2020-2022.

```
#junta a tabela de dias úteis com a de sábado e domingo
dataset = pd.concat([data_seg_sex_20,data_sabado_20,data_domingo_20,data_seg_sex_15,
data_sabado_15,data_domingo_15])
#faz com que todos os dados da coluna data sejam convertidos para datetime (um tipo de
variável próprio do pandas)
dataSet['Data'] = pd.to_datatime(dataSet['Data']).dt.date
dataSet = dataSet.drop_duplicates()]
dataSet = dataSet.sort_values(by=['Data','Hora Início'], ascending=True)
```

Fonte: do próprio autor (2022).

O código acima explica exatamente essa junção, como inicialmente as tabelas estavam separadas entre dias úteis e final de semana, primeiro era necessário juntar todas elas e por isso o comando `pd.concat` e definindo o novo `dataset`. Quando os nomes dos arquivos foram definidos, o final _20, sendo esse o dataset de 2020 até 2022, e o final _15, sendo esse o dataset de 2015 até 2019, foi dividido.

As datas duplicadas foram removidas, fazendo com que os dados pudessem ser colocados em ordem, tendo como parâmetro principal as datas e hora início como parâmetro secundário.

Se tratando de audiência de TV, quando é feriado, mais pessoas ficam em casa e por isso os números podem aumentar. Levando isso em consideração, foi decidido que era de extrema importância adicionar uma tabela com todos os feriados nacionais.

Código 2 - Feriados nacionais.

```
#Tabela de feriados
feriados = pd.read_csv('caminho do arquivo']
feriados = feriados.drop(['Dia da semana'], axis = 1)
#Adicionando os feriados
feriados['Data'] = pd.to_datetime(feriados['Data']).dt.date
dataSet = pd.merge(dataSet, feriados, how='left',left+on=['Data'], right_on=['Data'])
dataSet['Feriado'] = dataSet['Feriado'].notnull().astype('int')
```

Fonte: do próprio autor (2022).

Para os dados de todos os feriados, foi necessário buscar na internet e assim foi utilizado um arquivo csv com essas informações. [1]

A partir da introdução dos feriados, utilizou-se sistema binário para

Código 3 - Grade Horária com Audiência (2020-2022).

```
#junta grade horária dos dias úteis com o final de semana
gradeHoraria1 = pd.concat([grade_horaria_domingo,grade_horaria_sabado,grade_horaria_seg_sex], axis
= 0)
#faz com o que todos os dados da coluna "data" sejam convertidas datetime
gradeHoraria1 = gradeHoraria1.drop_duplicates()
gradeHoraria1[['Programa','Categoria']] = gradeHoraria1['Emissora 1'].str.split('/',expand=True)
gradeHoraria1 = gradeHoraria1.drop(['Emissora 1','Emissora 2','Emissora 3','Praça','Unnamed:
0'],axis = 1)
gradeHoraria1 = gradeHoraria1.sort_values(by=['Data','Faixa Horária'], ascending=True)
```

Fonte: do próprio autor (2022).

Código 4 - Grade Horária com Audiência (2015 - 2019).

```
grade_horaria_2015[['Programa','Categoria']] = grade_horaria_2015['NOME DA EMISSORA
1'].str.split('/',expand=True)
grade_horaria_2015 = grade_horaria_2015.drop(['NOME DA EMISSORA 1','NOME DA EMISSORA 2','NOME DA
EMISSORA 3','Praça','Unnamed: 0'],axis = 1)
grade_horaria_2015 = grade_horaria_2015.sort_values(by=['Data','Faixa Horária'], ascending=True)
```

Fonte: do próprio autor (2022).

[1] No momento de rodar o código deve-se colocar o link do caminho do arquivo na segunda linha: feriados = pd.read_csv('caminho do arquivo'].

Os dois códigos acima se referem ao mesmo processo de junção, porém como os dados de 2015 e de 2020 não estavam juntos, foi preciso fazer o processo separado para depois juntá-los.[2]

Para o modelo preditivo, **codes27**, o ideal era ter as informações sobre a grade horária, com todos os programas e suas respectivas categorias. Por isso todos os dados foram agregados, com dias úteis e finais de semana, no mesmo *dataset*.

Porém surgiu o seguinte problema: os programas e as categorias estavam na mesma coluna, porém informações precisavam estar separadas, por isso, na quinta linha, utilizamos o comando `str.split('/',expand=True)` que separa os dados que estão com "/" entre eles (**Código 4**).

Como *dataset* continham informações que não eram necessárias para o modelo preditivo, essas foram retiradas pelo seguinte comando: `gradeHoraria1.drop`.

Código 5 - Junção das Grades Horárias.

```
gradeHoraria = pd.concat([grade_horaria_2015,grade_horaria1])
```

Fonte: do próprio autor (2022).

Para dar continuidade no modelo preditivo, foi necessário juntar as tabelas citadas acima (`[grade_horaria_2015,grade_horaria1]`), por meio do comando `.concat`, como mostra o código acima (**Código 5**).

4.3.1.2. Etapa de Anonimização

Para essa etapa foi utilizado o CSV de Grade Horária (segunda a sexta, sábado e domingo). Por questões contratuais foi necessário anonimizar essas tabelas, como por exemplo os nomes das emissoras. Isso não foi feito em código pois era de extrema importância a preservação da identidade das emissoras no próprio código, como foi pedido.

Código 6 - Anonimização dos programas

```
#Ajustes no dataset da emissora 1 de segunda a sexta, sábado e domingo
dataSet = dataSet.drop(['Programa'], axis=1)
```

Fonte: do próprio autor (2022).

Como mostra o código 6, foi excluída a coluna "Programas" das tabelas de cada emissora. Para isso, foi utilizado o método "**`.drop()`**" da biblioteca Pandas, assim anonimizando os nomes dos programas das emissoras 1, 2 e 3.

2 No código 4 os nomes das emissoras foram anonimizados neste documento por questões contratuais.

4.3.1.3. Etapa de Label Encoding

Código 7 - Transformação das categorias em label encoding.

```
#dicionário para podermos manipular os dados da coluna categoria no pandas
dict_categoria = {'AUDITORIO' : '1', 'CARROS E MOTORES' : '2', 'CULINARIO' : '3', 'DEBATE' :
'4', 'DOCUMENTARIO' : '5', 'EDUCATIVO' : '6', 'ENTREVISTA' : '7',  'ESPORTE' : '8', 'FEMININO'
: '9', 'FILME' : '10', 'FUTEBOL' : '11', 'GAME SHOW' : '12', 'HUMORISTICO' : '13', 'JORNALISMO'
: '14','MINISSERIE' : '15', 'MUSICAL' : '16', 'NAO CONSTA' : '17','NOVELA' : '18', 'OUTROS' :
'19', 'POLITICO' : '20', 'PREMIACAO' : '21' 'REALITY SHOW' : '22', 'RELIGIOSO' : '24',
'REPORTAGEM' : '25',  'RURAL' : '26','SERIES' : '27', 'SHOW' : '28', 'SORTEIO' : '29', 'TELE
VENDAS' : '30', }
#Muda todos os nomes das categorias por números, pré definidos no dicionário acima
clean_merged.Categoria = clean_merged.Categoria.replace(dict_categoria)
```

Fonte: do próprio autor (2022).

No código mostrado acima, os nomes das categorias foram substituídos por números de 1 a 30, para facilitar o modelo preditivo. Para isso, foram listadas todas as categorias e assim direcionadas com o seu respectivo número, com o seguinte formato: `'Nome da Categoria' : 'Número da categoria'`

Durante a análise de dados, foi notado que tinham alguns programas que acabavam retornando dados viciados, já que esses programas não seguem nenhum padrão específico. Por isso foi tomada a seguinte decisão: uma categoria específica seria criada para cada programa que tivesse essas características. Esse processo foi realizado com cerca de 10 programas, o código continua o mesmo.

Código 8 - Transformação das categorias em label encoding.

```
#faz com que o programa se torne um número para podermos mudar sua categoria
dict_realities = { 'NOME DO PROGRAMA'[3] : NÚMERO DA CATEGORIA}
#muda na tabela o valor do programa
clean_merged.Programa = clean_merged.Programa.replace(dict_realities)
#faz com que todas linha que o programa tem um número específico mude sua categoria para esse
número
clean_merged.loc[clean_merged['Programa'] = NÚMERO DA CATEGORIA, 'Categoria'] = NÚMERO DA
CATEGORIA
```

Fonte: do próprio autor (2022).

4.3.1.4.Etapa de agregação:

Para conseguir utilizar as informações presentes na coluna "Data" foi necessário dividi-la em três novas colunas: "Ano", "Mês" e "Dia, utilizando o método "`.strftime()`". Separando-as pelo valor correspondente a cada uma e criando três novas colunas para agrupar essas informações separadamente.

Código 9 - Separação dia, mês e ano

[3] No espaço escrito `'NOME DO PROGRAMA'` , deve-se escrever o título daquele programa, e `'NÚMERO DO PROGRAMA'` deve-se escrever o número escolhido.

```
#Separa a coluna data em 3 colunas "Ano, mes e dia"
clean_merged['Dia'] = pd.to_datetime(clean_merged['Data']).dt.strftime('%d')
clean_merged['Mes'] = pd.to_datetime(clean_merged['Data']).dt.strftime('%m')
clean_merged['Ano'] = pd.to_datetime(clean_merged['Data']).dt.strftime('%Y')
```

Fonte: do próprio autor (2022).

TEXTO EXPLICANDO O 10

Código 10 - Separação hora, minuto e segundo

```
#Separa a coluna data em 3 colunas "Ano, mes e dia"
df[['Hora', 'Minuto', 'Lixo']] = df['Hora Início'].str.split('%d'),expan=True)
df[['Hora']] = df['Hora'].apply(lambda x: int(x))
df[['Minuto']] = df['Minuto'].apply(lambda x: int(x)/60)
df[['Hora Início']] = df.loc[:,['Hora','Minuto']].sum(axis=1)
df = df.drop(['Hora', 'Minuto', 'Lixo'], axis = 1)
```

Fonte: do próprio autor (2022).

4.3.1.5.Etapa de exclusão:

Essa etapa foi bem simples, pois era necessário retirar a coluna que mostrava a emissora, por motivos legais, e a coluna "Unnamed: 0" que o próprio Colab cria e não será utilizada.

Código 11 - Exclusão de colunas

```
#Retira as colunas das quais nao vamos usar
dataSet = dataSet.drop(['Unnamed: 0','Emissora'], axis=1)
```

Fonte: do próprio autor (2022).

4.3.1.6.Etapa de One Hot Encoding:

A técnica "One Hot Coding" consiste em transformar os dados da coluna em diversas colunas contendo 0 e 1, fazendo com que os dados fiquem mais apropriados para serem usados no modelo preditivo.

Código 12 - One Hot Coding - Categoria

```
#Importa as bibliotecas
from sklearn.preprossing import LabelEncoder
#cria uma lista com os dados que iremos usar
categorias = ['Categoria']
#Mapeamento dos valores
le = LabelEncoder()
test = pd.get_dummies(clean_merged[categorias], drop_first=True)
test = test.reset_index(drop=True)
```

Fonte: do próprio autor (2022).

EXPLICAR CÓDIGO 13

Código 13 - One Hot Coding - Dia da semana

```
dia_semana = ['Dia da Semana']
#Mapeamento dos valores
ds_hoc = pd.get_dummies(df[dia_semana], drop_first=True)
```

Fonte: do próprio autor (2022).

4.3.2. Como deve ser feita a agregação de registros e/ou derivação de novos atributos

Para serem realizadas as derivações dos atributos na ferramenta, foi utilizada a função *Split* do Pandas, a qual é responsável por dividir uma *string* em pequenos pedaços utilizando um "separador", que pode ser escolhido no código. Foi necessário realizar algumas manipulações nos registros disponibilizados.

A primeira manipulação foi na coluna "Programa/Categoria". Como dito anteriormente, era necessário realizar essa divisão para: 1. Seguir as questões contratuais, ou seja, os programas deveriam estar anonimizados para não ser possível a identificação da emissora. 2. Como a empresa gostaria de prever programas futuros, deveria ser listada todas as categorias separadas. Para isso foi utilizado a função citada acima, separando em duas colunas diferentes. Com isso, agregamos somente a coluna "Categoria" no modelo preditivo.

A segunda manipulação foi na coluna "Data". Os dados recebidos tinham um único problema: a coluna em questão, estava no formato "datetime", e o modelo não consegue ler esse tipo de dado. Por isso, tivemos que transformá-lo para números inteiros, sendo assim, cada um se tornou uma coluna: "dia", "mês" e "ano".

Fluxograma 2 - Etapas da elaboração do projeto.

Fonte: do próprio autor (2022).

Segue aqui uma explicação de cada etapa do fluxograma acima.

CSV: Essa etapa consiste em: 1. Transformar o arquivo XLSX, fornecido pelo cliente, para CSV, por meio de um código no Colab. 2. Extrair da tabela os dados sobre os últimos 7 anos da emissora.

Dados: A partir desses dados, focando nos últimos dois anos, foram realizadas diversas análises. Primeiramente, no próprio Sheets, com gráficos que mostravam os picos de audiência, categorizados pelas colunas fornecidas (classes sociais, idade, gênero). Em segundo, no Colab, onde foi calculado a média, desvio padrão, variância e moda. Todas essas análises geraram gráficos que estão nesse documento.

Extração: Agora deve-se extrair os dados que interessa nesse projeto (Emissora 1), para assim o modelo rodar certo.

Transformação: Todas as manipulações citadas no ponto 4.3.1, foram as utilizadas para o tratamento de dados. A primeira foi a junção dos dados de 2015-2019 e os dados de 2020-2022. Depois alguns dados foram anonimizados, para assim, serem transformados em Label Encoding. Após esses processos, foi necessário fazer agregações de colunas na tabela inicial, como por exemplo "Dia". Da mesma forma, era essencial excluir colunas repetidas. E, por último, alguns dados foram transformados para One Hot Coding.

Dataset: O DataSet são os dados que serão utilizados para rodar os modelos. Sendo um repositório central de informações que vão poder ser analisadas para tomar decisões corretas.

Processamento: Nesta etapa era preciso rodar alguns modelos que acreditávamos que seria o melhor para o modelo preditivo. Para isso, foram testados 3 modelos: KNN, LightGBM e Regressão Linear, todos eles são modelos de regressão.

Análise: É de extrema importância comparar os 3 modelos que foram rodados, para assim, poder escolher o mais adequado para essa situação. Para isso, deve se gerar gráficos comparando o R^2, erro médio quadrático, erro médio absoluto, desvio padrão e acurácia.

Predição de audiência: Idealmente, de acordo com os dados que serão inseridos, o modelo irá prever a audiência daquele programa,no horário e data escolhida.

4.3.3. Identificação das features selecionadas, com descrição dos motivos de seleção.

Inicialmente, foi feita uma seleção das features de acordo com o consenso do grupo, em relação às quais melhores cabiam para o modelo que seria utilizado. Pensando naquelas que mais agregavam e poderiam ter impacto no output da predição, optou-se por manter as features:

Tabela 2 - Features escolhidas.

Dados gerais	Rat%	Fid%	Shr%
Data	Total Domicílios	Rat%	Total Domicílios	Fid%	Total Domicílios	Shr%

Hora de início	AB I Rat%	AB I Fid%	AB I Shr%
Dia da semana	C1 I Rat%	C1 I Fid%	C1 I Shr%
Programa	C2 I Rat%	C2 I Fid%	C2 I Shr%
Categoria	DE I Rat%	DE I Fid%	DE I Shr%
Feriado	Masculino I Rat %	Masculino I Fid%	Masculino I Shr%
Ano	Feminino I Rat %	Feminino I Fid%	Feminino I Fid%
Dia	4-11 anos I Rat%	4-11 anos I Fid%	4-11 anos I Shr%
Mês	12-17 anos I Rat%	12-17 anos I FId%	12-17 anos I Shr%
	18-24 anos I Rat%	18-24 anos I Fid%	18-24 anos I Shr%
	25-34 anos I Rat%	25-34 anos I Fid%	25-34 anos I Shr%
	35-49 anos I Rat%	35-49 anos I Fid%	35-49 anos I Shr%
	50-59 anos I Rat%	50-59 anos I Fid%	50-59 anos I Shr%
	60+ anos I Rat%	60+ anos I Fid%	60+ anos I Shr%

Fonte: Elaborada pelos Autores (2022).

Como explicado anteriormente, o Rat% é a medida média de audiência, o Share (Shr%) é a participação dela em um evento, de um período específico, e a fidelidade (Fid%) é a permanência naquele programa. Usamos essas medidas no modelo preditivo, pois de acordo com a análise, foi constatado que são esses dados influenciam de maneira mais severa a predição, tornando-a mais precisa.

Também considerou-se a relevância do espaço/tempo em que o programa está encaixado, de acordo com o horário, dia, mês, ano, dia da semana e se esse é feriado ou não. Há uma mudança nos pontos atingidos e do perfil de telespectadores, podendo se estabelecer uma correlação direta entre essas variáveis e o resultado final.

Além disso, observou-se que os segmentos que cada programa possuía, afetavam o desempenho no *score* de audiência das emissoras, juntamente com o perfil do público. Dessa forma foi considerado cada um dos segmentos referentes às faixas horárias medidas. Ademais, após as avaliações, tornou-se claro a relevância de dados sobre o público que estava assistindo, uma vez que esses se mostravam os que mais sofriam alterações de acordo com as demais variáveis.

Neste sentido, após a escolha das features, foi feita uma análise do impacto de cada variável a partir do conceito de multicolinearidade, ou seja, conjunto de características. Foi considerado duas delas que carregam as mesmas ou similares informações, que possuem relação linear muito forte. Como consequência, em casos assim, o modelo pode acabar ignorando uma das características, tomando elas como redundantes.

Para conseguir obter os resultados, foram utilizadas as bibliotecas *numpy*, *pandas*, *seaborn* e *matplotlib.pylab* disponíveis para a linguagem Python, a partir do código abaixo:

Código 14 - Código para gerar o mapa de multicolinearidade.

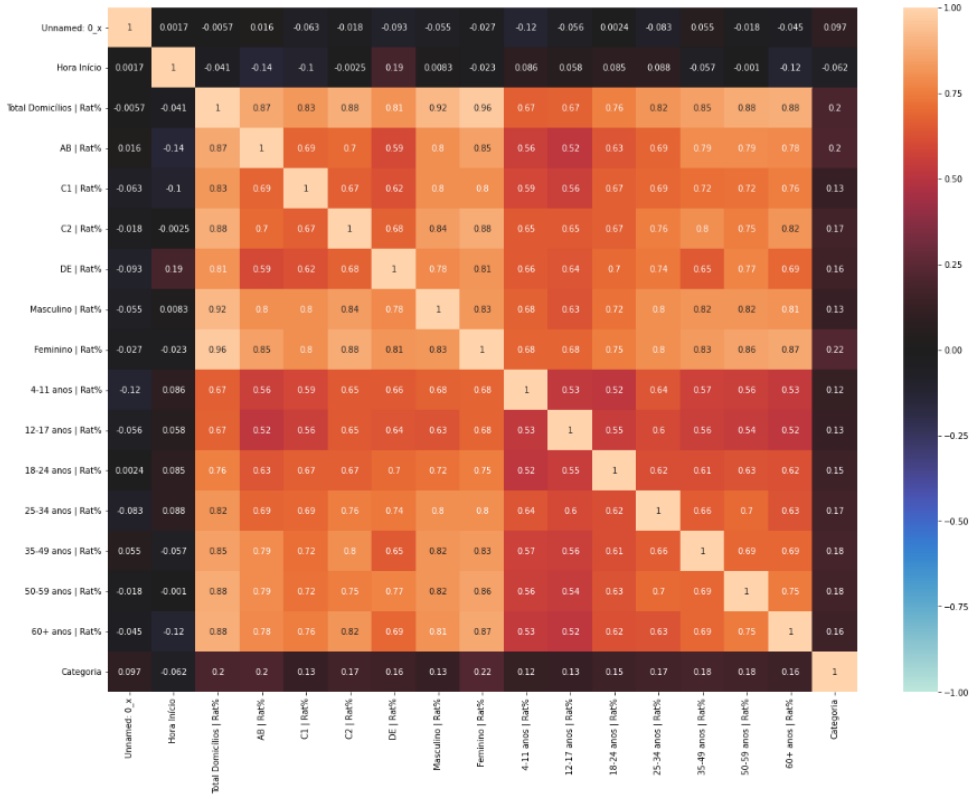
```
# Importação da biblioteca álgebra linear
import numpy as np
# Manipulação dos dados
import pandas as pd
# Visualização
import seaborn as sns
import matplotlib.pylab as plt
from matplotlib.pylab import rcParams

# Definindo tamanho da figura
rcParams['figure.figsize'] = 20, 15
# Matriz de correlação
matriz_correlacao = clean_merged.corr()
# Mapa de calor
sns.heatmap(matriz_de_correlacao, annot = True, vmin =-1, vmax=1, center=0)
# Definindo a posição dos tricks nos eixos
plt.yticks(rotation = 360)
plt.xticks(rotation = 90)
# Mostrando a figura
plt.show()
```

Fonte: Elaborada pelos Autores (2022).

Dessa forma, foi observado com a saída da execução, quais das features estariam com dados mais correlacionados, sendo as cores mais claras as associações mais similares, conforme observado no mapa abaixo.

Figura 20 - Mapa de multicolinearidade.



Fonte: Elaborada pelos Autores (2022).

4.4. Modelagem

Na modelagem dos dados foram utilizados 3 modelos para o teste de predição, são eles: KNN, Light GBM e a Regressão Linear, possuindo resultados diferentes em cada um deles.

4.4.1 Regressão Linear

No que tange a **regressão linear**, ela se apresenta como um modelo mais simples dentre todos os testados, por representar um modelo simples de *machine learning* para encontrar a reta que melhor explica a relação entre as *features* e um *target* contínuo, que, em linhas gerais, aproxima os dois por uma reta (uma função linear). Desse modo, podemos fazer afirmações sobre valores não disponíveis no *dataset*. Então, para sumarizar o processo, as tabelas e bibliotecas foram importadas para a manipulação dos dados:

Código 15 - Importação das bibliotecas e arquivos necessários.

```
#importando bibliotecas
import matplotlib.pyplot as plt
import pandas as pd
import datetime as dt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import *
from google.colab import drive
```

```
#montando o drive
drive.mount('/content/drive')

#importando tabela completa emissora 1
data = pd.read_csv("CAMINHO DO ARQUIVO", sep = ',')
data = data.drop(['Data','Unnamed: 0'], axis=1)
data[data['Feriado']== 1]
data.columns
```

Fonte: Elaborada pelos Autores (2022).

Observação: foi colocado "caminho do arquivo" no código pois foi retirado do drive dos autores.

Após isso, nós selecionamos quais seriam as variáveis utilizadas para o modelo preditivo e descartamos as outras e treinamos o modelo com o método `LinearRegression` da biblioteca do *Scikit-learn*:

Código 16 - Selecionando os inputs e outputs.

```
y = data[['Total Domicílios | Rat%','AB | Rat%','C1 | Rat%','C2 | Rat%','DE | Rat%','Masculino
| Rat%','Feminino | Rat%','4-11 anos | Rat%','12-17 anos | Rat%','18-24 anos | Rat%','25-34
anos | Rat%','35-49 anos | Rat%','50-59 anos | Rat%','60+ anos | Rat%']]
#remoção do output desejado
data.drop(['Total Domicílios | Rat%'], axis=1, inplace=True)
data.drop(['Total Indivíduos | Fid%'], axis=1, inplace=True)
data.drop(['Total Domicílios | Shr%'], axis=1, inplace=True)
data.drop(['Masculino | Rat%'], axis=1, inplace=True)
data.drop(['Masculino | Fid%'], axis=1, inplace=True)
data.drop(['Masculino | Shr%'], axis=1, inplace=True)
data.drop(['Feminino | Rat%'], axis=1, inplace=True)
data.drop(['Feminino | Fid%'], axis=1, inplace=True)
data.drop(['Feminino | Shr%'], axis=1, inplace=True)
data.drop(['AB | Rat%'], axis=1, inplace=True)
data.drop(['AB | Fid%'], axis=1, inplace=True)
data.drop(['AB | Shr%'], axis=1, inplace=True)
data.drop(['C1 | Rat%'], axis=1, inplace=True)
data.drop(['C1 | Fid%'], axis=1, inplace=True)
data.drop(['C1 | Shr%'], axis=1, inplace=True)
data.drop(['C2 | Rat%'], axis=1, inplace=True)
data.drop(['C2 | Fid%'], axis=1, inplace=True)
data.drop(['C2 | Shr%'], axis=1, inplace=True)
data.drop(['DE | Rat%'], axis=1, inplace=True)
data.drop(['DE | Fid%'], axis=1, inplace=True)
data.drop(['DE | Shr%'], axis=1, inplace=True)
data.drop(['4-11 anos | Rat%'], axis=1, inplace=True)
data.drop(['4-11 anos | Fid%'], axis=1, inplace=True)
data.drop(['4-11 anos | Shr%'], axis=1, inplace=True)
data.drop(['12-17 anos | Rat%'], axis=1, inplace=True)
data.drop(['12-17 anos | Fid%'], axis=1, inplace=True)
data.drop(['12-17 anos | Shr%'], axis=1, inplace=True)
data.drop(['18-24 anos | Rat%'], axis=1, inplace=True)
data.drop(['18-24 anos | Fid%'], axis=1, inplace=True)
data.drop(['18-24 anos | Shr%'], axis=1, inplace=True)
data.drop(['25-34 anos | Rat%'], axis=1, inplace=True)
data.drop(['25-34 anos | Fid%'], axis=1, inplace=True)
data.drop(['25-34 anos | Shr%'], axis=1, inplace=True)
data.drop(['35-49 anos | Rat%'], axis=1, inplace=True)
data.drop(['35-49 anos | Fid%'], axis=1, inplace=True)
```

```
data.drop(['35-49 anos | Shr%'], axis=1, inplace=True)
data.drop(['50-59 anos | Rat%'], axis=1, inplace=True)
data.drop(['50-59 anos | Fid%'], axis=1, inplace=True)
data.drop(['50-59 anos | Shr%'], axis=1, inplace=True)
data.drop(['60+ anos | Rat%'], axis=1, inplace=True)
data.drop(['60+ anos | Fid%'], axis=1, inplace=True)
data.drop(['60+ anos | Shr%'], axis=1, inplace=True)
x = data
```

Fonte: Elaborada pelos Autores (2022).

No código seguinte, foi dividido o treino do teste, e somente assim pode-se treinar o modelo em questão.

Código 17 - Treinamento e teste do modelo de regressão linear.

```
#dividindo treino e teste

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.15,  random_state =
100000)

#treinamento do modelo

model = LinearRegression().fit(x_train,y_train)


x_test
```

Fonte: Elaborada pelos Autores (2022).

Então, foi feito um teste do modelo preditivo utilizando o método "*model.predict()*" da biblioteca do *Scikit-learn*, foi verificado quais foram os resultados e assim, computado o R^2 com o método `_score(y_test, y_pred)` para verificar a acuracidade do modelo.

Código 18 - Teste e avaliação da predição da regressão linear.

```
#Predição usando a regressão linear
y_pred = model.predict(x_test)
y_pred

# R quadrado
r2_score(y_test, y_pred)
```

Fonte: Elaborada pelos Autores (2022).

A equação a seguir descreve como se deve calcular o R^2:

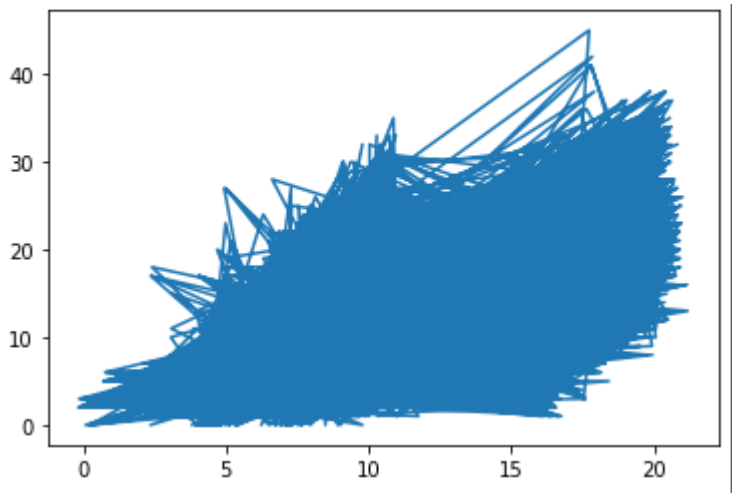
Equação 1 - Cálculo do R^2.

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Fonte: Medium

A seguir, o gráfico gerado pelo modelo preditivo:

Figura 21 - Gráfico resultante do mode



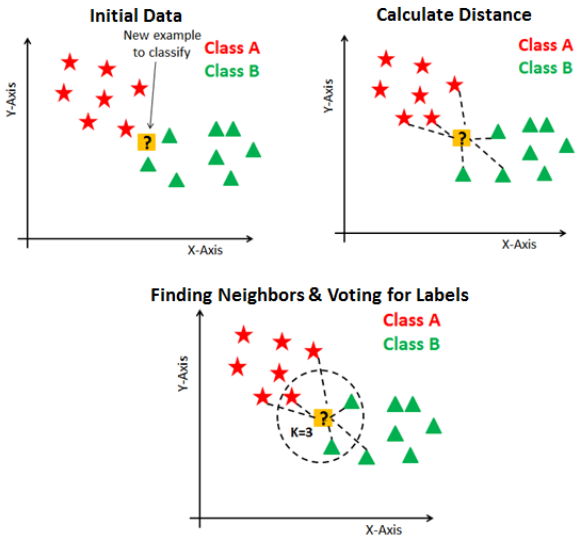
Fonte: Elaborada pelos Autores (2022).

Com esse gráfico pode-se chegar à conclusão de que tal modelo teve pouca acurácia, uma vez que a formação do gráfico se distancia muito do formato de uma reta, que, no caso, seria o modelo ideal.

4.4.2 K-Nearest Neighbors (KNN)

Já em outro teste realizado, foi utilizado o modelo **K-Nearest Neighbors (KNN)**, ou métodos vizinhos mais próximos, que consiste em uma ferramenta de *machine learning* supervisionado, que utiliza ferramentas matemáticas para comparar dados semelhantes entre si com o objetivo de inferir uma determinada característica sobre um data point desconhecido.

Figura 22 - Funcionamento do K-Nearest Neighbors (KNN).



Fonte: Inferir

Então, para expor o processo, as tabelas e as bibliotecas foram importadas para a manipulação dos dados:

Código 19 - Importação das bibliotecas e arquivos.

```
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
#------------------------ IMPORTAÇÃO DE TABELAS ----------------------------
from google.colab import drive
drive.mount('/content/drive')
dataset = pd.read_csv('CAMINHO DO ARQUIVO')
dataset.head()
dataset.columns
```

Fonte: Elaborada pelos Autores (2022).

Observação: foi colocado “caminho do arquivo” no código pois foi retirado do drive dos autores.

Após isso, os outputs desejados foram separados:

Código 20 - Definição dos inputs e outputs.

```
dataset_train = dataset.sample(len(dataset))
#remoção do output desejado
dataset_train.drop(['Total Domicílios | Rat%'], axis=1, inplace=True)
dataset_train.drop(['Masculino | Rat%'], axis=1, inplace=True)
dataset_train.drop(['Feminino | Rat%'], axis=1, inplace=True)
dataset_train.drop(['AB | Rat%'], axis=1, inplace=True)
dataset_train.drop(['C1 | Rat%'], axis=1, inplace=True)
dataset_train.drop(['C2 | Rat%'], axis=1, inplace=True)
dataset_train.drop(['DE | Rat%'], axis=1, inplace=True)
dataset_train.drop(['4-11 anos | Rat%'], axis=1, inplace=True)
dataset_train.drop(['12-17 anos | Rat%'], axis=1, inplace=True)
dataset_train.drop(['18-24 anos | Rat%'], axis=1, inplace=True)
dataset_train.drop(['25-34 anos | Rat%'], axis=1, inplace=True)
dataset_train.drop(['35-49 anos | Rat%'], axis=1, inplace=True)
dataset_train.drop(['50-59 anos | Rat%'], axis=1, inplace=True)
dataset_train.drop(['60+ anos | Rat%'], axis=1, inplace=True)
dataset_train.drop(['Data'], axis=1, inplace=True)
dataset_train.columns

#Definição de output
y = dataset['Total Domicílios | Rat%']
#Definição de input
X_train = dataset_train
#Definição de teste
X_train, X_test, y_train, y_test = train_test_split(X_train, y, test_size=0.33,
random_state=1000)
```

Fonte: Elaborada pelos Autores (2022).

Então, foi realizada a divisão dos eixos do modelo, separando os grupos de treino e de teste. Assim, rodando o algoritmo através do método *KNeighborsClassifier*(), verificando a acuracidade do modelo, a qual foi muito baixa no treino (99,6% de acuracidade) e no teste (99,02% de acuracidade).

Código 21 - Teste e treinamento do modelo KNN.

```
#Definição de output
y = dataset['Total Domicílios | Rat%']

#Definição de input
X_train = dataset_train

#Definição de teste
X_train, X_test, y_train, y_test = train_test_split(X_train, y, test_size=0.33,
random_state=1000)

# Instanciação do Algoritmo
knn = KNeighborsClassifier(n_neighbors=7)

# Treino,na qual x = Features, y = Label/Target

knn.fit(X_train, y_train.squeeze()) # squeeze() -> df para series
# Teste de Acuracidade (accuracy)
print('Acuracidade (treino): ', knn.score( X_train, y_train ))
print('Acuracidade (teste): ', knn.score( X_test, y_test ))
```

```
# realizando predições com o conjunto de teste
y_pred = knn.predict(X_test)
y_pred
```

Fonte: Elaborada pelos Autores (2022).

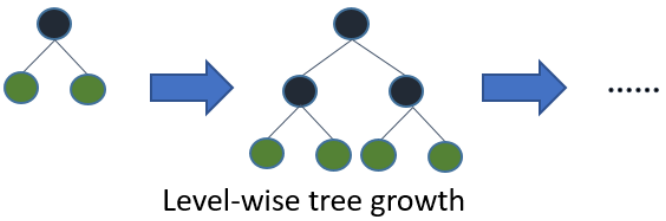
4.4.3 LightGBM

O teste com melhor resultado, foi o **LightGBM**, que consiste em um método baseado em *gradient boosting*, o qual é usualmente utilizado para modelos de regressão e classificação. Ela constrói o modelo em etapas, como outros métodos de *boosting*, e os generaliza, permitindo a otimização de uma função de perda diferenciável arbitrária.

LightGBM é um sistema o qual distribui a estrutura de aumento de gradiente usando uma estrutura de aprendizagem baseada em árvore, sendo assim, baseado em histograma, colocando valores contínuos em compartimentos discretos, o que leva a um treinamento mais rápido e um uso mais eficiente da memória.

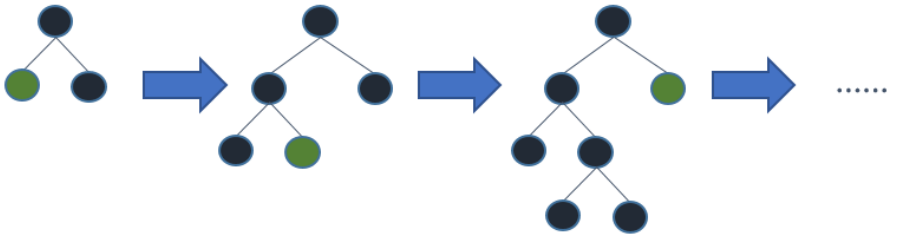
A estrutura usa um algoritmo de crescimento de árvore em folha que é diferente de muitos outros algoritmos baseados em árvore, os quais usam crescimento em profundidade. Os algoritmos de crescimento de árvores em folhas tendem a convergir mais rapidamente que os em profundidade. No entanto, eles tendem a ser mais propensos a sobreajuste, como pode ser percebido no gráfico abaixo a diferença entre as duas supracitadas:

Figura 23 - Crescimento de árvore em nível.



Fonte: Zephyrnet (2022).

Figura 24 - Crescimento de árvore em folha.



Fonte: Stack Exchange (2022).

Primeiramente, as bibliotecas e o arquivo CSV necessário foram importados para o modelo preditivo, além de conectar o Google Colab ao Drive em que os arquivos necessários estavam arquivados.

Código 22 - Importação de bibliotecas e de arquivos

```
#importação de bibliotecas
from google.colab import drive
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

#Acesso ao Google Drive
drive.mount('/content/drive')
pd.set_option('display.max_columns', None)

# csv
df_teste =
pd.read_csv('CAMINHO DO ARQUIVO')
df_teste
```

Fonte: Elaborada pelos Autores (2022).

Observação: foi colocado "caminho do arquivo" no código pois foi retirado do drive dos autores.

No código abaixo, os dados foram separados entre inputs e outputs.

Código 23 - Remoção das colunas que vão ser os Outputs.

```
#dataframes
data = df_teste.sample(len(df_teste))
#definindo o y
y = data['Total Domicílios | Rat%']
data.drop(['Total Domicílios | Rat%'], axis=1, inplace=True)
data.drop(['Total Indivíduos | Fid%'], axis=1, inplace=True)
data.drop(['Total Domicílios | Shr%'], axis=1, inplace=True)
data.drop(['Masculino | Rat%'], axis=1, inplace=True)
data.drop(['Masculino | Fid%'], axis=1, inplace=True)
data.drop(['Masculino | Shr%'], axis=1, inplace=True)
data.drop(['Feminino | Rat%'], axis=1, inplace=True)
data.drop(['Feminino | Fid%'], axis=1, inplace=True)
data.drop(['Feminino | Shr%'], axis=1, inplace=True)
```

```
data.drop(['AB | Rat%'], axis=1, inplace=True)
data.drop(['AB | Fid%'], axis=1, inplace=True)
data.drop(['AB | Shr%'], axis=1, inplace=True)
data.drop(['C1 | Rat%'], axis=1, inplace=True)
data.drop(['C1 | Fid%'], axis=1, inplace=True)
data.drop(['C1 | Shr%'], axis=1, inplace=True)
data.drop(['C2 | Rat%'], axis=1, inplace=True)
data.drop(['C2 | Fid%'], axis=1, inplace=True)
data.drop(['C2 | Shr%'], axis=1, inplace=True)
data.drop(['DE | Rat%'], axis=1, inplace=True)
data.drop(['DE | Fid%'], axis=1, inplace=True)
data.drop(['DE | Shr%'], axis=1, inplace=True)
data.drop(['4-11 anos | Rat%'], axis=1, inplace=True)
data.drop(['4-11 anos | Fid%'], axis=1, inplace=True)
data.drop(['4-11 anos | Shr%'], axis=1, inplace=True)
data.drop(['12-17 anos | Rat%'], axis=1, inplace=True)
data.drop(['12-17 anos | Fid%'], axis=1, inplace=True)
data.drop(['12-17 anos | Shr%'], axis=1, inplace=True)
data.drop(['18-24 anos | Rat%'], axis=1, inplace=True)
data.drop(['18-24 anos | Fid%'], axis=1, inplace=True)
data.drop(['18-24 anos | Shr%'], axis=1, inplace=True)
data.drop(['25-34 anos | Rat%'], axis=1, inplace=True)
data.drop(['25-34 anos | Fid%'], axis=1, inplace=True)
data.drop(['25-34 anos | Shr%'], axis=1, inplace=True)
data.drop(['35-49 anos | Rat%'], axis=1, inplace=True)
data.drop(['35-49 anos | Fid%'], axis=1, inplace=True)
data.drop(['35-49 anos | Shr%'], axis=1, inplace=True)
data.drop(['50-59 anos | Rat%'], axis=1, inplace=True)
data.drop(['50-59 anos | Fid%'], axis=1, inplace=True)
data.drop(['50-59 anos | Shr%'], axis=1, inplace=True)
data.drop(['60+ anos | Rat%'], axis=1, inplace=True)
data.drop(['60+ anos | Fid%'], axis=1, inplace=True)
data.drop(['60+ anos | Shr%'], axis=1, inplace=True)
data.drop(['Data'], axis=1, inplace=True)
```

Fonte: Elaborada pelos Autores (2022).

A separação dos dados foi realizada entre: 1.Os que seriam utilizados para treino através do método "*train_test_split()*", onde 85% dos dados foram utilizados para o treinamento do algoritmo, e 2. Os 15% restantes foram utilizados para o teste desse segundo descrito na parte *"test_size=0.15"*. Além disso, um dicionário foi criado para os parâmetros utilizados no modelo.

Código 24 - Divisão dos dados

```
#aqui estamos dividindo os dados que serão de treinamento e os dados que serão de teste
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data, y, test_size=0.15, random_state=42)

#dicionário dos parâmetros para o modelo
hyper_params = {
    'task': 'train', #função de treino
    'boosting_type': 'gbdt', #regressão lightgbm
    'objective': 'regression', #modelo de regressão
    'metric': ['l1','l2'], #métricas (erro médio quadrático e absoluto)
    'learning_rate': 0.08, #velocidade que o modelo aprende
    'feature_fraction': 1, #porcentagem do dataframe
    'verbose': 0, #debug
    'max_depth': 8, #tamanho de cada árvore
    'num_leaves': 128, #número de folhas
    'num_iterations': 20_000 #quantidade de tentativas do modelo
}
```

Fonte: Elaborada pelos Autores (2022).

A biblioteca do modelo preditivo utilizado foi importada e, assim, o modelo é criado. Após isso, os dados que foram trabalhados no modelo foram inseridos.

Código 25 - Criação do modelo e inserção dos dados

```
#criação do modelo
import lightgbm as lgb
gbm = lgb.LGBMRegressor(**hyper_params)
#inserindo os dados
gbm.fit(X_train, y_train,
        eval_set=[(X_test, y_test)],
        early_stopping_rounds=2_500)
```

Fonte: Elaborada pelos Autores (2022).

O método *score*() e a função *print*() do Python foi utilizada para verificar a acurácia do treino e do teste do modelo para, assim, decidir se valeria a pena usar aquele algoritmo. O resultado é demonstrado no final do código, se iniciando com "> > >".

Código 26 - Testes de acurácia

```
#Acurácia do treino
train_score = gbm.score(X_train,y_train)

#Acurácia do teste
test_score = gbm.score(X_test,y_test)

print('Training accuracy {:.4f}'.format(train_score)) #acurácia do treino
print('Testing accuracy {:.4f}'.format(test_score)) #acurácia do teste

>>> 0.9965598767625932
>>> 0.9874461885379952
>>> Training accuracy 0.9966
>>> Testing accuracy 0.9874
```

Fonte: Elaborada pelos Autores (2022).

No código abaixo, foi utilizado o método *predict*() e a função *print* do Python para verificar quais são os valores retornados pelo modelo. O resultado é demonstrado no final do código, se iniciando com "> > >".

Código 27 - Predições do treino e do teste

```
# Output do treino - predição
y_pred_train = gbm.predict(X_train, num_iteration=gbm.best_iteration_)
print(y_pred_train)

# Output do teste - predição
y_pred_test = gbm.predict(X_test, num_iteration=gbm.best_iteration_)
print(y_pred_test)

>>> [ 6.40760118 14.06741035  9.92614251 ...  6.5809714   9.16186486
  2.95479864]
[16.54097181  7.34349707 13.72431957 ... 10.04164805 11.28992151
 19.53877414]
>>> [16.54097181  7.34349707 13.72431957 ... 10.04164805 11.28992151
```

```
19.53877414]
```

Fonte: Elaborada pelos Autores (2022).

Algumas métricas foram importadas para a verificação da qualidade do modelo, verificando se o algoritmo não se encaixava em um caso de *overfitting* ou *underfitting*. Para isso, nós utilizamos os métodos "*r2_score()*", "*mean_absolute_error()*" e "*mean_squared_error()*", onde bons resultados foram apontados. O resultado é demonstrado no final do código, se iniciando com "> > >".

Código 28 - Métricas de avaliação do modelo

```
#R²
from sklearn import metrics
from sklearn.metrics import r2_score
print(metrics.r2_score(y_test, y_pred_test))

#erro médio absoluto
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_train, y_pred_train)

#erro médio quadrático
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred_test)

>>> 0.9874461885379952
>>> 0.28462998616900426
>>> 0.5340584656505355
```

Fonte: Elaborada pelos Autores (2022).

Para melhorar a visualização, um gráfico e uma tabela foram gerados, o segundo foi necessário para comparar os valores reais e obtidos. Para isso, foi utilizado o método *plot*().

Código 29 - Gráfico e tabela

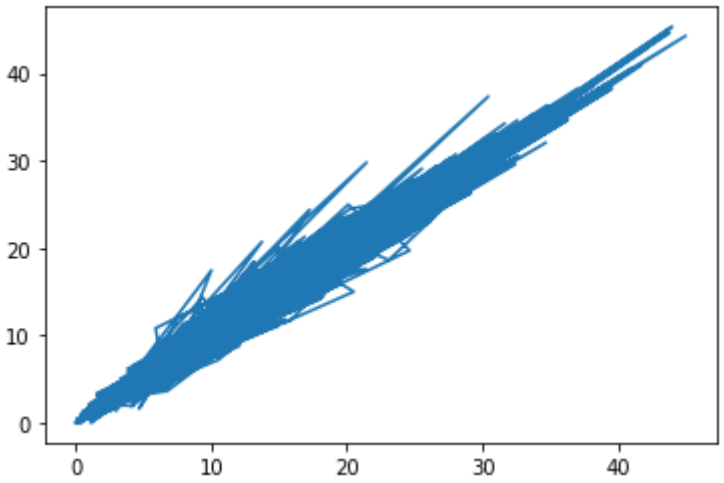
```
#usando a biblioteca para fazer o gráfico
plt.plot(y_pred_test, y_test)

results = {'Real Value': y_test, 'Prediction': y_pred_test}
df_results = pd.DataFrame(data=results)
df_results.head(100)
```

Fonte: Elaborada pelos Autores (2022).

Esse gráfico se assemelha mais com uma reta, e por isso demonstra que o modelo está com uma qualidade boa.

Figura 25 - Gráfico com a representação do modelo Light GBM.



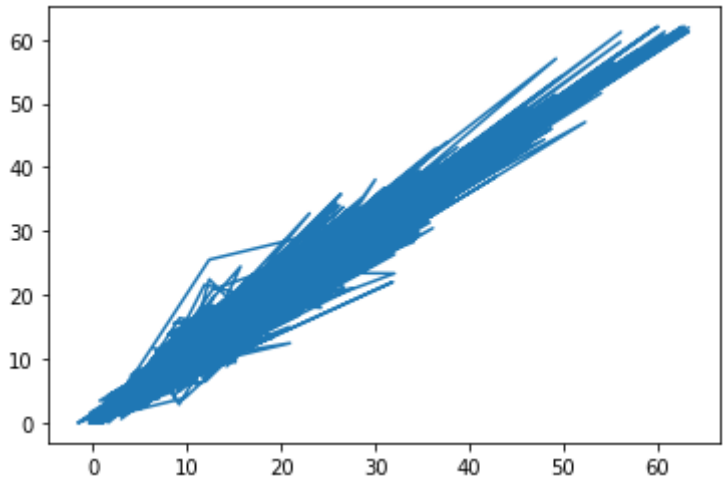
Fonte: Elaborada pelos Autores (2022).

4.5. Avaliação

4.5.1 Modelo escolhido

Após a análise do comportamento de cada modelo testado, a seguinte conclusão foi feita: o melhor modelo é o Light GBM, devido ao seu alto desempenho durante o teste com diversas métricas. Isso demonstra que tal modelo é o que consegue chegar aos valores mais próximos aos que correspondem à realidade. Tal fato pode ser visto através do gráfico gerado pelo modelo, o qual se assemelha muito a uma reta, validando o que foi dito anteriormente:

Figura 26 - Gráfico do desempenho do modelo Light GBM



Fonte: Elaborada pelos Autores (2022).

Para a avaliação do modelo, diversos métodos de validação foram utilizados: **R^2**, **erro médio absoluto**, **erro médio quadrático** e **acurácia** que se mostraram como os mais promissores. A partir do R^2 obteve se o seguinte resultado:

Código 30 - R^2

```
#R²
from sklearn import metrics
from sklearn.metrics import r2_score
print(metrics.r2_score(y_test, y_pred_test))

0.9888460035793034
```

Fonte: Elaborada pelos Autores (2022).

A fórmula para o cálculo do R^2.

Equação 2 - R^2.

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Fonte: Medium

Para o cálculo do erro médio absoluto, o método "*mean_absolute_error*" foi utilizado para a verificação da qualidade do modelo, o qual precisa apresentar valores mais baixos para que o modelo seja considerado bom.

Código 31 - Erro médio absoluto.

```
#Erro médio absoluto
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_train, y_pred_train)

0.5968758987603971
```

Fonte: Elaborada pelos Autores (2022).

Fórmula para o cálculo do erro médio absoluto:

Equação 3 - Erro médio absoluto.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} | \mathrm{Y}i - \hat{\mathrm{Y}}i |$$

Fonte: Medium

Para o cálculo do erro médio quadrático, utilizou-se o método "*mean_squared_error*" para a verificação da qualidade do modelo, o qual, assim como no caso anterior, precisa apresentar valores mais baixos para que se afirme uma qualidade maior no modelo.

Código 32 - Erro médio quadrático .

```
#Acurácia do treino Erro médio quadrático
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred_test)

0.7000935824826264
```

Fonte: Elaborada pelos Autores (2022).

Equação 4 - Fórmula do erro médio quadrático.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Fonte: Medium

Já quanto à acurácia, que é uma comparação percentual entre os valores reais e os valores encontrados, o teste indicou os seguintes resultados:

Código 33 - Medidas de acurácia.

```
#Acurácia do treino
trein_score = gbm.score(X_train, y_train)
print(train_score)


#Acurácia do teste
test_score = gbm.score(X_test, y_test)
print(test_score)
```

Fonte: Elaborada pelos Autores (2022).

4.5.2 Possíveis falhas e planos de contingência

No funcionamento do modelo escolhido, podem ocorrer alguns problemas relacionados à precisão do modelo, em que as previsões destoam da realidade, e isso pode ocorrer, principalmente, por alguma negligência nos dados coletados para o uso do modelo. Tais problemas podem ser os de dados desatualizados, dados viciados e de dados insuficientes.

Nesse sentido, primeiramente, para evitar o primeiro problema, é necessário que haja sempre a atualização do dataset com os dados disponibilizados pela pesquisa Kantar IBOPE para que as previsões utilizem sempre os dados mais próximos à atual realidade. Já quanto a questão dos dados viciados, é necessário que as pesquisas levem em conta populações de diversos nichos, sejam eles regionais, de renda, de idade, etc. Por último, é importante que sempre se mantenha um alto volume de dados coletados para o modelo, para que o dataset fique o mais abundante de informações possível. Dessa forma, os problemas relacionados à

precisão das previsões do modelo serão mantidos sob controle e o algoritmo funcionará muito bem.

4.6 Comparação de Modelos

4.6.1 Hiperparâmetros

4.6.1.1 Hiperparâmetros gerais

O *test size* representa qual será a porcentagem que será separada do *dataset* com a finalidade de teste. Portanto, se esse parâmetro é 0.20, ou seja 20%, o *train size* será 0.80, ou seja 80%, dividindo os dados do dataset na proporção descrita. De acordo com a tabela abaixo, realizou-se 3 testes diferentes para *test size*, com o valor padrão 0.15, que acaba recebendo a precisão de Rat% mais alta.

Testou-se aumentar e diminuir o valor do *test size*, porém essas modificações não melhoraram a precisão do modelo, como exemplificado com os valores de 0.05 e 0.20. A tabela a seguir apresenta apenas testes feitos no modelo de regressão linear, mas esse padrão de qualidade se repete para todos os outros modelos (KNN, LightGBM).

Tabela 3 - Testes de valores do "*Test_size*"

Rat%	Test_size
0.36098809594317066	0.05
0.36295387037658366	0.15
0.3625951626897451	0.20

Fonte: Elaborada pelos Autores (2022).

Código 34 - "*Test_size*" e "*random_state*".

```
#dividindo treino e teste
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.15,  random_state =
100000)
```

Fonte: Elaborada pelos Autores (2022).

4.6.1.2 Hiperparâmetros do Light GBM:

Para selecionar quais seriam os hiperparâmetros a serem utilizados no LightGBM, foram realizados diversos testes, avaliando métricas como acurácia e erro. A fim de conseguir o melhor desempenho na predição. Os hiperparâmetros escolhidos foram:

`"objective"`, que foi utilizado o hiperparâmetro *regression*, por ser o padrão para o modelo e o que mais se adequa ao projeto, que necessita de um modelo de regressão.

`"metric"`, foram utilizados o L1 e L2, representando o R^2, para que se tenha a medida de erro no treino e, com isso, padronizar com os outros modelos a serem comparados.

`"learning_rate"`, é usado o 0.32, representando a taxa de aprendizagem. Foi encontrado esse número, através do método de tentativa e erro, até que se atingiu o melhor resultado, onde foi combinado o teste de acurácia com o erro médio absoluto, como pode ser visto na Tabela 4 . Nesse contexto, o valor atua na normalização do peso das árvores, tendo como valor padrão de 0.1, determinando a contribuição de cada árvore no resultado final e controlando a rapidez com que o algoritmo aprende.

Tabela 4 - Learning Rate

Learning Rate	Acurácia teste	Erro médio absoluto
0.64	0.98878	0.2032
0.32	0.9893	0.2675
0.16	0.9886	0.3648
0.08	0.98622	0.4897
0.04	0.98241	0.6315

Fonte: Elaborada pelos autores (2022).

`"feature_fraction"` representa a porcentagem do dataframe a ser utilizado. No caso, foi escolhido o valor de 100%, ou seja, a totalidade dos dados disponíveis foram utilizados, ajudando o modelo a possuir maior efetividade.

`"max_depth"` representa um limitador para o tamanho de cada árvore. Foi utilizado o 8, um número relativamente baixo, mas que se justifica pelo tamanho do dataframe utilizado, no qual não se faz necessário um aumento do dataset por meio das árvores. Nos testes realizados, que foram explicitados na Tabela 5, pode ser notado o melhor desempenho com o `"max_depth"` em 8, apresentando menor erro médio absoluto e acurácia maior que com 4.

Tabela 5 - Comparação dos hiperparâmetros de acordo com o `"max_depth"`.

Max depth	Acurácia teste	Erro médio absoluto

16	0.98689	0.49540
8	0.98622	0.489799
4	0.94705	1.32398

Fonte: Elaborada pelos autores (2022).

`"num leaves"` foi utilizado o número 128. O número em questão é relativamente alto, haja vista o padrão utilizado pelo LightGBM, que é 31, porém a fim de evitar o overfitting, representando quando o modelo aprendeu as relações existentes no treino, esse aumento foi necessário. Se o valor padrão fosse colocado o modelo iria somente decorar, deixando-o viciado, e ao receber as informações das variáveis preditoras nos dados de teste, o modelo tenta aplicar as mesmas regras decoradas. A diferença nos parâmetros pode ser notada com facilidade ao observar a Tabela 6, em que analisando o erro médio absoluto, é notada uma grande diferença entre o escolhido e o outro parâmetro testado.

Tabela 6 - Num leaves

Num leaves	Acurácia teste	Erro médio absoluto
128 (escolhido)	0.9862	0.48979949
31	0.9728	0.901285

Fonte: Elaborada pelos autores (2022).

4.6.1.3 Hiperparâmetros do KNN

`"n_neighbors"` trata-se de um hiperparâmetro para a escolha dos vizinhos próximo, fazendo com que o número de vizinhos escolhido seja o total de comparações feitas entre eles. No modelo foi usado o n_neighbors igual a 2, pois com esse valor foi encontrada uma melhor acurácia, apesar de o padrão ser igual a 5.

Tabela 7 - N neighbors

n_neighbors	Acurácia teste	Erro médio absoluto
2(esco	0.9902	0.5008
5	0.9906	0.6987
7	0.9850	0.8176

Fonte: Elaborada pelos autores (2022).

4.6.2 Comparação de resultados

Quando colocados em comparação, os modelos apresentaram diferentes desempenhos, tornando-se claro a eleição do melhor para a predição baseada no dataset fornecido pelo cliente. Averiguando os valores obtidos, notou-se o melhor resultado para o modelo Light GBM, como visto anteriormente, notando-se a diferença díspar da acurácia alcançada, enquanto o modelo KNN, segue com a pior performance para o dataset usado.

Além disso, o erro médio quadrático (MSE) e o erro médio absoluto (MAE) também apresentaram diferenças significativas, onde o menor MSE foi obtido pelo modelo preditivo Light GBM, com o valor de 0.58, assim como o MAE, com o valor de 0.668. Também, pode-se constatar que o modelo de regressão linear, com 61.43, teve maior absorção de outliers, prejudicando a acurácia do modelo.

Tabela 8 - Comparação das métricas obtidas por cada modelo.

FATOR DE COMPARAÇÃO	MODELO LIGHT GBM	MODELO KNN	MODELO REGRESSÃO LINEAR
R^2	98,95%	-50,02%	36,29%
Erro médio quadrático	0,58	Não obtido	61,43
Erro médio absoluto	0,668	Não obtido	5,91
Acurácia do treino	99,79%	50,08%	36,31%
Acurácia do teste	98,95%	-50,02%	36,29%

Fonte: Elaborada pelos autores (2022).

Diante das métricas obtidas e após o teste dos melhores hiperparâmetros para cada modelo, foram feitas as comparações entre os resultados alcançados pelos mesmos, visando essa comparação por meios tangíveis e mensuráveis em uma mesma escala. Para isso, foi produzido um novo *dataset* de amostras a partir da base de dados original, de forma randômica, a fim de evitar viés no teste.

Para a criação destas amostras, utilizou-se do método "*.sample()*" , que sorteia aleatoriamente a quantidade desejada de linhas para o recorte do dataset. Com essa seleção realizada pelo método, foi gerado um novo arquivo com extensão CSV, posteriormente usado nos modelos.

Código 35 - Gerando amostras.

```
df_teste.sample(10).to_csv('/content/drive/Shareddrives/hefEStos: arquivo X/Dados - arquivo
x/amostraparadesvio2.csv')
```

Fonte: do próprio autor (2022).

Face a esse cenário, após selecionadas as amostras, as mesmas foram testadas apenas em dois dos três modelos escolhidos pela equipe, sendo esses o modelo preditivo Light GBM e o Regressão Linear, uma vez que o K-nearest neighbors (KNN) obteve uma acurácia negativa, assim descartado pela equipe.

Quando testado com o modelo de regressão linear (tendo 36,29% de acurácia), os resultados obtidos refletiram a baixa acurácia, dando a predição de algumas amostras com diferença acentuada, enquanto outras um pouco mais precisas. Esses valores foram reservados para posteriormente serem usados em análises de comparação com os resultados do outro modelo.

Tabela 9 - Valores reais vs valores previstos do modelo de regressão linear.

Valor real | Rat%	Total domicílios | Rat%
3.54	7.246786
7.98	6.563426
15.26	10.289991
8.06	11.354114
10.69	7.899833
5.74	10.739267
23.90	11.834498
4.75	11.083744
12.38	12.663980
17.29	11.447427

Fonte: elaborado pelo próprio autor (2022).

Contudo, quando testado pelo modelo Light GBM, foi possível notar o efeito da alta acurácia, obtendo predições bastante próximas do valor real e o baixo erro médio entre os preditos. Ademais, em comparação com as predições do regressão linear, notou-se uma preponderância do modelo corrente, em relação a adaptabilidade do dataset usado, de forma que até mesmo os outliers conseguiram de alguma forma, serem previstos de forma coerente, como observado na tabela 10.

Tabela 10 - Valores previstos no Light GBM.

Real Value	Prediction
22.46	22.142304
5.42	5.262246
10.78	10.318382
7.44	7.828568
6.19	6.478430
...	...
8.11	7.114564
19.68	19.409640
0.00	0.094853
20.44	20.469644
10.34	10.534386

Fonte: Elaborado pelos autores (2022).

5. Conclusões e Recomendações

Durante o período de 10 semanas, foram desenvolvidas soluções para a problemática trazida pelo parceiro. A equipe hefEStos promoveu a criação de 3 modelos preditivos a fim de obter a audiência de um programa lançado, auxiliando no desempenho da TV Gazeta na grade de programação.

Foram criados três modelos de predição com o método de regressão, sendo eles regressão linear, KNN e LightGBM. Embora o método com melhor precisão seja o Light GBM, os três modelos são aplicáveis para a previsão.

É recomendado que se utilize sempre o LightGBM, pois ele é o modelo que tem o melhor desempenho de acordo com os métodos de avaliação, ao usar coloque o máximo possível de informações do produto que será lançado, como o intervalo de horário que vai passar, data, categoria do programa e se o dia é um feriado ou não.

Ainda recomenda-se o uso da plataforma Colab, uma vez que os notebooks foram desenvolvidos na mesma. Caso haja a necessidade de aprimorar o dataset de treino, inserindo novas informações, a pasta disponibilizada no GitHub contém tanto a interface direta com o usuário, usando de modelos já treinados, como também os arquivos de treinamento e teste, na qual podem ser substituídos os datasets.

Ademais, em situação de dúvida, a equipe hefEStos se apresenta aberta para contato.

6. Referências

MEDIA, Kantar IBOPE. **Dados da audiência nas 15 praças praças regulares com base no ranking consolidado – 04/07 a 10/07.** Disponível em: <https://www.kantaribopemedia.com/dados-de-audiencia-nas-15-pracas-regulares-com-base-no-ranking-consolidado-0407-a-1007/>. Acesso em: 21 de agosto de 2022.

B2B. **What is the Value Proposition Canvas.** Disponível em: <https://www.b2binternational.com/research/methods/faq/what-is-the-value-proposition-canvas/#:~:text=The%20Value%20Proposition%20Canvas%20is,the%20customer%20values%20and%20needs.>. Acesso em: 20 de setembro de 2022.

MWITI, Derrick. **LightGBM: uma árvore de decisão de aumento de gradiente altamente eficiente.** Disponível em: <https://zephyrnet.com/pt/lightgbm-uma-%C3%A1rvore-de-decis%C3%A3o-de-aumento-de-gradiente-altamente-eficiente/>. Acesso em: 15 de setembro de 2022.

EXCHANGE, Stack. **Decision trees: leaf-wise (best-first) and level-wise tree traverse.** Disponível em: <https://datascience.stackexchange.com/questions/26699/decision-trees-leaf-wise-best-first-and-level-wise-tree-traverse>. Acesso em: 15 de setembro de 2022.

RIEPER, Marcos. **Feriados nacionais no Excel - Tabela de feriados.** Disponível em: <https://www.guiadoexcel.com.br/feriados-nacionais-no-excel-tabela-de-feriados-ate-2100/>. Acesso em: 22 de agosto de 2022.