

AGAMOTTO TV Gazeta

Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
08/08/2022	Antonio Nassar Arthur Prado Carolina Fricks Eduardo Porto Gabriela Matias Mateus Rafael Livia Bonotto	1.1	Criação do documento Começamos as seções: 2.1, 2.2, 3.1, 4.1.1 a, 4.1.1 b, 4.1.2, 4.1.3 d, 4.1.4,
26/08/2022	Arthur Prado Eduardo Porto Gabriela Matias Mateus Rafael Livia Bonotto	1.2	Preenchimento das seções 4.3. e 4.1.7.
29/08/2022	Livia Bonotto	1.3	Atualização da matriz de risco (seção 4.1.5)
08/09/2022	Livia Bonotto	1.4	Organização das seções 4.4 e 4.5.
09/09/2022	Livia Bonotto	1.5	Preenchimento das seções 4.4 e 4.5
10/09/2022	Carolina Favaro Fricks	1.6	Preenchimento das seções 4.5.1 e 4.5.3
11/09/2022	Mateus Rafael	1.7	Preenchimento das seções 4.4.2, 4.4.4, 4.5.2 e 4.5.4
11/09/2022	Gabriela Matias	1.8	Atualização das seções: 4.1.1. 4.1.3, 4.2.2 e 4.4.3.
19/09/2022	Livia Bonotto	1.9	Adição de novos tópicos e seções
19/09/2022	Eduardo Porto	2.0	Atualização da seção 4.3 e novos testes adicionados
20/09/2022	Livia Bonotto	2.1	Adição da "quarta etapa de testes"
21/09/2022	Mateus Rafael	2.2	Preenchimento das seções 4.4.6 e 4.4.8
25/09/2022	Livia Bonotto	2.3	Transferência do tópico "Resultados" de cada modelo da seção 4.4 para a 4.5, pois avaliamos ser mais didático desta forma;

25/09/2022	Livia Bonotto	2.3	Revisão e preenchimento das seções 4.4, 4.5 e 4.6.
26/09/2022	Mateus Rafael	2.4	Transportação do documento para terceira pessoa do plural até o tópico 3.1
29/09/2022	Mateus Rafael	2.5	Transportação do documento para terceira pessoa do plural até o tópico 4.5
03/10/2022	Mateus Rafael	2.6	Conclusão da transporte do documento para terceira pessoa do plural
04/10/2022	Mateus Rafael e Carolina Fricks	2.7	Preenchimento dos tópicos 3.2 e 3.3; Reavaliação e alterações no tópico 4.3.3 (Feature Engineering)
04/10/2022	Arthur Fraige; Livia Bonotto	2.8	Preenchimento do tópico 5; Preenchimento do tópico 6
05/10/2022	Mateus Rafael e Carolina Fricks	2.9	Reavaliação dos gráficos do documento
06/10/2022	Mateus Rafael	3.0	Revisão e preenchimento dos dados finais do documento.

Sumário

1. Introdução	5
2. Objetivos e Justificativa	5
2.1. Objetivos	5
2.2. Justificativa	5
3. Metodologia	6
3.1. CRISP-DM	6
3.2. Ferramentas	6
3.3. Principais técnicas empregadas	7
4. Desenvolvimento e Resultados	9
4.1. Compreensão do Problema	9
4.1.1. Contexto da indústria	9
4.1.2. Análise SWOT	10
4.1.3. Planejamento Geral da Solução	11
4.1.4. Value Proposition Canvas	12
4.1.5. Matriz de Riscos	14
4.1.6. Personas	14
4.1.7. Jornadas do Usuário	15
4.2. Compreensão dos Dados	16
4.3. Preparação dos Dados	25
4.4. Modelagem	30
4.5. Avaliação	40
4.6. Comparação de Modelos	52
5. Conclusões e Recomendações	65
6. Referências	66
Anexos	70

1. Introdução

Informação, entretenimento e prestação de serviços de comunicação focado no estado do Espírito Santo. A Rede Gazeta é o maior grupo de comunicação capixaba, possuindo oito estações de rádio e quatro emissoras de TV aberta afiliadas à Rede Globo.

A Rede Gazeta é uma empresa de grande importância para todo o estado do Espírito Santo, pois é a maior emissora de TV aberta local contando com mais de 500 funcionários.

O problema enfrentado pela emissora é poder criar novos programas de TV de forma assertiva com decisões baseadas em dados.

2. Objetivos e Justificativa

2.1. Objetivos

O objetivo geral do parceiro é prever a audiência em um período específico de tempo e, com base nessa métrica, prever o investimento que ele terá que fazer para o programa atingir a expectativa de telespectadores, aumentar a audiência e o tempo de permanência. Também há o objetivo de estimar qual programa passar em determinado horário para obter melhores resultados.

2.2. Justificativa

Pensando no problema, surge a necessidade de criar um modelo preditivo responsável por garantir uma sequência de informações que garanta uma visão prévia dos fatores e elementos que impactam no resultado final da audiência. Desse modo, com o modelo levantado por meio dos dados listados, podemos encontrar uma forma de prever como cada fator impacta no negócio e, assim, determinar um possível crescimento da audiência, tendo como base o horário (alcance e/ou tempo de permanência), fornecendo ao parceiro o score de audiência a partir do peso de cada variável na definição do resultado - dia; tempo (por hora); audiência; share; tempo de permanência; alcance e gênero do produto.

3. Metodologia

3.1. CRISP-DM

CRISP-DM, abreviação para Cross Industry Standard Process for Data Mining, é uma metodologia de planejamento para mineração de dados onde, por meio de um ciclo de etapas, é possível compreender o andamento/fluxo de um processo ou análise de dados de um projeto. As etapas consistem em fases, onde após o entendimento do modelo de negócios, se estabelece a maneira como dados são coletados e analisados. Após isso, estes dados são preparados para serem implementados e modelados conforme a necessidade de seu determinado uso. Por fim, as chamadas Instâncias dos Processos são a fase final onde esses dados são “sólidos” e assim estão prontos para serem utilizados.

3.2. Ferramentas

Para cada processo, desde a leitura do arquivo “.csv” até a conclusão do modelo preditivo, foram usadas algumas ferramentas para identificação, preparação, montagem, modelagem, teste e treinamento e predição de cada informação. Essas ferramentas e *softwares* serão apresentadas a seguir:

Google Collaboratory/Jupyter Notebook: É um *software* voltado para programação utilizando máquina virtual, onde o usuário só precisa ter como recurso um computador e conexão com a internet, uma vez que toda escrita de texto ou código são armazenadas no e-mail do usuário e utilizam de memória de armazenamento e de processamento “virtual”, ou seja, máquinas dos próprios servidores do Google.

Flask: É considerado um *microframework* web utilizado para criar um *front-end* integrado a partir da linguagem de programação Python e é considerado como *microframework*, pois não necessita de bibliotecas ou ferramentas específicas.

Visual Studio Code: Segundo o mecanismo de pesquisa Google, *Visual Studio Code*, ou “VS Code”, como é popularmente conhecido, é um editor de código-fonte feito pela Microsoft com o Electron Framework, para Windows, Linux e macOS. Os recursos incluem suporte para depuração, realce de sintaxe, conclusão de código inteligente, snippets, refatoração de código e Git incorporado.

PyCharm: É um ambiente de desenvolvimento integrado (Do Inglês “integrated development environment (IDE)), *software* gratuito e compatível com Windows, Linux e Mac OS que permite o usuário fazer seus desenvolvimentos utilizando a linguagem em Python.

Excel: É um *software* da Microsoft que permite ao usuário visualizar e manipular determinados dados em formato de tabelas, considerado como “planilhas” dentro do *software*.

3.3. Principais técnicas empregadas

Para que a recepção, visualização e manipulação de dados fosse realizada com a qualidade desejada pelo grupo, foram estudadas as melhores formas e abordagens para tratar e manipular os dados de forma individual e de forma coletiva, usando um *set* de dados e usando mais de um com necessidade de *merge* entre duas ou mais tabelas distintas. Para isto, o grupo tratou do set de dados com as seguintes técnicas que serão apresentadas com base no segmento do CRISP-DM:

Entendimento do negócio: nesta etapa, o grupo recebeu, leu e interpretou as informações de descrição da empresa parceira de projeto, da sua atuação e seus objetivos, problemas e possível solução, a qual foi desenvolvida ao decorrer do projeto.

Entendimento dos dados: considerada a segunda etapa do projeto, foi feita a ambientação com os softwares e recursos que seriam utilizados durante o desenvolvimento, interpretação inicial dos dados e abordagem com os dados para início do projeto. As técnicas utilizadas nesta etapa foram:

Preparação dos dados: identificada como terceira etapa da metodologia CRISP-DM, esta etapa foi planejada e organizada para preparar o ambiente e o *set* de dados para representação gráfica, busca por melhores informações de forma prática e primeiras análises para o que faria sentido ou não pelo grupo para ser utilizado ao decorrer do projeto. Aqui, o grupo usou as seguintes ferramentas e técnicas:

Mesclagem de planilhas: usado para realizar um “merge” nas informações que estavam separadas em mais de uma planilha e organizados no dataset que está sendo usado no modelo final.

Anonimização dos dados: usado para realizar a anonimização dos dados que não podem ser demonstrados abertamente ao público, como nome da emissora e os nomes dos programas de cada emissora.

Separação do *dataset*: foi uma ferramenta usada para separar a parte do *set* de dados para treino e para teste, ao qual seria usada para comparação de resultados, identificação de *outliers* e de *overfitting* ou *underfitting*.

Apresentação de gráficos: foi aplicado no modelo para trazer uma apresentação gráfica das audiências da planilha recebida, dos tratamentos realizados no *set* de dados e nas previsões realizadas durante o desenvolvimento do projeto.

Modelagem: nesta etapa, conhecida como a quarta etapa, o projeto já tem sua primeira forma, pois já se podem fazer as primeiras avaliações de gráficos e métricas, análise de variáveis e melhoria de atributos do(s) modelo(s). Nesta etapa do projeto, por conta dos cálculos e gráficos necessários para uma boa visualização e análise, foram utilizadas as seguintes ferramentas:

Métricas: foi aplicado para apresentar as métricas de r^2 do(s) modelo(s) usados na primeira etapa de testes.

One-hot encoder: uma técnica empregada foi o one-hot encoder que é usada para codificar recursos categóricos como uma matriz numérica, baseado em scikit-learn.

Avaliação: a quinta etapa do projeto é voltada para permitir ao grupo comparar e avaliar os modelos que foram criados anteriormente, suas métricas, gráficos e decidir qual o melhor modelo para aplicação/implementação. Para tal feito, foram necessárias as ferramentas que serão apresentadas a seguir:

Métricas: nesta etapa, o grupo utilizou da biblioteca de métricas e gráficos para se basear e análise e pensamento crítico para a avaliação de cada modelo até encontrar o modelo com as melhores estatísticas e parâmetros.

Aplicação: com o andamento do projeto seguindo de forma correta, o grupo chega à última etapa, que é a aplicação do melhor modelo de predição encontrado. Como esta é a etapa de entrega do projeto, o modelo estará pronto e utilizando todas as ferramentas e técnicas das etapas citadas anteriormente.

Os algoritmos usados para predição serão explicados detalhadamente no tópico 4.4, suas avaliações no tópico 4.5 e as comparações entre eles estarão no tópico 4.6.

4. Desenvolvimento e Resultados

4.1. Compreensão do Problema

4.1.1. Contexto da indústria

Principais Players

De acordo com a tabela indicada abaixo relativa aos principais *players* do mercado em concorrência, nota-se a TV 1 e TV 2. Porém, tendo como base os dados de audiência, foi possível entender que, apesar da força dos concorrentes dentro do mercado, atualmente o poder de alcance da audiência do parceiro é muito mais amplo. Com exceções muito pontuais de horários nos quais existe uma queda, mas o modelo preditivo busca, justamente, fornecer uma solução para que o parceiro possa ter um bom desempenho também nos horários de queda.

	TV 2	TV 1
Descrição	"A TV 2 possui alcance por todos 78 municípios do Espírito Santo. Com um único sinal, é transmitida uma cobertura completa com jornalismo, entretenimento, esporte, dramaturgia e coberturas nacionais e internacionais de qualidade. Mesmo com um sinal unificado, a TV Vitória consegue, com maestria, direcionar seus conteúdos a comunidades locais."	"TV 1 é uma emissora de televisão brasileira sediada em Vitória, capital do estado do Espírito Santo."
Pontos Fortes	- "A TV 2 recebeu o título de melhor TV regional do Brasil por sete vezes, eleita pela Academia Brasileira de Marketing"	- "A TV 1 conquistou o respeito e a audiência dos telespectadores e do mercado anunciante, consolidando-se como a 2ª maior emissora de TV do Espírito Santo. A grade regional é formada por 10 programas líderes em seus segmentos."
Pontos Fracos	- Abaixo da faixa padrão de audiência da TV Gazeta (Globo)	- Abaixo da faixa padrão de audiência da TV Gazeta (Globo)

Relação com Clientes:

O poder de barganha dos compradores diz respeito à capacidade de barganha dos clientes para com as empresas do setor. Neste caso, essa força é uma das mais atuantes pois a audiência diz respeito ao poder de escolha dos compradores/clientes em selecionar para qual rede de televisão ou canal vão dedicar sua atenção em determinado dia ou horário. O protótipo a ser desenvolvido irá determinar justamente os parâmetros que influenciam no poder de aderência dos compradores em relação a emissora e como isso pode influenciar no negócio.

Fornecedores:

No caso do mercado televisivo, não existe necessariamente uma grande concorrência entre fornecedores, pois se trata de fornecedores pulverizados. Logo, existem várias opções, como por exemplo: Atores, Repórteres, Fornecedores de Equipamentos, entre outros profissionais ou empresas que atuam com o fornecimento de equipamentos e pessoas para atuarem no mercado.

Novos Entrantes:

Para o mercado televisivo existe uma grande dificuldade para a atuação e implementação de novos entrantes, visto que é um mercado extremamente bem consolidado e regulado para emissoras tradicionais. Logo, novos entrantes não se apresentam como uma ameaça para o negócio, pois atualmente existem diversos aspectos e fatores, até mesmo burocráticos, que limitam a criação de novos canais televisivos. Desse modo, os principais *players* são grandes concorrentes, mas existe pouca probabilidade de novos entrantes que podem ser inseridos no mercado.

Tendências do Mercado:

Pensando que atualmente existe uma grande demanda em relação ao acesso a conteúdos por meio da Internet ou *Smartphones*, a indústria televisiva como um todo vem sofrendo uma queda. “De acordo com a Agência Nacional de Telecomunicações (Anatel), a queda de assinantes de TV paga tem sido impulsionada pela mudança no comportamento dos telespectadores, que estão optando por acompanhar filmes e séries em plataformas de *streaming*, como Netflix e Amazon, pois oferecem conteúdos originais e serviços com um custo menor aos usuários.” [DIGILAB, 2019]. Porém, tendo como base a própria análise dos dados de audiência, temos que os serviços como *Streaming*, por exemplo, possuem um nível de audiência individual, sendo que existem os *players* A, B ou C, cada um deles tem sua audiência determinada individualmente, o que em comparação com a rede televisiva segue sendo um nível abaixo da audiência do parceiro.

4.1.2. Análise SWOT

The Pythons		MATRIZ SWOT - FOFA	
	Fatores Positivos	Fatores Negativos	
Fatores Internos	Forças -Principal detentor de jornais de títulos da região -Associação com a Globo -Muitos recursos humanos e tecnológicos na área de jornalismo	Fraquezas -Base de dados não tratada -Provem de pouco recurso humano na área de inovação -Falta de uma ferramenta inovadora com relação a predição	
Fatores Externos	Oportunidades -Porcentagem alta da população tem conhecimento da emissora -Uma das principais emissoras do estado, no quesito relevância	Ameaças -Mar vermelho, grande concorrência de mercado -Falta de demanda de TV -Falta de fidelidade do público com a maioria dos programas de TV, hoje em dia -Aumento do uso de Streaming e Celulares, representa diretamente a queda da TV	

4.1.3. Planejamento Geral da Solução

a) O problema: Falta da possibilidade de atuar de forma preditiva em relação ao desempenho dos programas de televisão. Ausência de informações que permitam antecipar os problemas e intervir antes mesmo da estreia de determinado produto.

b) Dados disponíveis:

Mapeamento de Audiência: Planilha do Excel contendo informações sobre a audiência para o canal da Rede Gazeta e outros canais para comparação, assim como a audiência geral dos Canais Pagos e Serviços Não Identificados (*Streamings*).

Grade Horária de Programação: Grade contendo quais programas estão sendo transmitidos em relação a horário e mês, assim como as categorias em que esses programas se encaixam.

c) Proposta de solução:

A solução trata-se de um modelo preditivo, onde o usuário possa inserir no modelo um possível horário para um novo programa de TV. Assim, o modelo pode fornecer dados sobre a audiência daquele programa, como gênero dos telespectadores, faixa etária e tipo de programa que esse público tem preferência por assistir.

Por fim, auxiliará os produtores a criarem um programa que se adapte mais à possível audiência naquele horário e os profissionais da publicidade a organizarem campanhas de *marketing* mais assertivas, além de possibilitar a aplicação de investimentos com maior consciência dos resultados.

d) Tipo de tarefa (regressão ou classificação):

O tipo de tarefa da predição a ser desenvolvido inicialmente, é regressão, visto que trabalharemos com dados contínuos e receberemos um feedback de probabilidade. No entanto, a classificação também seria um caminho possível e interessante caso houvesse parâmetros de sucesso de audiência (baixo, médio e alto, por exemplo) a partir dos dados gerados pela predição.

e) Utilização da solução proposta:

O modelo preditivo a ser desenvolvido tem o objetivo de realizar uma previsão com uma boa taxa de precisão sobre o quanto uma programação pode obter sucesso ou a baixa, subindo ou caindo a taxa de audiência e o tempo em que o telespectador fique no mesmo canal.

O protótipo a ser desenvolvido visa impulsionar as estatísticas televisivas da TV Gazeta, em duas frentes, sendo a primeira em pico de telespectadores totais, seja em filmes, novelas,

noticiários, etc. Já a segunda frente o tempo de retenção e permanência do telespectador durante a exibição do programa.

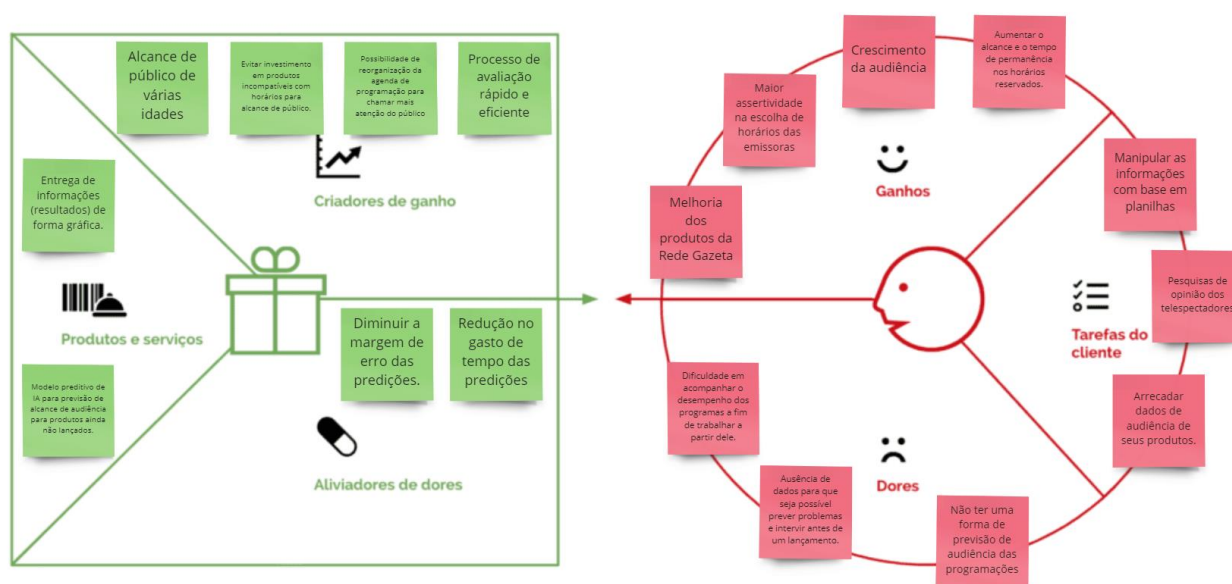
f) Benefícios da solução:

Possível crescimento de audiência a partir de uma análise prévia e dimensionamento dos fatores críticos da performance no horário (alcance e/ou tempo de permanência).

g) Critério de Sucesso:

Atualmente pode-se definir os critérios de sucesso do negócio tendo como base objetivo específico do projeto fornecido pelo cliente, no qual deve prever a audiência em um período específico de tempo e com base nesse resultado, pode prever quanto investimento ele terá que colocar para o programa atingir a expectativa de telespectadores, aumentar a audiência e o tempo de permanência. Considerando que obteve-se sucesso ao atingir o objetivo e isso pode ser metrificado por meio da existência de uma margem de erro pequena em relação da audiência prevista em comparação com o valor real. Quando falamos de margem de erro pequena se torna necessário definir inicialmente o que é uma margem pequena ou não em relação ao modelo de negócios. Para isso cabe ao grupo entender com o parceiro de negócios qual o nível de precisão desejado.

4.1.4. Value Proposition Canvas



Proposta de valor:

Produtos e serviços:

- Entrega de informações (resultados) de forma gráfica.
- Modelo preditivo de IA para previsão de alcance de audiência para produtos ainda não lançados.

Criadores de ganho:

- Alcance de público de várias idades
- Evitar investimento em produtos incompatíveis com horários para alcance de público.
- Possibilidade de reorganização da agenda de programação para chamar mais atenção do público
- Processo de avaliação rápido e eficiente

Aliviadores de dores:

- Diminuir a margem de erro das previsões.
- Redução no gasto de tempo das previsões

Perfil do cliente:

Ganhos:

- Crescimento da audiência.
- Aumentar o alcance e o tempo de permanência nos horários reservados.
- Maior assertividade na escolha de horários das emissoras.
- Melhores investimentos dos produtos da Rede Gazeta.

Tarefas do cliente:

- Manipular as informações com base em planilhas
- Pesquisas de opinião dos telespectadores
- Arrecadar dados de audiência de seus produtos.


Dores:

- Dificuldade em acompanhar o desempenho dos programas a fim de trabalhar a partir dele.
- Ausência de dados para que seja possível prever problemas e intervir antes de um lançamento.
- Não ter uma forma de previsão de audiência das programações.

4.1.5. Matriz de Riscos

		Matriz de Risco									
Probabilidade		Riscos					Oportunidade				
Muito Alta	5					Falta de clareza nos entregáveis (detalhamento)					
Alta	4				Falta de conhecimento necessário para uma melhor definição do modelo preditivo		Entrega de um modelo eficiente e de alta acurácia				
Médio	3		Por estar em estado inicial, o modelo pode não ser tão assertivo			Documentação sem estrutura científica	Programação mais assertiva em relação ao que o público demanda.	Todos os integrantes do grupo aprender e ter boa média final no módulo	Expandir o alcance da rede gazeta		
Baixa	2			Competição auto estudo x desenvolvimento	Modelo com poucas variáveis, assim não sendo suficiente assertivo	Desequilíbrio na divisão de tarefas e comprometimento.		Aumentar o tempo de permanência do público			
Muito Baixa	1										
		1	2	3	4	5	5	4	3	2	1
		Muito Baixo	Baixo	Médio	Alta	Muito Alta	Muito Alta	Alta	Médio	Baixo	Muito Baixo
		Impacto									

4.1.6. Personas



Nome
Cassia Ellen

Cargo
Analista de Dados

Era
25 to 34 years

Nível mais alto de educação
Mestrado em Tecnologia

Indústria
Tecnologia

Tamanho da organização
+500 Funcionários

Características

- Gostos: Rock Clássico, Heavy Metal, Pagode e filmes do Tarantino.
- Personalidade: Cassia é muito obstinada e gosta de otimizar seu tempo e processos no trabalho para que as ações sejam realizados de forma automatizada.
- Hobbies: Ler, aprender sobre Dados, ver filmes, passar tempo com sua família.
- Vida Pessoal: Cassia tem 2 filhos e gostaria muito de dedicar mais tempo com sua família, por conta disso tem interesse em otimizar seu trabalho.

Responsabilidades do trabalho

- Analisar os dados de audiência fornecidos pelo pela Kantar Ibope.
- Fornecer relatórios sobre o alcance de audiência em relação aos parâmetros fornecidos pela Kantar Ibope.

Ferramentas que eles precisam para fazer seu trabalho

- Situação Atual: Excel contendo a base dos dados.
- Situação Futura: Agamotto - Modelo de previsão de dados indicando o score de audiência.

Maiores desafios

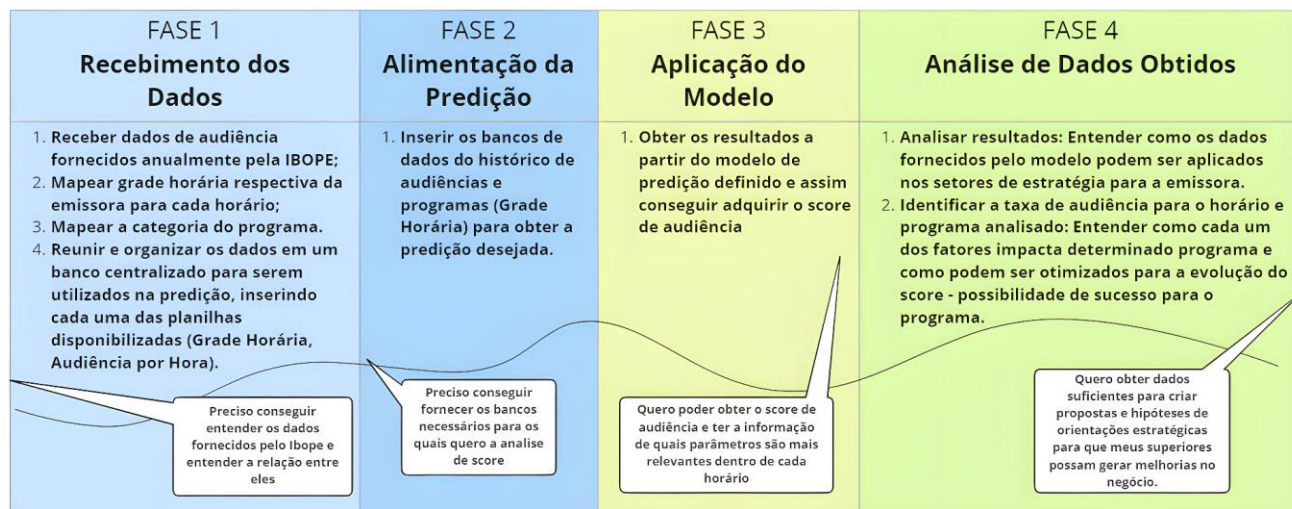
- Obter uma análise de dados em um grande volume com base em um histórico de evidências.
- Automatização e Praticidade no processo de gerar dados de referência
- Conseguir ter uma previsão antecipada de quanto de audiência o programa possivelmente irá ter.

4.1.7. Jornadas do Usuário



Cassia Ellen

Cenário e Expectativa : Gerar melhores opções estratégicas com base em uma predição de alcance de público de um produto em determinado horário.



Oportunidades

- Otimizar o input de Dados para inserção do Banco de Dados

Responsabilidades

- Criar um processo otimizado de inserção de banco de dados.
- Garantir que estamos fornecendo os resultados pertinentes para a definição do score de audiência.
- Mostrar o quanto cada feature influencia no resultado final da audiência.

4.2. Compreensão dos Dados

4.2.1 Descrição dos Dados:

- i. Fonte dos Dados: Kantar Ibope, 2022
- ii. Tipo de Arquivo: Planilha - CSV/ XLSX
- iii. Tamanho da Base de Dados: 427MB
- iv. Quantidade de Planilhas: 18 Planilhas
 - Canais Pagos (3 Planilhas [Sab., Dom., Seg - Sex])
 - NI Conteúdo (3 Planilhas [Sab., Dom., Seg - Sex])
 - TLE (3 Planilhas [Sab., Dom., Seg - Sex])
 - TV 0 (3 Planilhas [Sab., Dom., Seg - Sex])
 - TV 1 (3 Planilhas [Sab., Dom., Seg - Sex])
 - TV 2 (3 Planilhas [Sab., Dom., Seg - Sex])
- v. Colunas: 60 Colunas: Cada coluna representa as seguintes características para cada parâmetro de avaliação. Sendo que os parâmetros são dados por
 - Rat% (Rating): Número de indivíduos por domicílio acompanhando determinado evento -> Refere se a audiência.
 - Shr% (Share): Participação da audiência em um evento, em relação ao total de aparelhos ligados.
 - Rch% (Reach): Alcance da audiência nas programações em diferentes perfis.
 - Fid% (Fidelidade): Total de indivíduos diferentes atingidos por pelo menos 1 minuto por um conjunto de eventos.

Características avaliadas para cada parâmetro:

 - Gênero: Feminino e Masculino;
 - Classe Social: AB, C1, C2, DE;
 - Faixa Etária (anos): 4-11, 12-17, 18-24, 25-34, 35-49, 50-59, 60+.
- vi. Tipos de Dados: *Datetime* (dia e horário), String (texto), Inteiro (número).
- vii. Período de Análise: 2020 - 2022

Mesclando os dados, foi feita a confecção de três tabelas, a primeira medindo a relação entre audiência e idade, identificando quais faixas de idade apresentam maiores picos de audiência. A segunda identificando a relação de audiência e

gênero, podendo visualizar a porcentagem de telespectadores dividida entre público masculino e público feminino. Por último, uma tabela geral, que pega a média de audiência dos dias da primeira semana apresentada no banco de dados, e depois categorizando essa média em idade e gênero.

O risco que temos ao lidar com esses dados é o fato de que os dados da audiência são provenientes do IBOPE e a mesma base não pode ser compartilhada e deve ser anonimizada. Além disso, a diversidade não é tão grande, visto que temos acesso a grade de programação de apenas duas outras emissoras, gerando uma pequena base de dados para comparar a relevância do tipo de programa com a sua audiência.

Utilizando dos subconjuntos para visualizar o nível de audiência, divididos tanto pela faixa de idade, quanto pelo gênero, nos provê uma indicação consistente de quais tipos de programas atraem quais tipos de pessoas e utilizaremos isso como base para medir índice de audiência de um título que ainda não foi passado, ou qual seria o êxito de um título repetido.

Os dados disponibilizados para estudo apresentam restrição de segurança para preservar a privacidade dos dados das emissoras que foram usadas para fins de comparação. Sendo assim, não podem ser disponibilizados publicamente.

4.2.2. Descrição Estatística Básica dos Dados:

Números dos dados: baseado na base de dados, classificamos as estatísticas da seguinte forma:

Atributos de interesse (Tipos de dados):

a. Análise: Gênero X Horário

i. Média:

- Masculino: 4.03
- Feminino: 5.14;

ii. Mediana:

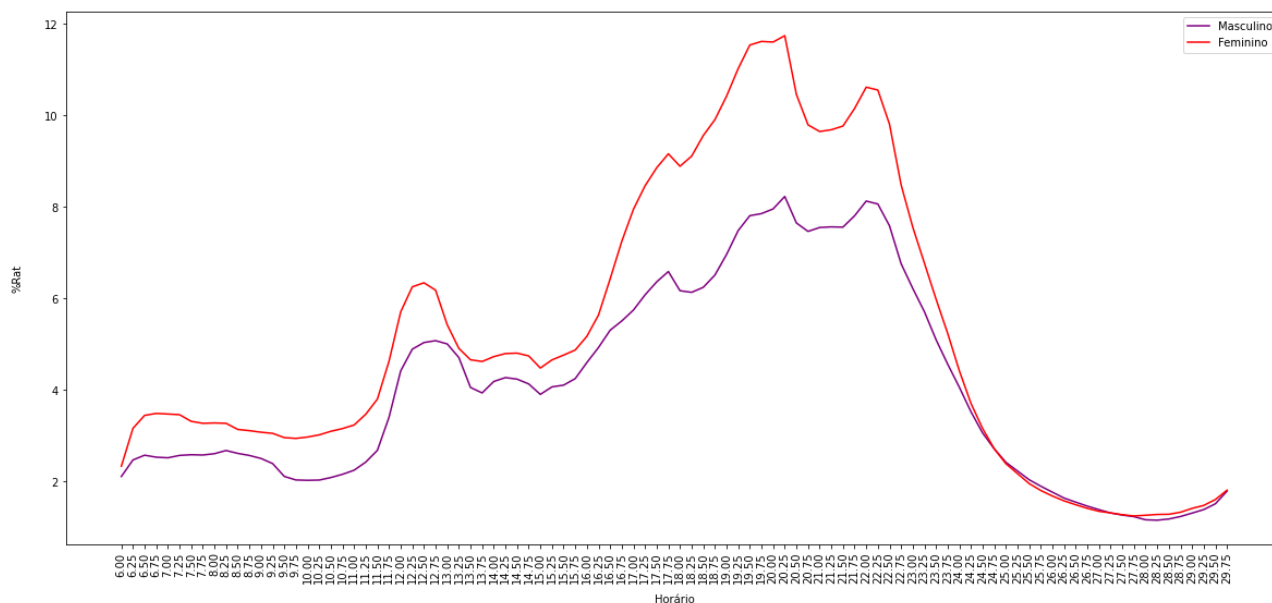
- *Masculino*: 3.41
- Feminino: 4.08;

iii. Desvio Padrão:

- Masculino: 2.785044321
- Feminino: 3.808778465;

Benefício da análise:

Entender qual o gênero de maior prevalência de forma macro em relação ao impacto na audiência e também poder visualizar como os gêneros podem ser um critério de padrão de audiência para cada horário.



Análise: Classe Social X Horário

iv. Média:

- AB: 4.03;
- C1: 3.66;
- C2: 5.17
- DE: 6.05;

v. Desvio Padrão:

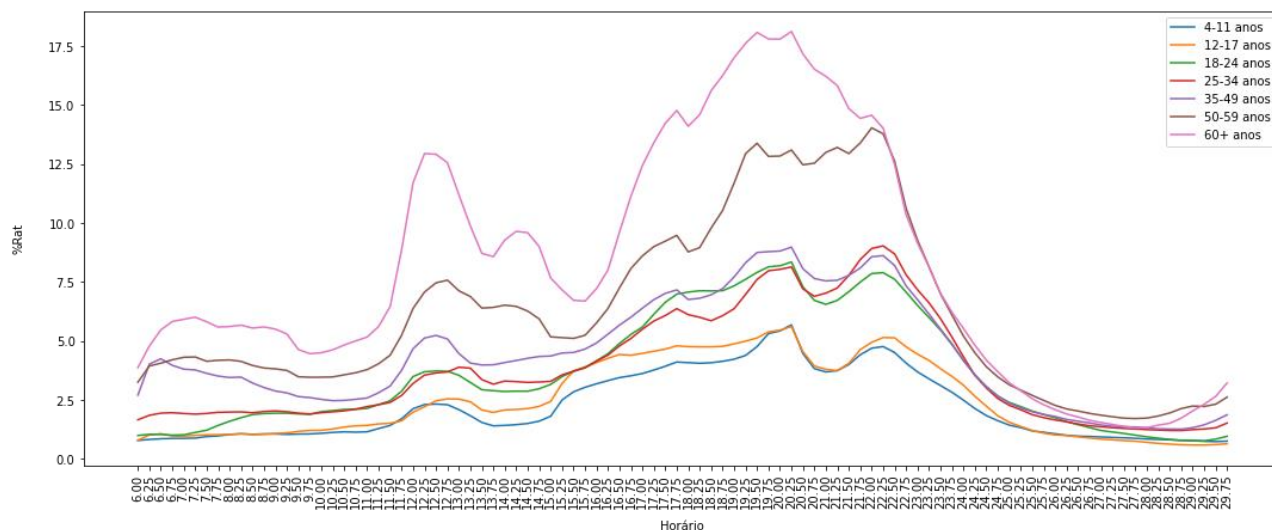
- AB: 2.903602071;
- C1: 3.151994451;
- C2: 4.207737178;
- DE: 4.985268943;

vi. Mediana:

- AB: 3.51;
- C1: 2.96;
- C2: 3.95;
- DE: 4.7;

Benefício da análise:

Entender qual a classe social de maior prevalência de forma macro em relação ao impacto na audiência e também poder visualizar se há uma variação entre a preferência de cada classe para determinarmos um critério de padrão de audiência para cada horário.



b. Análise: Faixa Etária X Horário:

i. Média:

- 4-11: 2.23;
- 12-17: 2.51;
- 18-24: 3.63;
- 25-34: 3.77;
- 35-49: 4.41;
- 50-59: 6.19 e
- 60+: 8.10

ii. Mediana:

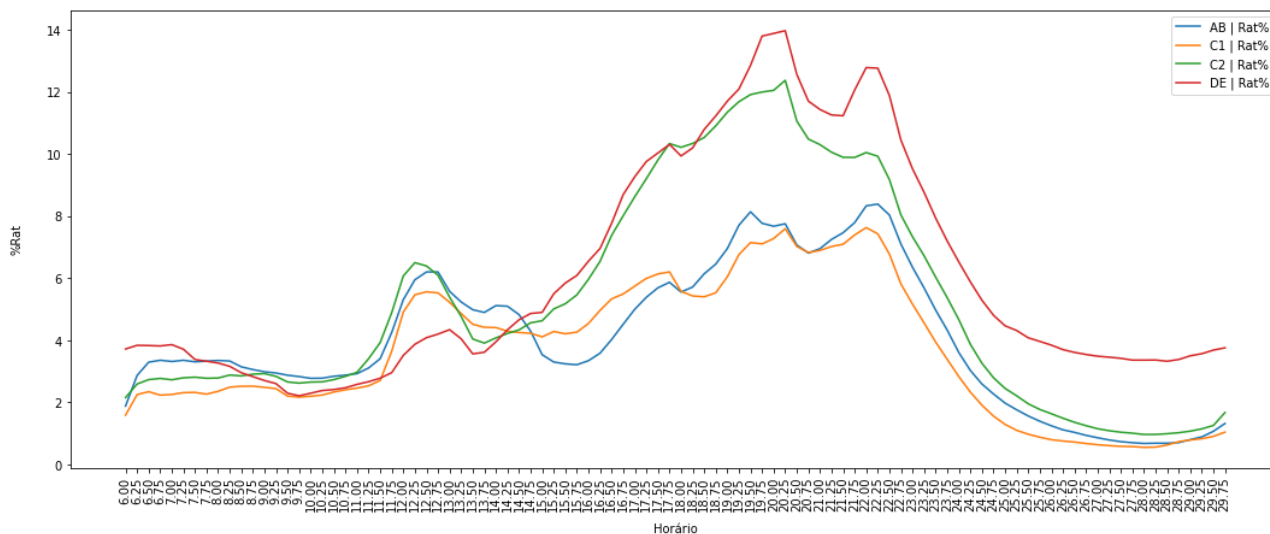
- 4-11: 1.47;
- 12-17: 1.57;
- 18-24: 2.61;
- 25-34: 2.89;
- 35-49: 3.78;
- 50-59: 5.01
- 60+: 6.68

iii. Desvio Padrão:

- 4-11: 2.49967781;
- 12-17: 2.909038769;
- 18-24: 3.639758362;
- 25-34: 3.349938817;
- 35-49: 3.095652657;
- 50-59: 4.892826927

● 60+: 6.032953907

Benefício da análise:

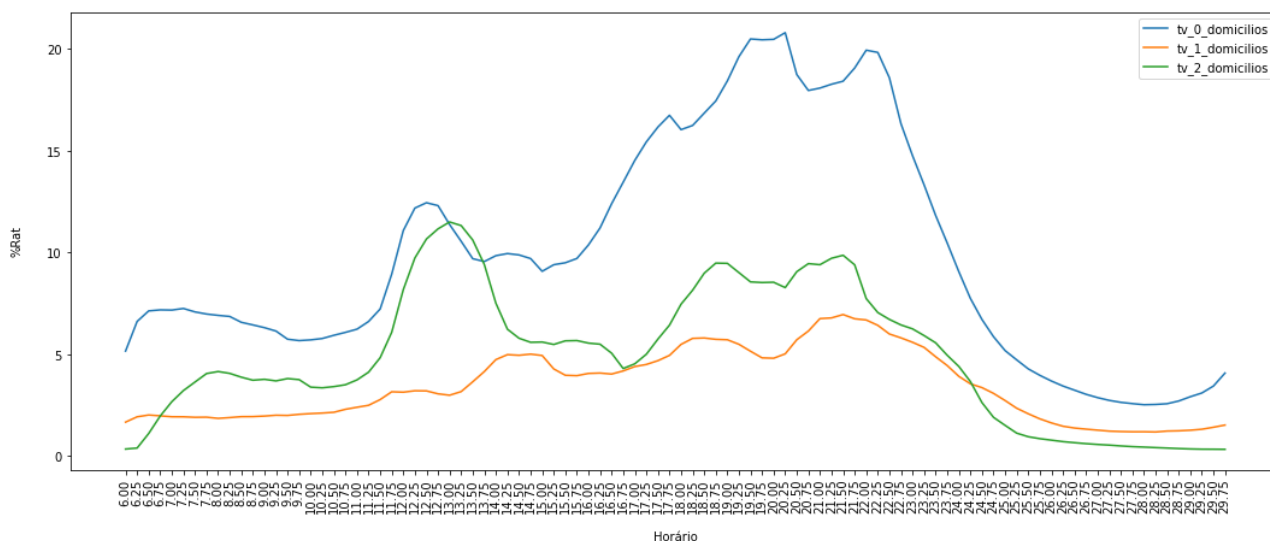


Entender qual a faixa etária de maior prevalência de forma macro em relação ao impacto na audiência e também poder visualizar se há uma variação entre a preferência de cada idade para determinarmos um critério de padrão de audiência para cada horário.

c. Análise: Rede Gazeta em Relação a Concorrentes

Benefício da análise:

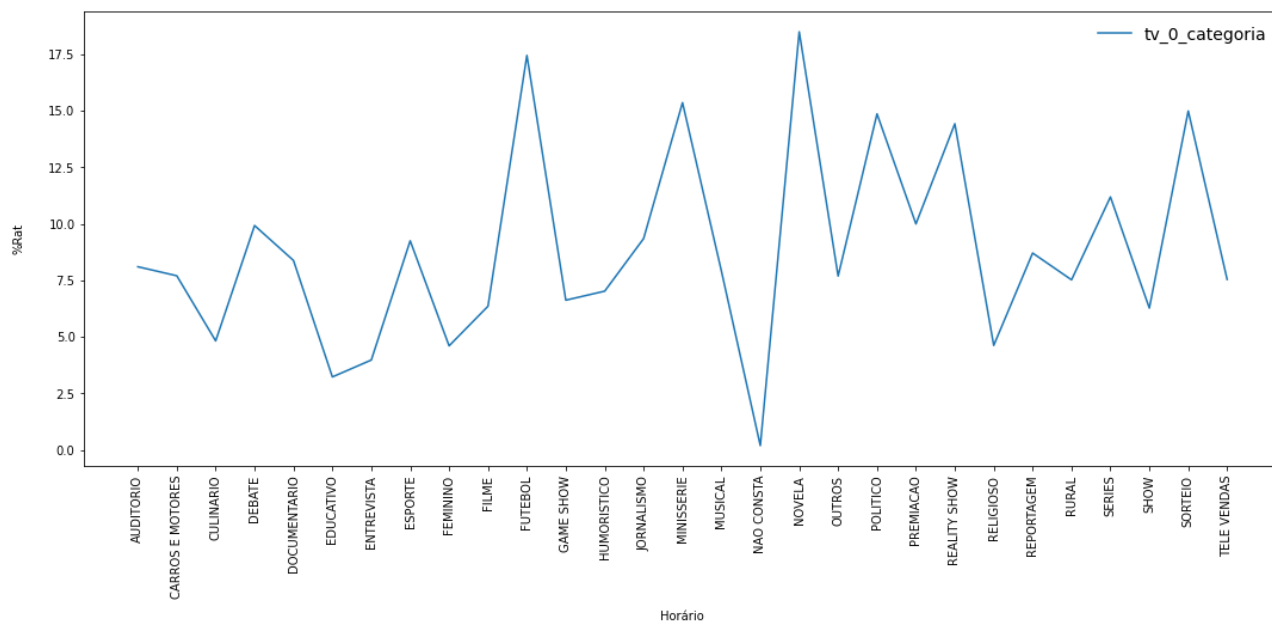
Comparação entre os horários de maior audiência podendo visualizar os dados de audiência em relação a Rede Gazeta e a outros canais para entendermos se os momentos em que temos uma menor audiência na TV Gazeta implica em uma maior audiência em outros canais ou são outros critérios que norteiam a redução da audiência.



d. Análise: Audiência X Categoria

Benefício da análise:

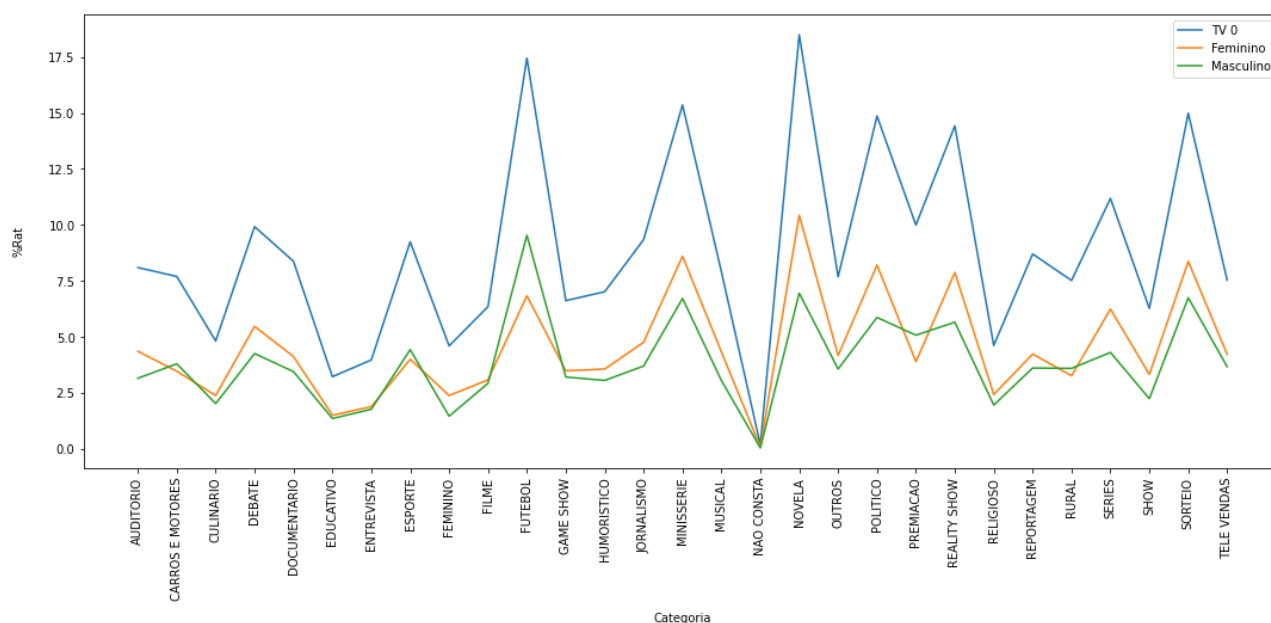
Visualização de qual categoria de programa possui uma maior audiência com base nos dados dos últimos 3 anos. Assim podemos ter uma boa métrica de quais programas ou serviços mais alcançam o público.



Análise: Gênero X Categoria

Benefício da análise:

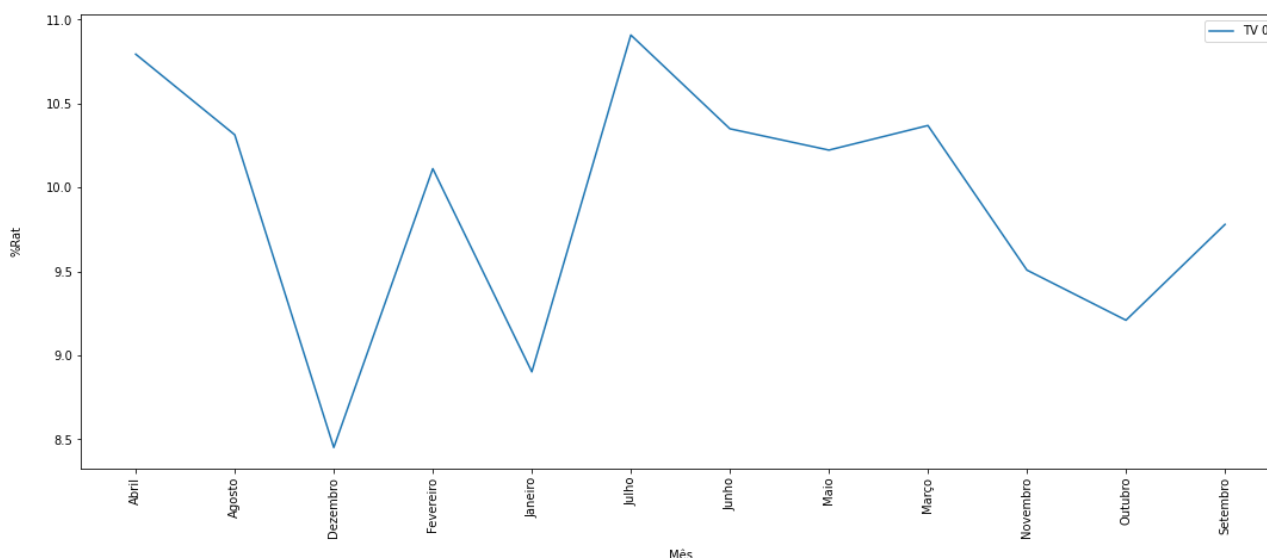
Relação entre a categoria do programa transmitido e gêneros predominantes em cada uma das categorias. Facilita analisar e validar novamente a questão de que é um critério de impacto na audiência e também ter algumas hipóteses sobre quais tipos de programas devem considerar mais ou menos o gênero que deverá ser o público alvo para as chamadas de *marketing*, por exemplo. Para os programas em que o gênero não varia muito em relação a categoria, podemos entender que esse critério é de menor peso para esse tipo de programa. Pois o público é unificado em relação ao gênero.



Análise: Audiência (Total de Domicílios) X Mês

Benefício:

Visualização dos meses nos quais existem maiores ou menores picos de audiência. O que posteriormente pode ser conectado com o tipo de programa (Categoria) ou até mesmo o evento que está sendo transmitido nesse período, assim garante que a visualização terá parâmetros generalizados e não apenas eventos específicos que impactam a audiência pontualmente.



Atributos de interesse (Visualização):

Para todas as situações analisamos a relação entre o parâmetro pré-determinado no item acima por meio do Rat% de audiência por domicílio. Assim têm-se sempre um mesmo parâmetro de referência para poder criar uma intersecção entre dados de um mesmo parâmetro.

Target da previsão:

O *target* da predição é a coluna "audiência", já que o software dará como resultado uma predição de *score* de audiência, taxa de permanência e alcance de público de um programa a ser lançado em determinado horário, definidos a partir dos dados de audiência previamente coletados.

O modelo consiste na entrada de um conjunto de exemplos (dados) que, como mencionado anteriormente, será a audiência com base no histórico e que se baseia em rótulos de valores conhecidos. O grupo tem como objetivo uma saída de um algoritmo de regressão, que é uma função que será usada para prever o valor de rótulo para qualquer novo conjunto de recursos de entrada.

Os dados da coluna de rótulo de entrada devem ser sempre do tipo *Float*. No caso do projeto em andamento, aplicamos a audiência como rótulo principal de entrada buscando definir como as características dos telespectadores.

Os treinadores para esta tarefa produzem a saída contendo a predição desejada de audiência e o quanto cada uma das variáveis (coeficientes de uma função, por exemplo) podem impactar no valor final de audiência.

4.3. Preparação dos Dados

4.3.1 Mesclagem das tabelas:

Para unificar as informações em uma mesma tabela na qual as informações necessárias para rodarmos os modelos de regressão foram centralizadas, realizou-se um processo de mesclagem:

[Link para o código de mesclagem.](#)

Passo 1: Definição dos Dados

Foi selecionada a tabela “Histórico.csv” fornecida pelo parceiro de negócios contendo as informações disponibilizadas pelo Kantar Ibope referentes ao histórico. Em seguida, foi escolhida a tabela “Grade Horária.csv” que contém a grade horária de todas as emissoras em que o grupo está trabalhando para a modelagem de dados, assim como as categorias determinadas pelo Kantar Ibope.

Passo 2: Mesclagem dos Dados (Colab)

Para realizar o processo de união em relação a esses dados, inicialmente foram definidos os intervalos dos bancos na qual será inserida a informação de qual programa está passando naquele período de tempo e qual é a categoria deste programa de acordo com o Kantar Ibope.

Em seguida, selecionadas quais bibliotecas seriam necessárias para realizar a mesclagem:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime
```

Feita a seleção e importação das bibliotecas selecionadas para a aplicação da mesclagem, foram importadas as tabelas referentes a:

- Grade Horária (Programas da: TV 0, TV 1 e TV 2)
- Histórico de Audiência (Dias da Semana, Sábado e Domingo para TV 0, TV 1 e TV 2)

Obtém-se então uma função que recebe um parâmetro genérico como o nome de uma emissora e fornece o banco de dados que mescla as informações de horários em relação a cada um dos programas e categorias.

Em seguida foi feita uma concatenação entre os dias da semana, sábado e domingo. Isso aplicado para o parâmetro “emissora”, que é uma entrada da função.

O próximo passo foi criar um novo banco no qual o grupo terá os dados de todos os dias concatenados e passarão as colunas de “Data” para o formato “*Datetime*” da biblioteca Pandas utilizada.

Feitas as concatenações e a passagem para o formato “*Datetime*”, o grupo ordenará os dados de acordo com a data. Com isso, será possível realizar a mesclagem entre a planilha de grade horária e a de histórico de programas, tendo como base de mesclagem a data e hora do programa.

Por fim, a função gera um arquivo “.csv” contendo uma versão final do banco com os dados todos mesclados em seus respectivos dias, horários e as audiências correspondentes a cada programa.

4.3.2. Anonimização dos dados:

Foram alterados os nomes das emissoras para números ordenados (tv_0, tv_1, tv_2), a fim de manter a confidencialidade dos dados e pela necessidade de descrição ao usar dados externos para comparação. Por motivos de anonimidade, também foram convertidos todos os programas para valores quantitativos, os relacionando com as suas emissoras.

Exemplo: o programa 1, da TV 0, foi convertido em “*programa 0.1*”. Já para o Programa 1 da TV 1, foi convertido para “*programa 1.1*”, e para o Programa 1 da TV 2, convertemos em “*programa 2.1*”. O mesmo padrão foi seguido para os outros programas de cada uma das emissoras.

4.3.3 Feature Engineering:

Padronização dos dados:

Para o início do projeto e durante o seu desenvolvimento, o grupo optou por converter as colunas “Mês”, “Data” e “Dia da Semana”, que originalmente são *strings*, para valores quantitativos, possibilitando a manipulação.

Dessa forma, os dados foram atribuídos da seguinte maneira: Janeiro : 1, Fevereiro: 2, Março: 3, Abril: 4, Maio: 5, Junho: 6, Julho: 7, Agosto: 8, Setembro: 9, Outubro: 10, Novembro: 11, Dezembro: 12. Em relação aos dias da semana, também foi criado um padrão de numeração: Segunda: 1, Terça: 2, Quarta: 3, Quinta: 4, Sexta: 5, Sábado: 6, Domingo: 7.

Em relação a coluna “Data” os dados estavam da seguinte maneira: 2020-06-01 e por isso foram convertidos para manter apenas o ano que neste caso específico é 2020. Quanto ao horário, foi feita a representação para análise por quarto de hora, ou seja, a cada 15 minutos. Assim pode-se obter uma organização horária indicando, por exemplo: 06:00:00, 06:05:00 e 06:10:00 foram convertidos para 6, 06:15:00, 06:20:00 e 06:25:00 foram convertidos para 6.25 (usando os horários em modelo hh:mm:ss e os números de

identificação convertidos como quarto de cem, pois a conversão é em formato de número e não de hora) até o último horário da base de dados, que se encerra com a conversão de 29.75, equivalente ao horário 29:55:00.

Durante a revisão dos atributos do projeto, o grupo notou viabilidade em outra abordagem com o *set* de dados e com o *software* usado para trabalhar as predições.

A anonimização dos dados, Mês, Data e Dia da Semana foram mantidos como na descrição inicial. A alteração feita para a versão final, foi voltada para a margem de tempo, a qual era de 15 em 15 minutos e passou a ser de 5 em 5 minutos, ou seja, o horário das 06:00:00 horas foi convertida para 6.0, o horário das 06:05:00 horas foi convertido para 6.05, 06:10:00 para 6.1, 06:15:00 para 6.15 e seguindo essa lógica até o horário das 29:55:00 horas, que foi convertida para 29.55, também usando os horários em modelo hh:mm:ss.

Valores ausentes ou em branco:

Não foi necessária a realização de nenhuma manipulação dos dados no intuito de tratar os valores ausentes ou nulos, visto que não há dados faltantes.

Seleção dos dados (colunas):

A planilha original foi manipulada de forma a manter somente as colunas que serão utilizadas para criar as diferentes regressões e hipóteses. Como foi decidido inicialmente de trabalhar com o Rat, retiramos as outras colunas que não tinham relação com o mesmo. Sendo assim, os dados mantidos foram: Data, Hora Início, Emissora, Mês, Dia do Mês, Dia da Semana, Total Domicílios, colunas de Rat para cada classe social, coluna de Rat para cada faixa de idade, Programa, Categoria e Faixa

Além disso, foi criada a coluna Horário, esta é feita com base na divisão dos horários em quartos de hora de acordo com a padronização dos dados que já havia sido estabelecida antes e segue os seguintes parâmetros:

- Manhã 1 : 6.0 - 9.75
- Manhã 2: 10.0 - 11.75
- Almoço: 12.0 - 14.75
- Tarde: 15.0 - 17.75
- Noite 1 : 18.0 - 20.75
- Noite 2 : 21.0 - 24.5
- Madrugada: 24.75 -29.75

<https://colab.research.google.com/drive/1LSIG9WWOoMTlwVSdMEcLDaz97mBSDzN9?usp=sharing>

Relação de colinearidade

Neste caso, foi feita a seleção das 3 emissoras, as quais foram executadas em um modelo que calcula a correlação entre todas as variáveis, a partir disso pôde-se ter alguns possíveis parâmetros para rodar a regressão linear.

<https://colab.research.google.com/drive/1gmglZnEj52AHufEtbdusmEGrZcB0Y4KZ?usp=sharing>

g

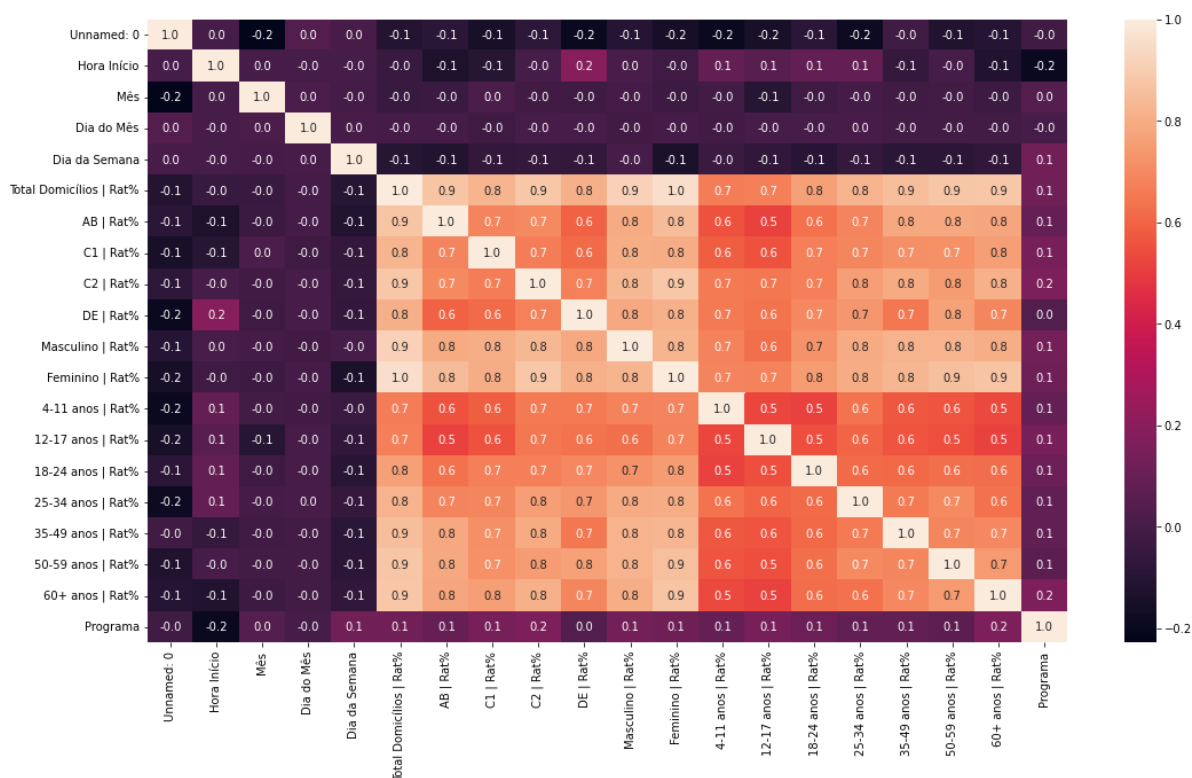


Figura que representa a TV 0

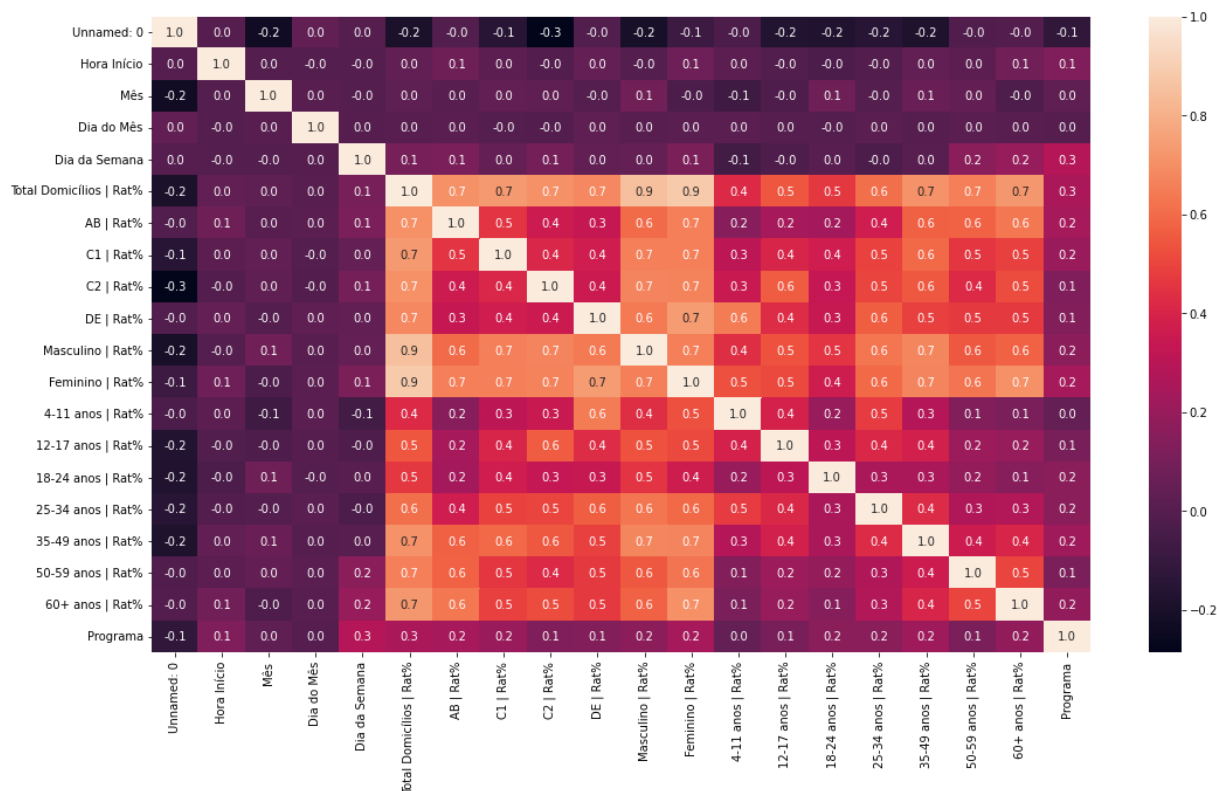


Figura que representa a TV 1

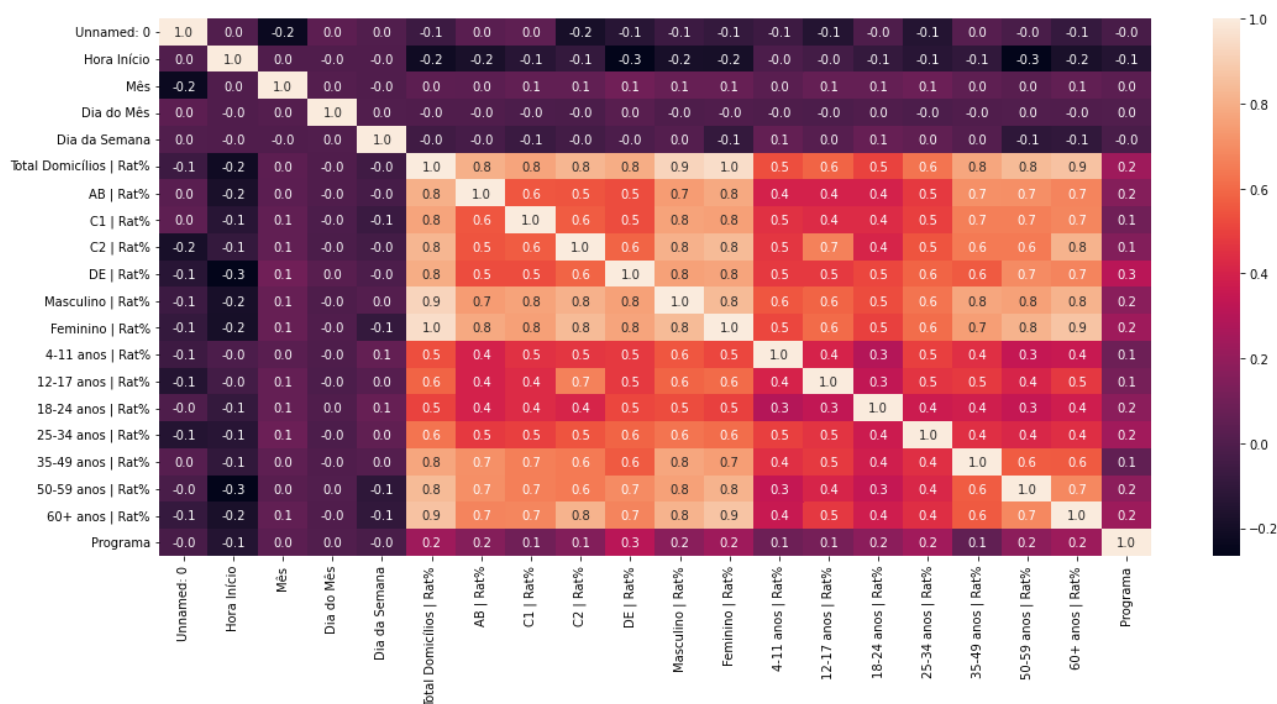


Figura que representa a TV 2

De acordo com o gráfico, quanto mais próximo de 1.0 maior a correlação entre as variáveis representadas no gráfico.

4.4. Modelagem

Primeira etapa de testes:

4.4.1. Primeiro Teste - Regressão Linear:

Modelo utilizado:

Os primeiros testes realizados foram feitos utilizando o modelo de regressão linear. A regressão linear é um algoritmo de predição mais simples e usual. Nesse modelo, utilizam-se duas variáveis para encontrar a relação entre elas. Sendo assim, é preciso definir o que é característica (x) e o que é o valor a ser predito (y). Além disso, também é preciso definir o que vai ser usado para treinar o modelo e o que será usado para testar sua performance.

Ao treinar o modelo, obtém-se uma representação gráfica dos resultados (previsões).

Após treiná-lo, é necessária a avaliação da performance dos resultados, ou seja, o quanto eles fazem sentido para uso. Nessa etapa, existem 3 formas de avaliação:

- **Erro quadrático médio:** essa métrica penaliza mais erros maiores, já que os erros (diferença entre o valor previsto e o correto) são elevados ao quadrado;
- **Erro absoluto médio:** essa métrica faz a média do erro absoluto de cada previsão. Facilita a interpretação no modelo real, mas no caso de existirem erros maiores (*outliers*), estes podem atrapalhar muito a ideia obtida pela média;
- R^2 : o "R quadrado" é uma métrica que varia entre $-\infty$ e 1 e é uma razão que indica o quão bom o nosso modelo está em comparação com um modelo "naive", que faz a predição com base no valor médio do *target*. Quanto mais próximo de 1, melhor é nosso modelo com relação a esse modelo mais simplista.

Nossa hipótese:

Em uma primeira análise de seleção de dados, houve consideração em analisar a influência do horário do programa em relação ao Rat observado durante aquele espaço de tempo. Assim, o grupo decidiu criar um modelo de regressão linear relacionando esses dois dados para cada uma das emissoras.

O modelo gerado:

Criado então um modelo de regressão linear usando a relação hora início x Rat para cada uma das emissoras:

Regressão TV 0:	Regressão TV 1:	Regressão TV 2:
Colab Regressão TV 0	Colab Regressão TV 1	Colab Regressão TV 2

Para construir cada uma das regressões, usou-se o *Dataset* já preparado após o processo de Feature Engineering, onde os dados foram mesclados, anonimizados, padronizados e selecionados.

Para cada uma das regressões, selecionam-se os valores de x e y , ou seja:

- X (características): definido como os valores da coluna “Hora Início”;
- Y (valor a ser predito): definido como os valores da coluna “Total Domicílios I Rat%”

A partir desses valores de x e y , dividem-se as variáveis como dados para treino do modelo e dados para testagem do nosso modelo.

Após isso, inicia o processo de treinamento do modelo e previsões para a variável y , gerando um gráfico dos valores previstos em relação à “hora início”.

Por fim, avalia-se a performance do modelo usando as métricas de erro quadrático médio, erro absoluto médio e R quadrado.

Este procedimento foi executado para cada um dos *datasets* - TV 0, TV 1 e TV 2 - a fim de avaliar a performance do algoritmo em cada uma das emissoras e comparar os resultados obtidos.

Segunda etapa de testes:

4.4.2 Segundo Teste - Random Forest Regressor:

Modelo utilizado:

Random Forest Regressor é um algoritmo que cria diversas árvores (fluxos) de decisão de forma aleatória e combina e compara os resultados delas para chegar em um resultado final do algoritmo. De maneira geral, esse algoritmo cria uma estrutura similar à uma árvore, onde os ramos são diferentes caminhos que o algoritmo segue para chegar em um valor previsto e em cada nó é verificada uma condição e, dependendo da resposta, o *fluxo* segue por um ramo específico.

Esse modelo atende tanto problemas de regressão - caso do projeto - quanto problemas de classificação. Em problemas de regressão, será realizada a média dos valores previstos e este será o resultado. Já para problemas de classificação, o resultado que foi apresentado com mais frequência será o escolhido.

Com esse modelo, é possível compreender de forma mais clara a importância atribuída para cada variável, possibilitando a medição do impacto de cada uma em um resultado final. No entanto, é um modelo limitado, já que apenas uma variável pode ser prevista.

Além disso, com a inserção de novos dados, o modelo irá analisá-los passando pelas diferentes árvores de decisão e cada uma delas dará um resultado.

Nossa hipótese:

Visto que nos primeiros testes realizados os resultados não foram satisfatórios para o objetivo do projeto, o grupo decidiu testar um novo algoritmo fornecendo novos parâmetros.

Dessa forma, espera-se ter uma maior acurácia com esse modelo. Inicialmente, realizou-se o teste deste algoritmo apenas para o dataset da TV 0. Em sequência, foi avaliada a possibilidade de aplicar para as outras emissoras e analisar o comportamento do modelo nesses *datasets*, a fim de comparar o desempenho em relação à TV 0.

O modelo gerado:

[Colab Random Forest - Seção 4.4.2](#)

Primeiro, se define o *dataset* de treino e o *dataset* inteiro, os valores x são os *inputs* que damos ao modelo e o y é a variável que queremos prever:

- X (características): definido como os valores das colunas de categoria do programa, mês, hora início, dias da semana e dia do mês.
- Y (valor a ser retornado): definido como os valores da coluna "Total Domicílios I Rat%"

Depois, foram importadas as bibliotecas necessárias para o modelo e definimos a quantia de estimadores n .

Então, o grupo executou e treinou o modelo dando um *fit* com as variáveis x e y .

O modelo também gerou um gráfico da distribuição dos valores preditos em relação aos valores testados, realizaram-se os cálculos dos erros do modelo e gerou um histograma desses erros calculados.

Ao final, foram calculadas as métricas de R^2 , MAE, MSE e o desvio padrão dos erros do modelo.

4.4.3 Terceiro teste - KNN:

Modelo utilizado:

KNN Regressor é um método não paramétrico que faz aproximação entre variáveis independentes e os valores futuros fazendo uma média das observações mais próximas.

Esse modelo é usado tanto para regressão como classificação, porém ao utilizar um *dataset* com muitas variáveis independentes o modelo começa a se tornar muito lento.

Nossa hipótese:

Com esse algoritmo esperamos que, ao analisar o conjunto de dados, sejam formadas relações de aproximação entre as variáveis independentes e montar uma previsão da audiência futura com base na aproximação desses valores. Inicialmente, testamos este algoritmo apenas para o *dataset* da TV 0. Em sequência, iremos avaliar a possibilidade de aplicar para as outras emissoras e avaliar o comportamento do modelo nesses *datasets*, a fim de comparar o desempenho em relação à TV 0.

O modelo gerado:

[Colab KNN Regressor - seção 4.4.3](#)

Primeiro, definimos o *dataset* de treino e o *dataset* inteiro, os valores x são os *inputs* que damos ao modelo e o y é a variável que queremos prever:

- X (características): foi definido como os valores das colunas de categoria do programa, mês, hora início, dia da semana e dia do mês.
- Y (valor a ser retornado): foi definido como os valores da coluna "Total Domicílios I Rat%"

Depois, importamos as bibliotecas necessárias para o modelo e definimos a quantia de estimadores k .

Para elaborarmos o KNN, seguimos o seguinte processo:

- a) Recebemos um dado não classificado e medimos a distância do novo dado em relação a cada um dos outros dados que já estão classificados;
- b) Selecionamos as K menores distâncias;
- c) Verificamos a(s) classe(s) dos dados que tiveram as K menores distâncias e contabilizamos a quantidade de vezes que cada classe que apareceu;
- d) Classificamos esse novo dado como pertencente à classe que mais apareceu.

Usaremos o *KVV* para entender se esse tipo de algoritmo se adequa ao nosso problema e o quão satisfatório ele será para o resultado esperado.

Nesse modelo, também utilizamos o r^2 como métrica de avaliação para o quanto os resultados obtidos podem ser considerados como bons parâmetros para a medição da audiência, também podendo ser aplicado o cálculo de desvio padrão.

Após, executamos o modelo dando um *fit* com as variáveis x e y de teste e calculamos o score do modelo utilizando as variáveis x e y de treino.

Então, passamos o valor da variável x e foi gerada a predição de y pelo modelo.

Ao final calculamos o valor de r^2 e comparamos os valores que foram preditos com os valores reais de audiência do dataset. Também geramos um gráfico da distribuição dos valores preditos em relação aos valores testados.

4.4.4 Quarto teste - Regressão Linear Múltipla:

Modelo utilizado:

O modelo de regressão linear múltipla se baseia em criar relações entre duas ou mais variáveis independentes para criar uma equação linear. Diferente da regressão linear simples, apresentada no tópico 4.4.1, a regressão linear múltipla utiliza mais variáveis preditoras (no nosso caso, utilizamos mais de 10 variáveis preditoras) para projetar o valor ou peso de uma

variável dependente e para analisar qual conjunto de variáveis traz uma explicação melhor para a variável dependente, que é a referência dos resultados apresentados ainda neste tópico.

Nossa hipótese:

Definindo conjuntos diferentes com as variáveis independentes, pode ser possível conseguir um modelo que consiga gerar previsões com uma boa acurácia. Os primeiros testes foram feitos com apenas um conjunto de variáveis e não obtivemos os resultados desejados. Os próximos passos serão reavaliar os conjuntos e refazer os testes para avaliar o que potencializa e o que prejudica as previsões. Inicialmente, testamos este algoritmo apenas para o *dataset* da TV 0. Em sequência, iremos avaliar a possibilidade de aplicar para as outras emissoras e avaliar o comportamento do modelo nesses *datasets*, a fim de comparar o desempenho em relação à TV 0.

O modelo gerado:

[Colab Regressão Linear Múltipla - seção 4.4.4](#)

Primeiro, definimos o dataset de treino e o dataset inteiro, os valores x são os inputs que damos ao modelo e o y é a variável que queremos prever:

- X (características): foi definido como os valores das colunas de categoria do programa, mês, hora início, dia da semana e dia do mês.
- Y (valor a ser retornado): foi definido como os valores da coluna "Total Domicílios I Rat%"

Depois, importamos as bibliotecas que o modelo usa e definimos uma constante, nesse caso foi a variável x e executamos o modelo dando um fit com as variáveis x e y.

Após, executamos o comando *predict* usando como parâmetro a variável x para prever o valor de y.

Ao final, printamos um sumário com os valores dos resultados da regressão múltipla juntamente com o cálculo das métricas do modelo, onde foi possível visualizar o valor de r^2 e o peso de cada variável na hora de construir a equação.

Terceira etapa de testes:

4.4.5 Quinto teste - XGBOOST

Modelo utilizado:

É uma biblioteca de código aberto de machine learning que atua com modelos de classificação e de regressão. Seu uso no projeto foi o de treinar, testar e realizar uma previsão do alcance de público de uma determinada programação, em uma determinada emissora de televisão. Para isso, foi trabalhado neste modelo, a importação do set de dados, o treinamento com 80% e os testes com 20% deste *set*. Após os treinos e os testes, o modelo realiza uma

predição, a qual foi comparada com o valor real que está no set de dados para comparar sua exatidão.

Nossa hipótese:

Utilizar o modelo com os set de dados, escolher os atributos e trabalhar para que o modelo entenda os dados, treine com eles, permita o grupo realizar os testes necessários com seleções de parâmetros e realize uma predição de acordo com os parâmetros escolhidos pelo grupo. Uma vez tendo-o feito, o grupo se reunirá para decidir se os resultados fornecidos pelo modelo são suficientes para o propósito do projeto.

O modelo gerado:

Para este modelo, foram importadas as bibliotecas necessárias para realizar treinos, testes, predições, geração de gráficos, atuação com métricas e similares, depois realizados os procedimentos de leitura e tratamento do set de dados em formato “.csv”, iniciado o treino do modelo com os dados e testes com parâmetros escolhidos pelos grupos, tais como: categorias, data, hora, mês, dia do mês, dia da semana e data como eixo “x” (variáveis preditoras) e o total domicílios I Rat% como eixo “y” (“*target*”). Após esse processo, foi realizada a predição com o modelo e, a partir daí, a análise dos dados.

Os hiperparâmetros usados para os testes foram: $n_splits = 10$, $n_repeats = 3$.

4.4.6 Sexto teste - CATBOOST Regressor

Modelo utilizado:

CatBoost é um modelo de machine learning usado para aprimorar árvores de decisão. Seu uso consiste no aprimoramento de algoritmos de regressão e seu foco é para pesquisadores e engenheiros, na área da computação e sistemas. Para o projeto desenvolvido, o modelo foi utilizado com o intuito de realizar predições de alcance de público para uma determinada programação, baseado na combinação de parâmetros de um *set* de dados fornecido pelos parceiros de projeto e organizado em categorias, datas e horas.

Nossa hipótese:

Ao utilizar o modelo com os parâmetros mencionados acima, o objetivo era que ele aprendesse a realizar análise dos dados. Após o treinamento com esses dados, o teste retornaria uma predição de audiência para uma determinada linha do *set* de dados.

O modelo gerado:

Para este, foi importada e instalada manualmente a biblioteca de regressão do CatBoost. Após, foi estruturada a parametrização de treino com 80% e de teste com 20% de do dataset.

Concluindo a parametrização, foi realizado o treinamento do modelo com os parâmetros de categorias, dias da semana, hora início, mês, dia do mês e data como eixo x (variáveis preditoras) e “total domicílios I Rat%” como eixo y (“*target*”).

Após o treinamento do algoritmo, foi realizado o teste de regressão do modelo com base no aprendizado e, em seguida, realizada a predição com base no teste, utilizando o parâmetro de saída “total domicílios I Rat%” (eixo y).

4.4.7 Sétimo teste - Gradient Boosting

Modelo utilizado:

Gradient Boosting é um modelo de machine learning supervisionado que consiste na utilização de algoritmos em aplicações de regressão e classificação. Este modelo foi idealizado com a intuição de prever e acertar baseado na combinação de modelos anteriores, o aprendizado com seus erros e a maximização de sua previsão de maneira gradativa.

Nossa hipótese:

Ao utilizar este modelo, o grupo visava utilizar todos os parâmetros e conseguir uma precisão de R^2 mais alta, para que este modelo pudesse ser utilizado com confiança em relação ao dataset fornecido.

Para testar esse modelo utilizamos na variável X as seguintes entradas: Ano, Mês, Dia da Semana, Dia do mês, Categoria e Hora Início. A variável Y é a coluna “Total Domicílios I Rat%”.

O modelo gerado:

A estrutura para a utilização deste modelo foi estruturada a partir da importação dos dados e das bibliotecas necessárias para realizar as métricas e os *plots* dos gráficos, logo após, foi feita a preparação dos dados para o modelo, onde foram separadas as variáveis preditoras de categorias: dia da semana, dia do mês, mês e hora início e a variável de y, que é o *target* foi o Total domicílios I Rat% e o *set* de dados foi organizado em 70% para treino e 30% para teste. Após a preparação dos dados, foi realizado o treino e o teste e, consequentemente, a predição através dos dados com o modelo, ao qual geraram resultados de r^2 , MAE, MSE e um gráfico de dispersão, que serão apresentados no tópico de avaliação 4.5.7.

4.4.8 Oitavo teste - Histogram Gradient Boosting

Modelo utilizado:

O histogram gradient boosting é um dos modelos de machine learning usado para classificação e para regressão, assim como alguns dos modelos testados e mencionados aqui no tópico 4.4*. Seu funcionamento é muito próximo ao do CatBoost Regressor, com um ponto negativo de que ele leva mais tempo para realizar os treinos e predições. Para o projeto desenvolvido, o modelo foi utilizado com o intuito de realizar predições de alcance de público para uma determinada programação, baseado na combinação de parâmetros de um *set* de dados fornecido pelos parceiros de projeto e organizado em categorias, datas e hora.

Nossa hipótese:

Utilizar o modelo com os parâmetros do *set* de dados para que ele aprenda a realizar a análise de dados, treinar com eles e realizar o teste e realizar uma predição de audiência com base em determinadas combinações de parâmetros do *set* de dados.

O modelo gerado:

Com este modelo, foram importadas as bibliotecas necessárias do modelo e dos cálculos e estruturados os parâmetros de treino com 80% e de teste com 20% dos dados. Após isso, foi feita a separação parâmetros do eixo x (categorias, dias da semana, hora início, mês, dia do mês e data) e do eixo y (total domicílios I Rat%), como o nosso *target*.

Com tudo montado, foi realizado o teste de regressão do modelo com base no aprendizado e realizada a predição, utilizado como parâmetro de saída, o "total domicílios I Rat%" (eixo y).

4.4.9 Nono teste - LightGBM

Modelo utilizado:

LGBM é uma estrutura de levantamento de gradiente rápida, distribuída e de alto desempenho que se baseia em um algoritmo de aprendizado de máquina popular - a árvore de decisão. Ele pode ser usado em classificação, regressão e muitas outras tarefas de aprendizado de máquina. Este algoritmo cresce em termos de folha e escolhe o valor delta máximo para crescer. LightGBM usa algoritmos baseados em histograma. As vantagens disso são as seguintes:

- Menos uso de memória
- Redução no custo de comunicação para aprendizagem paralela
- Redução no custo para calcular o ganho para cada divisão na árvore de decisão.

Assim, como o LightGBM é treinado muito mais rápido, também pode levar ao caso de sobreajuste às vezes. Portanto, vamos ver quais parâmetros podem ser ajustados para obter um modelo ideal melhor.

Para obter o melhor ajuste, os seguintes parâmetros devem ser ajustados:

1. `num_leaves`: Uma vez que LightGBM cresce em folha, este valor deve ser menor que $2^{\text{max_depth}}$ para evitar um cenário de sobreajuste.
2. `min_data_in_leaf`: Para grandes conjuntos de dados, seu valor deve ser definido em centenas a milhares.
3. `max_depth`: um parâmetro-chave cujo valor deve ser definido de acordo para evitar sobreajuste.

Para obter uma melhor precisão, os seguintes parâmetros devem ser ajustados:

- Mais dados de treinamento adicionados ao modelo podem aumentar a precisão. (também podem ser dados externos não vistos)
- `num_leaves`: aumentar seu valor aumentará a precisão conforme a divisão está ocorrendo em relação às folhas, mas também pode ocorrer overfitting.

- `max_bin`: o valor alto terá um grande impacto na precisão, mas acabará resultando em sobreajuste.

Nossa hipótese: Sabendo que o LightGBM consiste em um modelo de decisão pensando em uma árvore de decisão. Nosso objetivo com esse modelo foi conseguir definir bons parâmetros de testes (pensando nos parâmetros inicialmente estabelecidos pelo modelo anteriormente).

Após a definição de parâmetros buscamos aplicar nossa base de dados para que por meio dos valores de teste e treino pudéssemos obter o melhor resultado possível para o nosso R^2 . Focando em definir se o LightGBM é o melhor método para obtermos resultados de predição mais similares em relação aos dados de audiência reais que existem em nosso banco.

O modelo gerado: A partir da definição de parâmetros para o modelo e dos *sets* definidos para treino e teste, rodamos o LightGBM para o eixo x sendo: Categoria, Ano, Dia da Semana, Dia do Mês e Horário. E para o eixo y sendo: Total Domicílios I Rat%.

Realizados os testes, obtivemos os resultados de R^2 (TESTE): 98.51% e R^2 (TREINO): 99.43%.

Quarta etapa de testes:

Nesta etapa, apenas foi corrigido o dataset utilizado para os modelos da **segunda etapa de testes**, já que foi notada a falta da coluna 'Data' no dataset utilizado até aquele momento. Visto que essa coluna é uma característica muito importante para a predição, a mesma foi incluída e o dataset antigo foi substituído pelo novo. Além disso, a coluna 'Data' também foi incluída nos valores de X (características) para as predições.

Sendo assim, na tabela a seguir, podemos ver os valores de R^2 e erros antigos e os atualizados para os modelos feitos na **segunda etapa de testes** são:

	Random Forest Regressor	KNN	Regressão Linear Múltipla
Valores antigos	R^2 : 97,60% MAE: 0,65 MSE: 1,02	R^2 : 98,12% MAE: 0,60 MSE: 0,80	R^2 : 42%
Valor corrigido	R^2 : 98,33% MAE: 0,57 MSE: 0,71	R^2 : 98,14% MAE: 0,59 MSE: 0,79	R^2 : 44%

Tabela para os modelos com o dataset corrigido - incluindo 'Data' x Sem coluna 'Data'.

Ao término das etapas de testes e análise de cada modelo com testes de hiperparâmetros, o grupo decidiu que o modelo mais viável para ser usado foi o LightGBM, pois

possuiu a maior acurácia nos treinos e testes e as menores taxas de dispersão e desvio padrão dentre todos os modelos testados e avaliados. Seu funcionamento consiste em uma estrutura leve e eficiente para alavancar o desempenho do algoritmo produzido pelo usuário e melhorar a velocidade e a precisão do resultado de predição do algoritmo. O modelo opera primordialmente através de árvore de decisão e segue em direção aos valores mais próximos do que o último valor encontrado, aumentando assim a precisão e a acurácia de teste, treino e predição.

Tratando-se de informações técnicas sobre o modelo, veja que:

```
hyper_params = {
    'task': 'train' (essa é a função do algoritmo, que é o treino)
    'boosting_type': 'gbdt', (é a definição do tipo de modelo, definido como regressão gbd,
entre as opções "gbdt", "rf", "dart" e "goss")
    'objective': 'regression', (objetivo do modelo, executar regressão)
    'metric': ['l1', 'l2', 'rmse'], (atributos de métricas da biblioteca LightGBM, contendo erro
quadrático médio e médio absoluto)
    'learning_rate': 0.04, (é a velocidade com que o modelo vai desenvolver seu
aprendizado)
    'feature_fraction': 1, (é a porcentagem do dataframe que será utilizada)
    'verbose': 0, (é a ferramenta de debug do modelo, usada para identificar e retirar
possíveis bugs)
    'max_depth': 60, (é um parâmetro que limita o tamanho de cada árvore)
    'num_leaves': 2048, (é um parâmetro que limita a quantidade de folhas em cada árvore)
    'num_iterations': 5_000, (é a quantidade máxima de tentativas de árvores que o modelo
deve chegar)
}
```

Tendo essas informações, as taxas de erro obtidas com este modelo foram um MAE de 0,55 e MSE de 0,63 para o teste; e MAE de 0,35 e MSE 0,24 para o treino, com desvio padrão de 0,72.

4.5. Avaliação

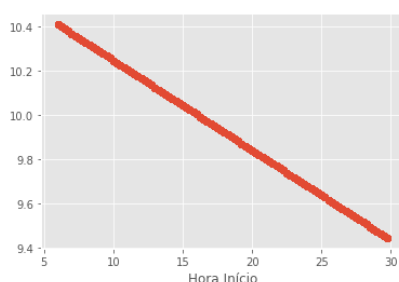
4.5.1: Avaliação dos resultados do primeiro teste:

Como foi visto na seção 4.4.1, utilizamos o modelo de regressão linear usando as colunas “Hora Início” e “Rat” e obtivemos os seguintes resultados:

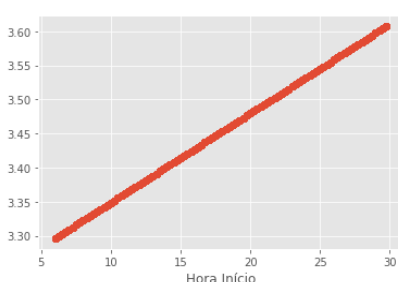
Resultados:

Na tabela a seguir, podemos observar os resultados da avaliação do modelo em cada dataset, usando as 3 métricas mencionadas anteriormente:

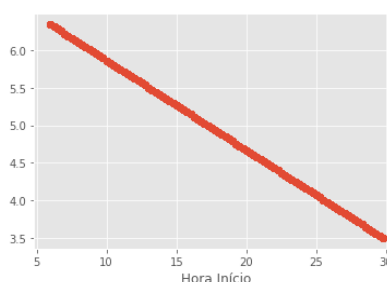
-	TV 0	TV 1	TV 2
Erro Quadrático	42.45	6.13	13.87
Erro Absoluto	5.23	1.92	3.08
R²	0.00%	0.17%	0.05%



(tv_0)



(tv_1)

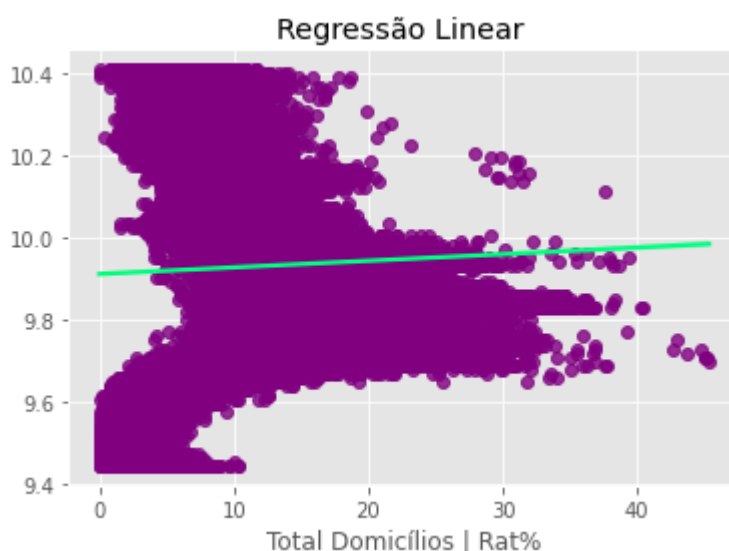


(tv_2)

Os gráficos acima apenas ilustram a reta dos valores reais de audiência.

Os resultados obtidos com esse algoritmo não possuem uma qualidade significativa, já que consideramos apenas uma variável como parâmetro (x) para a execução dos testes e das previsões. Além disso, não foram desconsiderados outliers e os dados também não foram normalizados, o que pode ter interferido na qualidade da predição.

É possível ver a seguir, o gráfico da regressão linear com apenas a hora início como entrada.



Como notado no gráfico acima, não há linearidade nenhuma em relação aos valores ideais, o que prova que tal modelo não é eficiente.

O modelo testado não resultou de forma satisfatória ao nosso problema, visto que tivemos uma acurácia muito baixa em todos os datasets e os parâmetros utilizados foram insuficientes e pouco relevantes para a nossa predição.

4.5.2: Avaliação dos resultados do segundo teste:

Como foi visto na seção 4.4.2, utilizamos o modelo Random Forest Regressor e obtivemos os seguintes resultados:

Resultados:

Ao final da predição, foi possível obter a acurácia e os erros para os conjuntos de treino e teste, além do cálculo do desvio padrão dos erros do conjunto de testes. O modelo foi testado e treinado com o dataset TV 0 e, na tabela apresentada a seguir, é possível ver os resultados dos cálculos:

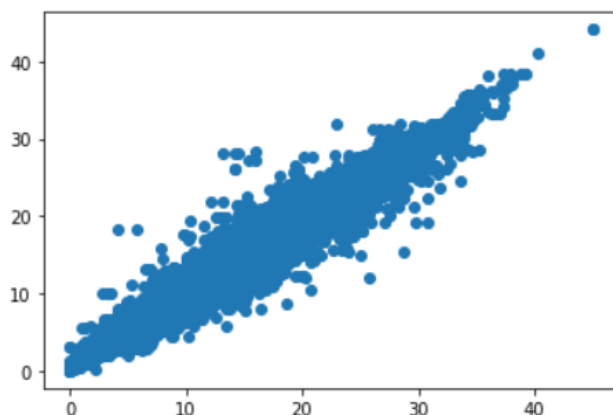
Acurácia, MAE e MSE para conjunto de treino e teste.

	R^2	MAE	MSE
y_train	99,43%	0,34	0,24
y_test	98,33%	0,57	0,71

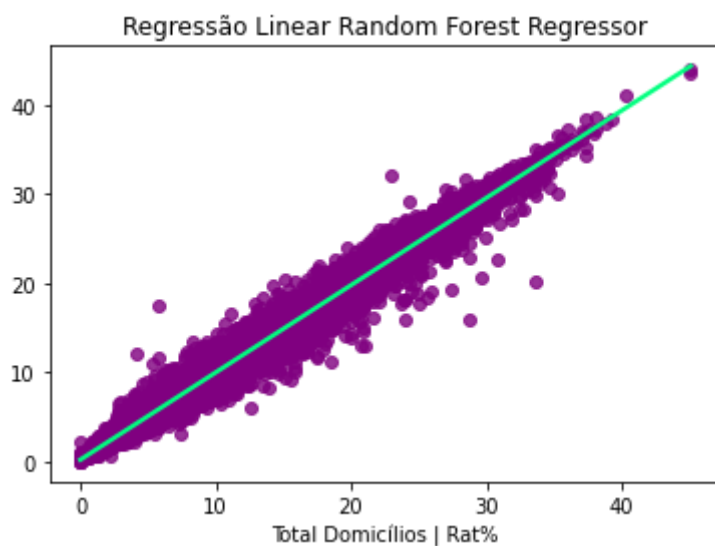
Desvio padrão: 0,84

Os resultados obtidos nessa regressão foram extremamente satisfatórios, visto que o mesmo apresentou uma alta acurácia, baixos índices de erro e um desvio padrão baixo para o nosso dataset.

Em sequência, é possível visualizar a primeira versão feita do gráfico de dispersão, onde no eixo y há o valor predito pelo modelo com o conjunto de teste e no eixo x os valores reais.



Já o próximo gráfico é o mais recente feito usando a biblioteca Seaborn. Este é um gráfico da regressão linear para o modelo. No eixo x são os valores reais e no eixo y são os valores preditos



Esse modelo é bastante promissor para o nosso problema, tendo em vista os resultados obtidos. Até então, este é o modelo que mais se adequa ao nosso problema e ao nosso objetivo de predição. Porém, ao lidarmos com mais de uma variável no eixo "y", será necessário realizar mais testes para medir a acurácia do algoritmo de forma mais detalhada.

Os nossos próximos passos serão trabalhar o modelo de teste e treino, avaliar o dataset para o modelo e vice-versa para aprimorarmos a implementação da solução. Além disso, foi identificado que as predições nestes primeiros testes tiveram seus desvios para mais quando comparados aos valores reais. Ainda não chegamos a uma resposta sobre esse acontecimento, mas estudaremos para entendê-la e corrigi-la.

4.5.3: Avaliação dos resultados do terceiro teste:

Como foi visto na seção 4.4.3, utilizamos o modelo KNN e obtivemos os seguintes resultados:

Resultados:

Ao final da predição, foi possível obter a acurácia e os erros para os conjuntos de treino e teste. Inicialmente, não foi possível calcular o valor do desvio padrão visto que o modelo utiliza mais memória RAM do que o disponível no Google Colab, impossibilitando o cálculo. Porém, após outros testes, foi possível realizar o cálculo do desvio padrão dos erros do conjunto de testes.

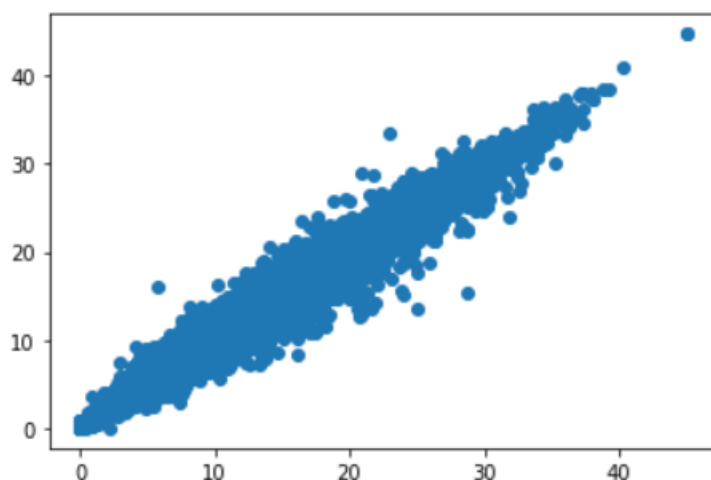
Acurácia, MAE e MSE para conjunto de treino e teste.

	R^2	MAE	MSE
y_train	99,32%	0,36	0,29
y_test	98,14%	0,59	0,79

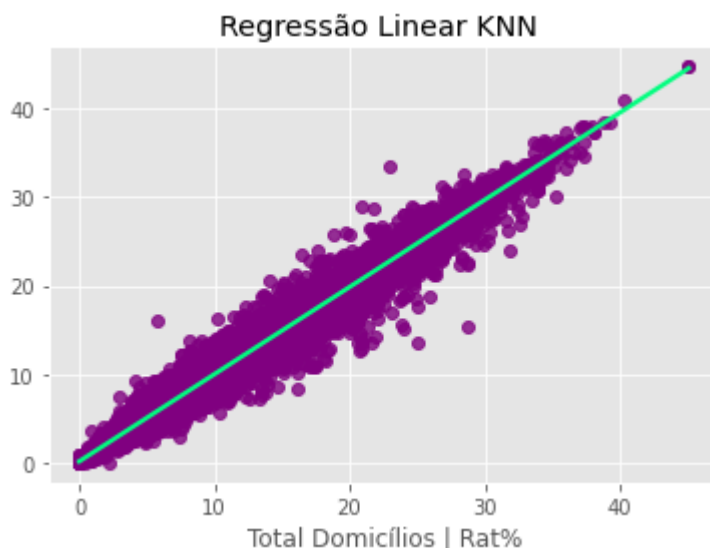
Desvio padrão: 0,94

Os resultados obtidos nessa regressão foram extremamente satisfatórios, visto que o mesmo apresentou uma alta acurácia, baixos índices de erro e um desvio padrão baixo para o nosso dataset.

No próximo gráfico a ser apresentado, será possível visualizar a primeira versão feita do gráfico de dispersão, onde no eixo y há o valor predito pelo modelo com o conjunto de teste e no eixo x os valores reais:



Já o próximo gráfico é o mais recente feito usando a biblioteca Seaborn. Este é um gráfico da regressão linear para o modelo. No eixo x são os valores reais e no eixo y são os valores preditos



Ao comparar esse gráfico com o do Random Forest, notamos que o mesmo é mais assertivo e tem menos outliers, apesar de ambos terem o R^2 similar.

O modelo KNN é promissor para o nosso objetivo, tendo em vista que o R^2 gerado resultou em um valor alto, porém faltam dados para verificar se ele realmente tem uma acurácia alta, será necessária a realização de mais testes para verificar sua acurácia e taxas de erro.

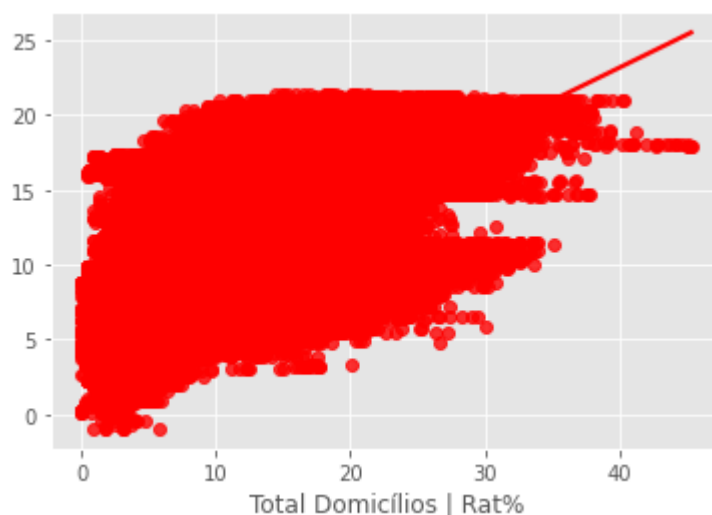
4.5.4. Avaliação dos resultados do quarto teste:

Como foi visto na seção 4.4.4, utilizamos o modelo de regressão linear múltipla e obtivemos os seguintes resultados:

Resultados:

Ao final da predição, foi possível obter a acurácia para o conjunto de testes. O modelo apresentou um R^2 de 44%. Isso mostra que tal modelo não é tão eficiente, visto que os modelos desenvolvidos até então apresentaram R^2 acima de 90%. Ambos, peso de y (“Total Domicílios | Rat%”) e peso de x (Múltiplas variáveis), têm como fundamento serem multiplicadores, que trabalham a eficácia de uma predição baseada em uma listagem prévia de dados, aplicando assim operações as quais influenciam no output. Além disso, se mostrou ser um modelo limitado, já que não foi possível calcular os erros para ele.

Esse modelo não será usado, visto que o mesmo apresentou uma acurácia muito baixa para o uso que precisamos e com os dados que temos.



Visto a quantidade de outliers, o grupo descartou o uso do modelo, visto que é algo que afeta diretamente na acurácia e na análise de dados

4.5.5. Avaliação dos resultados do quinto teste:

Como foi visto na seção 4.4.5, utilizamos o modelo XGBOOST e obtivemos os seguintes resultados:

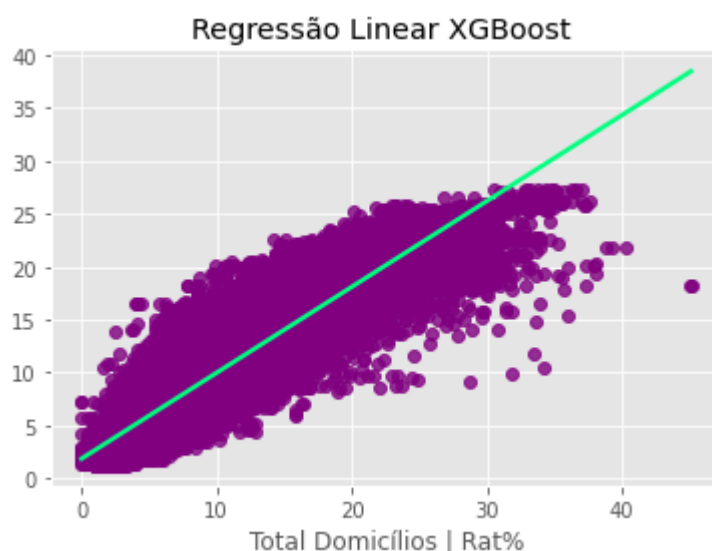
Resultados:

Ao final da predição, foi possível obter a acurácia e os erros para o conjunto de teste. O modelo foi testado e treinado com o dataset TV 0 e, na tabela, é possível ver os resultados dos cálculos:

Acurácia, MAE e MSE para o conjunto de teste.

R ²	MAE	MSE
84,71%	1,85	6,50

Com o gráfico a seguir e utilizando a biblioteca Seaborn, foi possível visualizar a dispersão dos dados em relação aos valores ideais (reta)



Desta forma ao avaliar esse modelo de machine learning não teve resultados satisfatórios e será desconsiderado.

4.5.6. Avaliação dos resultados do sexto teste:

Como foi visto na seção 4.4.6, utilizamos o modelo CATBOOST e obtivemos os seguintes resultados:

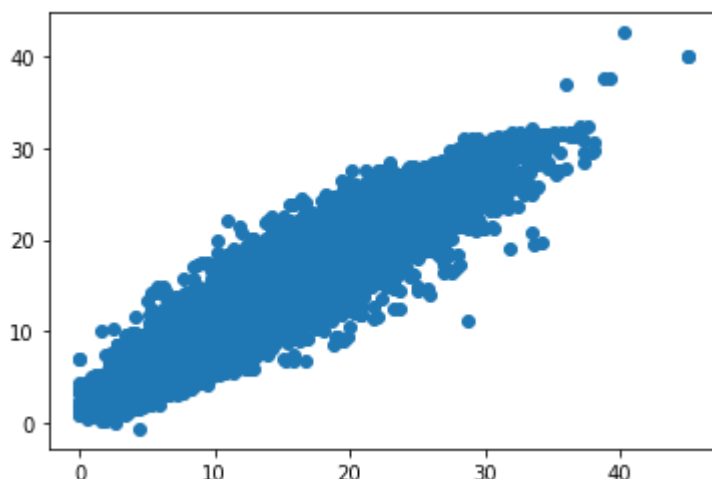
Resultados:

Ao final da predição, foi possível obter a acurácia e os erros para o conjunto de teste. O modelo foi testado e treinado com o dataset TV 0 e, na tabela, é possível ver os resultados dos cálculos:

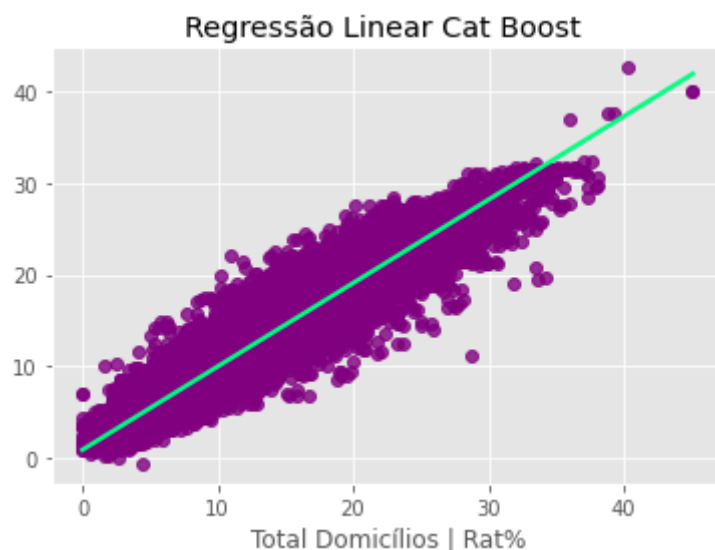
Acurácia, MAE e MSE para o conjunto de teste.

R ²	MAE	MSE
91,55%	1,42	3,59

Além disso, foi plotado um gráfico de dispersão dos dados no estilo *scatter*, tendo como resultado do gráfico com uma dispersão equilibrada em comparação aos modelos de Random Forest Regressor e KNN Regressor, como apresentado a seguir.



Um gráfico também foi plotado usando a biblioteca Seaborn e representa a regressão linear do modelo e como o mesmo se comporta comparando as previsões com o valor real.



Observando o gráfico, é possível notar que o modelo possui uma grande dispersão comparado aos outros.

Os resultados deste modelo de regressão não foram tão satisfatórios quando comparados aos resultados de outros modelos, visto que a acurácia foi menor que 95%.

Com os dados resultantes do treino e teste dos dados, as métricas foram insatisfatórias aos resultados já obtidos e isso tornou o modelo não propício para o projeto em execução.

4.5.7. Avaliação dos resultados do sétimo teste:

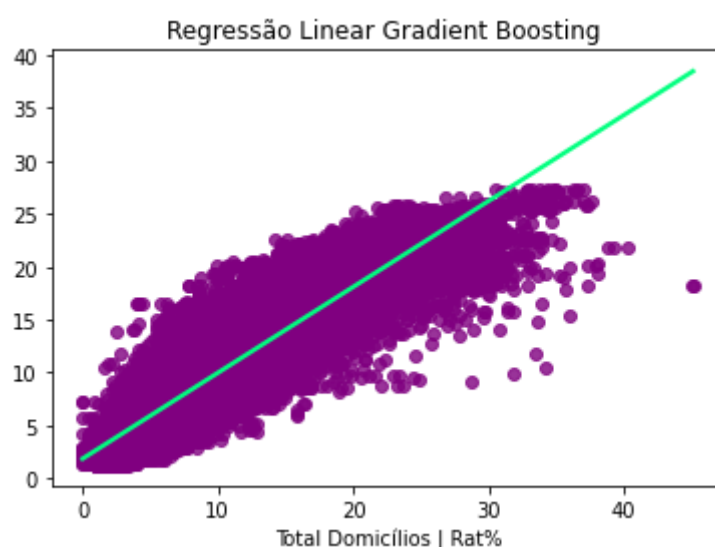
Como foi visto na seção 4.4.7, utilizamos o modelo Gradient Boosting e obtivemos os seguintes resultados:

Qualidade dos resultados:

Ao final da predição, foi possível obter a acurácia e os erros para o conjunto de testes. O modelo foi testado e treinado com o dataset TV 0 e, na tabela, é possível ver os resultados dos cálculos:

R ²	MAE	MSE
84.7 %	1.85	6.50

Plotando o gráfico da regressão linear para este modelo, obtivemos a seguinte dispersão:



É possível visualizar que o modelo gerou uma quantia significativa de *outliers*, o que impossibilita seu uso.

Este modelo possui uma certa facilidade em seu uso, porém com a acurácia abaixo dos 90% e as margens de erro tão altas, não foi eficaz o suficiente para ser continuado, pois já existem modelos com melhores resultados e com uma adaptação melhor ao *set* de dados.

4.5.8. Avaliação dos resultados do oitavo teste:

Como foi visto na seção 4.4.8, utilizamos o modelo Histogram Gradient Boosting e obtivemos os seguintes resultados:

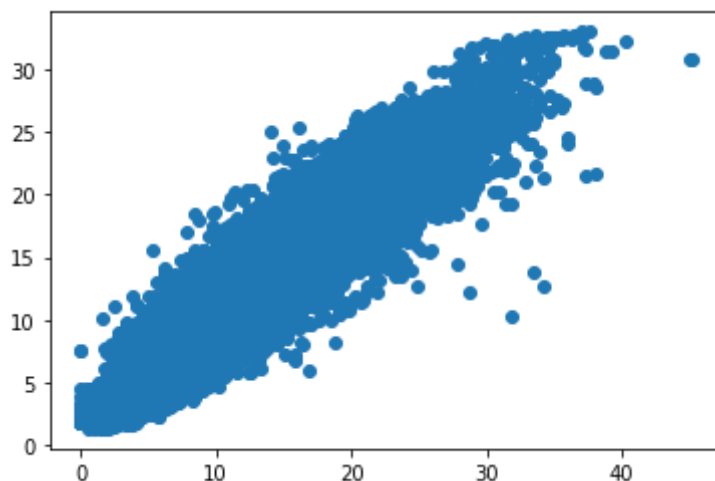
Resultados:

Ao final da predição, foi possível obter a acurácia e os erros para o conjunto de teste e, na tabela, é possível ver os resultados dos cálculos:

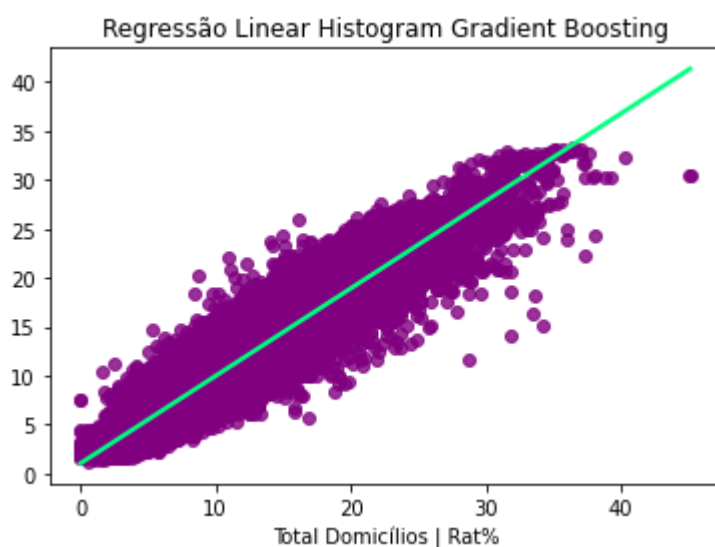
Acurácia, MAE e MSE para o conjunto de teste.

R^2	MAE	MSE
91,32%	1,44	3,69

Também foi gerado um gráfico de dispersão com uma margem de erro mais preenchida para a parte superior da reta de valores ideais, indicando que a margem de erro pode ser sempre para cima nas previsões do modelo, como pode ser visto à seguir:



Posteriormente, foi plotado um gráfico de regressão linear utilizando a biblioteca Seaborn, onde foram obtidos os seguintes resultados.



Apesar da acurácia ter sido relativamente alta, as taxas de erro se mantiveram muito altas em relação aos modelos anteriores.

Além disso, nota-se que o modelo é bem lento para execução de treino, teste e predição, além de não ter apresentado resultados bons o suficiente para dar continuidade ao projeto utilizando-o como um dos modelos principais.

4.5.9. Avaliação dos resultados do nono teste:

Assim como indicado na seção 4.4.9, utilizamos o modelo LightGBM e obtivemos os seguintes resultados:

Resultados:

Foi realizada a aplicação do modelo utilizando os seguintes parâmetros:

```
'task': 'train', #função
'boosting_type': 'gbdt', #tipo da regressão (gbdt, rf, dart, goss)
'objective': 'regression', #tipo do modelo
'metric': ['l1', 'l2', 'rmse'], #erro médio quadrático e erro médio absoluto
'learning_rate': 0.04, #velocidade do aprendizado
'feature_fraction': 1, #porcentagem do dataframe a ser utilizado
'verbose': 0, #usado para retirar possíveis bugs existentes (debug)
"max_depth": 60, #limita o tamanho de cada árvore
"num_leaves": 2048, #limita o número de folhas que cada árvore pode ter
"num_iterations": 5_000, #quantidade de tentativas
```

A partir desses parâmetros, o modelo foi executado para os valores de testes, obtendo os seguintes valores para os erros em relação ao valor de teste e os valores reais da base de dados:

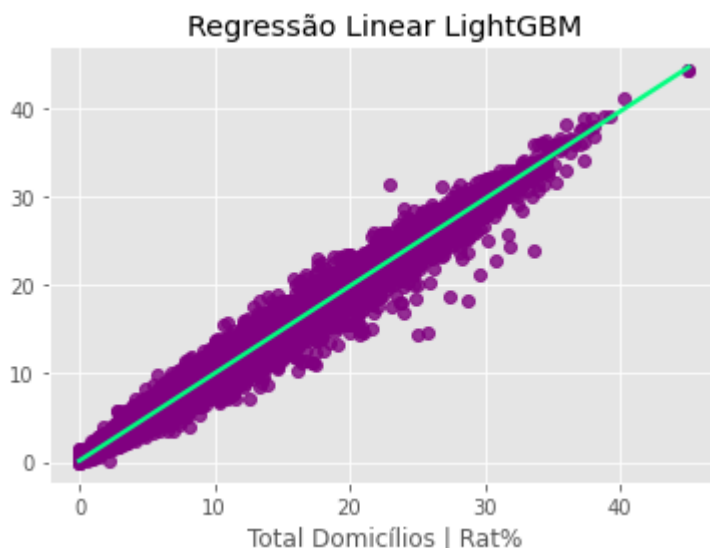
R ²	MAE	MSE
98,51	0,55	0,63

Da mesma forma, realizou-se a aplicação do modelo para os valores de X e Y de treino, seguindo os mesmos parâmetros de *input* e *output* da modelagem para os valores de X e Y de teste.

Obtendo então os seguintes resultados:

R ²	MAE	MSE
99,43%	0,35	0,24

A partir dos resultados obtidos em relação aos erros para os modelos tanto de teste quanto de treino, pudemos realizar a plotagem do seguinte gráfico:



O gráfico acima demonstra em seu centro a reta de regressão linear ideal para o nosso modelo em relação ao *dataset*. Enquanto os pontos em roxo são os valores estabelecidos para o modelo de teste e assim pode-se verificar o quanto os valores preditos se aproximam dos valores reais para o modelo.

Verificando então de forma visual a relação já previamente estabelecida pelos valores de R^2 (98,51%) que são responsáveis por fornecer o quanto os valores preditos se aproximam dos valores reais, ou seja, a acurácia do modelo.

Analisando cada um dos nove modelos testados durante o módulo, foi possível avaliar diferentes parâmetros para o mesmo modelo e realizar comparações entre eles, comparar os modelos entre si e discutir entre o grupo os 3 melhores modelos. Após essa escolha, foi feita uma nova análise e comparação intra e inter modelos e decidido através das métricas, qual seria o modelo ideal para utilização do parceiro do projeto após a entrega. Assim foi definido o modelo LightGBM, por sua acurácia alta e taxas de erro mais equilibradas e baixas, além de uma dispersão muito menor que os outros dois modelos de alta precisão obtidos pelo grupo.

A seguir, é possível saber quais os possíveis erros que podem ocorrer ao utilizar o modelo e como lidar com cada um deles:

- **Rodar o modelo sem um parâmetro:** caso o usuário realize os *inputs* sem definir um dos parâmetros, o modelo rodará com sua respectiva informação em "0". Para evitar isso, analise se todos os *inputs* estão inseridos corretamente;

4.6 Comparação de Modelos

4.6.1 Interpretando as métricas de erro:

Para a avaliação da performance de cada modelo, foram utilizadas algumas métricas para cálculo de erro: o MSE, MAE e desvio padrão. Assim, foi possível fazer uma comparação mais aprofundada de cada modelo feito.

- **Calculando o Erro Médio Absoluto (MAE):**

O erro médio absoluto irá calcular uma média de todos os valores obtidos, e para cada valor obtido é calculado o módulo da subtração *valor - média*. Ao final, os resultados são somados e resultam no erro médio absoluto.

No entanto, caso haja muitos valores *outliers*, essa métrica não dará peso a eles na e não impactarão tanto no resultado final.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i|$$

Equação do MAE

- **Calculando o Erro Quadrático Médio (MSE):**

O erro quadrático médio é mais uma métrica utilizada para avaliar a acurácia dos modelos. O MSE dá um peso maior aos erros do modelo, visto que cada erro é elevado ao quadrado e, por fim, é calculada uma média desses erros quadráticos.




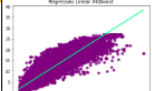
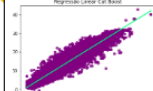
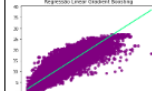

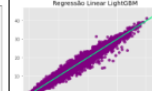
Essa métrica é bastante sensível a *outliers* (valores discrepantes), já que esses erros serão elevados ao quadrado. Além disso, caso haja muitos erros significativos, o valor obtido pelo cálculo poderá ser maior.

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

Equação do MSE

4.6.2 Tabela e colab de todos os modelos

Foi gerada uma tabela de comparação de todos os modelos testados, com seus respectivos parâmetros, R^2 resultado, erros (MSE e MAE) e gráficos de dispersão:

Características	Modelo 2: Random Forest	Modelo 3: KNN	Modelo 4: Regressão Multipla	Modelo 5 : XGBoost	Modelo 6: CATBOOST	Modelo 7: Gradient Boosting	Modelo 8: Histogram Gradient Boosting	Modelo 9: LightGBM
Entrada (x)	Categoria_AUDITORIO', 'Categoria_CARROS E MOTORES', 'Categoria_CULINARIO', 'Categoria_DEBATE', 'Categoria_DOCUMENTARIO', 'Categoria_EDUCATIVO', 'Categoria_ENTREVISTA', 'Categoria_ESPORTE', 'Categoria_FEMININO', 'Categoria_FILME', 'Categoria_FUTEBOL', 'Categoria_GAME SHOW', 'Categoria_HUMORISTICO', 'Categoria_JORNALISMO', 'Categoria_MINISERIE', 'Categoria_MUSICAL', 'Categoria_NAO CONSTA', 'Categoria_NOVELA', 'Categoria_OUTROS', 'Categoria_POLITICO', 'Categoria_PREMIACAO', 'Categoria_REALITY SHOW', 'Categoria_RELIGIOSO', 'Categoria_REPORTAGEM', 'Categoria_RURAL', 'Categoria_SERIES', 'Categoria_SHOW', 'Categoria_SORTEIO', 'Categoria_TELE VENDAS', 'Mês', 'Hora Início', 'Dia da Semana_1', 'Dia da Semana_2', 'Dia da Semana_3', 'Dia da Semana_4', 'Dia da Semana_5', 'Dia da Semana_6', 'Dia da Semana_7', 'Dia do Mês', 'Data'							
Entrada (y)								
Parâmetro Utilizado	n_estimators = 301, random_state = 1	n_neighbors = 2, random_state = 1	-	n_splits=10, n_repeats=3, random_state=1	verbose = 0 n_estimators=100 n_splits=10 n_repeats=3 random_state=1	n_splits = 10 n_repeats = 3 random_state = 1	n_splits = 10 n_repeats = 3 random_state = 1	learning_rate': 0.04, 'feature_fraction': 1, verbose': 0, 'max_depth': 60, 'num_leaves': 2048, 'num_iterations': 5_000.
R^2 (treino)	99,43%	99,32%	-	-	-	-	-	99,43%
MAE (treino)	0,34	0,36	-	-	-	-	-	0,35
MSE (treino)	0,24	0,29	-	-	-	-	-	0,24
R^2 (teste)	98,33%	98,14%	44%	84,85%	91,55%	84,71%	91,25%	98,51%
MAE (teste)	0,57	0,59	-	1,85	1,42	1,85	1,44	0,55
MSE (teste)	0,71	0,79	-	6,43	3,59	6,5	3,72	0,63
Gráfico de dispersão (TESTE)								

Dessa forma, foi possível definir quais modelos tiveram uma melhor adaptação ao nosso problema.

[Link para a tabela de comparação - todos os modelos](#)

Além disso, todos os modelos testados foram reunidos em um único colab, onde foram organizados por etapa de teste - assim como neste documento - e, para cada modelo, há sua biblioteca, a execução do algoritmo e os resultados e erros gerados.

[Link para o colab com todos os modelos](#)

4.6.3 Definição dos melhores modelos

A partir da visualização e dos resultados obtidos em cada um dos algoritmos, foi possível definir os melhores modelos que apresentaram uma maior acurácia e menor taxa de erros. Foram eles: **Random Forest Regressor, KNN e LightGBM**.

4.6.4. Testes de hiperparâmetros

Após selecionar os melhores modelos, a próxima etapa foi encontrar os melhores valores para os hiperparâmetros de cada algoritmo.

Cada modelo possui seus próprios parâmetros que precisam ser definidos antes de realizar o treinamento do modelo - os chamados hiperparâmetros. Por padrão, alguns hiperparâmetros já possuem um valor definido, no entanto, esses valores podem ser ajustados

de forma que melhores resultados de acurácia sejam alcançados. Assim, existem diferentes maneiras de testar esses valores e visualizar os resultados obtidos, otimizando o modelo.

O primeiro método utilizado foi o Grid Search. Esse método é muito utilizado e, de maneira geral, apresenta um bom desempenho.

O Grid Search:

Esse algoritmo testa todas as combinações possíveis dos hiperparâmetros. Para executá-lo, é necessário fornecer um *range* de valores a serem testados para cada hiperparâmetro e o modelo irá testar todas as combinações possíveis. Ao final, um gráfico é gerado. Em seguida, ele selecionará os hiperparâmetros que obtiveram o menor erro.

No entanto, esse processo leva bastante tempo e demanda um alto esforço computacional. Após diversos testes, onde o algoritmo não finalizava ou gerava um valor que diminuía o desempenho da predição, esse método foi descartado e outros métodos foram buscados. Foi decidido, então, criar um laço de repetição para testagem dos hiperparâmetros.

Laço para teste de hiperparâmetros:

Para esse teste, os 3 principais algoritmos foram selecionados: KNN, Random Forest Regressor e LightGBM. Para cada um deles, foi definido um *range* de valores a serem testados e a predição foi realizada para cada um deles. Ao final, foram obtidos: R^2 , erro absoluto médio e erro quadrático médio - para cada valor de parâmetro testados.

Para o Random Forest Regressor, o hiperparâmetro testado foi o *n_estimators* (definido como o número de árvores na floresta), já que ele influencia significativamente na performance do algoritmo. Já para o KNN, testamos valores para o *n_neighbors* (definido como número de vizinhos a serem usados).

Exemplo do algoritmo (KNN):

```
k_range = range(1,6)
scores_list = []
mae_list = []
mse_list = []
for k in k_range:
    knn = KNeighborsRegressor(n_neighbors=k)
    knn.fit(x_train, y_train)
    print(k)
    y_pred = knn.predict(x_test)
    scores_list.append(metrics.r2_score(y_test, y_pred))
    mae_list.append(mean_absolute_error(y_test, y_pred))
    mse_list.append(mean_squared_error(y_test, y_pred))
for k in range(len(k_range)):
```

```
print(f"Valor de n_estimators: {k_range[k]} -> R² resultado: {scores_list[k] * 100:.2f}% -> Erro médio absoluto (MAE): {(mae_list[k]):.2f} -> Erro médio quadrático (MSE): {(mse_list[k]):.2f}")
```

Como o LightGBM possui vários hiperparâmetros, não foi possível aplicar este teste para ele, então os testes foram realizados manualmente:

Teste de hiperparâmetro do LightGBM:

a) Teste 1:

Descrição: Modelo padrão fornecido como uma primeira versão de hiperparâmetros em diversas orientações sobre a aplicação do LightGBM. Desse modo, os parâmetros seguiram valores padrões de aplicação e serviu como uma base para variações posteriores.

```
'task': 'train', #função
'boosting_type': 'gbdt', #tipo da regressão (gbdt, rf, dart, goss)
'objective': 'regression', #tipo do modelo
'metric': ['l1', 'l2', 'rmse'], #erro médio quadrático e erro médio absoluto
'learning_rate': 0.04, #velocidade do aprendizado
'feature_fraction': 1, #porcentagem do dataframe a ser utilizado
'verbose': 0, #usado para retirar possíveis bugs existentes (debug)
'max_depth': 60, #limita o tamanho de cada árvore
'num_leaves': 2048, #limita o número de folhas que a árvore pode ter
'num_iterations': 5_000, #quantidade de tentativas
```

b) Teste 2:

Descrição: Houve uma variação em relação ao modelo inicial para os parâmetros de velocidade de aprendizado e também para a quantidade de tentativas para o aprendizado do modelo.

```
'task': 'train', #função
'boosting_type': 'gbdt', #tipo da regressão (gbdt, rf, dart, goss)
'objective': 'regression', #tipo do modelo
'metric': ['l1', 'l2', 'rmse'], #erro médio quadrático e erro médio absoluto
'learning_rate': 0.02, #velocidade do aprendizado
'feature_fraction': 1, #porcentagem do dataframe a ser utilizado
'verbose': 0, #usado para retirar possíveis bugs existentes (debug)
'max_depth': 60, #limita o tamanho de cada árvore
'num_leaves': 2048, #limita o número de folhas que cada árvore pode ter
'num_iterations': 6_000, #quantidade de tentativas
```

c) Teste 3:

Descrição: Diferente dos modelos anteriores, para esse caso há uma ampliação na quantidade de iterações para o teste de modelo. Com o objetivo de buscar um resultado de maior precisão em relação a predição.

```
'task': 'train', #função
'boosting_type': 'gbdt', #tipo da regressão (gbdt, rf, dart, goss)
'objective': 'regression', #tipo do modelo
'metric': ['l1', 'l2', 'rmse'], #erro médio quadrático e erro médio absoluto
'learning_rate': 0.04, #velocidade do aprendizado
'feature_fraction': 1, #porcentagem do dataframe a ser utilizado
'verbose': 0, #usado para retirar possíveis bugs existentes (debug)
"max_depth": 60, #limita o tamanho de cada árvore
"num_leaves": 2048, #limita o número de folhas que a árvore pode ter
"num_iterations": 8_000, #quantidade de tentativas
```


Comparação de hiperparâmetros de todos os modelos:

Ao final dos testes de hiperparâmetro para cada um dos modelos, os dados obtidos foram agrupados na tabela a seguir:

Modelos testados:	Parâmetro Utilizado	R ² Gerado	MAE	MSE
Random Forest Regression	n_estimators = 298	98,3283%	0,5722	0,7107
	n_estimators = 299	98,3286%	0,5722	0,7106
	n_estimators = 300	98,3288%	0,5722	0,7105
	n_estimators = 301	98,3290%	0,5721	0,7104
	n_estimators = 302	98,3287%	0,5722	0,7105
	n_estimators = 303	98,3284%	0,5722	0,7107
KNN	n_neighbors = 1	97,78%	0,65	0,95
	n_neighbors = 2	97,93%	0,63	0,88
	n_neighbors = 3	97,77%	0,65	0,95
	n_neighbors = 4	97,54%	0,68	1,05
	n_neighbors = 5	97,25%	0,73	1,17
LightGBM	learning_rate': 0.04, 'feature_fraction': 1, verbose': 0, "max_depth": 60, "num_leaves": 2048, "num_iterations": 5_000,	98,515%	0,55	0,63
	learning_rate': 0.02, feature_fraction': 1, verbose': 0, "max_depth": 60, "num_leaves": 2048, "num_iterations": 6_000,	98,511%	0,55	0,63
	learning_rate': 0.04, 'feature_fraction': 1, verbose': 0, "max_depth": 60, "num_leaves": 2048, "num_iterations": 8_000,	98,515%	0,55	0,63

Ainda que as variações de performance tenham sido baixas, foi possível ter uma ideia de como os modelos se comportam com diferentes hiperparâmetros e foi possível definir os melhores valores para cada modelo - **destacados em negrito**.

[Link para a tabela de hiperparâmetros](#)

4.6.5 Valor Predito X Valor Real:

Para podermos realizar uma análise estatística em relação aos valores dos dados reais e os valores preditos por cada um dos 3 modelos (KNN, LightGBM e Random Forest Regressor), criamos um código que gera 100 valores aleatórios de posições de linhas para nossa base de dados de aproximadamente 21000 dados. Assim, para esses 100 valores aleatórios rodamos os 3 modelos e geramos uma base de comparação entre o valor real, o predito pelo KNN, pelo LightGBM e pelo RFR.

Na tabela a seguir determinamos 10 entre esses 100 valores para obtermos qual deles possui uma menor relação de erro e assim, tem uma precisão que atende melhor o valor real em relação ao modelo:

Dessa forma, é possível ver como cada modelo se comporta e a variação da predição feita em relação ao valor real de Rat.

Nº da Coluna	Valores Reais	Predição LightGBM	Predição KNN	Predição RFR
31579	12.63	10.867100	10.725	10.962131
19918	17.23	17.305710	16.655	16.684293
21616	2.25	1.715464	2.250	1.898887
12395	19.23	18.118155	17.695	17.852216
28873	10.65	10.899154	11.195	11.072821
40923	5,95	5.671059	5.785	5.772822
4486	16.44	17.336065	17.450	17.407785
36015	6.47	4.925302	5.080	5.057654
16575	4.20	3.938453	3.660	3.760673
35889	4.51	4.288603	4.335	4.428532

A partir da tabela anterior, podemos perceber a partir de 10 valores as relações existentes entre valores aleatórios da nossa base, cada um dos modelos e o valor real para a audiência em cada uma das linhas mapeadas acima.

A partir dos valores de cada modelo em relação ao real, pudemos obter também o erro para cada uma das linhas, e por fim indicar o desvio padrão da soma dos erros em cada um dos modelos. Como indicado na tabela a seguir:

Nº da Coluna	Erro LGBM	Erro KNN	
--------------	-----------	----------	--

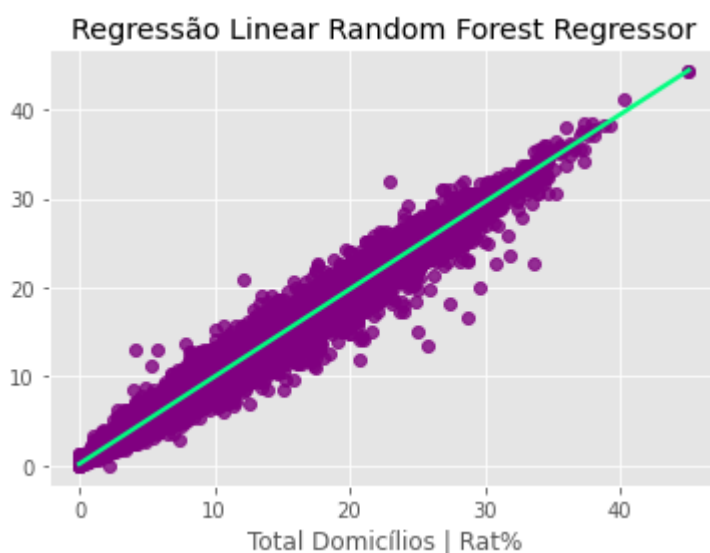
31579	1,246558545	1,347038418	
19918	0,0535350544	0,4065863992	
21616	0,3779740304	0,00000000	
12395	0,7861931391	1,085408909	
28873	0,176178483	0,3853731957	
40923	0,1972410727	0,1166726189	
4486	0,6336136379	0,714177849	
36015	1,092266431	0,9828784258	
16575	0,1849416573	0,3818376618	
35889	0,15655132	0,1237436867	
Soma dos Desvios	0,4262172371	0,4575289912	

4.6.6 Interpretação dos gráficos

Nesta etapa, foram gerados alguns gráficos usando a biblioteca Seaborn. Para cada modelo, foram plotados um gráfico de dispersão e um histograma.

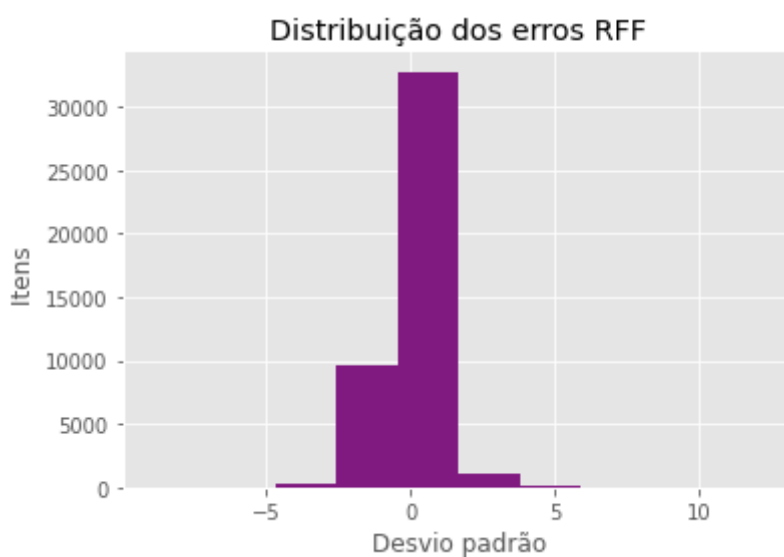
- **Random Forest Regressor**

Ao comparar os dados preditos do modelo Random Forest Regressor com os dados reais, foi possível obter o seguinte gráfico de regressão linear, onde a linha verde representa a reta da equação linear no eixo y estão os valores preditos e no eixo x os valores reais.



$$R^2 = 98,33\%$$

Também foi gerado um histograma comparando a distribuição total de erros do modelo e o desvio padrão.

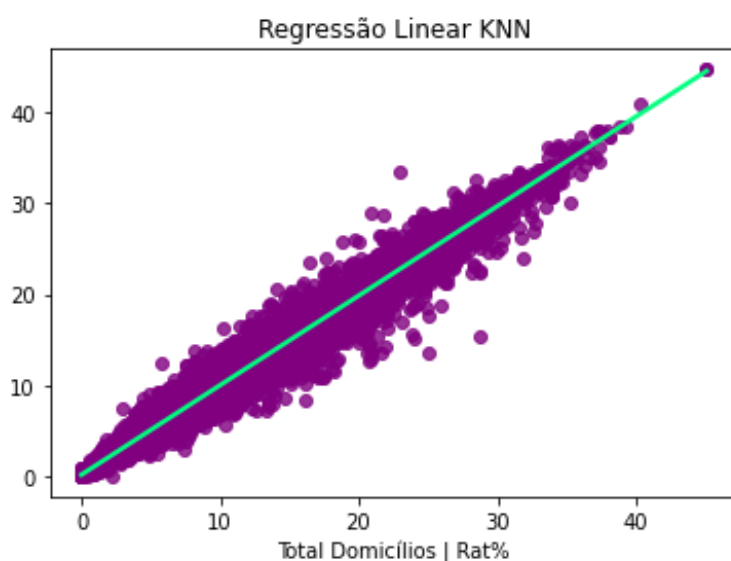


Desvio padrão 0.84

Ao analisar tais gráficos nota-se que este modelo tem uma desvio padrão negativo, o que permite concluir que em alguns casos esse modelo tem a tendência de estimar uma audiência menor que o futuro programa tende a atingir.

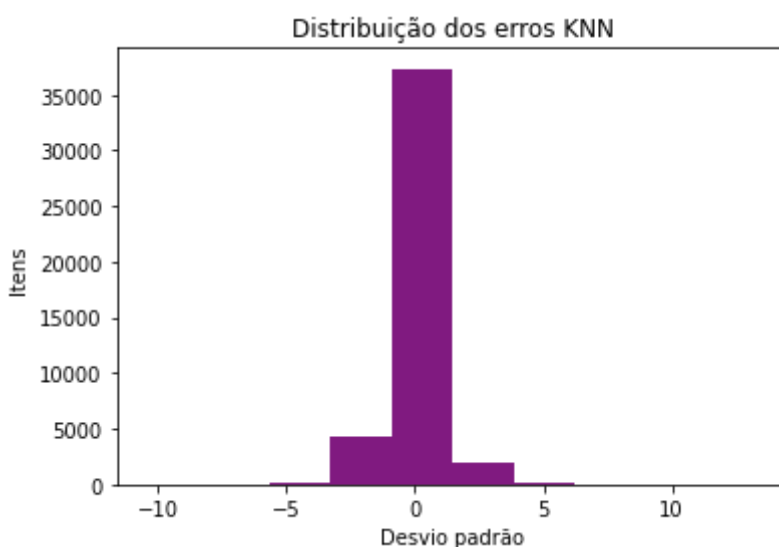
- KNN

Ao comparar os dados preditos do modelo KNN com os dados reais, foi possível obter o seguinte gráfico de regressão linear, onde a linha verde representa a reta da equação linear, no eixo y estão os valores preditos e no eixo x os valores reais.



$$R^2 = 97,93\%$$

Também foi gerado um histograma comparando a distribuição total de erros do modelo e o desvio padrão.

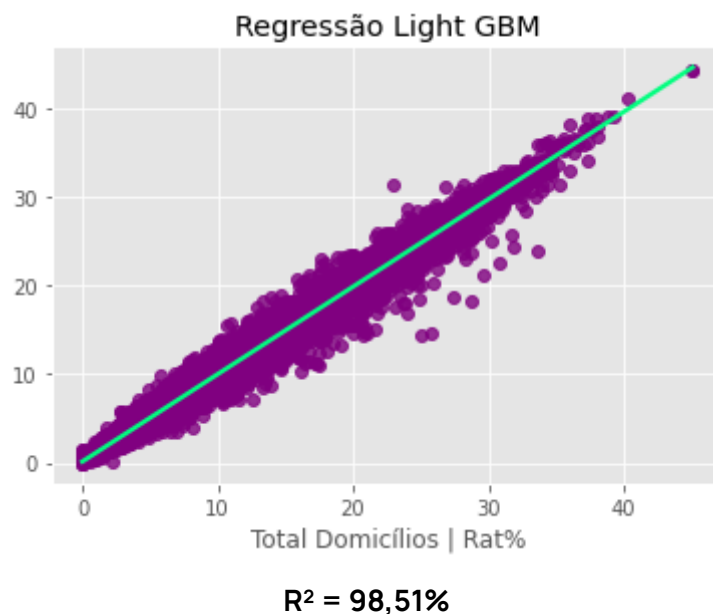


$$\text{Desvio padrão } 0,89$$

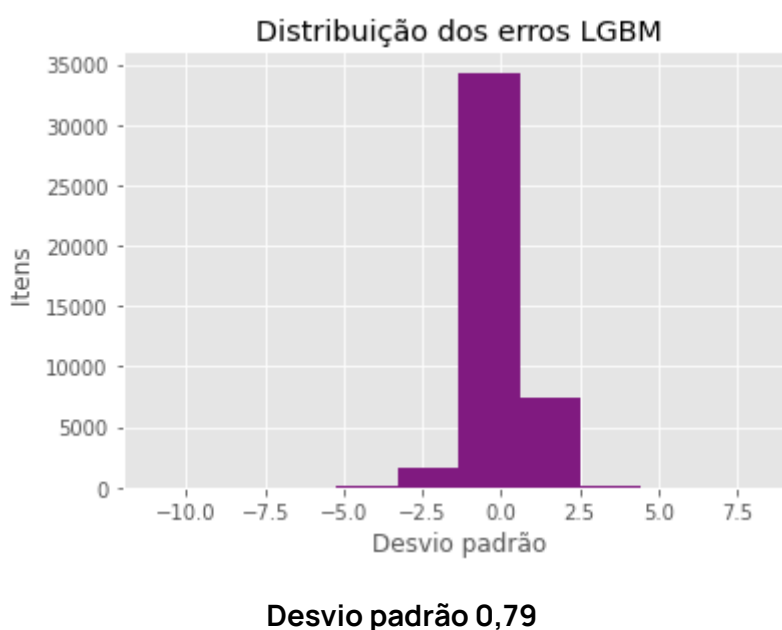
Ao analisar os gráficos do KNN o mesmo tem uma boa distribuição de acertos comparados com os valores preditos, sem outliers significativos. Ao se tratar do desvio padrão este modelo tem a tendência de estimar para mais os valores da audiência.

- **LightGBM**

Ao comparar os dados preditos do modelo LightGBM com os dados reais, foi possível obter o seguinte gráfico de regressão linear, onde a linha verde representa a reta da equação linear, no eixo y estão os valores preditos e no eixo x os valores reais.



Também foi gerado um histograma comparando a distribuição total de erros do modelo e o desvio padrão.

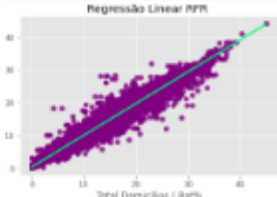
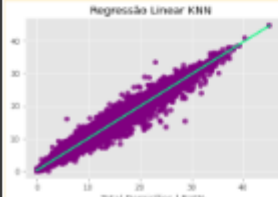
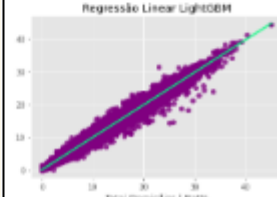


A regressão linear do LightGBM foi onde foi obtido o melhor resultado de r^2 98.51%, apesar que algumas vezes surgem pontos outliers. Em relação ao desvio padrão o modelo tem tendência de errar a predição para valores maiores e em raros casos outliers acima de 2.5 desvio padrões.

4.6.7 Comparação final dos principais modelos

Os principais modelos foram agrupados em um único colab, onde podem ser executados e gerados suas acurácias e erros (para valores de treino e teste). Além disso, é possível, também, dar *input* de dados para execução do modelo LightGBM (o melhor até então), podendo ser testada a interação do cliente com o produto - [Colab - Agamoto](#).

Após todos os testes realizados, foi gerada uma tabela para agrupar esses resultados e facilitar a visualização deles, possibilitando uma comparação da performance dos 3 modelos selecionados:

<div> <div>Comparação</div> <div>Random Forest x KNN x LightGBM</div> </div>			
Características	Random Forest Regressor	KNN	LightGBM
R ² (treino)	99,43%	99,32%	99,43%
MAE (treino)	0,34	0,36	0,35
MSE (treino)	0,24	0,29	0,24
R ² (teste)	98,33%	98,14%	98,51%
MAE (teste)	0,57	0,59	0,55
MSE (teste)	0,71	0,79	0,63
Desvio Padrão (teste)	0,84	0,94	0,79
Gráfico de dispersão (TESTE)			

[Comparação de Modelos](#)

A partir da tabela de comparações e as tabelas referentes aos testes de linha elaborados e especificados na seção 4.6.5 foram estabelecidos os desvios padrões e a acurácia para cada modelo em relação aos valores reais da base de dados utilizada.

Obtemos assim os drivers que nortearão a escolha do modelo a ser utilizado para a predição final.

Definidos os *drivers*, pode-se definir então qual modelo será aplicado a partir da leitura da métrica de cada *driver* utilizado e a priorização de quais impactam mais os resultados finais. Para

isso, se tornam necessários os seguintes passos finais para a definição do modelo e conclusão do projeto:

- i. Ampliação do *Range* de Teste de Linha: Atualmente utilizam-se apenas 100 valores para a realização do teste, uma amostra reduzida em comparação a base de aproximadamente 21000 dados.
- ii. Definição do modelo que possui um menor desvio padrão da amostra de acordo com o teste de linha.
- iii. Melhorias na interface de interação com usuário a partir da definição do melhor modelo de predição.

5. Conclusões e Recomendações

Analisando os resultados das predições a gráficos gerados pelo Agamotto, o grupo definiu que os parâmetros de “Categoria” são os que mais interferem nos resultados, já outros fatores como “horário” e “dia do mês” e “dia da semana”, tem uma interferência bem menor do que o esperado. As análises possuem poucos *outliers*, a programação possui certa linearidade com poucos desvios que precisaram ser normalizados nos dados.

No eventual uso ou alteração do algoritmo, é importante ter em mente a relevância dos parâmetros de categoria no resultado da predição e levar em conta outros parâmetros, como que podem ser utilizados (Data, mês, horário de início e horário de encerramento), como influenciadores secundários no resultado da predição. Em caso de alteração, ter em mente que o algoritmo afeta diretamente o público que já foi registrado anteriormente, a principal função da ferramenta é determinar a permanência e pico de telespectadores, dado a condições específicas de tempo e data, logo deve-se assumir uma constância em parâmetros como gênero, idade, etc.

6. Referências

NumPy. NumPy: The fundamental package for scientific computing with Python. [S. l.]: NumPy, Disponível em: <https://numpy.org/>. Acesso em: 10 ago. 2022.

Pandas. Pandas. [S. l.], Disponível em: <https://pandas.pydata.org/>. Acesso em: 10 ago. 2022.

WEHRSTEIN, Linda. CRISP-DM ready for Machine Learning Projects: Structure your Machine Learning projects using the strengths of a well-known Data Mining process model amended by roles and responsibilities.. [S. l.]: Towards Data Science, 19 dez. 2020. Disponível em: <https://towardsdatascience.com/crisp-dm-ready-for-machine-learning-projects-2aad9172056a>. Acesso em: 10 ago. 2022.

SCIKIT LEARN. 1.1. Linear Models. [S. l.]. Disponível em: https://scikit-learn.org/stable/modules/linear_model.html?highlight=linear+regression. Acesso em: 10 ago. 2022.

VELASQUEZ, Luis Henrique. Uma visão geral sobre machine learning – Classificação. [S. l.]: Statplace, Disponível em: <https://psicometriaonline.com.br/o-que-e-regressao-linear-multipla/>. Acesso em: 11 ago. 2022.

PATEL, Harshil. What is Feature Engineering – Importance, Tools and Techniques for Machine Learning: Feature engineering techniques for machine learning are a fundamental topic in machine learning, yet one that is often overlooked or deceptively simple.. [S. l.]: Towards Data Science, 30 ago. 2021. Disponível em: <https://towardsdatascience.com/what-is-feature-engineering-importance-tools-and-techniques-for-machine-learning-2080b0269f10>. Acesso em: 22 ago. 2022.

YADAV, Dinesh. Categorical encoding using Label-Encoding and One-Hot-Encoder. [S. l.]: Towards Data Science, 6 dez. 2019. Disponível em: <https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd>. Acesso em: 24 ago. 2022.

Psicometria Online. O que é Regressão Linear Múltipla?. [S. l.], Disponível em: <https://psicometriaonline.com.br/o-que-e-regressao-linear-multipla/>. Acesso em: 5 set. 2022.

SCIKIT LEARN. Sklearn.tree.DecisionTreeRegressor. [S. l.], Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html?highlight=decision+tree+regressor>. Acesso em: 6 set. 2022.

SCIKIT LEARN. Sklearn.neighbors.KNeighborsRegressor. Scikit Learn. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html?highlight=kneighbors#sklearn.neighbors.KNeighborsRegressor.kneighbors>. Acesso em: 6 set. 2022.

SCIKIT LEARN. RandomForestRegressor. Scikit Learn. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html?highlight=random+forest+regressor>. Acesso em: 7 set. 2022.

Editor Minitab. Basta! Lidando com a multicolinearidade na análise de regressão. [S. l.]: Minitab, 19 abr. 2019. Disponível em: <https://blog.minitab.com/pt/basta-lidando-com-a-multicolinearidade-na-analise-de-regressao#:~:text=Em%20regress%C3%A3o%2C%20%22multicolinearidade%22%20refere,mas%20tamb%C3%A9m%20uns%20aos%20outros>. Acesso em: 10 set. 2022.

JÚNIOR, ICMC. RANDOM FOREST: O que é Random Forest?. [S. l.]: ICMC JÚNIOR, 30 jun. 2021. Disponível em: <https://icmcjunior.com.br/random-forest/>. Acesso em: 10 set. 2022.

SCIKIT LEARN. Metrics and scoring: Quantifying the quality of predictions. Scikit Learn. Disponível em: https://scikit-learn.org/stable/modules/model_evaluation.html. Acesso em: 15 set. 2022.

SOARES, Alícia. O que é desvio padrão e como calculá-lo?: Saiba mais sobre essa variante estatística!. [S. l.]: Voitto, 24 dez. 2021. Disponível em: <https://www.voitto.com.br/blog/artigo/o-que-e-desvio-padrao>. Acesso em: 16 set. 2022.

HOW TO, Statistics. Absolute Error & Mean Absolute Error (MAE). [S. l.], Disponível em: <https://www.statisticshowto.com/absolute-error/>. Acesso em: 16 set. 2022.

BROWNLEE, Jason. XGBoost for Regression. Machine Learning Mastery, 12 mar. 2021. Disponível em: <https://machinelearningmastery.com/xgboost-for-regression/>. Acesso em: 16 set. 2022.

KAROLINE, Penteado. Métricas de avaliação para séries temporais. [S. l.]: Alura, 9 jun. 2021. Disponível em: https://www.alura.com.br/artigos/metricas-de-avaliacao-para-series-temporais?utm_source=awin. Acesso em: 16 set. 2022.

SCIKIT LEARN. Sklearn.ensemble.HistGradientBoostingRegressor. Scikit Learn. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingRegressor.html?highlight=lightgbm>. Acesso em: 17 set. 2022.

GUIMARÃES, Alysso. Entendendo CatBoost: Um guia (quase) definitivo. Data Hackers, 13 fev. 2022. Disponível em: <https://medium.com/data-hackers/entendendo-catboost-um-guia-quase-definitivo-b28bc153a78>. Acesso em: 18 set. 2022.

SCIKIT LEARN. Sklearn.experimental.enable_hist_gradient_boosting. Scikit Learn. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.experimental.enable_hist_gradient_boosting.html?highlight=gradient+boosting#module-sklearn.experimental.enable_hist_gradient_boosting. Acesso em: 18 set. 2022.

Seaborn. Seaborn: statistical data visualization. [S. l.], Disponível em: <https://seaborn.pydata.org/>. Acesso em: 19 set. 2022.

SCIKIT LEARN. Sklearn.metrics.plot_confusion_matrix. Scikit Learn. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.plot_confusion_matrix.html?highlight=pyplot. Acesso em: 20 set. 2022.

SCIKIT LEARN. Sklearn.model_selection.GridSearchCV. Scikit Learn. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html?highlight=grid+search#sklearn.model_selection.GridSearchCV. Acesso em: 22 set. 2022.

DYOURI, Abdelhadi. Como criar um aplicativo Web usando o Flask em Python 3. [S. l.]: Digital Ocean, 12 maio 2020. Disponível em: <https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3-pt>. Acesso em: 27 set. 2022.

Anexos

Para maior comodidade sobre os assuntos abordados pelo grupo no documento, ficarão disponíveis alguns *links* que servirão de facilitadores para os parceiros de projeto separados por etapa de desenvolvimento ou, como popularmente chamado, *Sprint*.

Sprint 1:

Documentos:

- **Primeira entrevista com o parceiro:**
https://docs.google.com/document/d/1PENCaCLOGvyeuePkIM9TN_ZxZZ-LH2HVwnkKjbQ7KvM/edit
- **Business Model Canvas, Análise SWOT, Jornada do Usuário e Personas:**
<https://miro.com/app/board/uXjVOhWavQY=/>
- **Matriz de Risco:**
https://docs.google.com/spreadsheets/d/1drHGd_pV2bNOUbtPm1hATa6fKYm0osp8yMOyHcvvR6E/edit#gid=0

Sprint 2:

Documentos:

- **Business Model Canvas, Análise SWOT, Jornada do Usuário e Personas:**
<https://miro.com/app/board/uXjVOhWavQY=/>
- **Diário The Pythons:**
<https://docs.google.com/document/d/1037htUBmZtWjmFO6kBOtT6XoBF1Dlcf1aTevbyVteTY/edit?usp=sharing>

Principais Colabs:

- **Plotagem de gráficos:** <https://colab.research.google.com/drive/1xrFZdqBK5at1g-2ibzDAulpeeGOlvzKa?usp=sharing>
- **Multicolinearidade:**
<https://colab.research.google.com/drive/1gmglZnEj52AHufEtbdusmEGrZcB0Y4KZ?usp=sharing>
- **Anonimização dos dados das emissoras:**
https://colab.research.google.com/drive/1e9TegVQB_FY7k_f_yF_YWk6PLlmtF3uK?usp=sharing
- **Feature Engineering:**
<https://colab.research.google.com/drive/1LSIG9WWOoMTlwVSdMEcLDaz97mBSDzN9#scrollTo=mfXqV0ZrAFbt>
- **Regressão Linear TV 0 (tentativa 1):**
<https://colab.research.google.com/drive/1zi5V-0yZLeE7qCM6RVMcXwCWE-GL25aa?usp=sharing>

- **Regressão Linear TV 1 (tentativa 1):**
<https://colab.research.google.com/drive/10vhOh0u3nRoZceG-wBaD5Lc892laEXvB?usp=sharing>
- **Regressão Linear TV 2 (tentativa 1):**
<https://colab.research.google.com/drive/1XUizIPEaqwd5iYOn5iPReLsPXDRrrpUt?usp=sharing>

Apresentação:

- **Slides da segunda Sprint:**
https://www.canva.com/design/DAFJCKLh3Ds/Y1FnxIFqE-9NnKSbtKES6g/edit?utm_content=DAFJCKLh3Ds&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Sprint 3:

Documentos:

- **Diário The Pythons:**
<https://docs.google.com/document/d/1037htUBmZtWjmFO6kBOtT6XoBF1DlcF1aTevbyVteTY/edit?usp=sharing>

Principais Colabs:

- **Primeiro teste de parâmetros (falho):**
https://colab.research.google.com/drive/1BEtQ_GiA6-uZbRXU1eHIDOPo6qBNHzP8?usp=sharing
- **Regressão Linear TV 0 (tentativa 2):**
https://colab.research.google.com/drive/1zd2J14_BlxgK-sIDqQWQXW9fFbdosNm3?usp=sharing
- **Regressão Linear TV 1 (tentativa 2):**
https://colab.research.google.com/drive/1sAx7HuwFYR7_hci8OJ9zqsaP8iddoDt4?usp=sharing
- **Regressão Linear TV 2 (tentativa 2):**
https://colab.research.google.com/drive/1ecEy0_0HfO58p-gNTbQ5yGEq1TIH0O9u?usp=sharing

Apresentação:

- **Slides da terceira Sprint:**
https://www.canva.com/design/DAFLqu863yk/COH5Ev6fx6tStweDRUBEdg/edit?utm_

content=DAFLqu863yk&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Sprint 4:

Documentos:

- **Diário** **The** **Pythons:**
<https://docs.google.com/document/d/1037htUBmZtWjmFO6kBO6T6XoBF1DlCF1aTevbyVteTY/edit?usp=sharing>
- **Planilha** **de** **comparação** **dos** **modelos:**
https://docs.google.com/spreadsheets/d/1qWvWPAvjGau2p7SILpS_tCgEnBWzslWZeOPAOTK40Dw/edit#gid=0

Principais Colabs:

- **Teste** **de** **Modelos:**
<https://colab.research.google.com/drive/1RDmlomfT03LUNn36aSSWltHu1V9UXJ4K#scrollTo=SMkodX0yKh0a>
- **Colab** **com** **todos** **os** **modelos** **do** **Agamotto:**
<https://colab.research.google.com/drive/1mm2veTcL35IUujnT7Gx-y5-leZhHtFCF?usp=sharing>

Apresentação:

- **Slides** **da** **quarta** **Sprint:**
https://www.canva.com/design/DAFM-f3rXQk/X81vXuxtPD8Obiw5Pw9v9w/edit?utm_content=DAFM-f3rXQk&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Sprint 5:

Documentos:

- **Diário** **The** **Pythons:**
<https://docs.google.com/document/d/1037htUBmZtWjmFO6kBO6T6XoBF1DlCF1aTevbyVteTY/edit?usp=sharing>
- **Planilha** **de** **comparação** **dos** **modelos:**
https://docs.google.com/spreadsheets/d/1qWvWPAvjGau2p7SILpS_tCgEnBWzslWZeOPAOTK40Dw/edit#gid=0

Principais Colabs:

- **Plotagem** **de** **gráficos:**

<https://colab.research.google.com/drive/1xrFZdqBK5at1g-2ibzDAulpeeGOlvzKa#scrollTo=-QClgOpK-UeP>

- **Todos** **os** **modelos** **-** **Agamotto:**

<https://colab.research.google.com/drive/1mm2veTcL35IUujnT7Gx-y5-leZhHtFCF?usp=sharing>

- **Modelo** **final:**

<https://colab.research.google.com/drive/14fuzR4yiF9SjRrqE-Rm9kllia7wk8Sby?usp=sharing>

Apresentação:

- **Slides** **da** **quarta** **Sprint:**

<https://prezi.com/p/edit/miuexowfd5x3/>