



inteli

GIA
Rede Gazeta



Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
09/08/2022	João Marques e Raphael Antunes	0.1	Canvas de proposta de valor
04/08/2022	Raduan Muarrek e Luana Parra	0.1.2	Análise de indústria
04/08/2022	Vitor Oliveira e João Suarez	0.1.3	Matriz de riscos
08/08/2022	João Marques, Raphael Antunes, Luana Parra	0.1.4	Matriz SWOT
09/08/2022	João Marques e Raphael Antunes	0.1.5	Canvas de proposta de valor
10/08/2022	Raduan Muarrek e João Marques	0.1.6	Descrição da predição
10/08/2022	Luana Parra e Vitor Oliveira	0.1.7	Descrição dos dados
11/08/2022	Elisa Flemer e Raphael Antunes	0.1.8	Criação dos gráficos para a análise de dados
11/08/2022	Luana Parra	0.1.9	Revisão dos artefatos da Sprint 1
12/08/2022	Elisa Flemer	0.2.0	Revisão final para Sprint 1
16/08/2022	João Suarez e Elisa Flemer	0.2.1	Correção das legendas dos gráficos e início da preparação de dados
28/08/2022	Luana Parra e Elisa Flemer	1.0	Desenvolvimento e revisão dos artefatos da Sprint 2.
11/08/2022	Luana Parra e Elisa Flemer	2.0	Desenvolvimento e revisão dos artefatos da Sprint 3.

Sumário

1. Introdução	6
2. Objetivos e Justificativa	7
2.1. Objetivos	7
2.2. Justificativa	7
3. Metodologia	8
3.1. CRISP-DM	8
3.2. Ferramentas	8
3.3. Principais técnicas empregadas	9
4. Desenvolvimento e Resultados	9
4.1. Compreensão do Problema	9
4.1.1. Contexto da indústria	9
Análise aprofundada seguindo o modelo de 5 forças de Porter	10
4.1.2. Análise SWOT	12
4.1.3. Planejamento Geral da Solução	12
4.1.4. Value Proposition Canvas	13
4.1.5. Matriz de Riscos	14
4.1.6. Personas	15
4.1.7. Jornadas do Usuário	17
4.2. Compreensão dos Dados	18
Análise preliminar dos dados	19
Audiência por hora por emissora (Seg a Sex)	19
Audiência por hora por emissora (Sábado)	20
Audiência por hora por emissora (Domingo)	21
Audiência da Emissora 0 por dia da semana	22
Audiência por dia do mês por emissora	25
Audiência da Emissora 0 por mês	26
Audiência por emissora por ano	27

Audiência da Emissora 0 por classe socioeconômica	28
Audiência da Emissora 0 por faixa etária	29
Audiência total da Emissora 0 por gênero	30
Audiência da Emissora 0 por gênero	31
Considerações sobre o resultado desejado	31
4.3. Preparação dos Dados	32
Pré-processamento dos dados	32
Anonimização dos dados	32
Otimização de processamento dos arquivos	33
Formatação de datas	33
Merge com grade horária	34
Checando a existência de outliers	34
Checando valores nulos e ausentes	35
Checando categorias de baixa frequência	35
Seleção de features	36
Teste de hipóteses	37
Hipótese 1	38
Hipótese 2	38
Hipótese 3	38
4.4. Modelagem	40
Rodada 1	41
Rodada 2	43
KNN	44
Decision trees (árvores decisórias)	46
Random forest generator	48
Rodada 3	50
Regressão linear	50
KNN	51
Decision trees	51

Random forest generator	53
Rodada 4	54
LightGBM	55
CatBoost	56
XGBoost	56
Rodada 5	56
Regressão linear	57
KNN	57
Decision tree	58
Random forest generator	60
LightGBM	60
Aplicando LightGBM em outras saídas	61
Share	61
Reach	61
Fidelidade	62
Interface gráfica	62
4.5. Avaliação	65
5. Conclusões e Recomendações	68
6. Referências	69
Anexos	70

1. Introdução

A Rede Gazeta de Comunicações é a maior empresa de comunicação do Espírito Santo, com mais de 500 funcionários. Fundada em 1928 com o jornal *A Gazeta*, tem como maior propósito informar, entreter e prestar serviços de comunicação aos capixabas com qualidade, ética e inovação, contribuindo para o desenvolvimento socioeconômico, cultural e de cidadania. Atualmente, o grupo é formado por um site de notícias (www.agazeta.com.br); oito rádios; quatro emissoras de TV aberta (TV Gazeta) afiliadas à Rede Globo; e dois portais de notícias locais (G1 Espírito Santo e o Globo Esporte Espírito Santo).

No que tange às emissoras, devido à sua conexão com a Rede Globo, devem coordenar a programação local com os eventos nacionais produzidos pela rede. Assim, com slots limitados para veicular seus próprios conteúdos, torna-se imprescindível prever a audiência de novos eventos com acurácia para maximizar o retorno de seus investimentos. Objetiva-se, com isso, escolher sempre o melhor evento para cada horário disponível e alocar recursos eficientemente, priorizando campanhas de marketing para programas com menor audiência esperada.

No momento, apesar de existirem registros históricos de score de audiência para diferentes horários e demográficos, não há análises de correlação entre programa (com suas características principais) e slot. Consequentemente, faltam evidências para prever e justificar a transmissão de certos eventos em certos horários, causando significativa incerteza para os stakeholders da empresa a cada modificação na grade de programação.

2. Objetivos e Justificativa

2.1. Objetivos

O principal objetivo da Rede Gazeta é maximizar o retorno financeiro de cada evento veiculado. Essa monetização se dá através da venda de slots publicitários, cuja precificação, por sua vez, é fortemente influenciada do score de audiência do programa sendo veiculado durante o slot comercial. Assim, em termos mais específicos, para maximizar seus retornos, a Rede Gazeta deve maximizar a audiência de cada conteúdo a fim de atrair mais anunciantes.

2.2. Justificativa

A GIA (Gazeta Inteligência Artificial) é um modelo preditivo inovador que calcula o score de audiência esperado, geral e por demográfico, para novos programas em diferentes dias e horários. É disruptiva na indústria por aplicar machine learning para o grande volume de dados produzido pelo Kantar IBOPE e oferecer uma análise detalhada substanciando a previsão de audiência. Antes dele. Assim, a GIA não só produz um diagnóstico objetivo, baseado em tendências históricas, como também elenca as variáveis utilizadas para se chegar a esse resultado, a fim de que o usuário possa validar a linha de raciocínio.

Essas funcionalidades são desconhecidas na indústria televisiva, que costuma delegar essas tarefas inteiramente para funcionários. Estes acabam dedicando muita energia mental e tempo para a análise e tomada de decisão manuais, sendo, portanto, muito mais vulneráveis a erros de interpretação e viés pessoal.

Assim, a GIA é um algoritmo destinado a alavancar a produtividade, assertividade e acurácia do Departamento de Programação da Gazeta, colocando-a, desse modo, a frente dos concorrentes no que tange à escolha e agendamento de novos eventos. A médio e longo prazo, espera-se que as sugestões da GIA, quando aplicadas, aumentem significativamente o sucesso de audiência da emissora e o retorno financeiro de seus investimentos.

3. Metodologia

Descreva as etapas metodológicas que foram utilizadas para o desenvolvimento, citando o referencial teórico. Você deve apenas enunciar os métodos, sem dizer ainda como ele foi aplicado e quais resultados obtidos.

3.1. CRISP-DM

Descreva brevemente a metodologia CRISP-DM e suas etapas de processo

O CRISP-DM (Cross Industry Standard Process for Data Mining) é uma metodologia ágil voltada para projetos envolvendo Machine Learning, mineração e análise de dados. Sendo assim, é um processo cíclico dividido nas seguintes etapas:

- 1."Business Understanding"(estudo do projeto ou negócio, atendendo os objetivos e interesses do cliente);
- 2."Data Understanding (identificar, coletar e analisar os conjuntos de dados que podem ajudar a atingir os objetivos do projeto);
- 3."Data Preparation" (ocorre a manipulação de dados, filtrando quais dados serão usados para a modelagem);
- 4."Modeling" (desenvolver um modelo e selecionar a técnica de modelagem);
- 5."Evaluation" (avaliar a qualidade, fidedignidade e segurança dos resultados obtidos da etapa de Modelagem);
- 6."Deployment" (se inicia o processo de desenvolvimento dos modelos criados e avaliados nas etapas anteriores)

3.2. Ferramentas

Descreva brevemente as ferramentas utilizadas e seus papéis (Google Collaboratory)

Nome	O que é	Em que foi utilizado	Versão
Google Collaboratory	É um serviço de nuvem gratuito hospedado pelo próprio Google para incentivar a pesquisa de Aprendizado de Máquina e Inteligência Artificial.		

--	--	--	--

3.3. Principais técnicas empregadas

Descreva brevemente as principais técnicas empregadas, algoritmos e seus benefícios

4. Desenvolvimento e Resultados

4.1. Compreensão do Problema

4.1.1. Contexto da indústria

A indústria televisiva aberta opera, primordialmente, em modelo B2B através da venda de slots publicitários para anunciantes diversos nos intervalos de cada evento (publicidades, merchandising, comerciais). A receita dessa atividade é diretamente relacionada ao score de audiência atingido pelos programas veiculados pela emissora. Nesse sentido, a atenção dos telespectadores é a mercadoria oferecida a seus clientes, isto é, às empresas que buscam divulgar seus produtos e serviços para um público de larga escala.

Dentre os principais players do mercado, tem a Globo, RecordTV, SBT, Band e TV Brasil. Destes, a Globo apresenta as maiores taxas de audiência em quase todas as situações, beirando um monopólio da indústria. A lista abaixo apresenta mais detalhes sobre cada uma delas.

Globo: Assistida por mais de 200 milhões de pessoas diariamente (no Brasil e no exterior), é a segunda maior rede de televisão comercial do mundo cobrindo 99,55% do total da população brasileira. Tem como diferenciais o jornalismo (especialmente o Jornal Nacional e Fantástico) e a produção de telenovelas. Possui 120 afiliadas e inúmeros canais, incluindo plataformas de streaming e transmissão para outros países.

Rede Record: Fundada em 1993 pelo empresário Paulo Machado de Carvalho, a Rede Record é uma rede de televisão aberta que passa em canal aberto e tem como nicho conteúdo religioso. Está na segunda colocação de audiência nacional pelo IBOPE e possui um trabalho de digitalização bem avançado, já tendo uma plataforma de streaming e uma grande presença nas redes sociais.

SBT: O SBT foi fundado pelo empresário e comunicador Sílvio Santos em 1981 após o Grupo Silvio Santos ganhar uma licitação do Governo Federal para comprar a então “TV Tupi”. Hoje, ela ocupa a terceira posição na média da audiência nacional medida pelo IBOPE. Podemos considerar que o SBT ocupa dois nichos: o de programa de auditório, como programa do Ratinho e o próprio programa do Sílvio Santos, e o de programas infantis, como a novela *Chiquititas* e desenhos animados.

Rede Bandeirantes: A rede Bandeirantes foi criada em 1967 por João Jorge Saad após a compra da Rádio Bandeirantes. Hoje, a quarta posição na média nacional do IBOPE. Atua historicamente com esportes em geral e também com política, sendo tradição a produção dos primeiros debates de cada temporada de eleição no Brasil.

TV Brasil: É a emissora aberta oficial do Poder Executivo Brasileiro, além de ser obrigatoriamente transmitida em operadoras de TV paga. Atualmente, televisiona uma mistura de programas independentes, seriados e transmissão de eventos esportivos.

No que tange às tendências na indústria, a maior delas é a transformação digital. Com o advento de plataformas de streaming, a indústria televisiva tem se reinventado para disponibilizar sua grade na internet ao vivo e sob demanda. Um exemplo é a Globo Play, que oferece não só produções originais como também filmes e séries internacionais para competir com gigantes tais quais Netflix e Amazon Prime.

Além disso, tem-se o crescimento significativo da personalização no consumo de conteúdo. Através de algoritmos de predição, os principais streamings e redes sociais já conseguem recomendar o conteúdo mais relevante para cada usuário de modo certeiro. Assim, há grande pressão para que a indústria televisiva tradicional também agregue mais personalização em suas operações.

Ademais, o Brasil hoje oferece solo fértil para a produção de séries ficcionais e não ficcionais. Seguindo o sucesso desse modelo em outros países, existe significativa motivação para investir em documentários e minisséries nacionais, como demonstrado pelo sucesso de títulos tais quais “Irmandade” e “3%”.

Análise aprofundada seguindo o modelo de 5 forças de Porter

Ameaça de novos entrantes: O estabelecimento de uma nova emissora de televisão exige grande capital inicial – com 70% de acionistas brasileiros –, uma concessão da Anatel e uma aprovação do Ministério de Comunicações quanto à sua proposta de programação e condições técnicas e financeiras. Sem dúvida, é uma empreitada que exige muitos recursos, paciência e competência. Além disso, uma vez estabelecida, uma nova emissora teria ainda o desafio de concorrer com as grandes redes de televisão brasileiras por audiência. Nesse sentido, percebe-se que há barreiras significativas para novos entrantes, de modo que a ameaça destes é baixa. No mercado atual existem grandes players, o que cria uma barreira de

novos entrantes com alta capacidade de crescimento. Desse modo, por mais que hoje a produção de conteúdo seja de fácil acesso, a distribuição (caso da rede gazeta) é um espaço com poucas empresas e que possuem quase um monopólio no segmento. Por fim, a possibilidade de novos entrantes é baixa, pois existe uma demanda de capital inicial, caixa e competência de desafiar grandes competidores nesse mercado.

Poder de barganha dos fornecedores: Os fornecedores são os produtores de conteúdo nacionais (incluindo roteiristas, diretores, atores, etc) e internacionais (distribuidores de filmes, séries e desenhos animados). Nesse sentido, tem-se um baixo poder de barganha no cenário brasileiro devido às poucas possibilidades de artistas televisionarem suas criações em emissoras de grande porte. Entretanto, quando se trata de empresas internacionais, o panorama muda, pois estas têm a possibilidade de vender para inúmeras emissoras, plataformas de streaming e consumidores diretos por todo o globo. Portanto, nesse caso, seu poder de barganha aumenta consideravelmente.

Poder de barganha dos compradores: O cliente, para a Rede Gazeta, pode ser tanto a “audiência” quanto os “patrocinadores”. Do lado da audiência, percebe-se um aumento de poder de barganha nos últimos anos devido à polarização político-ideológica que o Brasil tem enfrentado. Assim, telespectadores cobram atitudes condizentes com suas opiniões por parte das emissoras e tendem a migrar para outros canais quando não recebem o que esperam. Já do lado dos patrocinadores, a TV deixou de ser a primeira opção de praça para muitos. Com os avanços da tecnologia e dos algoritmos de marketing segmentado, existem hoje muitos outros meios de atingir clientes com maior eficácia e menor capital investido. Assim, os anunciantes passam a ter um maior poder de barganha para com as emissoras.

Ameaça de serviços/produtos substitutos: A indústria televisiva tem encontrado agressiva competição nas plataformas de streaming e redes sociais. Cita-se, por exemplo, Netflix e Youtube como importantes concorrentes, atraindo 33% e 64% da população brasileira atualmente. Entretanto, dado que, segundo a pesquisa do Kantar IBOPE, a televisão ainda persiste em 97% dos lares nacionais e teve seu consumo intensificado durante a pandemia, infere-se que a ameaça de substitutos é mediana, pois os dados mostram que ela ainda domina o momento de lazer do brasileiro médio.

Rivalidade entre concorrentes: Historicamente, emissoras de TV aberta foram sempre muito competitivas, lutando por certos artistas e punindo aqueles que migravam para uma rival. Entretanto, esse cenário tem se attenuado. Hoje, já se vê artistas e atores passando da Globo para o SBT, ou do SBT para a Record, sem grandes polêmicas. Uma exceção é o caso Globo-Record, as quais ainda não compartilham artistas e apresentadores. Ainda assim, é fato que a competição por direitos de transmissão, especialmente esportivos, é acirrada, assim como por scores de audiência absolutos.

4.1.2. Análise SWOT

MATRIZ SWOT		
	Fatores Internos (Controláveis)	Fatores Externos (Incontroláveis)
Pontos Fortes	Forças <ul style="list-style-type: none"> - Rede televisiva com maior alcance no Espírito Santo; - 16 veículos de comunicação que se conectam todos os dias com os capixabas; - Afiliada da Rede Globo; - Alta qualidade de equipamentos técnicos e de filmagem. 	Oportunidades <ul style="list-style-type: none"> - Popularização de serviços on-demand para conteúdo televisivo, abrindo espaço para investimentos em streaming de programas da TV Gazeta; - Baixa incidência de conteúdo culturalmente capixaba para a população local.
Pontos Fracos	Fraquezas <ul style="list-style-type: none"> - Empresa tradicional; - Poucos funcionários focados na área de inovação; - Dependência de dados do People Meter, que não discriminam demográficos e grade de programação com acurácia satisfatória; - Limitados pelas decisões executivas da Rede Globo. 	Ameaças <ul style="list-style-type: none"> - Alta concorrência no mercado de comunicação, intensificada pelos artistas de redes sociais; - Preferência por outros aparelhos eletrônicos e mídias sociais para entretenimento por partes dos telespectadores; - Crescimento das plataformas de streaming; - Enfraquecimento da mídia tradicional (em especial noticiários) por conta da polarização informacional do país; - Certo desinteresse da população brasileira por noticiários incisivamente políticos.

4.1.3. Planejamento Geral da Solução

Para nosso projeto, recebemos um documento Excel com 18 abas, cada qual contendo taxas de audiência colhidas pelo PeopleMeter em diferentes períodos e divididas por gênero, faixa etária e classe socioeconômica.

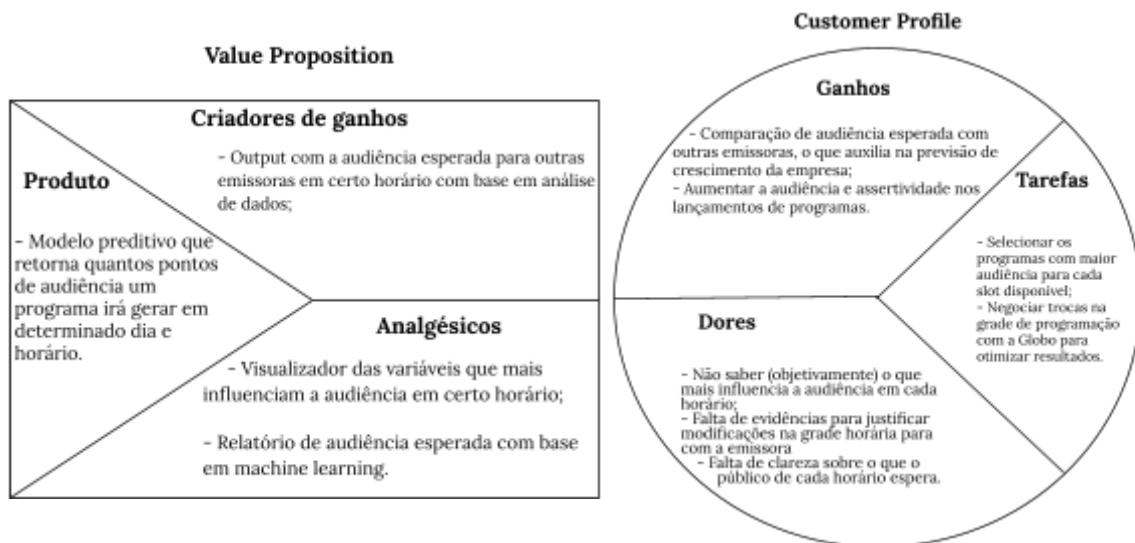
Com esse documento, pretendemos construir um modelo preditivo em que o usuário (funcionário da Rede Gazeta) possa inserir a data e horário desejados para um novo programa, assim como o gênero do programa em questão. Com isso, o objetivo é prever o score de audiência nessa data e slot, indicando as variáveis que mais tiverem peso para o modelo para, assim, criar ações mais efetivas para maximizar o retorno financeiro do evento.

Nesse sentido, nosso produto trata de regressão, pois nós vamos retornar um valor dentre infinitas opções. O output secundário, no entanto, relacionado às variáveis utilizadas na predição, talvez seja classificatório. Ainda não temos como saber.

Além disso, em termos práticos, a solução proposta será utilizada pelas equipes de Programação e Marketing da Rede Gazeta. Desse modo, de acordo com os dados fornecidos por esse modelo, a Rede Gazeta pode definir um esforço (maior ou menor) necessário para divulgar certo programa, realizar trocas na grade de horário ou mesmo negociar slots com a Globo. Tudo isso, se bem aplicado, contribuirá para um maior retorno financeiro para os investimentos da empresa.

Por fim, o critério de sucesso ainda não foi bem definido, dado que não começamos a implementar o modelo. Entretanto, antecipamos que ele será determinado de acordo com uma tolerância máxima para o erro da predição. Isso, por sua vez, será aferido na fase de testes com um training set derivado dos dados que recebemos. Logo, consideramos o modelo preditivo um sucesso ele retorna uma margem de erro pequena/desconsiderável.

4.1.4. Value Proposition Canvas



4.1.5. Matriz de Riscos

Matriz de Risco											
Probabilidade		Riscos					Oportunidade				
Muito Alta	5			Atrasos por conta de membros do grupo faltando							
Alta	4						Entregar análises fidedignas considerando as capacidades técnicas do grupo	Realizar os autoestudos antes das aulas e desenvolvimento s relacionados			
Médio	3			Parceiros de projeto com expectativa diferente de produto, superestimando.	Concentração de tarefas em pessoas específicas do grupo	Nosso produto não se destacar em relação aos outros grupos	A solução ser adotada pela Rede Gazeta			Expandir a audiência da rede Gazeta	
Baixa	2				Não conseguir entregar o projeto a tempo	Não conseguir analisar todos os dados e trazer vieses no modelo preditivo					
Muito Baixa	1			Grupo não se entrosar	Mudanças na LGPD, impactando as amostras que o aparelho pode coletar	Violação dos dados do ibope Não entregar previsões assertivas					
		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											

 Matriz de risco - Grupo 5

4.1.6. Personas

Persona 1



Marta Lopes, 32 anos, Analista de Comunicação.

"Comunicativa, gosta de trabalhar com as pessoas, viajar e conhecer diferentes culturas"

Biografia: Migrou do Marketing para a área de Comunicação da Rede Gazeta; Decidida, prática e ótima negociadora.

Dores/Motivações atuais com o problema: Não consegue negociar com os anunciantes de forma que correlacione o horário com a classe social que é alvo do anúncio.

Objetivos/necessidades específicas em relação ao problema: Focalizar os anunciantes para horários com maior potencial de compradores; Ter evidências para negociar taxas maiores com anunciantes fora do tradicional "horário nobre".

Persona 2



Giovanna Mattos, 28 anos, Gerente Geral de Marketing.

"Analítica, adora dados e quer transformar o ambiente em que trabalha através da tecnologia."

Biografia: Recentemente contratada e ainda se familiarizando com a empresa e suas particularidades; Experiência com Google Ads e análise de dados; Trabalhou como freelancer por bastante tempo.

Dores/Motivações atuais com o problema: Não há a possibilidade de prever a necessidade de reforço de campanhas de marketing direcionadas à horários; Falta de evidências para aprovar o orçamento de marketing.

Objetivos/necessidades específicas em relação ao problema: Reavaliar a divulgação de novos eventos que tiveram um menor potencial de audiência; Identificar fatores que mais contribuem para a audiência de um programa a fim de enfatizá-los na campanha de marketing; Ter evidências para negociar orçamentos maiores e promoções para o departamento de marketing, a partir da comparação da audiência esperada e audiência consolidada após campanhas.

Persona 3



Rodrigo Souza, 35 anos, Gerente de Operação e Programação.

"Funcionário de longa-data desde o período de estágio. Enfrenta dificuldades para justificar os seus planos de ação de forma objetiva desde cedo."

Biografia: Funcionário de longa-data muito familiarizado com a grade atual; Opiniões fortes sobre os programas existentes e muitas ideias de melhoria; Intuitivo, tende a tomar decisões com base em suas impressões subjetivas.

Dores/Motivações atuais com o problema: Não saber (objetivamente) o que mais influencia a audiência em cada horário; Falta de evidências para justificar modificações na grade horária para com a Globo; Falta de clareza sobre o que o público de cada horário espera.

Objetivos/necessidades específicas em relação ao problema: Selecionar os programas com maior audiência para cada slot disponível; Identificar o que faz um programa ter sucesso em cada slot para produzir melhores eventos; Negociar troca de programação em certos slots com a Globo; Suprir expectativas do público de determinado slot.

4.1.7. Jornadas do Usuário



Rodrigo, Gerente de Operação e Programação

Cenário: Rodrigo quer realizar um plano de ação mais objetivo para a audiência de um novo programa.

Expectativas				
FASE 1 (Organização de dados)	FASE 2 (Análise)	FASE 3 (Deduzindo)	FASE 4 (Preparando)	FASE 5 (Conferir e Estabelecer)
<p>1.Começa a organizar (quase que manualmente) os números de audiência, características dos telespectadores e horários que possam se relacionar com o novo programa;</p> <p>2. Para acessar esses dados (tabelas em Excel) precisa pedir acesso para outros setores.</p> <p>'É essencial organizar todos os dados que possam se relacionar com o contexto de um novo programa, mesmo o processo sendo extremamente cansativo'</p> 	<p>1.Confere mais cuidadosamente os dados selecionados, com viés de análise;</p> <p>2.Checa o número de audiência, características do público e horário de antigos programas que se relacionam com o novo;</p> <p>3.Imagina o quanto bem esse novo programa poderia ser encaixado com base em experiências anteriores.</p> <p>'Pelo o que já se passou até hoje, como seria o desempenho de um novo programa nesse cenário? Imagino o quanto bem esse programa e nossa rede possa se tornar '</p> 	<p>1.A partir do que se foi imaginado e concluído (tendendo ao subjetivo), temos as primeiras propostas de encaixe da nova programação expostas, a partir da audiência;</p> <p>2.Discorre as novas propostas com o que foi formulado.</p> <p>'Bom, contudo, seria ótimo ter argumentos mais sólidos para a implementação de um novo programa na grade. Ainda tenho minhas dúvidas em relação a essas novas propostas'</p> 	<p>1.Organiza os dados coletados anteriormente para serem inseridos no modelo preditivo;</p> <p>2.Apresenta as circunstâncias e dúvidas trabalhadas para o modelo preditivo.</p> <p>'Inserir todos os dados no modelo preditivo é bem cansativo/chato'</p> 	<p>1.Compara as suposições subjetivas já feitas com aquelas desenvolvidas no modelo preditivo, fazendo questionar o que se foi concebido anteriormente ou apenas reforçando;</p> <p>2.Com esse estudo mais completo, sugestões dessa nova grade de horário são apresentadas.</p> <p>'Agora sim, embasado e completo. Estou satisfeito com o resultado do modelo preditivo!'</p> 

Oportunidades

- Automatizar e deixar mais rápida a organização inicial de dados;
- Deixar mais prática a inserção de dados no modelo preditivo.

miro



Giovanna Mattos, Gerente Geral de Marketing

Cenário: Giovanna quer identificar exatamente (ou o mais próximo disso) o melhor horário para divulgar comerciais e propagandas dos novos programas.

Expectativas				
FASE 1 (Destrichando/Organizando)	FASE 2 (Análise)	FASE 3 (Deduzindo)	FASE 4 (Preparando)	FASE 5 (Conferir e Estabelecer)
<p>1.Após receber as informações do novo programa, confere quais programações já estabelecidas mais se assemelham a ele;</p> <p>2.Pede acesso aos dados (tabelas Excel) desses programas semelhantes (horário, público atingido, antiga forma de divulgação) para outros setores.</p> <p>'Esse é um trabalho muito manual e burocrático, tenho que pedir diversas tabelas e informações para outros setores do trabalho '</p> 	<p>1.Confere mais cuidadosamente os dados selecionados, com viés de análise;</p> <p>2.Checa em quais programas e horários o público alvo está mais ativo;</p> <p>3.Compara como antigas divulgações desse gênero de programa foram feitas e seus impactos, mantendo acertos e evitando erros.</p> <p>'Gosto muito de poder exercer esse meu lado analista, mas um olhar pessoal ainda é necessário'</p> 	<p>1.A partir do que se foi imaginado e concluído na análise, temos as primeiras propostas de encaixe da divulgação do novo programa;</p> <p>2.Discorre as novas propostas com o que foi formulado.</p> <p>'Bom, mas ainda é algo um tanto que subjetivo, queria ter ainda mais exatidão nessa minha proposta de marketing em divulgação'</p> 	<p>1.Organiza os dados coletados anteriormente para serem inseridos no modelo preditivo;</p> <p>2.Apresenta as circunstâncias e dúvidas trabalhadas para o modelo preditivo.</p> <p>'Inserir todos os dados no modelo preditivo é bem cansativo/chato'</p> 	<p>1.Compara as suposições subjetivas já feitas com aquelas desenvolvidas no modelo preditivo, fazendo questionar o que se foi concebido anteriormente ou apenas reforçando;</p> <p>2.Com esse estudo mais exato e completo, a forma de divulgação desse novo programa é apresentada.</p> <p>'Agora sim, ainda mais próxima da exatidão. Estou satisfeita, com o resultado do modelo preditivo!'</p> 

Oportunidades

- Deixar mais prático e evitar intermediários na organização inicial de dados;
- Tornar a inserção de dados no modelo preditivo menos manual e mais rápida.

Obs: Caso esteja com dificuldade de visualização, a imagem está com dimensões maiores na seção Anexos.

4.2. Compreensão dos Dados

Recebemos duas planilhas em formato XLSX que agrupam dados coletados pelo Kantar IBOPE Media a partir de um aparelho de depuração de audiência chamado “People Meter”. Esse aparelho torna possível registrar o perfil de quem está assistindo com os demográficos de que faz parte, o número total de indivíduos atingidos e o tempo de consumo.

O primeiro arquivo, denominado “TV_Histórico”, contém 18 abas elencando dados de audiência por emissora e dia da semana. Mais especificamente, cada emissora é dividida em três abas: uma para dias úteis (156.673 linhas), uma para sábado (31.105 linhas) e uma para domingo (também 31.105 linhas). O período analisado, por sua vez, é de 6 de junho de 2020 a 25 de junho de 2022, sendo que cada dia é tabelado de cinco em cinco minutos. Uma peculiaridade, nesse sentido, é que o horário a cada dia começa em 24:00:00 (correspondendo à meia-noite) e vai até 29:55:00 (correspondendo às 5:55:00 da manhã); depois, a contagem se reinicia às 6:00:00.

Cada aba contém as porcentagem de três métricas de audiência: Rating (considerado o score de audiência), Fidelidade (parcela dos televisores que ficou no mesmo canal por pelo menos um minuto), Reach (parcela de televisores alcançada) e Share (parcelas de televisores sintonizados com certa emissora dentre todos os ligados em dado momento). Esse valores são, por sua vez, subdivididos em classe socioeconômica (AB, C1, C2 e DE), sexo (masculino e feminino) e idade (4-11 anos, 12-17 anos, 18-24 anos, 25-34 anos, 35-49 anos, 50-59 anos e 60+ anos).

Já o segundo arquivo, “grade_Diária_06_2020_a_06_2022.xlsx”, traz uma coluna “Praça”, correspondente ao local de transmissão, “Data”, “Faixa Horária” e uma coluna para cada emissora, contendo a programação para cada slot de junho de 2020 a junho de 2022.

Para fins de análise de dados, esse documento foi agregado com o primeiro através da adição de duas colunas (“Programa” e “Gênero”) seguindo a paridade de horário entre as tabelas. Além disso, os horários foram formatados em intervalos de 30 minutos através de média aritmética para repetição. O campo de data também foi dividido em colunas de ano, mês e dia.

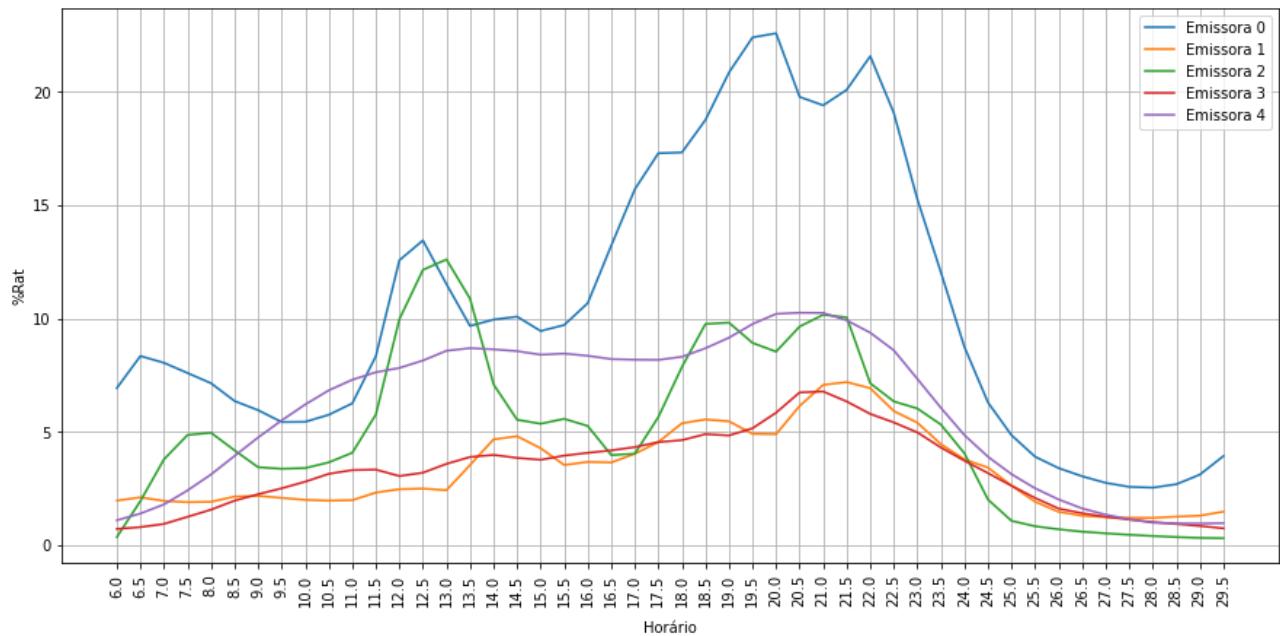
Ademais, é importante notar que não há contingências em relação à qualidade e preservação dos dados, ou seja, células vazias ou “NaN” não são um problema até o momento.

Por fim, os dados disponibilizados pelo Kantar IBOPE são de natureza sigilosa, de modo que o nome das emissoras não deve ser mencionado em qualquer documento público.

Análise preliminar dos dados

Para todas as análises, foram consideradas apenas as audiências em Rat, conforme recomendação do cliente. Isso se dá porque o Rat é justamente um cálculo mais significativo sobre os valores de Fid e Shr.

Audiência por hora por emissora (Seg a Sex)



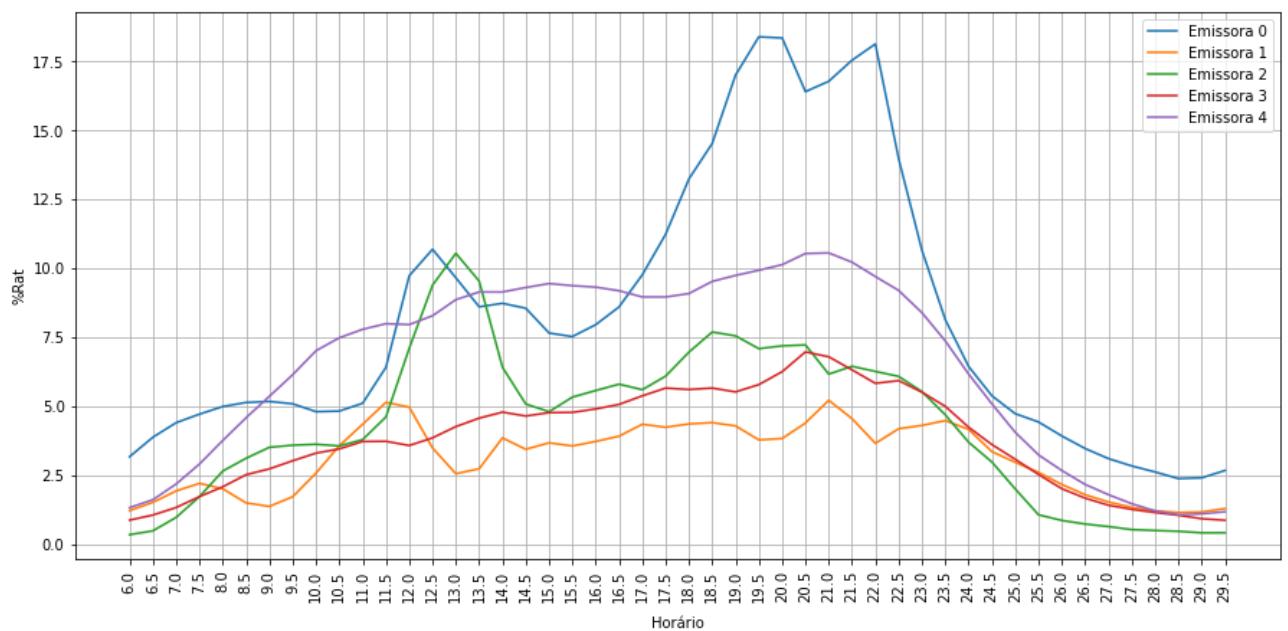
Este gráfico apresenta cada emissora como uma linha e mostra a audiência de Rat de domicílios totais a cada hora do dia durante a semana. Nesse sentido, há uma dominância clara da Emissora 0 durante todo o período, exceto das 9h30 às 11h15, quando a Emissora 4, representando itens não identificados, como streaming, a supera. Uma hipótese que explica esse comportamento é o fato de este ser o horário preferido por donos(as) de casa para realizar tarefas domésticas; nesse contexto, é provável que elas utilizem os desenhos animados disponíveis em plataformas de streaming para distrair as crianças, segundo insight do parceiro de projetos.

Ademais, o primeiro pico relativo ocorre às 6h30. Sugere-se que isso ocorra devido à veiculação do primeiro evento jornalístico do dia justamente no momento em que a maioria das pessoas acorda para ir à escola ou ao trabalho. Isso volta a ocorrer ao meio-dia, aproveitando a atenção das pessoas em seu horário de almoço. Aqui, há uma competição acirrada entre a Emissora 0 e a Emissora 2, dado que ambas apresentam o segundo jornal do dia nesse horário. Percebe-se, portanto, que a audiência se divide entre as duas, o que indica a necessidade de se diferenciar e segmentar o conteúdo a fim de melhor atender a população capixaba. Uma possibilidade seria dedicar mais tempo a assuntos locais e temas culturais para gerar mais identificação no público, instigando um sentimento de orgulho e pertencimento que os fará voltar para o programa dia a dia.

A partir daí, há uma baixa de audiência durante o final do bloco jornalístico e bloco de filmes, provavelmente por ser um horário em que a maioria das pessoas está ocupada com suas tarefas diárias, profissionais ou não.

A audiência volta a subir no final da tarde com a transmissão das primeiras telenovelas e aumenta consistentemente até cerca das 20h, quando começa o principal programa jornalístico do dia. Nesse momento, há uma queda súbita com grandes chances de ser motivada por divergências ideológicas. Por fim, o último pico se dá às 22h30, em que ora passam filmes, ora reportagens, ora reality shows populares.

Audiência por hora por emissora (Sábado)



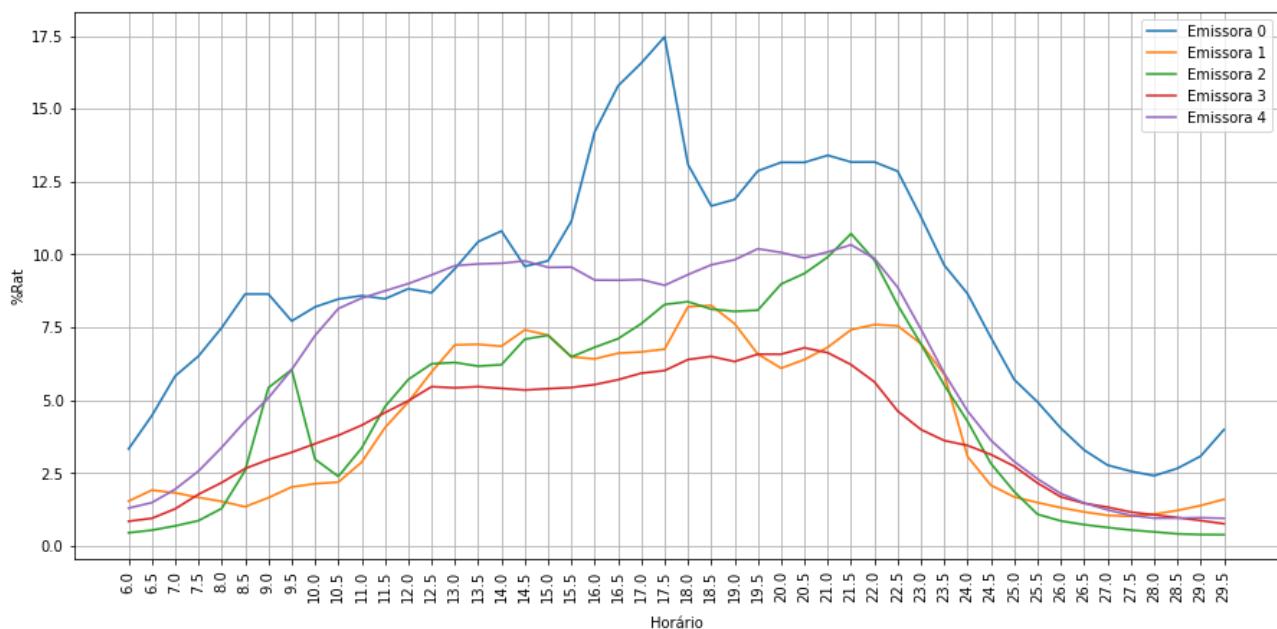
Aos sábados, nota-se uma configuração relativa semelhante de audiência entre as emissoras. Porém, ao tratar de valores absolutos, percebe-se que as porcentagem de Rat são claramente menores aos sábados, chegando à máxima geral de 18 contra a de 21 durante a semana.

Ademais, dentre outras particularidades, há uma presença considerável da Emissora 4, representando plataformas de streaming e outros conteúdos não identificados pelo PeopleMeter. Isso é visível pela primeira vez das 8h45 até as 11h30. Nossa hipótese, nesse caso, segue a linha da anterior em que pais e guardiões utilizam essas ferramentas para distrair as crianças durante o período da manhã. Outra explicação é um maior consumo de entretenimento durante o período da manhã em finais de semana, em que a família se reúne para tomar café e pode compartilhar conteúdos no tablet e no celular. Já os picos na hora do almoço seguem a mesma lógica do resto da semana, ainda que com valor absoluto inferior a 20%.

Durante a tarde, a Emissora 4 novamente sobe em audiência comparativamente, apesar de se manter no mesmo valor absoluto dos dias úteis. Isso indica que as pessoas assistem a conteúdo de streaming nesse horário mais consistentemente do que à TV aberta, isto é, em seu tempo de descanso, elas preferem streaming aos filmes e shows de auditório característicos dos sábados.

Por fim, o resto do dia segue o comportamento de segunda a sexta.

Audiência por hora por emissora (Domingo)



Domingo é um dia atípico, como pode ser claramente visto pelas flutuações nas linhas. O dia já começa com audiência considerável e consistente das 8h30 às 9h para a Emissora 0, que veicula eventos rurais nesse período. Um fato curioso é que a queda da Emissora 0 entre 9h e 9h30 se alinha quase que perfeitamente com a subida da Emissora 2, indicando uma migração dos telespectadores para assistir à programação mais decididamente capixaba dessa emissora para esse slot.

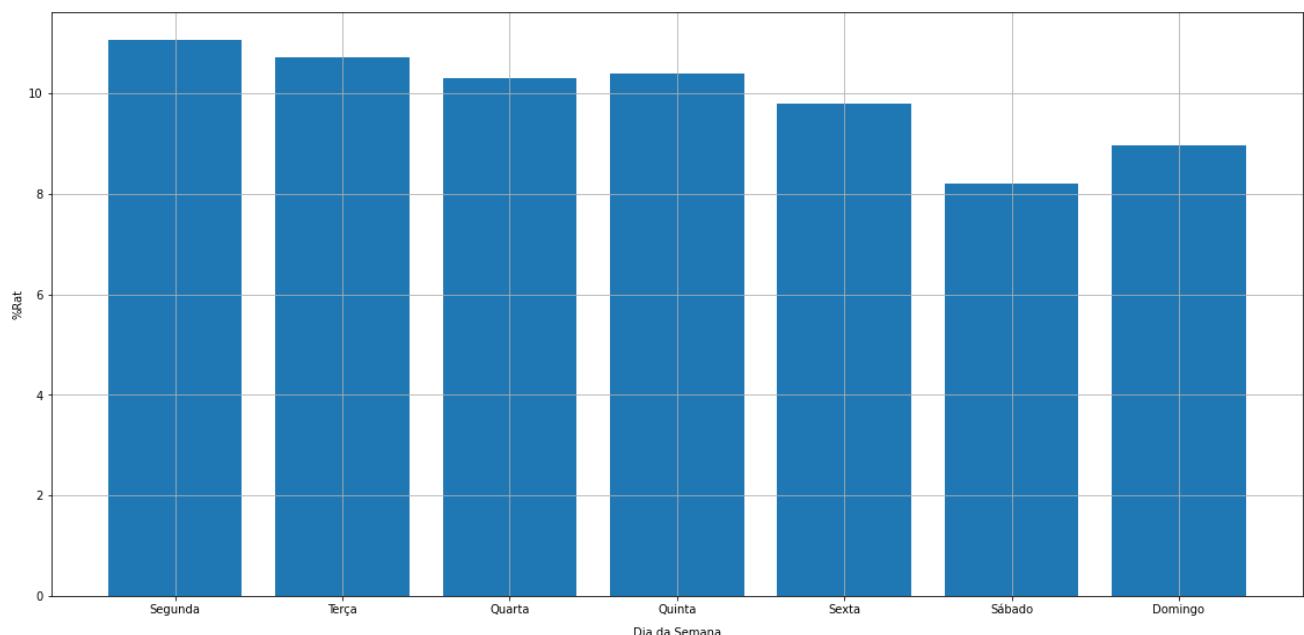
No horário do almoço, os picos não ocorrem mais devido à ausência de eventos jornalísticos. Ainda assim, o leve aumento às 13h30 para a Emissora 0 mostra que a população gosta de sentar para assistir a filme – conteúdo comum para esse período aos domingos – após se alimentar.

O pico principal acontece mais cedo, às 17h30, quando começam os programas de auditório mais populares de cada emissora. Além disso, existe um público cativo consistente para a programação das 20h às 23h, quando são transmitidos blocos de reportagens e reality

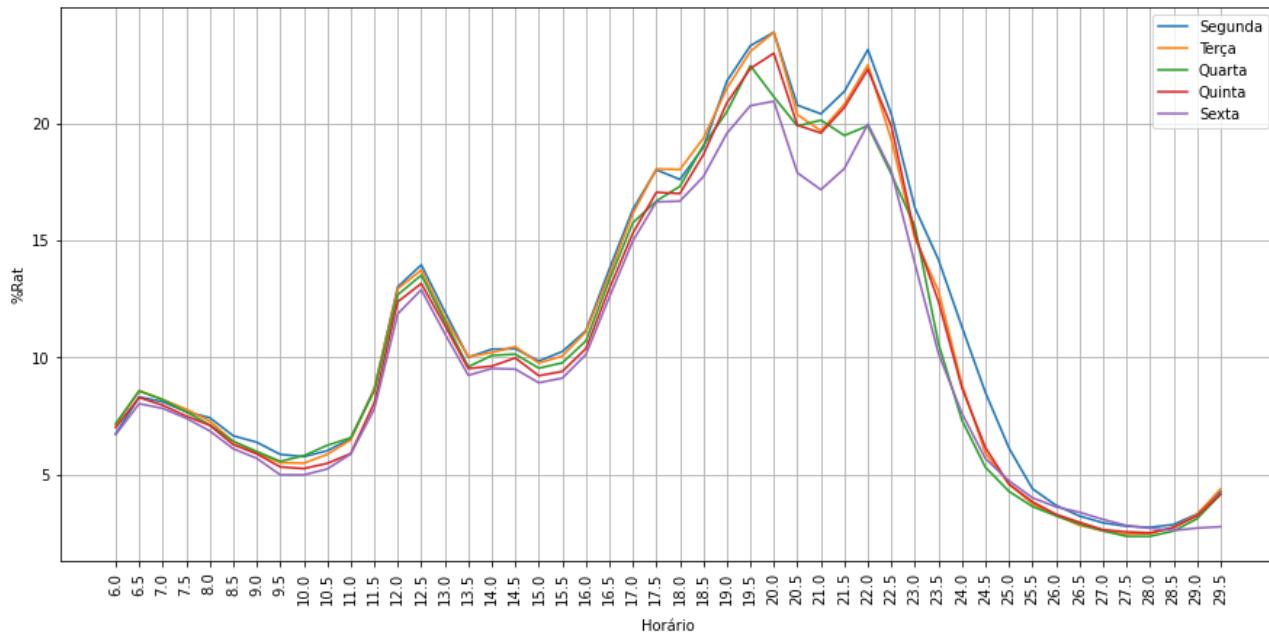
shows. Ainda assim, em segundo nível, a Emissora 4 ainda alcança uma audiência competitiva com seus eventos de auditório mais tradicionais.

Audiência da Emissora 0 por dia da semana

Dada a dominância geral da Emissora vista nos gráficos anteriores, decidimos seguir as análises apenas com a audiência dela, pois outras emissoras não apresentam valores significativos em comparação.



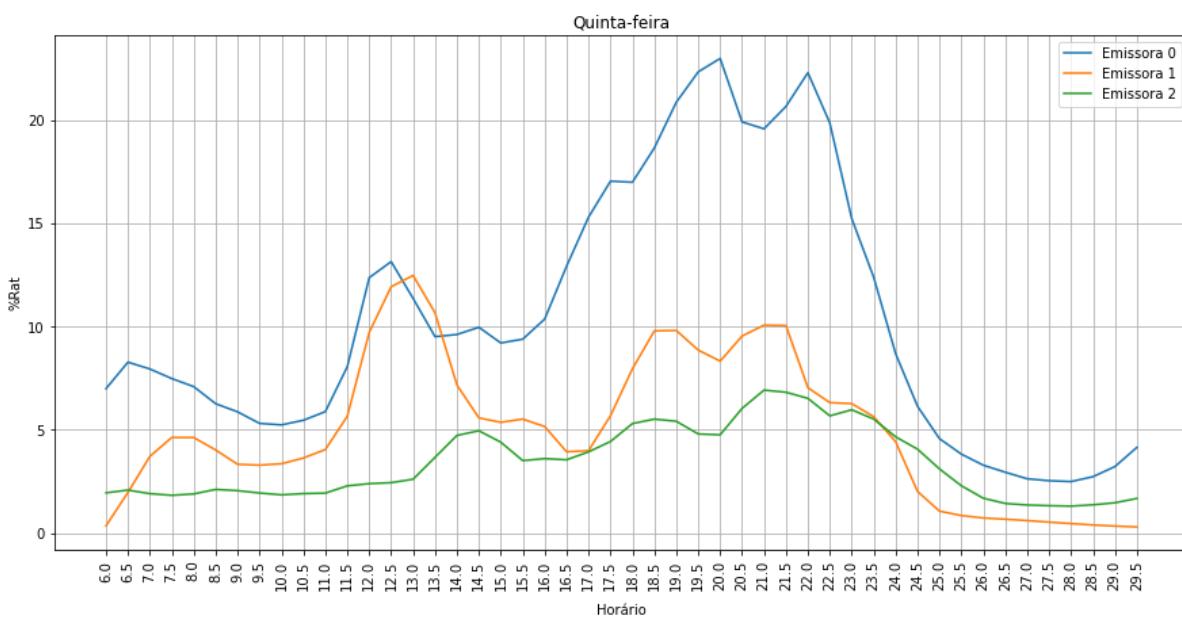
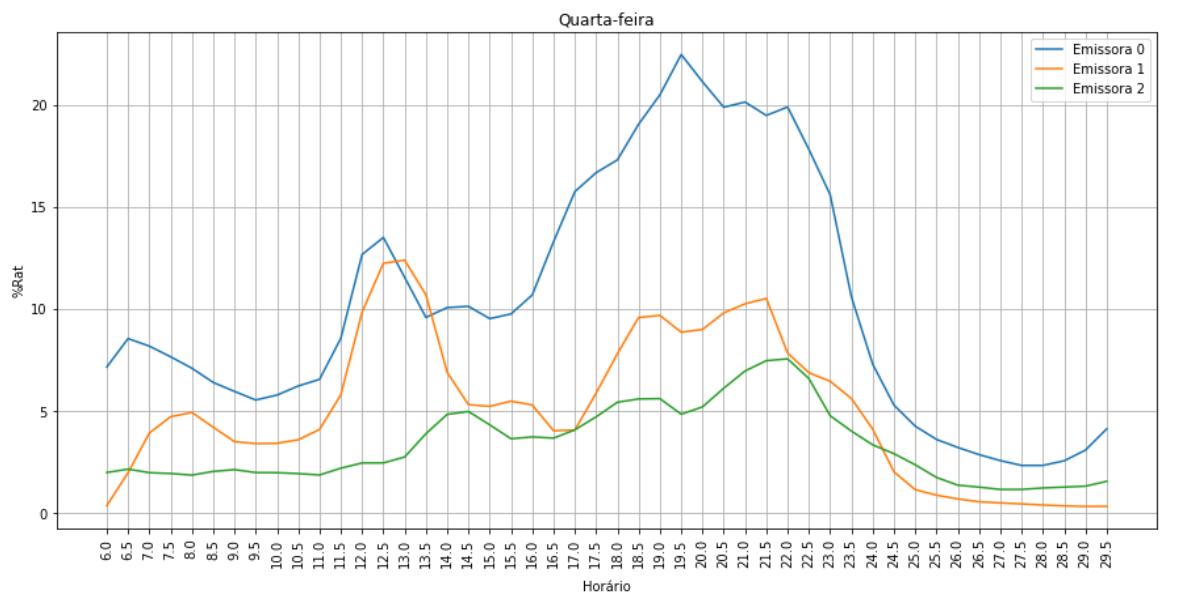
A primeira coisa a se notar é que as audiências para cada dia, exceto sábado e domingo, apresentam pouca variabilidade, ficando sempre em torno de 10-10.5 Rat%. Nesse sentido, as baixas no final de semana são fáceis de explicar: as pessoas têm mais tempo livre nesses dias e buscam outras formas de lazer além da televisão. Por outro lado, explicar por que mais pessoas sintoniza às segundas tem se provado um desafio. A programação semanal em certo mês tende a se manter regular, de modo que não grandes mudanças entre segunda e terça, por exemplo. Para aumentar nosso entendimento nesse aspecto, geramos também os gráficos de horário para cada dia da semana, a fim de visualizar quais blocos e, consequentemente, quais programas mais atraíam os telespectadores.



Como sugerido, as audiências são muito semelhantes durante a semana. Ainda assim, olhando com mais atenção, percebe-se que os valores para sexta-feira tendem a cair cada vez mais conforme o dia progride, decerto porque as pessoas preferem sair após o trabalho nessas noites em antecipação do final de semana.

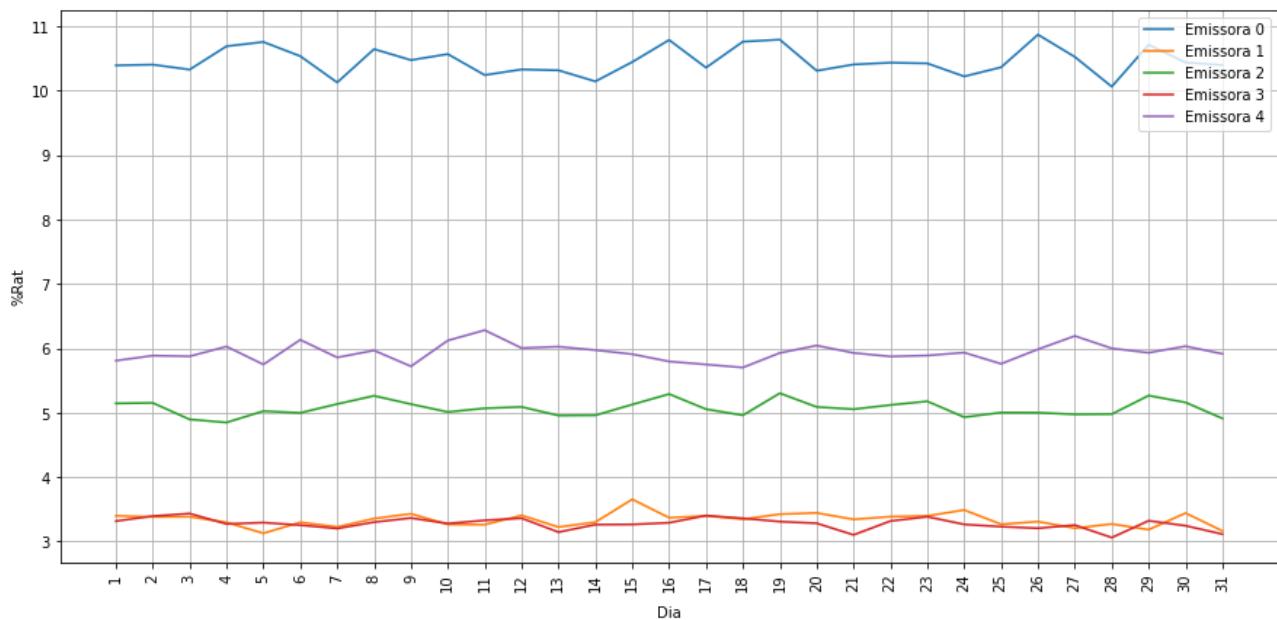
Já a terça-feira segue o padrão de segunda com muita precisão, exceto no final da noite e início da madrugada. Um ponto curioso aqui é que, pelo jeito como as horas são apresentadas no gráfico, temos a impressão de que a hora 29.0 no gráfico de segunda-feira representa a madrugada de terça-feira. Porém, na realidade, esse horário se refere à madrugada entre domingo e segunda-feira, pois, na planilha, todos os dias iniciam com horário 24.0 e só depois retornam para 6. Com esse insight, percebemos que um motivo para a maior audiência na madrugada de segunda-feira é simplesmente o fato de esta ser uma “extensão” de domingo, de modo que a muitas pessoas ainda não voltaram a seu horário de sono habitual e ficam acordadas até mais tarde.

Ademais, para examinar as baixas relativas nas quartas e quintas, plotamos também as audiências por horário para cada um desses dias considerando também outras emissoras. O objetivo era identificar se havia migração por algum programa específico.



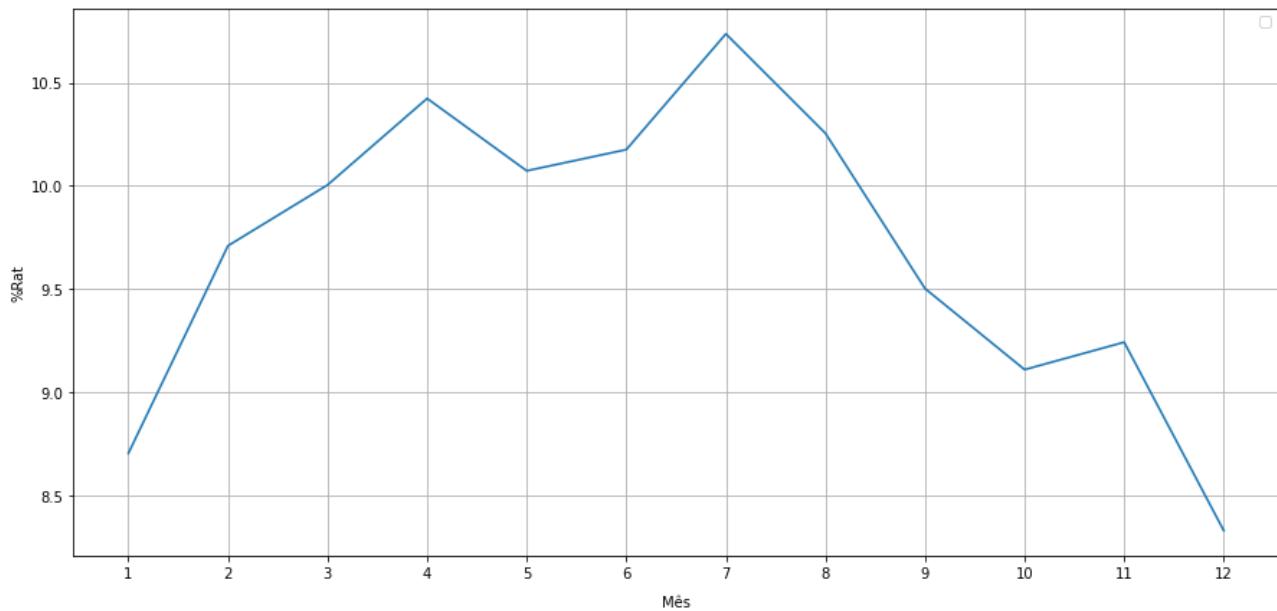
Ambos os dias, entretanto, apresentam flutuações semelhantes às dos outros dias da semana, como aferido pela média semana no primeiro gráfico mostrado nessa análise. Assim, ainda nos falta informação para induzir especificamente o que pode influenciar esse comportamento diferenciado às quartas-feiras e, em menor grau, às quintas.

Audiência por dia do mês por emissora



Novamente, a dominância da Emissora 0 é clara, e as audiências flutuam em um intervalo muito pequeno. Nesse sentido, ao comparar as datas dos picos relativos com a programação vigente, há resultados inconclusivos. Em alguns casos, realmente houve eventos especiais, como jogos de futebol, reality shows e blocos de reportagens populares. Em outros, no entanto, não há explicação clara do porquê esse dia em particular atrairia mais público. Por isso, deduzimos que, salvo raras exceções, não há muita variação de comportamento com o passar do mês.

Audiência da Emissora 0 por mês

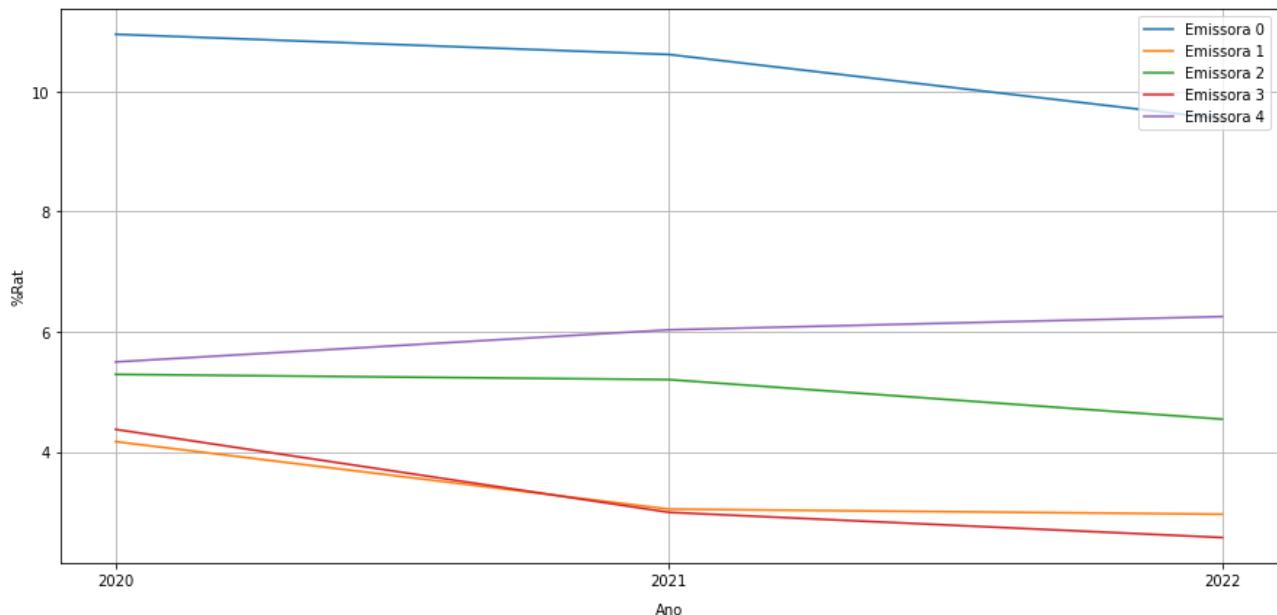


Há dois picos importantes na análise mensal de audiência. O primeiro ocorre em abril, provavelmente porque esse foi o mês em que o reality show mais popular da Emissora 0 foi televisionado nos últimos dois anos. Já o segundo se dá em julho, decerto pela veiculação das últimas Olimpíadas nesse período.

A queda que se segue pode ser explicada, em parte, pela usual troca de novelas e séries no segundo semestre. Nesse contexto, vê-se que a audiência ainda se mantém relativamente alta no mês de agosto, talvez pelo interesse da população nas novidades, mas diminui com o passar dos meses quando essa novidade inicial se esvai.

Há um leve aumento em novembro, que nós atribuímos como hipótese à finalização de atividades profissionais e escolares. É o momento em que as responsabilidades diminuem e as pessoas começam a tirar férias. Já em dezembro, com os planejamentos para as festas e possíveis viagens e outras formas de lazer presencial, a audiência volta a cair.

Audiência por emissora por ano



Nota-se uma leve queda de audiência da Emissora 0 de 2020 para 2022. Isso é explicado pela normalização pós-pandemia. Durante a crise de COVID-19, as pessoas ficaram muito mais tempo em casa e, portanto, tiveram mais oportunidades de assistir à televisão, além de terem mais interesse nas notícias quanto ao status da doença. Com a reabertura comercial, os níveis de audiência voltaram para o que acreditamos serem níveis pré-pandemia. Entretanto, como não temos dados de anos anteriores, não podemos validar essa hipótese.

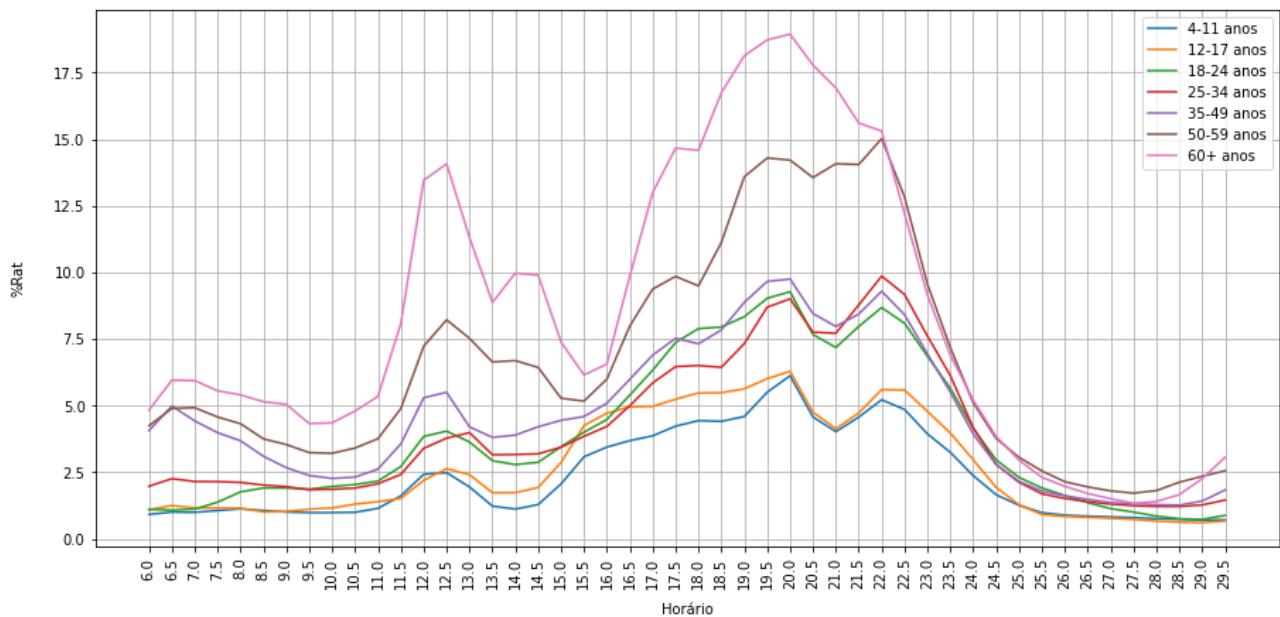
Percebe-se também que a Emissora 4, representando plataformas de streaming, teve certo crescimento nos últimos anos. Isso é corroborado por pesquisas que mostram que o consumo de streaming aumentou em 34% com a criação de mais conteúdo exclusivo ("State of Mobile 2022 - data.ai", 2022).

Audiência da Emissora 0 por classe socioeconômica



Na análise socioeconômica, o pico geral pertence à classe DE no bloco jornalístico do período da noite. Assim, deduz-se que essa classe tem preferência pelo jornalismo tradicional, enquanto as outras classes consomem uma programação mais variada e menos política, com programas de auditório, jornais locais e novelas, durante a manhã e tarde.

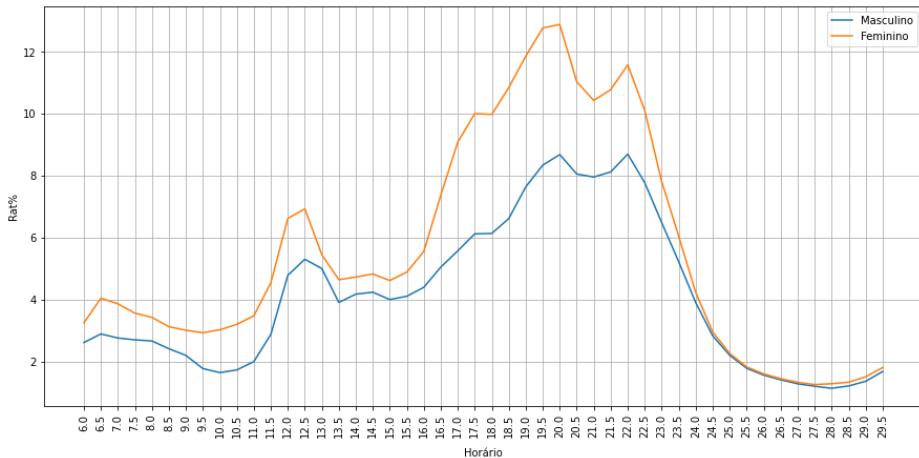
Audiência da Emissora 0 por faixa etária



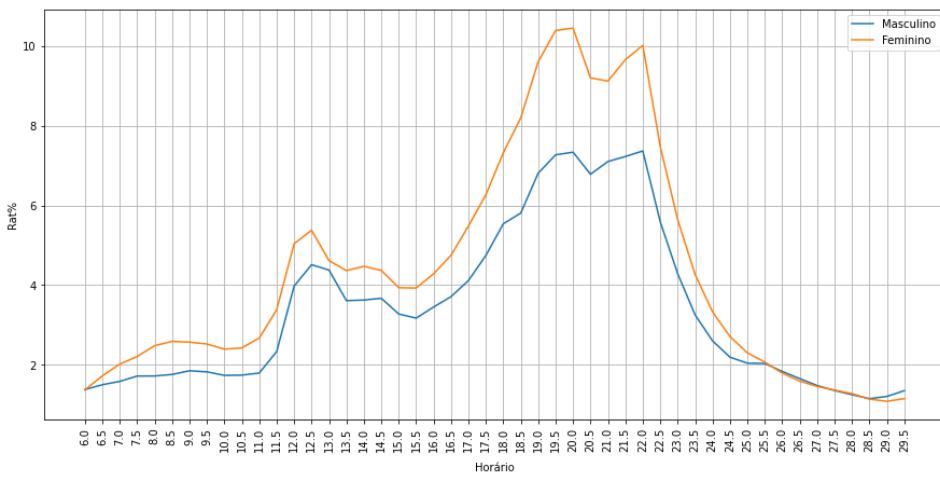
No que tange à faixa etária, a terceira idade domina em quase todos os casos e especialmente durante a manhã, provavelmente pela incidência de aposentadoria e horários mais flexíveis, além da tendência a acordar mais cedo (AMANDA CHAN 12 APRIL 2011, [s.d.]). Apesar disso, todas faixas apresentam comportamento e flutuações parecidas, ainda que em níveis diferentes, indicando que a categoria dos eventos a cada faixa horária agrada a população em geral de forma consistente. Um ponto interessante é que crianças aparecem com muito menos frequência, provavelmente pela escola e preferência por Youtube e outras plataformas.

Audiência total da Emissora 0 por gênero

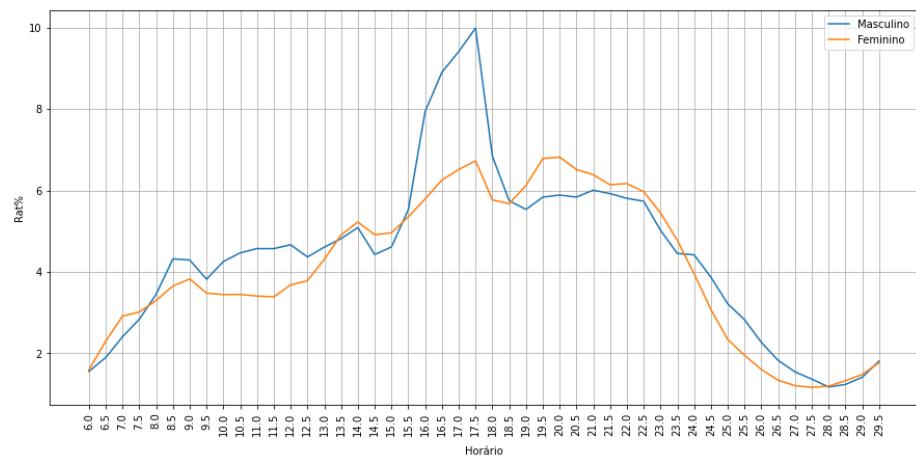
Segunda a sexta



Sábado

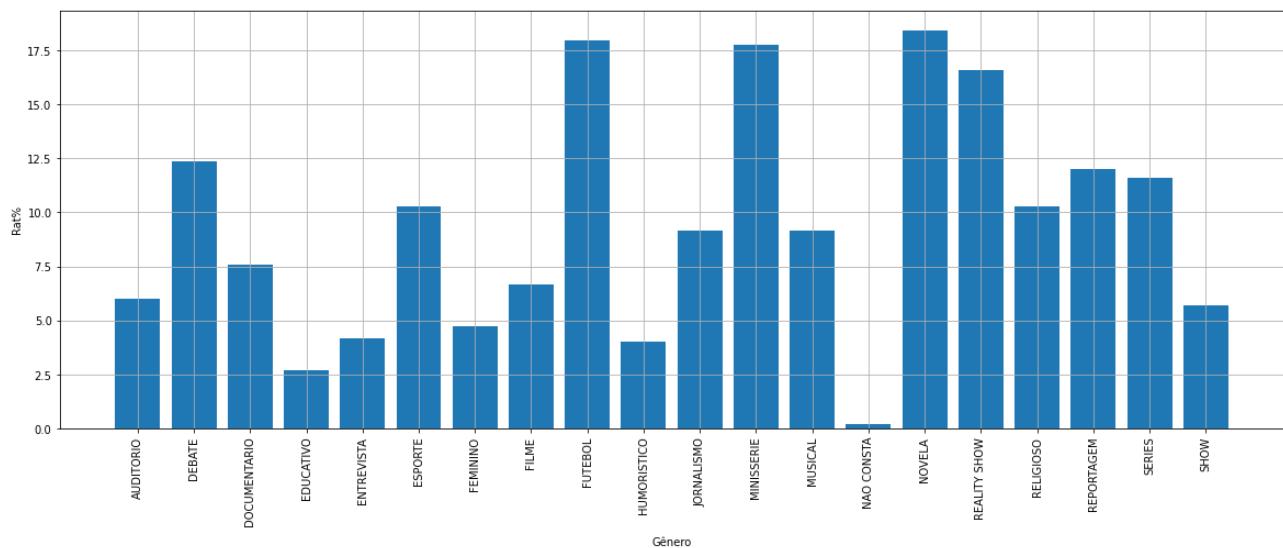


Domingo



A análise desses três gráficos mostra que, em geral, mulheres predominam na audiência geral por cerca de 1.5 Rat% durante todo o dia e quase 4 pontos durante o horário de pico. Esse resultado é surpreendente, dado que hoje apenas 7% das mulheres se declaram donas de casa e têm acesso a uma televisão o dia inteiro (“Parcela da população que se declara dona de casa cai para 7% em 26 anos”, 2019). Outra hipótese não validada é um maior interesse feminino por telenovelas. A exceção é o dia de domingo, quando homens ultrapassam as mulheres no horário de jogos de futebol.

Audiência da Emissora 0 por gênero



Geramos esse gráfico com o objetivo de entender quais gêneros eram mais assistidos. Nesse sentido, definimos uma lista de “preferidos” pelo público com base nos picos relativos, chegando aos itens de Debate, Futebol, Minissérie, Novela, Reality Show, Reportagem e Séries. Curiosamente, o gênero Jornalismo, do qual esperávamos uma das maiores audiências, não figura nessas máximas relativas. Isso é tão inusitado para nós que suspeitamos que talvez o PeopleMeter não esteja classificando todos os blocos jornalísticos corretamente, dado que são eles que atraem as maiores taxas de audiência nas análises por hora.

Considerações sobre o resultado desejado

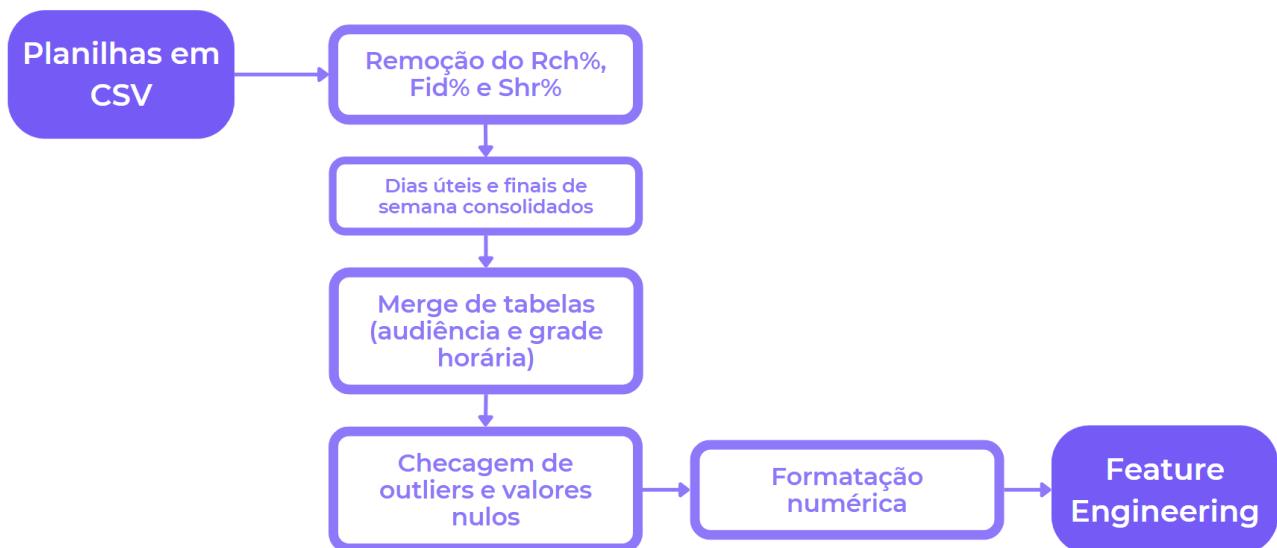
O resultado desejado da predição (saída) tem dois componentes. O primeiro e principal é o score de audiência esperado para um evento de determinadas características em certo dia e horário. Por ser um número dentre inúmeras opções, sua natureza é contínua (float). O segundo output planejado é uma lista das variáveis consideradas para chegar ao resultado principal e o peso de cada uma delas. Até o momento, não ficou claro se esse resultado será classificatório, partindo do princípio de que nem todas as variáveis possíveis serão utilizadas em cada predição, ou contínuo, caso os pesos de cada variável possam mudar de predição em

predição. Será necessário estudar mais sobre modelos preditivos e fazer as primeiras implementações para sanar essa dúvida.

4.3. Preparação dos Dados

Pré-processamento dos dados

Os passos descritos abaixo referem-se às manipulações efetuadas para explorar dados, gerar gráficos e decidir quais features utilizar em nosso projeto. Nesse sentido, o objetivo era diminuir o tempo de processamento das planilhas no código, eliminar informações desnecessárias e formatar certos dados para otimizar filtros e diferentes visualizações antes de selecionar features, conforme demonstrado no fluxograma abaixo.



O código associado a cada passo pode ser conferido em nosso notebook do [Colab](#) e as planilhas resultantes de cada etapa estão disponíveis no Drive do grupo Jupyter.

Anonimização dos dados

Dado que os dados fornecidos pelo Kantar IBOPE são confidenciais e exigem limitações contratuais com a TV Gazeta, principalmente no que tange às associações entre valores de audiência e emissora, realizamos um processo de anonimização sobre os arquivos antes de adicioná-los ao Google Drive e ao Colab. Este processo teve duas etapas:

1. No arquivo "TV_Histórico.xlsx", substituímos todas as menções a emissoras por codinomes seguindo o padrão "Emissora A", "Emissora B" e "Emissora C";
2. No arquivo "grade_Diária_06_2020_a_06_2022.xlsx", não só realizamos as mesmas substituições de nomes de emissoras como também transformamos os títulos dos programas em "PROGRAMA 1", "PROGRAMA 2", "PROGRAMA 3", etc. Deve-se mencionar que esse método repete codinomes quando o programa em si se repete (por exemplo,

um evento de jornalismo diário denominado "PROGRAMA 56" será substituído por esse codinome todos os dias na planilha) e que a numeração reinicia a cada emissora (ou seja, "PROGRAMA 56" corresponde a programas diferentes em diferentes emissoras). Por fim, o arquivo original trazia programas e gêneros concatenados em uma única coluna, no formato "PROGRAMA / CATEGORIA". Porém, para facilitar a anonimização dos programas, segregamos, desde o início, essas informações em duas colunas para cada emissora: "EMISSORA X (Programa)" e "EMISSORA X (Categoria)".

OBS: Será enviado um dicionário com os dados e novas nomenclaturas ao parceiro.

Otimização de processamento dos arquivos

O documento "audiencia_original.csv", continha cerca de um milhão de linhas quando somadas todas as suas abas. Logo, o carregamento desse arquivo no código demandava muito tempo e recursos computacionais, além de gerar dataframes mais difíceis de manipular.

Sendo assim, nossa primeira ação em preparar os dados foi separar esse documento Excel em dezoito arquivos CSV, um para cada aba. Também eliminamos as colunas de audiência e demográficos que não se referissem ao "Rat%", métrica mais relevante. Desse modo, por enquanto, removemos os valores de "Shr%", "Fid%" e "Rch%", para ter acesso facilitado aos dados desejados e, também, diminuir o overhead do algoritmo através de um formato mais leve que o XLSX.

Contudo, durante o workshop com o parceiro da Sprint 2, aprendemos que também há interesse por parte da TV Gazeta em modelos que considerem os outros tipos de audiência. É algo que pretendemos explorar mais a fundo na próxima sprint.

Formatação de datas

As datas no arquivo original estão formatadas como "dd/mm/aaaa", seguindo o padrão brasileiro. Inicialmente, consideramos desmembrar essa coluna em colunas de "Dia", "Mês" e "Ano"; entretanto, após alguns testes, percebemos que isso dificultava algumas análises mais profundadas dos dados no código. Por exemplo, se quiséssemos filtrar a primeira semana de cada mês, teríamos de aplicar diversas condições sobre as coluna tanto de dias quanto de meses. Em vez disso, decidimos deixar a planilha como está e apenas converter os valores das colunas em objetos de data no Python, através do método "pd.to_datetime", quando fosse necessário manipular as datas de alguma forma. Um detalhe importante é que a conversão do XLSX para CSV com encoding UTF-8 formatou todas as datas, automaticamente, para "aaaa-mm-dd". Isso não afeta a conversão para objetos de data mencionada anteriormente.

Agregando emissoras

Para agregar informações semelhantes, facilitar a análise por emissora e favorecer a separação de sets de treinamento no futuro, juntamos os CSVs de dias úteis, sábados e domingos de cada emissora em um único CSV para cada uma delas. Em mais detalhes, dado que com a etapa de

otimização conseguimos três arquivos por emissora (“Seg a Sex”, “Sáb”, “Dom”), decidimos, nesta fase, juntar esses arquivos em um só para cada emissora, ordenado cronologicamente.

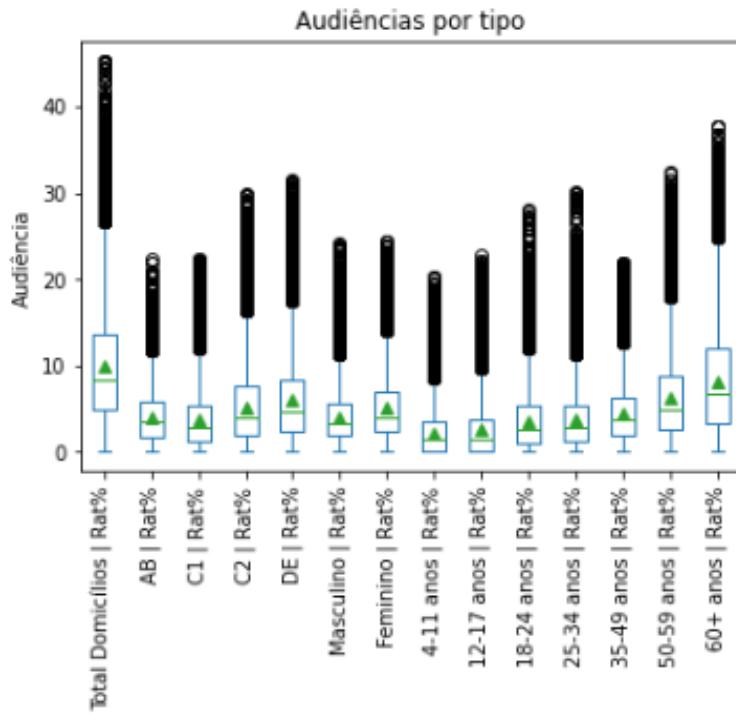
Merge com grade horária

Com todos os dados de audiência por emissora compilados, restou mapear cada faixa horária de cada dia com o programa (e categoria) veiculados nesse período. Essas informações estão disponíveis no CSV “grade.csv”, que detalha o evento e o categoria para cada faixa horária e dia do período de 2020 a 2022. Sendo assim, agregamos uma coluna de “Programa” e uma de “Categoria” às planilhas das três emissoras contempladas no documento de grade horária, fazendo um match com as datas e horários presentes na origem e destino dessa transformação.

Checando a existência de outliers

Para determinar se existem valores muito extremos nos dados que poderiam enviesar o modelo preditivo, tiramos a média e desvio-padrão de cada coluna em cada planilha. Então, conferimos se existia algum valor mais distante da média do que três desvios-padrão, pois este é o critério mais utilizado para identificar outliers. Para nossa surpresa, isso ocorreu com muita frequência, chegando a 21.000 ocorrências, o que nos causou estranheza. Parecia que estávamos perdendo dados demais nessa operação.

Para avaliar melhor o cenário, decidimos, então, gerar gráficos box-plot de todas as colunas. Esse tipo de gráfico é útil porque explicita não só a média de um dataset como também os percentis 25 e 75, a amplitude dos dados e os outliers de forma muito mais visual e intuitiva. Nesse modelo, outliers são valores além de $1,5 * \text{IQR}$ (a amplitude entre os percentis 25 e 75) da média e geralmente aparecem como pontos isolados acima ou abaixo das caudas do gráfico, o que demonstra sua falta de encaixe com o resto do dataset.



Ao analisar os gráficos de nossos dados, entretanto, vemos que não há apenas alguns pontos isolados, e sim de uma incidência tão alta de valores extremos que mais parecem uma continuação das caudas do gráfico. Nesse sentido, é fácil visualizar que esses valores formam um padrão distinto no dataset e que eliminá-los descaracterizaria nossos dados. Afinal, outliers são, por definição, exceções. Aqui, é evidente que audiências altas estão longe de serem excepcionais.

Uma explicação contextualizada do porquê isso acontece é que os picos de audiência, apesar de regulares, acontecem com baixa frequência quando comparados com toda a amplitude de faixa horária. Em outras palavras, dentre as 24 horas possíveis de programação, apenas uma ou duas atingem esses picos vertiginosos. Assim, temos valores altíssimos e regulares, porém não frequentes o suficiente para puxar a média em sua direção, dado que as outras 22 horas do dia marcam scores muito menores. Logo, temos que a média permanece baixa, porém há uma significativa frequência de valores elevados além da amplitude esperada. Nesse contexto, decidimos manter todos os dados, pois não há outliers reais a serem removidos.

Checando valores nulos e ausentes

A checagem de valores nulos, ausentes e/ou vazios foi feita através do método “`isnull()`”, da biblioteca Pandas. O resultado foi falso para as três planilhas de emissoras. Portanto, não foi necessário nenhum tratamento nesse quesito.

Checando categorias de baixa frequência

Ainda na busca por outliers e exceções, percebemos que não deveríamos examinar apenas valores absolutos, mas também a frequência de certas variáveis categóricas. Nesse sentido, quisemos garantir que os gêneros de programa mencionados nas planilhas apareciam

com frequência suficiente para que conclusões pudessem ser tiradas de forma estatisticamente válida. Por isso, realizamos a contagem de cada categoria presente na grade horária e descobrimos que algumas delas possuíam frequências baixíssimas, com apenas 2 ou 7 ocorrências.

Assim, acabamos eliminando as categorias “NAO CONSTA”, “SORTEIO”, “OUTROS” e “TELE VENDAS”, pois não continham mais de 12 incidências no melhor dos casos. Nossa critério para essa decisão foi eliminar todas as categorias até a de “DEBATE”, que traz um aumento relativo de aparições, chegando a 36, e é inherentemente relevante às emissoras por suas ramificações políticas.

Seleção de features

A finalidade da GIA é clara: prever o score de audiência para futuros programas com certas características (gênero) em certos dias e horários.

Nesse contexto, com a preparação anteriormente detalhada executada, pudemos começar a selecionar nossas features. Iniciamos pelas datas, pois seriam mais fáceis de manipular e preparar através das bibliotecas do Python. Em nossos estudos prévios, percebemos que o dia do mês não exercia influência significativa nas audiências (vide gráficos da seção anterior), contudo o dia da semana e o mês em si, sim. Assim, determinamos que essas duas informações (dia da semana e mês) deveriam ser contempladas nas features.

A fim de utilizar esses dados em um modelo preditivo, no entanto, precisávamos transformá-los em valores numéricos. Por isso, transformamos as variáveis categóricas associadas aos dias da semana (“Segunda”, “Terça”, etc.) em números de 1 a 7, seguindo a técnica de label encoding. Isso garantiu uma relação ordinal entre os valores, a qual desejávamos por conta da natureza ordinal dos dias da semana em si (segunda-feira está mais relacionada à terça-feira do que à sexta, por exemplo, e isso poderia trazer insights inesperados para o AI). Além disso, destrinchamos a coluna “Data” em uma de mês, composta por inteiros de 1 a 12, através da transformação das datas em objetos datetime no Python e o subsequente acesso ao atributo “mês” de cada um deles.

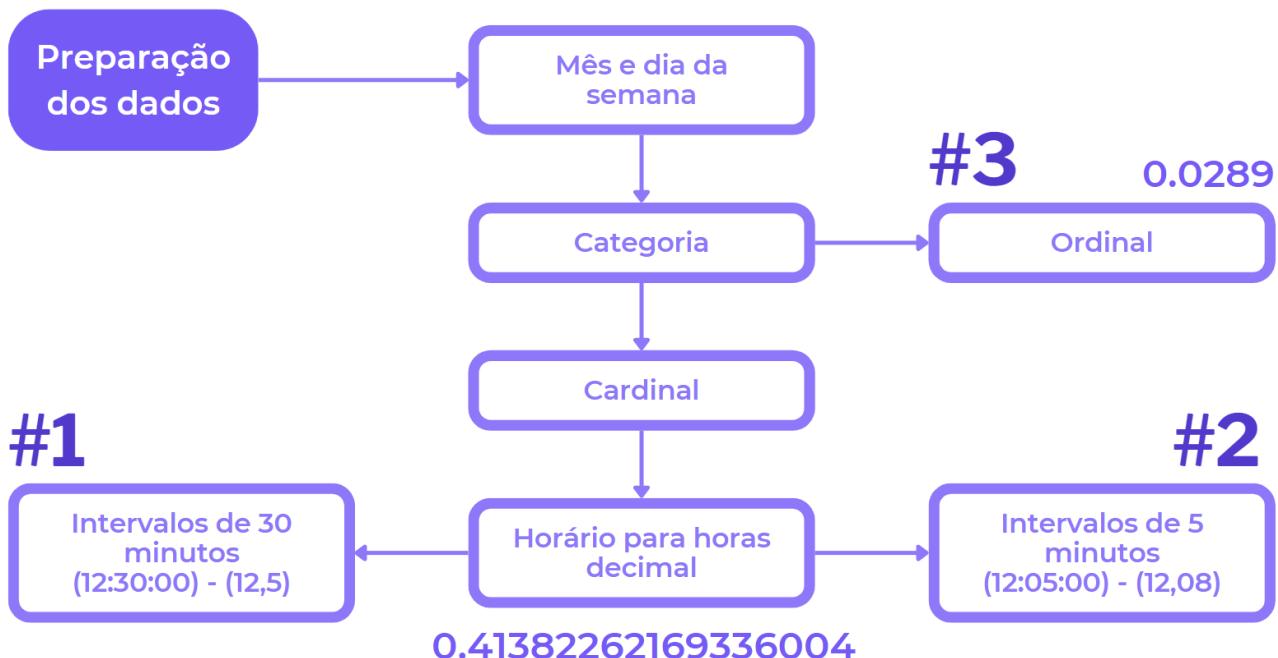
A próxima feature, por sua vez, deveria contemplar, de alguma forma, a faixa horária do programa. A seleção desse atributo causou muitas dúvidas e hipóteses. Pensamos em quebrá-lo em hora de início e hora de término, adicionar uma feature de duração, desmembrar o horário em um inteiro de hora e um inteiro de minuto, e muitas outras opções. Para fins de simplicidade e de ter um lugar pelo qual começar, todavia, resolvemos implementar dois tipos de entradas relacionadas a horário:

1. Horário em número decimal com intervalos de 30 minutos (12:30:00 se torna 12,5);
2. Horário em número decimal com intervalos de 5 minutos (12:05:00 se torna 12,08).

Não houve um raciocínio lógico muito extenso para a escolha desses formatos específicos além da noção de que precisávamos começar a testar alguma hipótese. Na teoria, todas as opções pareciam igualmente válidas. Portanto, escolhemos as mais simples para começar a medir e comparar desempenhos.

Por fim, para contemplar a categoria do programa, decidimos testar tanto a transformação por label encoding (ordinal, que dá um valor crescente começando em 1 a cada categoria) quanto a por one-hot encoding (cardinal, que cria colunas novas para cada variável única e categoriza os valores como 1 ou 0). Esperávamos que a versão cardinal tivesse resultados superiores desde o início, dado que categorias não possuem ordem inerente e essa premissa poderia confundir o AI. Ainda assim, a discrepância foi tão grande que acabamos testando essa abordagem apenas uma vez e desistimos logo em seguida.

Assim, ao final, idealizamos e testamos três hipóteses para feature engineering. A saída para todas elas consistia no Rat, a principal métrica analisada pelo parceiro e um bom ponto de partida para nossas explorações. Abaixo, detalhamos as hipóteses.



1. Entradas de dia da semana, mês, categoria cardinal e hora de início de meia em meia hora. Saída de Rat%. Predição com regressão linear;
2. Dia da semana, mês, categoria ordinal e hora de início de meia em meia hora. Saída de Rat%. Predição com regressão linear;
3. Dia da semana, mês, categoria cardinal e hora de início de cinco em cinco minutos. Saída de Rat%. Predição com regressão linear.

Teste de hipóteses

A partir das features selecionadas, resolvemos testar cada hipótese para averiguar a eficácia das seleções em si e das transformações utilizadas (one-hot encoding/label encoding). Essa avaliação foi feita através de três métricas: erro médio quadrado, que penaliza mais erros maiores; erro médio, que mostra a discrepância média entre valor predito e valor real; e R², que representa a eficácia da hipótese quando comparada a um modelo naive. Nessa última, objetiva-se chegar o mais próximo possível de 1.

Os resultados estão resumidos abaixo.

Hipótese 1

As features da hipótese 1 foram dia da semana, mês, categoria cardinal e hora de início. Contudo, a hora de início teve uma redução de linhas de 30 em 30 minutos, tirando a média de cada período.

Erro médio quadrado: 32.804852251521474

Erro médio: 4.62607344055152

R²: 0.4184967210608739

Aqui, o erro médio é de quase 5 pontos de audiência. Para uma amplitude de cerca de 40 pontos, isso é um erro significativo, passando de 10%. Já o R² nem chega a 0,5, mostrando-se insuficiente.

Hipótese 2

As features da hipótese 2 foram dia da semana, mês, categoria cardinal e hora de início. Contudo, a hora de início manteve a divisão de 5 em 5 minutos.

Erro médio quadrado: 24.80568954239913

Erro médio: 3.8310018190852384

R²: 0.4158580211459625

Para nossa surpresa, esse conjunto de features resultou em erros e R² similares à proposta anterior. Analisando o fato com mais atenção, percebemos que isso se dá porque a única diferença entre as hipóteses 1 e 2 é o fato de a primeira conter as médias da segunda a cada meia-hora. Logo, elas possuem valores proporcionais, cuja semelhança foi preservada no modelo preditivo. Apesar de o erro médio ter sofrido uma alteração maior, vemos que isso se dá apenas pela característica desse tipo de erro de penalizar mais erros maiores. Assim, percebemos que simplesmente mudar a periodização dos horários no futuro não altera significativamente o desempenho do modelo.

Hipótese 3

As features da hipótese 3 foram hora de início (de 30m em 30m), dia da semana, mês e categoria ordinal.

Erro médio quadrado: 55.67440408917043

Erro médio: 6.238326034220553

R²: 0.028903147388637307

O resultado, apesar de conter erro médio similar, mostra-se evidentemente fracassado quando o R² é analisado – muito, muito longe de 1. Isso provavelmente se deu porque o processo de regressão detectou “padrões” não existentes entre os valores ordinais das

categorias, como uma maior semelhança entre categorias 1 e 2, mesmo que elas não tenham nada em comum na realidade. Esse teste nos mostrou, portanto, a importância de se utilizar one-hot encoding em variáveis não ordinais.

Em suma, nossos testes de hipótese foram vitais em entender melhor o que devemos priorizar em nossas features e modelos. Com base nisso, pensamos em mais opções de entradas que pretendemos explorar nas próximas sprints, como a adição de duração, hora de término e também a conglomeração de horários em períodos maiores, tais quais “MANHÃ 1”, “ALMOÇO”, etc.

4.4. Modelagem

O processo de modelagem foi longo e com diversas reviravoltas. Iniciamos com três hipóteses de features (detalhadas na seção 4.3), algumas ideias de algoritmos e vagas suposições sobre quais métricas utilizar.

Ao final, tínhamos testado mais de 70 modelos, selecionado quais algoritmos tendiam a performar melhor com nosso dataset e entendido profundamente como cada métrica funcionava e flutuava perante mudanças de feature ou parâmetro. Assim, esse processo todo se deu em três rodadas de experimentação, cada uma rendendo novos insights e direcionando (ou redirecionando) nosso planejamento.

A tabela abaixo resume essas etapas, que serão mais detalhadas no decorrer desta seção.

#	Features	Algoritmos	Métricas	Insights
Rodada 1	Hora de início (15m), dia da semana, mês e categoria	Regressão linear	R ² , R ² ajustado, erro médio percentual e erro médio absoluto.	Regressão linear não é um bom algoritmo para nosso dataset.
Rodada 2	Hora de início (15m), dia da semana, mês e categoria	KNN, árvore decisória e random forest	R ² , R ² ajustado, erro médio percentual e erro médio absoluto.	85% de R ² é um objetivo possível para algoritmos baseados em árvore.
Rodada 3	Hora, minuto, dia da semana, dia do mês, mês, categoria, BBB e feriado	Regressão linear, KNN, árvore decisória e random forest	R ² , R ² ajustado, erro médio percentual e erro médio absoluto.	BBB e feriado são features extremamente relevantes para nosso modelo.
Rodada 4	Hora, minuto, dia da semana, dia do mês, mês, categoria, BBB e feriado	LightGBM, CatBoost e XGBoost	R ² , R ² ajustado, erro médio percentual e erro médio absoluto.	Gradient boosting oferece os melhores resultados.
Rodada 5	Hora em blocos, dia da semana, dia do mês, mês, categoria, BBB e feriado	Regressão linear, KNN, árvore decisória e random forest	R ² , R ² ajustado, erro médio percentual e erro médio absoluto.	A divisão do horário em blocos resulta em péssima performance.

Rodada 1

Nossa primeira rodada teve como objetivo inicial testar as features escolhidas na Sprint 2 (**hora de início, dia da semana, mês e categoria cardinal**, tendo o “**Total Domicílios I Rat%**” da Rede Gazeta como saída) no algoritmo de **regressão linear**.

A regressão linear trata-se da descoberta, através da álgebra linear, de uma função que descreva, com o menor erro quadrado possível, a relação entre valores de entrada (x) e valores de saída (y). Assim, traça-se uma curva aproximada a partir da estimativa dos coeficientes, que servem também para explicar como o modelo funciona.

Uma ressalva importante é que, seguindo recomendações de padronização do orientador de turma, decidimos periodizar a coluna de "Hora Início" de 15 em 15 minutos, em vez de utilizar as versões de 5 em 5 min ou 30 em 30 min da Sprint 2.

Já no que tange a **métricas**, começamos analisando, nessa ordem de importância, **R²**, **R² ajustado**, **erro médio percentual**, **erro médio absoluto**. A justificativa para essa seleção se dá, porque R avalia o quanto determinado modelo explica as variações de um dataset. Como buscamos um modelo que se assemelhe ao máximo com a realidade, precisamos justamente daquele que melhor explique e se adeque aos dados recebidos.

Porém, sabemos também que o R² nem sempre é confiável. Por ser uma métrica associada à variância, quanto mais features são adicionadas, mais ele cresce, visto que um maior número de entradas resulta em maior variância mesmo que as adições não sejam relevantes. Por esse motivo, achamos importante sempre consultar o R² ajustado, que retira esse viés e mostra a adequação real do modelo, independentemente do número de features.

$$R^2 = \frac{SSR}{SST} = \frac{\sum (\hat{y}_i - \bar{y})^2}{\sum (y_i - \bar{y})^2} \quad \bar{R}^2 = 1 - (1 - R^2) \frac{n - 1}{n - p - 1}$$

Fórmula do R² e do R² ajustado

Ademais, determinamos interessante também o erro médio absoluto, porque ele, estando na escala do output desejado, oferece uma perspectiva mais intuitiva do quão bem o modelo performa. Assim, torna-se possível definir, com mais precisão, uma faixa de tolerância de erro. Nessa mesma linha, resolvemos dar peso considerável também ao erro médio percentual, obtido através da divisão do erro médio absoluto pela média os valores reais. Essa é, claramente, uma métrica imperfeita, pois não calcula a média de cada erro dividido pelo valor real associado a ele, e sim a média dos erros dividido pelas médias das saídas corretas. Apesar disso, é uma maneira de entender, mais intuitivamente, o que o erro representa na escala do dataset, de modo que consideramos relevante utilizá-lo.

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

Fórmula de erro médio absoluto

Não consideramos o erro médio quadrado, pois, apesar de útil em muitos casos, não nos serviria tão bem quanto as outras métricas devido aos motivos explicitados acima; assim, para fins de simplicidade, dado que já tínhamos quatro métricas elencadas, decidimos ignorá-lo.

Quanto a faixas de tolerância ou objetivos para essas métricas, nessa primeira rodada ainda não as tínhamos. Como não havíamos testado quase nada, não tínhamos expectativas embasadas do que poderíamos alcançar em cada etapa. Portanto, estávamos, em outras palavras, apenas experimentando diferentes modelos para descobrir o que era possível.

Nesse contexto, começamos aplicando o método de regressão linear da biblioteca Scikit Learn nas features originais, isto é, sem nenhuma modificação de escala. Os resultados foram os seguintes:

Erro médio: 5.063221162450242
Erro médio percentual: 35.21%
R²: 0.4722920738828278
R² ajustado: 0.4722920738828278

Baseando-nos nos resultados preliminares da Sprint 2, já esperávamos que a regressão linear não passasse de 0,5 para valores “crus”. Com tão poucas features, também não causou surpresa que o R² tivesse o mesmo valor; sua importância viria, na verdade, através da comparação com outros modelos, conforme mais entradas adicionadas. Ademais, mesmo sem metas objetivas para as métricas, sabíamos que 35% de erro é um valor muito elevado.

Por isso, resolvemos tentar novamente, porém normalizando os dados (isto é, forçando-os para uma escala de 0 a 1) dessa vez. Essa técnica evita que discrepâncias de escala entre colunas prejudiquem as previsões. Assim, dado que nossos campos têm certa variação de escala (0 e 1 para categorias; 1 a aproximadamente 45 para audiência, por exemplo), supomos que a normalização poderia nos ajudar a melhorar nossas métricas.

Erro médio: 0.11222507194200755
Erro médio percentual: 35.21%
R²: 0.47229207388282823
R² ajustado: 0.47229207388282823

Apesar de os erros terem caído bruscamente, isso só se deu porque eles também foram adaptados para a escala de 0 a 1. Na realidade, a adequação do modelo permaneceu a mesma,

como pode ser visto no erro percentual e métricas de R². Logo, não foi a diferença de escalas que causou resultados negativos.

A próxima etapa, nesse sentido, foi tentar lidar com a variância individual de cada coluna através da padronização, que substitui valores pelos seus **z-scores** (número de desvios-padrão da média). A suposição era que essa performance era tão boa quanto, senão melhor do que a última tentativa, por ser apenas outra maneira de lidar com variância, tal qual a normalização. Fazendo isso, obtivemos:

```
Erro médio: 248378030.81966096
Erro médio percentual: 100.0%
R2: -2.0044504114610543e+20
R2 ajustado: -2.0044504114610543e+20
```

Para nossa surpresa, essa abordagem foi tão ruim que resultou em um R² negativo e um erro percentual de 100%. Indubitavelmente este foi nosso pior modelo de todo o projeto. Uma possível explicação para essas métricas é o fato de todas as colunas de categoria, antes binárias (0 ou 1), terem se transformado em decimais conforme seus z-scores. Assim, criou-se novamente uma ordinalidade que provavelmente distorceu as previsões.

Analizando esses resultados, chegamos à conclusão de que a regressão linear não era o melhor modelo para nossas features. Mesmo que alguma manipulação melhorasse os resultados, esse aumento dificilmente chegaria a um valor absoluto de R² desejável. Assim, decidimos focar nossos esforços em outros modelos.

Rodada 2

Nossa segunda rodada teve como objetivo testar as mesmas features (**hora de início, dia da semana, mês e categoria cardinal**, tendo o “**Total Domicílios I Rat%**” da Rede Gazeta como saída) no algoritmos de **KNN, árvore decisória e random forest**.

Inicialmente, havíamos entendido que o método “score” do KNN e árvore de decisão (mais recomendado para avaliar esses algoritmos) retornasse a “acurácia” do modelo, isto é, o número de acertos sobre o número total de tentativas. Porém, através da comparação desse método com os resultados de R² para cada modelo, descobrimos que ambos os métodos sempre retornavam o mesmo valor.

Pesquisando mais a fundo, aprendemos que eles, de fato, calculam o mesmo R². Portanto, mantivemos, para esta rodada, as mesmas métricas de **R²** (representado pelo método “score” em alguns modelos), **R² ajustado, erro médio percentual e erro médio absoluto**. Ademais, novamente iniciamos os experimentos sem metas objetivas em mente, pois nos faltava ainda exploração para saber o que era possível esperar dos modelos.

As features para todos os modelos foram as mesmas utilizadas na Rodada 1, sem normalização ou padronização.

KNN

O algoritmo KNN (K-Nearest Neighbors) é um algoritmo de aprendizado supervisionado não paramétrico, que usa a proximidade de um ponto com seus vizinhos para fazer previsões. Desse modo, a escolha do modelo KNN se deve à sua natureza de comparar diretamente os valores independentes com os seus vizinhos mais próximos, o que, por vezes, pode performar melhor do que modelos de regressão linear.

Nesse contexto, resolvemos aplicar este algoritmo ao dataframe considerando diferentes números de vizinhos (n-neighbors) para obter resultados diferentes. Começamos 2 e 3 e vimos que as métricas melhoraram com o acréscimo de vizinhos. A partir daí, testamos ímpares crescentes até que o R^2 parasse de subir. Escolhemos apenas ímpares porque a literatura indicava que eles tendem a performar melhor com esse algoritmo. Assim, obtivemos o número de vizinhos ideal por tentativa de erro. Os resultados foram os seguintes:

```
evaluate_knn(2)

Erro médio: 2.2127987847745305
Erro médio percentual: 15.39%
R2: 0.8731274463543462
R2 ajustado: 0.8731274463543462

evaluate_knn(3)

Erro médio: 2.0993115396203677
Erro médio percentual: 14.6%
R2: 0.8843116748839566
R2 ajustado: 0.8843116748839566

evaluate_knn(5)

Erro médio: 2.041335241538616
Erro médio percentual: 14.13%
R2: 0.8899105209974141
R2 ajustado: 0.8899105209974141

evaluate_knn(7)

Erro médio: 2.040835521313075
Erro médio percentual: 14.07%
R2: 0.8891245924911518
R2 ajustado: 0.8891245924911518
```

```
evaluate_knn(9)
```

```
Erro médio: 2.0652401437291465  
Erro médio percentual: 14.22%  
R²: 0.8860147743061687  
R² ajustado: 0.8860147743061687
```

```
evaluate_knn(11)
```

```
Erro médio: 2.0782107608169675  
Erro médio percentual: 14.29%  
R²: 0.8844572548118802  
R² ajustado: 0.8844572548118802
```

Naturalmente, ficamos surpresos com o aumento dramático do R² em todos os casos e vimos que era possível alcançar métricas expressivas com nossos dados. Nesse sentido, analisando com mais profundidade, percebemos que o erro médio cai rapidamente do segundo ao quinto vizinho, atingindo seu ápice em k = 7. Daí em diante, no entanto, o R² começa a cair e o erro volta a subir.

Isso ocorreu, provavelmente, porque até 7 vizinhos, havia muito poucas comparações, de modo que o modelo não chegava a contemplar toda a diversidade dos dados. A partir de 7, no entanto, o excesso de comparações deve ter introduzido ruído nas previsões, tornando-as sensíveis demais à variância natural dos dados. Logo, tivemos como melhor modelo aquele em que k = 7, seu valor ótimo para nosso dataset.

Por fim, notamos que o R² ajustado continua acompanhando o R². Isso é esperado pois o número de features não mudou.

Dessa maneira, quisemos, também, avaliar a influência da normalização nesse processo. Ignoramos a padronização porque já sabíamos, da rodada 1, que essa técnica tende a distorcer as colunas cardinais. Assim, com apenas a normalização, tivemos:

```
evaluate_knn(2)
```

```
Erro médio: 0.04957258769042309  
Erro médio percentual: 15.57%  
R²: 0.867808873505403  
R² ajustado: 0.867808873505403
```

```
evaluate_knn(3)
```

```
Erro médio: 0.04772620903056398  
Erro médio percentual: 14.96%  
R²: 0.878753581954773  
R² ajustado: 0.878753581954773
```

```
evaluate_knn(5)
```

```
Erro médio: 0.04658526603441716  
Erro médio percentual: 14.57%  
R²: 0.8849486555678554  
R² ajustado: 0.8849486555678554
```

```
evaluate_knn(7)
```

```
Erro médio: 0.046632804557406766  
Erro médio percentual: 14.53%  
R²: 0.8861322061908479  
R² ajustado: 0.8861322061908479
```

```
evaluate_knn(9)
```

```
Erro médio: 0.04785548923046109  
Erro médio percentual: 14.92%  
R²: 0.8820795339502258  
R² ajustado: 0.8820795339502258
```

```
evaluate_knn(11)
```

```
Erro médio: 0.04963317413482603  
Erro médio percentual: 15.45%  
R²: 0.8733015932980688  
R² ajustado: 0.8733015932980688
```

Na versão padronizada, k = 7 continua sendo o valor ótimo para vizinhos, porém, mesmo nele, o R² ainda é um pouco menor, e o erro é um pouco maior quando comparado ao R² não normalizado. Conclui-se, portanto, que a versão não normalizada também prejudica a performance do modelo no KNN.

Decision trees (árvore decisórias)

As árvores decisórias são uma representação visual de todos os possíveis caminhos de ações que se pode seguir para tomar uma decisão, ou seja, trata-se de uma ferramenta que estabelece regras e condições para se chegar em uma predição.

Nesse contexto, testamos o algoritmo de árvore decisória tanto com a profundidade automática do método quanto com valores refinados manualmente. Seguimos a abordagem de tentativa e erro mais uma vez, aumentando a profundidade enquanto as métricas melhoraram.

Os parâmetros automáticos resultaram em:

```
Erro médio: 2.395798241964256
Erro médio percentual: 16.7%
R2: 0.8486702591169911
R2 ajustado: 0.8486702591169911
```

Assim como no KNN, o modelo de árvore de decisão traz um R² elevado e um erro médio aceitável. Percebe-se, consequentemente, que esse algoritmo também apresenta um desempenho muito superior ao da regressão linear. Contudo, como as métricas ainda são inferiores às obtidas na versão ótima de KNN, decidimos modificar a profundidade máxima da árvore, um dos parâmetros mais importantes do algoritmo, manualmente, por tentativa e erro.

Fazendo isso:

```
evaluate_dt(2)
```

```
Erro médio: 3.766926702147079
Erro médio percentual: 26.53%
R2: 0.6641555580741112
R2 ajustado: 0.6641555580741112
```

```
evaluate_dt(3)
```

```
Erro médio: 3.3297598933451176
Erro médio percentual: 23.4%
R2: 0.7210137045622873
R2 ajustado: 0.7210137045622873
```

```
evaluate_dt(10)
```

```
Erro médio: 2.1244049572824313
Erro médio percentual: 14.84%
R2: 0.8757071325208261
R2 ajustado: 0.8757071325208261
```

Os valores iniciais (com 2 e 3 de profundidade) são muito inferiores aos que vínhamos conseguindo nos últimos modelos. Porém, a melhora relativa entre eles nos incentivou a testar valores de profundidade mais altos, esperando, com isso, diminuir mais rapidamente o erro médio e aumentar o R². Assim, com 10 de profundidade, já atingimos resultados comparáveis aos anteriores. Tentando mais um pouco:

```
evaluate_dt(12)
```

```
Erro médio: 2.0835230640672546  
Erro médio percentual: 14.54%  
R2: 0.8807213098408481  
R2 ajustado: 0.8807213098408481
```

```
evaluate_dt(13)
```

```
Erro médio: 2.1334118419168546  
Erro médio percentual: 14.89%  
R2: 0.8720882174647683  
R2 ajustado: 0.8720882174647683
```

Atingimos o ápice na profundidade 12, com 88% de R² e 14,54% de erro médio. Isso reflete uma performance levemente inferior aos ápices do KNN, mas ainda assim muito positiva.

Random forest generator

O algoritmo de random forest, em português, floresta aleatória, cria muitas árvores de decisão, de maneira aleatória, formando o que podemos enxergar como uma floresta. Assim, cada árvore é utilizada na escolha do resultado final, em uma espécie de votação. Isso garante maior equilíbrio nas previsões, pois a maior amostra minimiza possíveis vieses.

Nesse contexto, testamos o algoritmo de floresta aleatória com diversos números de estimadores (isto é, número de árvores). Como feito anteriormente, aumentamos gradualmente os argumentos para, através de tentativa e erro, encontrar os valores ótimos. O resultado foi o seguinte:

```
evaluate_forest(20)
```

```
Erro médio: 2.105253526278549  
Erro médio percentual: 14.67%  
R2: 0.8846517417059605  
R2 ajustado: 0.8846517417059605
```

```
evaluate_forest(100)
```

```
Erro médio: 2.10019891767052  
Erro médio percentual: 14.65%  
R2: 0.8846832941424568  
R2 ajustado: 0.8846832941424568
```

```
evaluate_forest(200)
```

```
Erro médio: 2.100815732129745  
Erro médio percentual: 14.65%  
R2: 0.8841834707002605  
R2 ajustado: 0.8841834707002605
```

```
evaluate_forest(800)
```

```
Erro médio: 2.0961493910422955  
Erro médio percentual: 14.62%  
R2: 0.8848536516427098  
R2 ajustado: 0.8848536516427098
```

```
evaluate_forest(1600)
```

```
Erro médio: 2.0954377012320022  
Erro médio percentual: 14.61%  
R2: 0.8849557287021126  
R2 ajustado: 0.8849557287021126
```

```
evaluate_forest(5000)
```

```
Erro médio: 2.0952422409624205  
Erro médio percentual: 14.61%  
R2: 0.8850830990502894  
R2 ajustado: 0.8850830990502894
```

Nesses modelos, tivemos, pela primeira vez, um caso evidente de rendimentos decrescentes. De 20 para 5000 estimadores, obtivemos uma melhora de apenas um milésimo no R² e 7 centésimos no erro percentual. No entanto, o tempo para computar o último modelo foi vinte vezes maior do que o tempo utilizado no primeiro. Portanto, percebemos que não valeria a pena testar estimadores maiores, ainda que as métricas estivessem crescendo, pois esse crescimento seria muito baixo quando comparado aos recursos computacionais necessários para alcançá-lo.

Assim, chegamos a um modelo cujo R² é superior às árvores decisórias e KNNs otimizados, mas inferior em erro médio nesses mesmos exemplos. Todavia, a discrepância é tão baixa que pode ser desconsiderada. Para fins práticos, o desempenho é o mesmo.

Rodada 3

Na rodada 2, já havíamos atingido resultados aceitáveis, chegando sempre a valores superiores a 80% no R². Entretanto, sob a recomendação dos TVCoders (Grupo 4), descobrimos que poderíamos melhorar ainda mais nossas métricas através da adição de mais features (rodada 3) e aplicação de algoritmos de Gradient Boosting (rodada 4).

Assim, voltamos aos nossos arquivos intermediários, da fase de feature engineering, e modificamos os códigos para que fossem preservadas as colunas de “Hora Início (15 em 15 m)”, “Dia da Semana”, “Dia do Mês”, “Mês”, “Categoria”, “Feriado” e “BBB”.

A penúltima foi implementada utilizando a biblioteca “holiday” do Python, que já trazia uma lista datada de feriados brasileiros. Com isso, pudemos comparar os itens com as datas em nosso dataset e criar uma coluna binária indicando se certo programa fora televisionado em um feriado ou não.

Já a coluna “BBB” necessitou de algumas manipulações locais com as planilhas não anonimizadas. Isso se deu porque as versões pós-anonimização haviam substituído os títulos dos programas, nos quais apareciam a flag “BBB”, pela numeração padrão “PROGRAMA X”. Por isso, modificamos localmente nossos arquivos originais e produzimos um CSV simplificado que trazia apenas a coluna binária de BBB. Assim, foi possível concatenar esse arquivo com os outros já presentes no Drive da forma mais transparente possível e ainda preservar o sigilo contratual deste projeto.

Com essas features agregadas, tentamos, novamente, aplicar os algoritmos de **regressão linear, KNN, decision tree e random forest**. Ignoramos a regressão linear porque ela já tinha se provado pouco eficiente nas rodadas passadas. As métricas, contudo, permaneceram as mesmas: **R², R² ajustado, erro médio percentual e erro médio absoluto**.

Regressão linear

```
Erro médio: 3.833022223914597
Erro médio percentual: 38.63%
R2: 0.4155973645673756
R2 ajustado: 0.4155973645673756
```

Mais uma vez, ficamos desapontados com a regressão linear. Mesmo sem normalização, o resultado foi inferior ao da Rodada 1. Comparado aos modelos da Rodada 2, é uma péssima performance. A partir daqui, decidimos nos focar em outros algoritmos para os próximos testes.

KNN

```
evaluate_knn(2)
```

```
Erro médio: 2.2830145759717313  
Erro médio percentual: 22.91%  
R2: 0.7577304698122289  
R2 ajustado: 0.7577304698122289
```

```
evaluate_knn(3)
```

```
Erro médio: 2.161453484830023  
Erro médio percentual: 21.69%  
R2: 0.7851972597619992  
R2 ajustado: 0.7851972597619992
```

```
evaluate_knn(7)
```

```
Erro médio: 2.0839173440790963  
Erro médio percentual: 20.94%  
R2: 0.8035037793380517  
R2 ajustado: 0.8035037793380517
```

```
evaluate_knn(9)
```

```
Erro médio: 2.0999127777100846  
Erro médio percentual: 21.11%  
R2: 0.8002665151256181  
R2 ajustado: 0.8002665151256181
```

Curiosamente, o aumento de features diminui o desempenho de modelos em KNN. Embora o pico ainda seja $k = 7$, o R^2 nesse ponto é de apenas 80% com 21% de erro, contra os 88% e 14%, respectivamente, da rodada 2. Nesse panorama, ficou claro que KNN também não era o melhor algoritmo para essas features.

Decision trees

Assim como na rodada 2, para o algoritmo de árvore decisória, nós experimentamos tanto parâmetros (profundidade máxima) automáticos quanto manuais. Desse modo, para a tentativa automática, tivemos:

```
Erro médio: 0.8957613927135373  
Erro médio percentual: 9.04%  
R²: 0.9407370668809228  
R² ajustado: 0.9407370668809228
```

Surpreendente, conseguimos, dessa maneira, nosso melhor resultado até então. Com um erro de 0,9, já estávamos muito satisfeitos. Ademais, no começo da sprint, acreditávamos não ser possível passar de 0,5 no R²; agora, já chegamos a 0,94. Porém, ainda pairava a dúvida: é possível fazer mais? Por isso, começamos a testar mais argumentos.

```
evaluate_dt(2)
```

```
Erro médio: 3.4516499111460406  
Erro médio percentual: 34.8%  
R²: 0.47485939087072637  
R² ajustado: 0.47485939087072637
```

```
evaluate_dt(3)
```

```
Erro médio: 2.8741950000216105  
Erro médio percentual: 28.98%  
R²: 0.6185183584281578  
R² ajustado: 0.6185183584281578
```

```
evaluate_dt(10)
```

```
Erro médio: 1.867807201709382  
Erro médio percentual: 18.8%  
R²: 0.8408070221941255  
R² ajustado: 0.8408070221941255
```

```
evaluate_dt(20)
```

```
Erro médio: 1.1178841484224193  
Erro médio percentual: 11.27%  
R²: 0.9263491264588971  
R² ajustado: 0.9263491264588971
```

```
evaluate_dt(25)
```

```
Erro médio: 0.9300363700525779  
Erro médio percentual: 9.39%  
R²: 0.9388862470616853  
R² ajustado: 0.9388862470616853
```

```
evaluate_dt(30)
```

```
Erro médio: 0.8962892672799581  
Erro médio percentual: 9.05%  
R2: 0.9406934030255313  
R2 ajustado: 0.9406934030255313
```

```
evaluate_dt(35)
```

```
Erro médio: 0.8961219501935049  
Erro médio percentual: 9.05%  
R2: 0.9404783719792429  
R2 ajustado: 0.9404783719792429
```

```
evaluate_dt(37)
```

```
Erro médio: 0.8950737175581821  
Erro médio percentual: 9.04%  
R2: 0.9406658534469396  
R2 ajustado: 0.9406658534469396
```

```
evaluate_dt(40)
```

```
Erro médio: 0.8976920616546851  
Erro médio percentual: 9.06%  
R2: 0.940432851741446  
R2 ajustado: 0.940432851741446
```

Com base nos resultados, vemos que foi necessário muito mais esforço computacional para melhorar as métricas desta vez, conforme nos aproximávamos do valor alcançado pelo método automatizado. De 2 a 20 de profundidade, a queda foi brusca, porém, a partir daí, tivemos de dobrar esse número para aumentar o R² 2 pontos percentuais. De fato, conseguimos taxas finais menores do que as automáticas, porém o custo-benefício, considerando tempo, RAM e GPU, não parece ter valido a pena. Em grande escala, seria mais útil dedicar esse overhead de recursos em novos modelos, e não no refinamento exacerbado de um modelo que já estava bom.

Random forest generator

Assim como na rodada 2, resolvemos testar vários argumentos para estimadores a fim de encontrar um valor ótimo. A suposição era de que esse algoritmo tivesse melhores resultados do que a decision tree, dado que ele tira a média de diversas árvores decisórias.

```
evaluate_forest(20)
```

```
Erro médio: 0.8974944331058854  
Erro médio percentual: 9.05%  
R2: 0.9569516976192453  
R2 ajustado: 0.9569516976192453
```

```
evaluate_forest(100)
```

```
Erro médio: 0.8747982865236993  
Erro médio percentual: 8.82%  
R2: 0.9595642765366462  
R2 ajustado: 0.9595642765366462
```

```
evaluate_forest(200)
```

```
Erro médio: 0.8721312286767394  
Erro médio percentual: 8.79%  
R2: 0.9598968867968866  
R2 ajustado: 0.9598968867968866
```

Como esperado, mesmo o modelo mais simples, com apenas 20 árvores (estimadores), trouxe um R² quase tão bom quanto a melhor decision tree. Aumentando os parâmetros, alcançamos novos recordes de erro médio percentual, chegando a 8,79%. O contraponto, todavia, é que cada tentativa levou muito mais tempo do que na Rodada 2 devido ao maior número de features. Mais que isso, tivemos que aumentar os estimadores em 10 vezes para conseguir 0,35 pontos percentuais a menos no erro. Assim, apesar de acreditar que poderíamos atingir resultados ainda melhores com esse algoritmo, resolvemos parar em 200 estimadores para maximizar recursos. Afinal, argumentos maiores tomariam muito mais tempo e renderiam melhorias mínimas, de modo que não valeria a pena.

Rodada 4

Ainda seguindo as orientações dos TVCoders, decidimos aplicar as features da Rodada 3 em três algoritmos de **Gradiente Boosting: Light GBM, CatBoost e XGBoost**. As métricas utilizadas foram as mesmas das rodadas anteriores, pois, assim como no caso do KNN e decision trees, os métodos de “score” de gradient boosters, quando aplicados a regressões, retornam o R² do modelo.

O algoritmo Gradient Boosting é uma técnica de aprendizado de máquina que produz um modelo de previsão na forma de um conjunto de modelos de previsão fracos, geralmente árvores de decisão. Desse modo, ele constrói o modelo em etapas, como outros métodos de

reforço, e os generaliza, permitindo a otimização de uma função de perda diferenciável arbitrária. Assim, as três implementações de algoritmos de boost que utilizamos são:

1. Light GBM
2. CatBoost
3. XGB

LightGBM

LightGBM é uma framework de gradient boosting otimizada para ter um tempo de treinamento mais curto, maior eficiência, menor uso de memória, aprendizado paralelizável e maior acurácia. Em nossos testes, utilizamos os hiperparâmetros sugeridos pelo grupo 4.

```
hyper_params = {
    'task': 'train', #função
    'boosting_type': 'gbdt', #tipo da regressão
    'objective': 'regression', #tipo do modelo
    'metric': ['l1','l2'], #erro médio quadrático e erro médio absoluto
    'learning_rate': 0.08, #velocidade do aprendizado
    'feature_fraction': 1, #porcentagem do dataframe a ser utilizado
    'verbose': 0, #usado para retirar possíveis bugs existentes (debug)
    "max_depth": 8, #limita o tamanho de cada árvore
    "num_leaves": 128, #limita o número de folhas que cada árvore pode ter
    "num_iterations": 20_000 #quantidade de tentativas
}
```

Com isso, tivemos os seguintes resultados:

```
Erro médio: 0.7244104210475191
Erro médio percentual: 7.31%
R2: 0.9759282035342731
R2 ajustado: 0.9759282035342731
```

Indubitavelmente, este é o melhor modelo que desenvolvemos dentre todas as rodadas. Com cerca de 98% de R² e apenas 7% de erro, o LightGBM superou e muito nossas expectativas, reformulando o que pensávamos ser possível neste projeto. Entretanto, foi também o algoritmo mais demorado, levando em torno de 15 minutos para rodar completamente. Nesse sentido, assim que a execução terminou, fizemos questão de salvar o modelo final em .txt para que pudéssemos acessá-lo futuramente sem precisar aplicar todo o algoritmo mais uma vez.

Contudo, apesar de já estarmos bastante satisfeitos com o desempenho do LightGBM, quisemos, a fim de exploração, também experimentar outros algoritmos de gradient boosting, diga-se o CatBoost e o XGBoost.

CatBoost

Catboost é outra framework de gradient boosting notória pelo suporte a variáveis categóricas através de permutação. Ainda assim, já que todo o nosso processamento e modelos anteriores haviam sido feitos, com sucesso, considerando apenas variáveis numéricas, resolvemos seguir esse padrão também neste algoritmo. Dessa maneira, apenas rodamos o método de regressão da biblioteca “catboost” nos splits de teste e treino que já tínhamos e obtivemos o seguinte resultado:

Erro médio: 1.580566972563443
Erro médio percentual: 15.92%
 R^2 : 0.8898189911874532
 R^2 ajustado: 0.8898189911874532

Apesar de certamente não tão bom quanto as métricas obtidas previamente, o CatBoost ainda nos rendeu uma boa performance. Ficou, assim, marcado em nossas notas internas como um algoritmo com grande potencial para futuros testes com outras features e/ou saídas.

XGBoost

XGBoost atende os mesmos objetivos do LightGBM, diferenciando-se no modo de crescimento da árvore. Enquanto o LightGBM cresce pelas folhas, o XGBoost cresce pela profundidade. Nesse caso, as variáveis precisam ser obrigatoriamente numéricas. Assim, novamente inserimos o dataset conforme o tínhamos. Obtivemos, com isso:

Erro médio: 2.0194466511913003
Erro médio percentual: 20.34%
 R^2 : 0.8171118501051915
 R^2 ajustado: 0.8171118501051915

É um resultado aceitável, quando analisado individualmente. Em muitos projetos, qualquer R^2 acima de 70% ou 75% já é satisfatório. No entanto, esse modelo ainda é muito inferior a outros que conseguimos até com algoritmos mais simples (como KNN ou árvore decisória). Logo, apesar de ter servido para ampliar nossos horizontes, o XGBoost não serviu aos nossos propósitos.

Com isso, terminamos as etapas de experimentações da Sprint 3, tendo testado a maioria dos algoritmos de regressão presentes na biblioteca “Scikit Learn” com diferentes tipos de features. Em suma, tivemos como melhor resultado o Light GBM, com cerca de 98% de R^2 e 7% de erro, muito melhor do que tínhamos imaginado.

Rodada 5

A quinta rodada foi uma experiência com um tipo de feature que nos interessava já há algum tempo: a divisão de horário em blocos, isto é, o agrupamento das 6h às 9h em “MANHÃ 1”, das 11h às 13h em “ALMOÇO”, etc. Acreditamos que isso nos daria mais pontos em cada grupo, possivelmente trazendo resultados mais acurados devido ao aumento no volume de amostras. Por isso, fizemos a substituição dos horários de início segundo a tabela abaixo:

Horários	Blocos
6:00:00 - 8:55:00	Manhã 1
9:00:00 - 11:55:00	Manhã 2
12:00:00 - 13:25:00	Almoço
13:30:00 - 15:55:00	Tarde 1
16:00:00 - 17:55:00	Tarde 2
18:00:00 - 22:25:00	Horário Nobre
22:30:00 - 24:25:00	Noite
24:30:00 - 29:55:00	Madrugada

Depois, aplicamos **regressão linear**, **KNN**, **decision tree** e **random forest** sobre o dataset. As métricas permaneceram as mesmas.

Regressão linear

```
Erro médio: 3.879666885112597
Erro médio percentual: 39.06%
R²: 0.4075769499489936
R² ajustado: 0.4075769499489936
```

A rodada 5 teve um resultado ainda pior que a regressão linear inicial, com as features da rodada 1. Mesmo assim, por termos pouca confiança nos desempenhos em regressão linear, resolvemos testar essas features nos outros algoritmos.

KNN

```
evaluate_knn(2)

Erro médio: 3.6531999360302185
Erro médio percentual: 36.83%
R²: 0.3546405818810551
R² ajustado: 0.3546405818810551
```

```
evaluate_knn(3)
```

```
Erro médio: 3.490243237480201  
Erro médio percentual: 35.07%  
R2: 0.42432885063303727  
R2 ajustado: 0.42432885063303727
```

```
evaluate_knn(7)
```

```
Erro médio: 3.2889777889954566  
Erro médio percentual: 33.19%  
R2: 0.5101615387575982  
R2 ajustado: 0.5101615387575982
```

No caso do KNN, além de demorar muito mais do que as outras rodadas, chegando a minutos com o k ainda baixo (k=2 ou k=3), o R² foi péssimo. Paramos os experimentos no k=7, pois percebemos que as métricas não melhoraram o suficiente para justificar os recursos computacionais.

Decision tree

```
Erro médio: 3.1041876566921367  
Erro médio percentual: 31.26%  
R2: 0.5644199770181695  
R2 ajustado: 0.5644199770181695
```

A imagem acima se refere aos parâmetros automáticos do algoritmo de árvore decisória. Embora ele supere, como esperado, o KNN anterior e até as regressões lineares, não se compara aos outros resultados obtidos nas rodadas anteriores. Testando profundidades manualmente, temos:

```
evaluate_dt(2)
```

```
Erro médio: 4.1665017153066355  
Erro médio percentual: 41.99%  
R2: 0.32804906563363345  
R2 ajustado: 0.32804906563363345
```

```
evaluate_dt(3)
```

```
Erro médio: 4.075862282002839  
Erro médio percentual: 41.08%  
R2: 0.3577074753702594  
R2 ajustado: 0.3577074753702594
```

```
evaluate_dt(10)
```

```
Erro médio: 3.5570220860496615  
Erro médio percentual: 35.81%  
R2: 0.48336846268927947  
R2 ajustado: 0.48336846268927947
```

```
evaluate_dt(20)
```

```
Erro médio: 3.177393936808667  
Erro médio percentual: 31.98%  
R2: 0.5556955052492014  
R2 ajustado: 0.5556955052492014
```

```
evaluate_dt(30)
```

```
Erro médio: 3.105919444722716  
Erro médio percentual: 31.28%  
R2: 0.5643286113691521  
R2 ajustado: 0.5643286113691521
```

```
evaluate_dt(100)
```

```
Erro médio: 3.1041668632786603  
Erro médio percentual: 31.26%  
R2: 0.5644163057808337  
R2 ajustado: 0.5644163057808337
```

Como podemos ver, existe uma melhora no R² e erro médio conforme a profundidade máxima aumenta; porém, esse acréscimo é muito pequeno para justificar o gasto computacional e temporal. Ao nos aproximarmos de 100 como argumento, esse ganho passa a ser de apenas milésimos no R², mesmo quando a profundidade torna-se três vezes maior. Assim, a árvore decisória também não oferece resultados adequados.

Random forest generator

```
evaluate_forest(20)
```

```
Erro médio: 3.107251345700086
Erro médio percentual: 31.28%
R²: 0.564047996324722
R² ajustado: 0.564047996324722
```

```
evaluate_forest(100)
```

```
Erro médio: 3.104838331916731
Erro médio percentual: 31.26%
R²: 0.564462505416807
R² ajustado: 0.564462505416807
```

```
evaluate_forest(400)
```

```
Erro médio: 3.1047739153911933
Erro médio percentual: 31.27%
R²: 0.5644984185387432
R² ajustado: 0.5644984185387432
```

Baseando-nos nos resultados anteriores, não esperávamos que o algoritmo de floresta retornasse resultados comparáveis aos sucessos das outras rodadas com as features atuais. Ainda assim, a baixa melhora relativa em relação à árvore decisória nos surpreendeu, pois esse algoritmo deveria justamente otimizar o potencial de decision trees. Nossa conclusão, frente a isso, é que de fato existe um limite de performance para modelos baseados em árvores de decisão para as features que escolhemos, de modo que não podemos esperar por muito mais do que 55% de R² com esses algoritmos.

LightGBM

```
Erro médio: 3.105252071494387
Erro médio percentual: 31.17%
R²: 0.5645468235985043
R² ajustado: 0.5645468235985043
```

Com esse modelo, percebemos que o conjunto de features desta rodada realmente não rende bons resultados. Nem o LightGBM, que havia sido nosso melhor algoritmo até agora, alcançou métricas satisfatórias. Concluímos, portanto, que deveríamos manter a divisão de horário em hora e minuto em vez de agregá-los em blocos.

Rodada 5.1: Aplicando LightGBM em outras saídas

Apesar dos péssimos resultados na Rodada 5, o sucesso da Rodada 4 nos motivou a investir o restante de tempo da sprint na criação de modelos para outros tipos de audiência (“Total Domicílios | Shr%”, “Total Indivíduos | Fid%” e “Total Indivíduos | Rch%”). Dada a performance superior do LightGBM frente a todos os outros algoritmos, resolvemos aplicar somente ele nesta etapa.

Antes disso, entretanto, tivemos de retornar aos arquivos da seção de pré-processamento. Naquele período, havíamos removido as colunas relacionadas à Share, Fidelidade e Reach, por não serem o objetivo principal do projeto. Agora, tínhamos de resgatá-las e manipulá-las para que tivessem as features formatadas conforme fora utilizado nas rodadas desta seção.

Ademais, devido a alguns erros que tivemos ao tentar utilizar as colunas dessas taxas como saídas, percebemos que os dados nesses campos não estavam padronizados. Em mais detalhes, os números tipo float eram representados ora com vírgula, ora com ponto. Logo, foi preciso adicionar um passo extra de preparação para padronizar esses valores.

Feito isso, foi possível aplicar o LightGBM livremente.

Share

Erro médio: 2.3811322952875527

Erro médio percentual: 7.65%

R²: 0.9029153060018735

R² ajustado: 0.9029153060018735

Para o Share, conseguimos 90% de R² e também 7% de erro médio percentual. Isso ocorre, mesmo o erro médio sendo maior do que o encontrado no modelo de Rat, porque o Share alcança valores maiores dos que o limite superior do Rat. Por nem sequer ser um requisito do projeto, consideramos esse resultado mais que satisfatório e também salvamos esse modelo.

Reach

Erro médio: 0.41743243733821916

Erro médio percentual: 8.55%

R²: 0.9690144458599838

R² ajustado: 0.9690144458599838

O modelo de Reach também performou surpreendentemente bem, alcançando 96% em R² de primeira. O erro percentual acompanha os valores definidos pelos últimos modelos, configurando sucesso. Por conseguinte, este modelo de Reach com LightGBM também foi salvo.

Fidelidade

Erro médio: 3.796477367101829
Erro médio percentual: 4.02%
 R^2 : 0.1133581576364362
 R^2 ajustado: 0.1133581576364362

A saída de Fidelidade foi a única em que o LightGBM teve um desempenho ruim. O motivo até o momento não é claro para nós e será explorado com mais afinco na Sprint 4; afinal, modelar o Fid% não era o objetivo da Sprint 3. Ainda assim, é curioso como o R pode ser tão baixo com um erro tão insignificante. Suspeitamos que possa haver algum bug no código ou problema de formatação nos dados que esteja introduzindo essa distorção. Nesse sentido, não salvamos esse modelo pelo fracasso em atingir métricas aceitáveis.

Rodada 6

A rodada 6 foi desenvolvida na Sprint 4. Revisitamos a etapa de modelagem nesse ponto porque percebemos que a coluna de BBB, utilizada a partir da rodada 3, tinha chances de estar enviesando nesses dados. Em resumo, esse campo associava um valor binário positivo às entradas cuja programação era uma edição de BBB, formando uma coluna de 0s e 1s que enfatizavam a presença do reality show naquele slot.

Porém, ao analisar essa abordagem com mais atenção, percebemos que estávamos considerando essa característica de modo duplicado, pois o one-hot encoding das categorias já apresentava uma coluna binária para a programação de BBB. Além disso, essa implementação também indicava para o modelo que o televisionamento desse evento em um horário do dia afetava o dia inteiro, o que não fazia sentido, conforme notado pelo parceiro. Assim, entendemos que essa feature poderia prejudicar os dados e, portanto, testamos os modelos da Rodada 3 sem ela.

Por fim, ignoramos os modelos de gradient boosting devido à baixa explicabilidade desse tipo de algoritmo. Como

Regressão linear

Erro médio: 3.830982933226498
Erro médio percentual: 38.65%
 R^2 : 0.4158912371517206
 R^2 ajustado: 0.4158912371517206

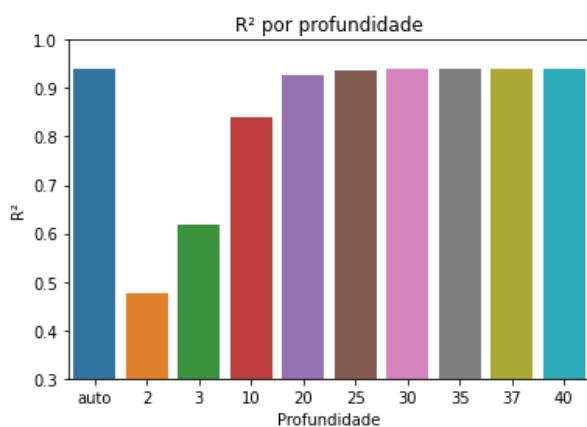
A conclusão, com esse teste, foi clara: regressão linear é um péssimo modelo para nosso projeto, independentemente das features. Com apenas 41% de R^2 , determinamos que não valia a pena utilizá-lo em novas rodadas.

KNN



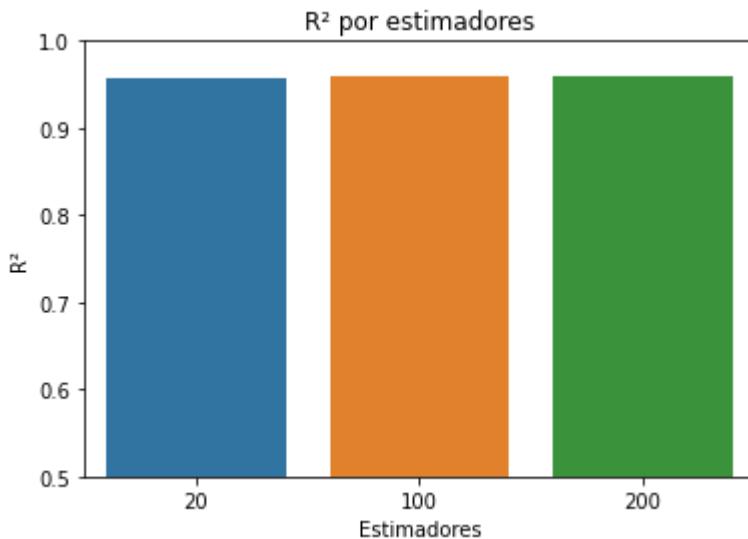
Nos testes de KNN, resolvemos nos focar apenas nos R² porque os erros já estavam próximos demais para serem significativos. Assim, geramos o gráfico acima para melhor visualizar as discrepâncias no coeficiente de determinação. Percebemos, com isso, que o ápice de desempenho permanecia em k = 7, mas que não havia uma diferença tão grande com valores vizinhos, como k = 9. Nesse contexto, a meta passou a ser priorizar o menor k possível que ainda mantivesse esse nível de acurácia, pois k menores exigem menos tempo de predição.

Árvore decisória



Semelhantemente ao modelo anterior, isolamos o R² para os testes de árvore decisória em um gráfico de barras. Com isso, percebemos que, a partir de 30 níveis de profundidade, não há grande diferença no desempenho positivo do modelo. Portanto, decidimos também almejar pela menor profundidade possível, nesse algoritmo, que ainda preserve essas taxas de R² a fim de diminuir o tempo de treinamento.

Random forest



No random forest, a discrepância entre R² é muito menor mesmo quando o número de estimadores cresce em 10 vezes. Assim, vemos que não é necessário utilizar valores muito grandes para floresta aleatória para se conseguir um bom resultado. Novamente, priorizamos parâmetros menores para otimizar o tempo.

Rodada 7

Na Sprint 4, descobrimos ser possível otimizar parâmetros através de Grid e Random Search. Essas ferramentas testam diferentes combinações de modo exaustivo ou aleatório, respectivamente, dentro de intervalos passados para cada hiperparâmetro do algoritmo de indução. Inicialmente, essa abordagem pareceu redundante, pois já havíamos feito esse processo de forma manual nas sprints passadas, testando diferentes permutações de hiperparâmetros em busca do melhor custo-benefício de tempo e R². Ainda assim, observando melhor, percebemos que essas técnicas poderiam ser muito úteis em validar nosso raciocínio.

Desse modo, decidimos executar o Random Search para nossa melhor rodada até então (rodada 6). Escolhemos apenas o Random para economizar tempo, dado que já tínhamos boas hipóteses sobre quais seriam os melhores intervalos e queríamos apenas confirmar nossas ideias em vez de selecionar hiperparâmetros do zero.

No que tange aos modelos testados, restringimo-nos a apenas KNN, árvore decisória e random forest, pois regressão linear havia performado de forma consistentemente ruim. Ademais, seguindo o raciocínio da rodada 6, consideramos algoritmos de gradient boosting muito complexos para atender nosso requisito de explicabilidade para o cliente final. Como demonstrado na seção de “Avaliação”, o gráfico de LightGBM, por exemplo, é tão extenso que o computador mal consegue renderizar uma imagem inteligível. Nesse sentido, decidimos

priorizar os modelos de “Árvore decisória” e “Random Forest”, pois seus gráficos, via de regra, de escolha são mais claros e concisos.

KNN

Para KNN, escolhemos o intervalo de 5 a 30. Até então, nosso melhor modelo havíamos testado k de 1 a 9, tendo como ápice k = 7. Queríamos, portanto, garantir que não existiam picos ainda melhores com valores maiores; ainda assim, devido a limites computacionais e de tempo, limitamos o máximo a 30.

```
RandomizedSearchCV(cv=3, estimator=KNeighborsRegressor(),
                    param_distributions={'n_neighbors': [5, 6, 7, 8, 9, 10, 11,
                                                       12, 13, 14, 15, 16, 17,
                                                       18, 19, 20, 21, 22, 23,
                                                       24, 25, 26, 27, 28,
                                                       29]},  
                    verbose=2)
```

Nessas restrições, validamos que k = 7 realmente é o melhor hiperparâmetro para este modelo:

```
✓ [104] print(randomized_search.best_params_)  
{'n_neighbors': 7}
```

Com isso,

Erro médio: 2.0870419197897263
Erro médio percentual: 21.06%
R²: 0.80326066204431
R² ajustado: 0.80326066204431

Decision Tree

Na rodada 6, nosso melhor modelo de decision tree havia sido o de 40 níveis de profundidade. Portanto, determinamos o intervalo de 25 a 50 para o random search da rodada 7, para contemplar os valores próximos ao ápice e averiguar se era possível conseguir resultados semelhantes com um menor número de níveis.

```
RandomizedSearchCV(cv=3, estimator=DecisionTreeRegressor(),
                    param_distributions={'max_depth': [25, 26, 27, 28, 29, 30,
                                                       31, 32, 33, 34, 35, 36,
                                                       37, 38, 39, 40, 41, 42,
                                                       43, 44, 45, 46, 47, 48,
                                                       49]},  
                    verbose=2)
```

O resultado foi uma profundidade máxima de 28, o que diminui bastante o tempo de execução do treino do modelo:

```
print(randomized_search.best_params_)

{'max_depth': 28}
```

Assim, tivemos um resultado aceitável com maior custo-benefício:

```
Erro médio: 0.9053855191545851
Erro médio percentual: 9.13%
R²: 0.9399598874368328
R² ajustado: 0.9399598874368328
```

Random forest

Para random forest, consideramos um intervalo de 20 a 200 estimadores, pois nosso melhor resultado, até então, havia ocorrido com 200, mas demandado tempo demais com treinamento. Queríamos, portanto, alcançar um resultado semelhante com menos árvores.

Rodamos o random search localmente para otimizar o tempo, pois já estávamos chegando a duas horas de execução no Colab.

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
[CV 1/3] END .....n_estimators=166;; score=0.937 total time= 1.7min
[CV 2/3] END .....n_estimators=166;; score=0.939 total time= 1.3min
[CV 3/3] END .....n_estimators=166;; score=0.937 total time= 1.2min
[CV 1/3] END .....n_estimators=102;; score=0.937 total time= 38.5s
[CV 2/3] END .....n_estimators=102;; score=0.938 total time= 26.2s
[CV 3/3] END .....n_estimators=102;; score=0.936 total time= 26.7s
[CV 1/3] END .....n_estimators=36;; score=0.935 total time= 9.1s
[CV 2/3] END .....n_estimators=36;; score=0.936 total time= 9.8s
[CV 3/3] END .....n_estimators=36;; score=0.935 total time= 18.9s
[CV 1/3] END .....n_estimators=82;; score=0.937 total time= 47.6s
[CV 2/3] END .....n_estimators=82;; score=0.938 total time= 30.5s
[CV 3/3] END .....n_estimators=82;; score=0.936 total time= 30.8s
[CV 1/3] END .....n_estimators=74;; score=0.936 total time= 34.3s
[CV 2/3] END .....n_estimators=74;; score=0.938 total time= 46.3s
[CV 3/3] END .....n_estimators=74;; score=0.936 total time= 30.6s
[CV 1/3] END .....n_estimators=77;; score=0.937 total time= 19.3s
[CV 2/3] END .....n_estimators=77;; score=0.938 total time= 23.4s
[CV 3/3] END .....n_estimators=77;; score=0.936 total time= 51.8s
[CV 1/3] END .....n_estimators=100;; score=0.937 total time= 44.4s
[CV 2/3] END .....n_estimators=100;; score=0.939 total time= 1.1min
[CV 3/3] END .....n_estimators=100;; score=0.936 total time= 1.1min
[CV 1/3] END .....n_estimators=76;; score=0.937 total time= 31.1s
[CV 2/3] END .....n_estimators=76;; score=0.938 total time= 52.3s
[CV 3/3] END .....n_estimators=76;; score=0.936 total time= 51.7s
[CV 1/3] END .....n_estimators=64;; score=0.936 total time= 37.2s
[CV 2/3] END .....n_estimators=64;; score=0.938 total time= 18.9s
[CV 3/3] END .....n_estimators=64;; score=0.935 total time= 45.8s
[CV 1/3] END .....n_estimators=139;; score=0.937 total time= 44.6s
[CV 2/3] END .....n_estimators=139;; score=0.938 total time= 35.2s
[CV 3/3] END .....n_estimators=139;; score=0.937 total time= 40.9s
{'n_estimators': 166}
```

Com isso, chegamos a um estimador ótimo de 166, resultando em:

Erro médio: 0.8722437501926801

Erro médio percentual: 8.8%

R²: 0.9598184105363474

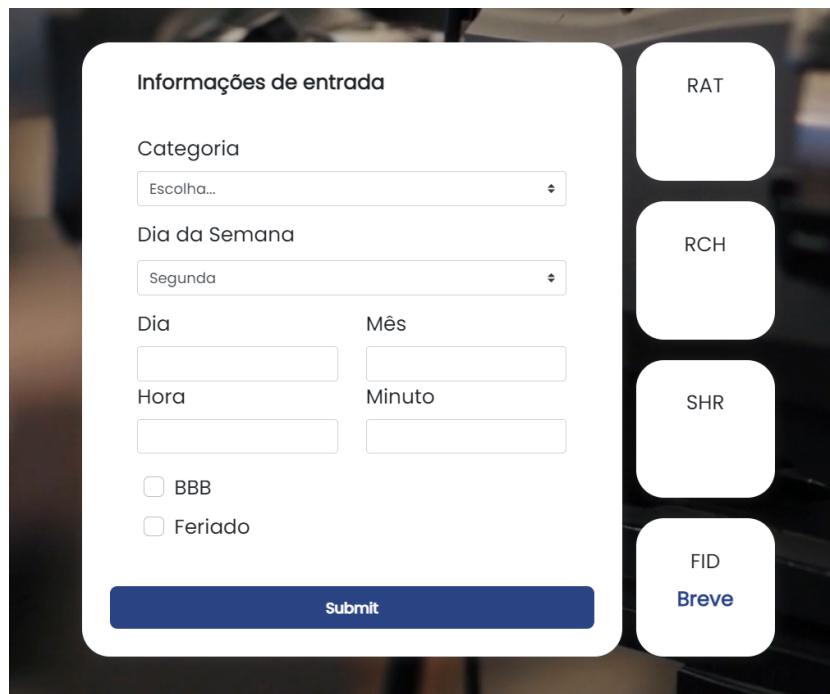
R² ajustado: 0.9598184105363474

Esse resultado não é muito diferente do atingido com 200 estimadores, porém rodou em bem menos tempo. Em suma, conseguimos parâmetros otimizados para todos os nossos modelos, fazendo desta rodada um sucesso.

Interface gráfica

A despeito do fracasso do modelo de Fid%, o fato de termos três modelos muito bons, salvos e acessíveis (Rat%, Shr% e Rch%), além de mais algum tempo até o fim da Sprint, nos motivou a experimentar com interfaces gráficas. A ideia inicial era utilizar os recursos de formulário nativos ao Colab; porém, isso logo se provou pouco intuitivo do ponto de vista do usuário final, além de prejudicar a disposição estética dos resultados.

Por esse motivo, resolvemos criar uma mini aplicação web que recebesse as entradas por HTML, fizesse uma requisição HTTP a uma API e então apresentasse o resultado de forma visualmente agradável.



The screenshot shows a mobile-style interface. On the left, a large white card contains input fields for 'Categoria' (with a dropdown menu), 'Dia da Semana' (with a dropdown menu showing 'Segunda'), and time inputs for 'Dia' and 'Mês'. Below these are dropdowns for 'Hora' and 'Minuto'. At the bottom of this card are two checkboxes: 'BBB' and 'Feriado'. A blue 'Submit' button is at the very bottom. To the right of this card are four vertical rounded rectangles, each containing a label: 'RAT', 'RCH', 'SHR', and 'FID Breve'.

Screenshot da interface gráfica

Isso foi atingido através de um micro servidor local, criado em Flask, que recebe requisições de um formulário feito com HTML, CSS e Javascript. Esse micro servidor está conectado a um módulo que contém os modelos previamente salvos. Assim, os parâmetros da requisição são utilizados para calcular a predição, a qual é enviada de volta à página web.

▼ Request Payload [view source](#)

```
▼ {Dia da Semana: 3, BBB: 1, Mês: 3, Dia: 4, Hora: 14, Minuto: 30, feriado: 0, Cat  
    BBB: 1  
    Categoria_AUDITORIO: 0  
    Categoria_CARROS E MOTORES: 0  
    Categoria_CULINARIO: 0  
    Categoria_DEBATE: 0  
    Categoria_DOCUMENTARIO: 0  
    Categoria_EDUCATIVO: 0  
    Categoria_ENTREVISTA: 1  
    Categoria_ESPORTE: 0  
    Categoria_FEMININO: 0  
    Categoria_FILME: 0  
    Categoria_FUTEBOL: 0  
    Categoria_GAME SHOW: 0  
    Categoria_HUMORISTICO: 0  
    Categoria_JORNALISMO: 0  
    Categoria_MINISSERIE: 0  
    Categoria_MUSICAL: 0  
    Categoria_NOVELA: 0  
    Categoria_POLITICO: 0  
    .  
    .
```

Payload de uma requisição em JSON para a API

	Headers	Payload	Preview	Response	Initiator	Timing
1				8.92084940657203,4.038493835096491,25.414120605871293		

Resposta da requisição, contendo predições para Rat, Rch e Shr

O código dessa aplicação está disponível no GitHub do projeto. Pretendemos, até o final do projeto, hospedar toda essa arquitetura em algum provedor gratuito para acesso remoto.

4.5. Avaliação

Ao todo, foram quase 90 modelos testados em cinco rodadas, divididos entre regressão linear, KNN, decision trees, random forest, LightGBM, CatBoost e XGB. Como métricas, utilizamos, para todos eles, R², R² ajustado, erro médio absoluto e erro médio percentual.

Nesse panorama, alguns modelos tiveram um desempenho surpreendentemente positivo, atingindo mais de 90% em coeficiente de determinação e erros percentuais de menos de 10%, ou 0,7 em valor absoluto. Por outro lado, algumas tentativas resultaram em métricas baixíssimas, chegando a R² negativos e erros de 100%.

Ademais, vale ressaltar que tivemos um divisor de águas na Sprint 4 com o entendimento de que o BBB estava duplicado e que poderíamos otimizar nossos hiperparâmetros com Random Search. Principalmente devido às mudanças de features na Rodada 6, percebemos que não poderíamos mais comparar resultados com as rodadas anteriores, pois estaríamos lidando com modelos fundamentalmente diferentes. Por isso, separamos nossa análise em duas partes – Rodadas 1-5 e Rodadas 6-7.

Rodadas 1 a 5

Melhores resultados

Algoritmo	Melhor modelo (R ² ajustado)	Pior modelo (R ² ajustado)
Regressão linear	0.47229207388282823 (Rodada 1 - Valores Normalizados)	-2.0044504114610543e+20 (Rodada 1 - Valores Padronizados)
KNN	0.8891245924911518 (Rodada 2 - n_neighbors = 7)	0.3546405818810551 (Rodada 5 - n_neighbors = 2)
Árvore decisória (Decision Tree)	0.9409509640864655 (Rodada 3 - max_depth = 35)	0.32804906563363356 (Rodada 5 - max_depth = 2)
Florestas aleatórias (Random Forest)	0.9598968867968866 (Rodada 3 - max_depth = 200)	0.564047996324722 (Rodada 5 - max_depth = 20)
Gradient Boosting	0.975951121438195 (Rodada 4 - LightGBM)	0.5645468235985043 (Rodada 5 - LightGBM)

Com base nos resultados obtidos, pudemos, finalmente, determinar nossas métricas de sucesso para os modelos atuais e os futuros, pois, agora, sabemos o que é possível atingir. Nesse sentido, determinamos como critério mínimo de qualidade um R² de pelo menos 85%, preferencialmente maior ou igual a 90%. A partir dessa faixa, consideramos com igual peso o

erro médio e, também, a explicabilidade do algoritmo, a fim de selecionar o melhor modelo. Por exemplo, apesar do algoritmo LightGBM consistemente render os melhores resultados, ele é muito mais difícil de explicar e entender do que a árvore decisória. Assim, para R^2 acima de 85%, consideramos também o fator explicabilidade para que o modelo final instigue maior confiança no usuário final.

Já no fator explicabilidade, o mais fácil de ilustrar é, sem dúvidas, a regressão linear, pois exige apenas o elencamento dos coeficientes. No entanto, como vimos nos experimentos, esse é o algoritmo de pior performance para os nossos dados. A próxima opção são as árvores decisórias, que podem ser explicadas por gráficos de decisão construídos através do método “plot_tree”. Tentamos gerar esse gráfico para nosso melhor modelo, porém o código rodou por mais de 30m antes de falhar. Enquanto trabalhamos em maneiras de otimizar essa função, geramos um gráfico mais simples, baseado em um modelo com menor profundidade, para demonstrar como essa representação funciona.

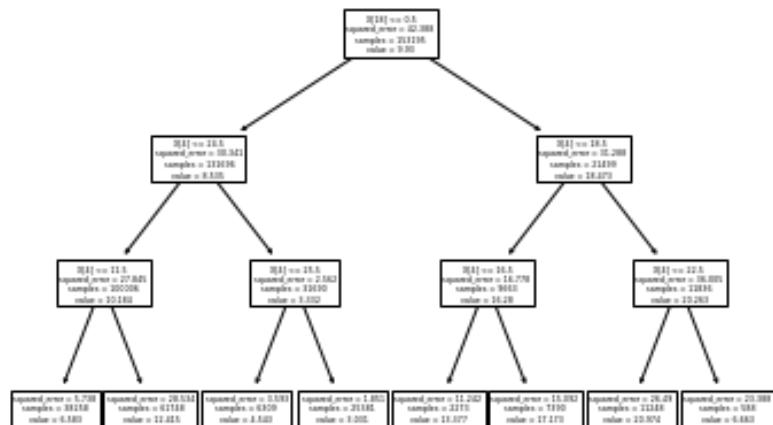


Gráfico de árvore decisória

Além disso, o LightGBM também contém um método que ilustra as regras do modelo. No entanto, ainda não conseguimos gerá-lo com resolução suficiente para ser inteligível, pois ele é muito grande. Até o momento, só conseguimos criar versões minúsculas, como a disponível abaixo.



Gráfico de LightGBM

Rodadas 6 e 7

Melhores resultados

Algoritmo	Rodada 6	Rodada 7
Régressão linear	0.4158912371517206	n/a
KNN	0.80326066204431 ($k = 7$)	0.80326066204431
Árvore decisória (Decision Tree)	0.940872888840858 ($\text{max_depth} = 40$)	0.9399598874368328 ($\text{max_depth} = 28$)
Florestas aleatórias (Random Forest)	0.9599161485095925 ($\text{estimators} = 200$)	0.9598184105363474 ($\text{estimators} = 166$)

Observando os resultados acima, entendemos que o melhor modelo dessas rodadas foi o de floresta aleatória. Algoritmos de gradient boosting provavelmente teriam performado ainda melhor, julgando pelo 98% atingido com LightGBM anteriormente. Porém, 94%-95% é mais que o suficiente quando se leva em consideração a explicabilidade do modelo e os gráficos associados. Afinal, não adianta nada ter um AI extremamente acurado que não passa confiança para seu usuário final nem pode ser defendido de modo transparente no meio legal (em caso de processos, por exemplo).

Assim, escolhemos como modelo final para Total Domicílios / Rat% o algoritmo de **Árvore Decisória com 28 de profundidade**. O R² é praticamente indistinguível entre 28 e 40 de max_depth; logo, escolhemos menor número para que a execução dos métodos seja mais curta. Suas métricas exatas são:

Erro médio: 0.9053855191545851
Erro médio percentual: 9.13%
R²: 0.9399598874368328
R² ajustado: 0.9399598874368328

Além disso, seu gráfico é muito mais inteligível do que o de LightGBM, por exemplo. Infelizmente, ainda não conseguimos gerar esse gráfico em tempo hábil no Colab. Pretendemos fazer isso localmente na Sprint 5.

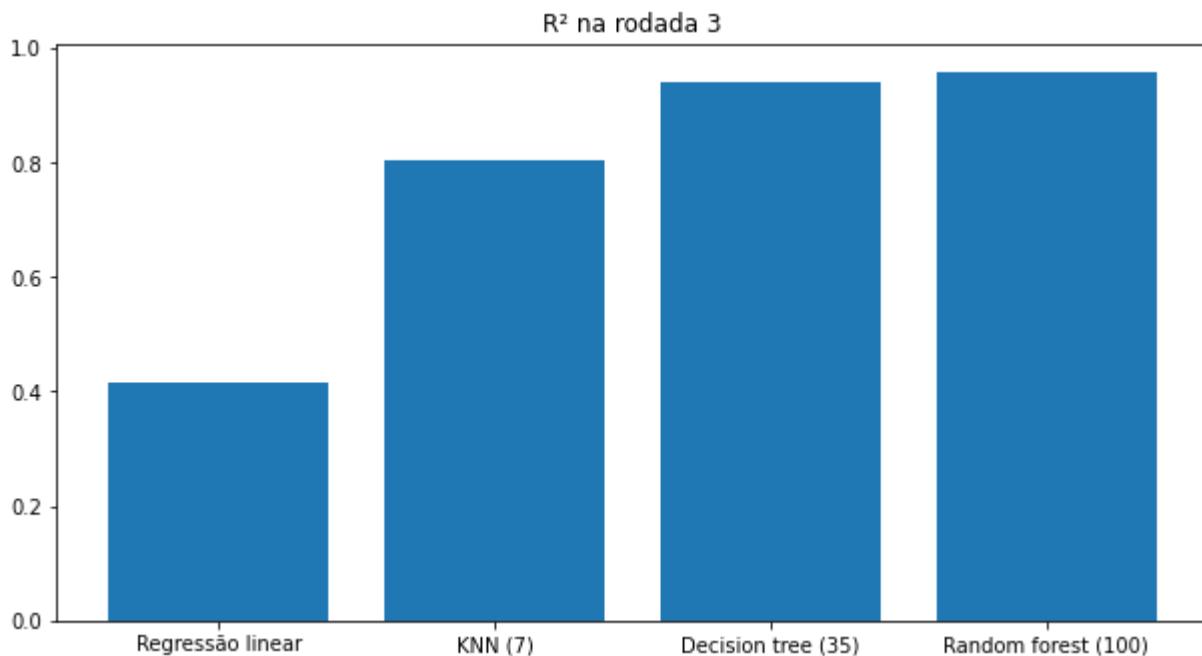
Quanto aos outros modelos, deixamos a escolha deles em aberto até o fim do projeto. Dizemos modelos porque, apesar do escopo do projeto prever apenas um tipo de predição (no caso, de Rat%), pretendemos entregar uma interface que agrupe tanto Rat% quanto Fid%, Shr% e Rch%, possivelmente segmentados por demográficos também.

4.6. Comparação de modelos

Nossa comparação de modelos consistiu no método de k-fold cross-validation disponível na biblioteca Scikit Learn. Essa ferramenta calcula a média de R^2 com base em várias amostras de treino e teste. Nesse sentido, o dataset original é dividido em k conjuntos. Então, a cada iteração, $k - 1$ conjuntos são utilizados para treino, com o restante servindo de teste. Essa divisão muda a cada repetição, o que garante que todos os subconjuntos sejam teste pelo menos uma vez. Assim, diminui-se o risco de overfitting e tem-se uma visão mais assertiva da qualidade de um modelo, pois levamos em consideração mais de um treinamento.

No entanto, devido às limitações de tempo da Sprint 4 e as modificações necessárias na etapa de modelagem (leia-se, remoção de BBB e testes com random search), decidimos focar a comparação apenas no modelo principal para este projeto: o de Total Domicílios | Rat%. Nesse contexto, realizamos comparações para a Rodada 3 (nossa melhor rodada da Sprint 3), para a Rodada 6 (baseada na Rodada 6, mas sem BBB) e para a Rodada 7 (baseada na Rodada 6 com random search). Excluímos as outras porque elas não incluíam as features mais recentes e, portanto, tinham grande chance de estarem enviesadas. Desse modo, as Rodadas 6 e 7 atuam não como alternativas às outras, mas como aprimoramentos a elas, podendo substituí-las nas análises.

Rodada 3

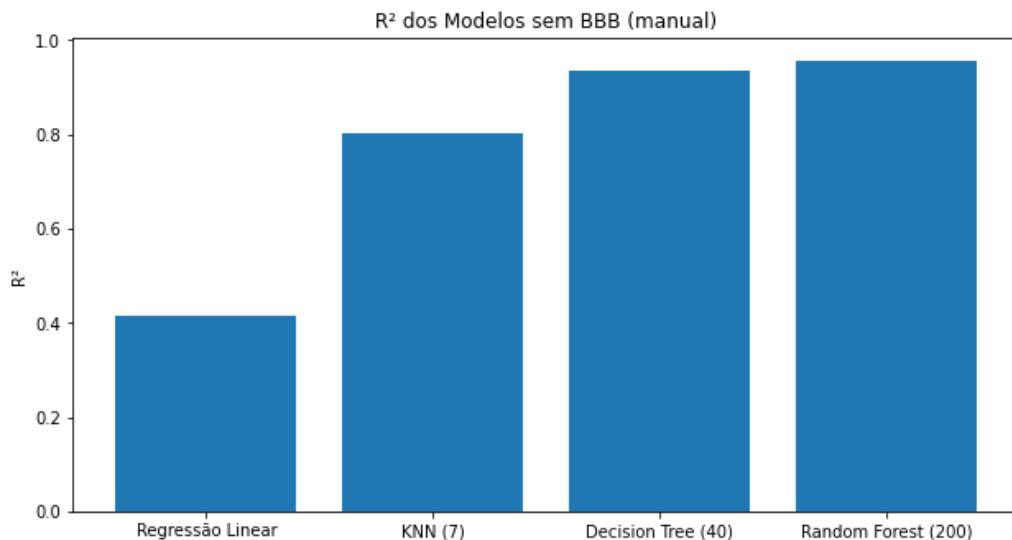


Na Rodada 3, temos a esperada má performance de regressão linear. Já o KNN se apresenta como superior, chegando a 80% de R², mas não se sustenta frente ao decision tree e random forest, ambos acima de 90% de acurácia. A diferença entre esses dois últimos, principalmente, é mínima, como se pode ver no output do cross-validation abaixo:

```
Decision Tree (35): 0.960295 (0.001758)
/usr/local/lib/python3.7/dist-packages/j
    "timeout or by a memory leak.", UserWa
Random Forest (200): 0.971594 (0.000792)
KNN (7): 0.814859 (0.002862)
Linear regression: 0.414827 (0.005880)
```

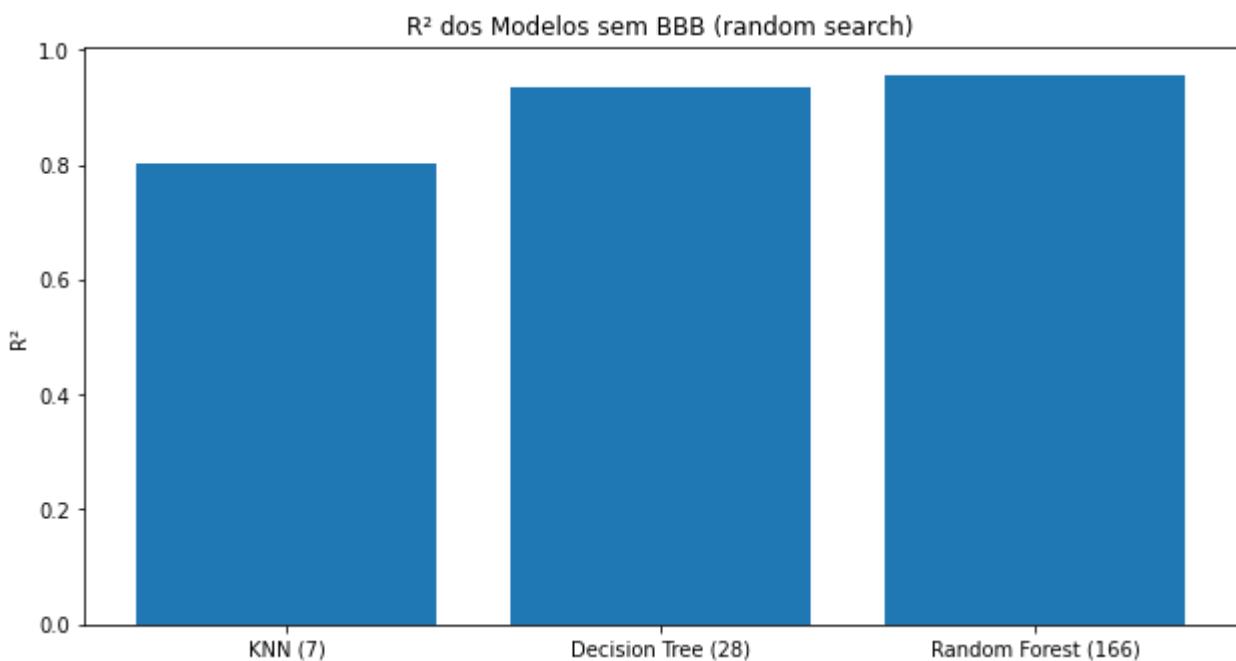
Em geral, são os resultados esperados com base nos desempenhos vistos em outras rodadas. Entre árvore decisória e floresta aleatória, com apenas 1 ponto percentual de diferença, damos preferência à árvore pela sua maior explicabilidade.

Rodada 6



Similarmente à rodada 3, a versão sem BBB oferece como melhor resultado o random forest e, como pior, a regressão linear. Os valores em si de R² não diferem muito, ficando acima dos 90% e em torno de 40%, respectivamente. Ainda assim, o padrão de se ter o decision tree em níveis comparáveis persiste, chegando a mais de 90% de R² com 40 de profundidade. Considerando a explicabilidade, é o melhor modelo, sem contar o fato de rodar mais rapidamente do que o random forest.

Rodada 7



A rodada 7 é a mais importante, pois otimiza parâmetros em nossa melhor seleção de features. Portanto, sabíamos já antes da comparação que nosso modelo escolhido viria desse conjunto. Visualmente, tem-se novamente a quase equiparação de decision tree com random forest:

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.  
[Parallel(n_jobs=-1)]: Done  3 out of  3 | elapsed:  7.9min finished  
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.  
KNN (7): 0.801285 (0.001136)  
[Parallel(n_jobs=-1)]: Done  3 out of  3 | elapsed:    2.1s finished  
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.  
Decision Tree (28): 0.935978 (0.003671)  
Random Forest (166): 0.957182 (0.000687)  
[Parallel(n_jobs=-1)]: Done  3 out of  3 | elapsed:  3.2min finished
```

Decision tree conseguiu 93% de R², enquanto random forest chegou a 95%. Essa diferença percentual é compensada, entretanto, pelo gráfico mais simples da árvore decisória juntamente de seu tempo de execução menor. Quanto ao KNN, apesar de muito bom, não oferece benefícios significativos o suficiente para justificar o R² de 80%.

5. Conclusões e Recomendações

Escreva, de forma resumida, sobre os principais resultados do seu projeto e faça recomendações formais ao seu parceiro de negócios em relação ao uso desse modelo. Você pode aproveitar este espaço para comentar sobre possíveis materiais extras, como um manual de usuário mais detalhado na seção “Anexos”.

Não se esqueça também das pessoas que serão potencialmente afetadas pelas decisões do modelo preditivo, e elabore recomendações que ajudem seu parceiro a tratá-las de maneira estratégica e ética.

6. Referências

State of Mobile 2022 - data.ai. Disponível em: <<https://www.data.ai/en/go/state-of-mobile-2022/>>.

AMANDA CHAN 12 APRIL 2011. Discovery Reveals Why Old People Go to Bed Early. Disponível em: <<https://www.livescience.com/13666-older-people-sleep-wake-early.html>>.

Parcela da população que se declara dona de casa cai para 7% em 26 anos. Disponível em: <<https://www1.folha.uol.com.br/mercado/2019/08/parcela-da-populacao-que-se-declara-dona-de-casa-cai-para-7-em-26-anos.shtml>>. Acesso em: 14 ago. 2022.

Anexos

Utilize esta seção para anexar materiais como manuais de usuário, documentos complementares que ficaram grandes e não couberam no corpo do texto etc.

Jornada do Usuário

Expectativas

- Fazer uma melhor seleção de programas;
- Identificar os fatores que levam um programa ser popular;
- Suprir demandas desconhecidas do público;
- Desenvolver dados argumentativos mais sólidos.



Rodrigo, Gerente de Operação e Programação

Cenário: Rodrigo quer realizar um plano de ação mais objetivo para a audiência de um novo programa.

FASE 1 (Organização de dados) <p>1.Começa a organizar (quase que manualmente) os números de audiência, características dos telespectadores e horários que possam se relacionar com o novo programa; 2. Para acessar esses dados (tabelas em Excel) precisa pedir acesso para outros setores.</p>	FASE 2 (Análise) <p>1.Confere mais cuidadosamente os dados selecionados, com viés de análise; 2.Checa o numero de audiência, características do público e horário de antigos programas que se relacionam com o novo; 3.Imagina o quanto bem esse novo programa poderia ser encaxado com base em experiências anteriores.</p>
FASE 3 (Deduzindo) <p>1.A partir do que se foi imaginado e concluído (tendendo ao objetivo), temos as primeiras propostas de encaixe da nova programação expostas, a partir da audiência; 2.Discute as novas propostas com o que foi formulado.</p>	FASE 4 (Preparando) <p>1.Organiza os dados coletados anteriormente para serem inseridos no modelo preditivo; 2.Apresenta as circunstâncias e dúvidas trabalhadas para o modelo preditivo.</p>
FASE 5 (Conferir e Estabelecer) <p>1.Compara as suposições subjetivas já feitas com aquelas desenvolvidas no modelo preditivo, fazendo questionar o que se foi concebido anteriormente ou apenas reforçando; 2.Com esse estudo mais completo, sugestões dessa nova grade de horário são apresentadas.</p>	FASE 5 (Conferir e Estabelecer) <p>'Agora sim, embasado e completo. Estou satisfeito com o resultado do modelo preditivo!'</p> <p>'Inserir todos os dados no modelo preditivo é bem cansativo/chato'</p> <p>'Bom, contudo, seria ótimo ter argumentos mais sólidos para a implementação de um novo programa na grade. Ainda tenho minhas dúvidas em relação a essas novas propostas'</p> <p>'Pelo o que já se passou até hoje, como seria o desempenho de um novo programa nesse cenário? Imagino o quanto bem esse programa e nossa rede possa se tornar'</p> <p>'É essencial organizar todos os dados que possam se relacionar com o contexto de um novo programa, mesmo o processo sendo extremamente cansativo'</p>

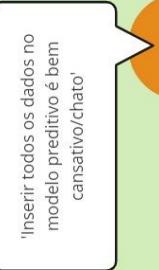
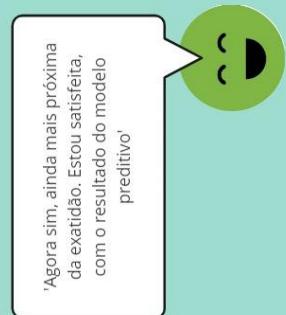
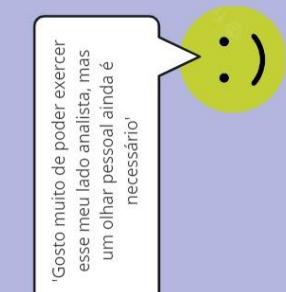
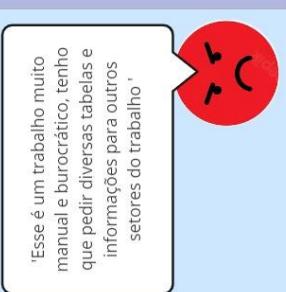
Oportunidades

- Automatizar e deixar mais rápida a organização inicial de dados;
- Deixar mais prática a inserção de dados no modelo preditivo.

Giovanna Mattos, Gerente Geral de Marketing
Cenário: Giovanna quer identificar exatamente (ou o mais próximo disso) o melhor horário para divulgar comerciais e propagandas dos novos programas.

Expectativas

- Identificar fatores que mais chamam a atenção dos telespectadores em uma propaganda;
- Melhores evidências para alocação de propagandas.

<p>FASE 1 (Destrichando/Organizando)</p> <p>1.Após receber as informações do novo programa, confere quais programações já estabelecidas mais se assemelham a ele; 2.Pede acesso aos dados (tabelas Excel) desses programas semelhantes (horário, público atingido, antiga forma de divulgação) para outros setores.</p>	<p>FASE 2 (Análise)</p> <p>1.Confere mais cuidadosamente os dados selecionados, com viés de análise; 2.Checa em quais programas e horários o público alvo está mais ativo; 3.Compara como antigas divulgações desse gênero de programa foram feitas e seus impactos, mantendo acertos e evitando erros.</p>	<p>FASE 3 (Deduzindo)</p> <p>1.A partir do que se foi imaginado e concluído na análise, temos as primeiras propostas de encaixe da divulgação do novo programa; 2.Discorre as novas propostas com o que foi formulado.</p>	<p>FASE 4 (Preparando)</p> <p>1.Organiza os dados coletados anteriormente para serem inseridos no modelo preditivo; 2.Apresenta as circunstâncias e dúvidas trabalhadas para o modelo preditivo.</p>	<p>FASE 5 (Conferir e Estabelecer)</p> <p>1.Compara as suposições subjetivas já feitas com aquelas desenvolvidas no modelo preditivo, fazendo questionar o que se foi concebido anteriormente ou apenas reforçando; 2.Com esse estudo mais exato e completo, a forma de divulgação desse novo programa é apresentada.</p>
			 <p>'Inserir todos os dados no modelo preditivo é bem cansativo/chato'</p>	 <p>'Agora sim, ainda mais próxima da exatidão. Estou satisfeita, com o resultado do modelo preditivo'</p>
			 <p>'Bom, mas ainda é algo um tanto que subjetivo, queria ter ainda mais exatidão nessa minha proposta de marketing em divulgação'</p>	 <p>'Gostei muito de poder exercer esse meu lado analista, mas um olhar pessoal ainda é necessário'</p>
			 <p>'Esse é um trabalho muito manual e burocrático, tenho que pedir diversas tabelas e informações para outros setores do trabalho'</p>	

Oportunidades

- Deixar mais prático e evitar intermediários na organização inicial de dados;
- Tornar a inserção de dados no modelo preditivo menos manual e mais rápida.