



Modelo preditivo da variabilidade
da evolução do câncer de mama -
Faculdade de Medicina da
Universidade de São Paulo

Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
04/08/2022	Gabriela Barretto, Pedro Romão e Wagner Estevam	1.0	Criação do documento.
10/08/2022	Gabriela Barretto e Wagner Estevam	1.5	Preenchimento das seções 1, 2, 4.1 e 4.2.
14/08/2022	Elias Biondo	1.9	Refinamento geral pré-entrega.
26/08/2022	Gabriela Barretto, Elias Biondo, Matheus Fidelis	2.0	Preenchimento da seção 4.3.
06/09/2022	Gabriela Barretto, Wagner Estevam, Matheus Fidelis	3.0	Preenchimento das seções 4.3.2, 4.4 e 4.5
20/09/2022	Luca Giberti e Wagner Estevam	4.0	Atualização das seções 4.4.1, 4.4.3, 4.4.4, 4.4.5 Preenchimento da seção 4.5
22/09/2022	Elias Biondo, Matheus Fidelis e Wagner Estevam	4.5	Refinamento geral pré-entrega.

Sumário

1. Introdução	4
2. Objetivos e Justificativa	5
2.1. Objetivos	5
2.2. Proposta de solução	5
2.3. Justificativa	5
3. Metodologia	6
3.1. CRISP-DM	6
3.2. Ferramentas	6
3.3. Principais técnicas empregadas	6
4. Desenvolvimento e Resultados	7
4.1. Compreensão do Problema	7
4.1.1. Contexto da indústria	7
4.1.2. Análise SWOT	9
4.1.3. Planejamento Geral da Solução	9
4.1.4. Value Proposition Canvas	10
4.1.5. Matriz de Riscos	10
4.1.6. Personas	11
4.1.7. Jornadas do Usuário	11
4.2. Compreensão dos Dados	12
4.3. Preparação dos Dados	14
4.4. Modelagem	15
4.5. Avaliação	16
4.6. Comparação de Modelos	17
5. Conclusões e Recomendações	18
6. Referências	19
Anexos	20

1. Introdução

A Universidade de São Paulo (USP) é a melhor universidade da América Latina de acordo com o ranking Best Global Universities (2021), recebendo destaque na área de pesquisa científica nacional e internacional. É, também, a melhor universidade pública do país. A Faculdade de Medicina da Universidade de São Paulo (FM-USP) é a principal parceira de negócio desse projeto. Associada ao Hospital das Clínicas (HC/FM-USP), uniram forças para a criação de uma solução inovadora. Posicionando-se sempre à favor da vida, sua área de atuação, no contexto desse produto, é a saúde humana e a oncologia (especialidade médica que estuda os cânceres e a forma como essas doenças se desenvolvem).

A evolução do câncer de mama e suas respostas a tratamentos convencionais é muito variável, o que faz com que médicos não saibam tratar, de forma objetiva, casos particulares e, ainda, identificar o grau de risco dos pacientes e o seu tempo de sobrevida estimado. Torna-se interessante, a partir disso, o surgimento de alguma forma de tecnologia para estimar essas variáveis visando uma distribuição mais eficaz e eficiente dos recursos públicos do Sistema Único de Saúde (SUS). Dessa maneira, além da economia de aportes, vidas poderão ser salvas através de um tratamento direcionado e personalizado.

2. Objetivos e Justificativa

2.1. Objetivos

O principal objetivo do parceiro de negócio é viabilizar a criação de uma solução capaz de analisar, filtrar e classificar dados de pacientes com câncer a fim de identificar a variabilidade da evolução da doença, bem como as possíveis respostas a tratamentos já implantados, para, assim, detectar padrões que indiquem aos profissionais de saúde uma possível estimativa de risco e grau de sobrevida de um paciente.

Como objetivo secundário, destaca-se o direcionamento de recursos disponíveis de maneira mais eficaz e eficiente, explica-se: com a identificação de padrões que indiquem aos profissionais de saúde uma possível estimativa de risco e grau de sobrevida de um paciente, será possível conduzir menos ou mais consultas e exames a depender da necessidade do paciente, tornando o processo totalmente personalizável às variáveis identificadas.

Não fica de fora, também, o aumento da qualidade de vida dos enfermos, uma vez que eles poderão se deslocar menos ou planejar mais idas ao hospital dependendo de suas condições clínicas.

2.2. Proposta de solução

Dado o exposto, a partir da matéria-prima disponibilizada pelo parceiro - os dados de pacientes com câncer - a proposta de solução da equipe é um modelo preditivo da variabilidade da evolução do câncer de mama, isto é, um modelo de aprendizado de máquina e inteligência artificial que identifique padrões da variabilidade da evolução da doença em relação aos tratamentos e terapias aplicados sob os enfermos, a fim de retornar um prognóstico com padrões explícitos que indiquem aos profissionais de saúde o risco e grau de sobrevida de um paciente específico com base em seus dados clínicos.

2.3. Justificativa

Um algoritmo de análise preditiva, como proposto, assegurará a consideração de todos os dados relevantes à variabilidade da evolução da doença. Apesar de ainda pouco explorados, esses algoritmos possuem a capacidade de mudar como a humanidade pensa e executa a medicina nos tempos atuais. Essas análises nunca poderiam ser feitas, de maneira generalizada, por seres humanos, uma vez que a quantidade de dados existentes é extremamente massiva. Tendo como premissa básica a qualidade da coleta dos dados disponibilizados e a ausência de vieses no banco, é garantida uma boa assertividade nas predições do algoritmo, viabilizando, dessa maneira, um sistema completo e rápido de análise de casos de pacientes com a doença e o seu direcionamento eficiente.

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

3.2. Ferramentas

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

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

Não foram identificados concorrentes diretos do parceiro de negócio, uma vez que o mesmo se posiciona como uma instituição pública a favor da sociedade e em serviço por ela, explica-se: toda produção de conhecimento interna é passível de uso por qualquer outra organização, desde que sejam respeitados os devidos direitos de uso e direitos comerciais. No entanto, é possível destacar algumas outras empresas que atuam na mesma fatia de mercado, como por exemplo:

- O Hospital A.C.Camargo, especializado no diagnóstico, tratamento e pesquisa de câncer em humanos;
- O Instituto de Câncer do Estado de São Paulo (ICESP), maior centro oncológico da América Latina, especializado no tratamento de casos de câncer de alta complexidade;
- O Instituto Nacional de Câncer (INCA), órgão auxiliar do Ministério da Saúde no desenvolvimento e coordenação das ações integradas para a prevenção e o controle de câncer no Brasil

Como uma instituição pública sem fins lucrativos, fica evidente o modelo de negócio do parceiro quanto ao seu pioneirismo e excelência nos âmbitos de ensino, pesquisa e extensão universitária, além da modernização e inovação tecnológica. Não cedendo a pressões do mercado, a instituição é capaz de, por meio de investimentos da União, inovar e estar na vanguarda da ciência, sempre com as últimas e mais modernas soluções de problemas.

Sobre o uso crescente de algoritmos de aprendizado de máquina e inteligência artificial, emerge-se a tendência da substituição ou o aumento das capacidades humanas para a compreensão objetiva de conceitos científicos, matemáticos e humanos. Em 2021, o setor de saúde digital alcançou um recorde de US\$ 57,2 bilhões de dólares, segundo os dados do relatório State of Digital Health da plataforma norte-americana de inteligência de mercado CB Insights, correspondendo a um aumento de 79% em relação ao ano anterior.

Em uma visão macro, segue abaixo a análise da indústria realizada de acordo com as diretrizes de Michael Porter e suas Cinco Forças:

1. **Rivalidade entre os concorrentes:** por ser uma instituição governamental sem fins lucrativos, o parceiro não possui concorrentes diretos, mas sim hospitais associados em pesquisa e tratamento da doença. Muito para além disso, identificam-se organizações privadas que, no sentido figurado, concorrem por pacientes. Como fator de observação, identifica-se os altos investimentos em educação e saúde por parte de outras organizações.
2. **Poder de negociação dos fornecedores:** por necessitar de produtos extremamente refinados e de alto valor agregado, o parceiro possui fornecedores com alto poder de barganha, uma vez que certos medicamentos e aparelhos de pesquisa não são ofertados de maneira massiva e possuem suas produções e precificação controladas por um pequeno grupo de empresas. Como fornecedores de dados, os hospitais também podem negar o fornecimento da matéria prima das atividades desenvolvidas pelo parceiro: os dados. E, por fim, pelas tecnologias criadas necessitarem de um extenso processo de licenciamento, exigências quanto ao custo da solução podem ser feitas por aqueles que viabilizaram o processo de sua criação.
3. **Poder de negociação dos clientes:** como detentores de seus dados, os pacientes podem negar o uso de seus dados ao parceiro, inviabilizando, assim, a sua atuação em pesquisa e no desenvolvimento de novas tecnologias. Em outra perspectiva, eles também podem recusar o uso das soluções desenvolvidas impactando, também, à sua aderência e popularidade.
4. **Ameaça de novos entrantes:** com os recentes altos investimentos da iniciativa privada no desenvolvimento de novas tecnologias e serviços na mesma fatia de mercado do parceiro, identifica-se um risco moderado de aparecimento de soluções mais adequadas visto a abrangência e disponibilidade de recursos quase que inesgotável de fundos financeiros interessados.
5. **Ameaça de produtos substitutos:** criação de terapias alternativas para a manipulação de genes que causam a tendência do câncer, inviabilizando, assim, a necessidade das soluções propostas pelo parceiro.

4.1.2. Análise SWOT

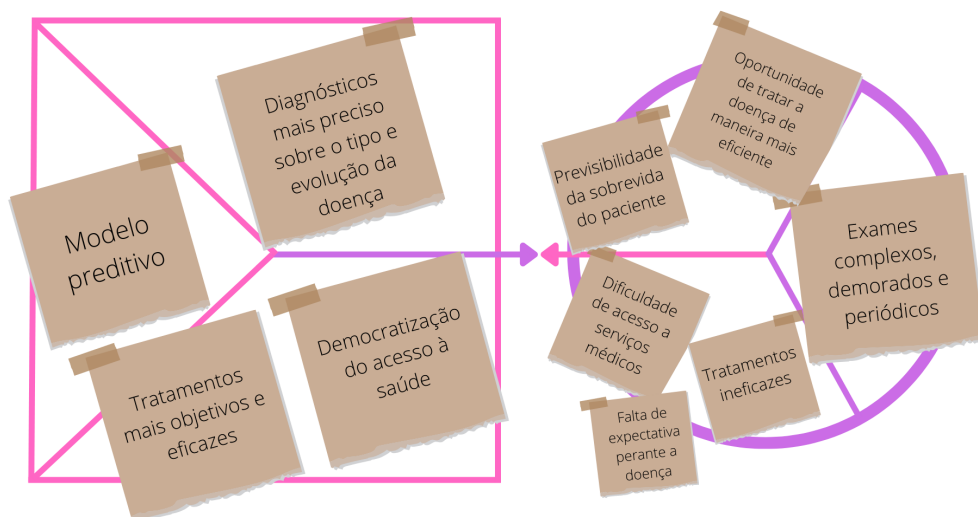


Anexo 1 - Análise Swot.

4.1.3. Planejamento Geral da Solução

- a) quais os dados disponíveis (fonte e conteúdo - exemplo: dados da área de Compras da empresa descrevendo seus fornecedores)
- b) qual a solução proposta (pode ser um resumo do texto da seção 2.2)
- c) qual o tipo de tarefa (regressão ou classificação)
- d) como a solução proposta deverá ser utilizada
- e) quais os benefícios trazidos pela solução proposta
- f) qual será o critério de sucesso e qual medida será utilizada para o avaliar

4.1.4. Value Proposition Canvas



Anexo 2 - Canvas Value Proposition.

4.1.5. Matriz de Riscos

		Ameaças					Oportunidades				
Probabilidade	90%			Ameaça 018: nenhum padrão de predição detectado	Ameaça 019: análise SWOT incompleta	Ameaça 020: análise de mercado imprecisa	Oportunidade 008: refinar conhecimentos sobre desenvolvimento de algoritmos de aprendizado de máquina em geral	Oportunidade 009: refinar conhecimentos sobre design e experiência do usuário em modelos preditivos	Oportunidade 010: entendimento da falta de mercado de atuação do cliente	Oportunidade 011: refinar conhecimentos sobre documentações de projetos de inteligência artificial	
	70%				Ameaça 016: falta de alinhamento com o cliente no que diz respeito ao significado de algumas especificidades médicas	Ameaça 017: definição de persona imprecisa no que diz respeito ao uso da solução, vontades, pensamentos e expectativas	Oportunidade 005: desenvolvimento de power-skills relacionadas ao setor da saúde	Oportunidade 006: aprender a lidar com imprevistos de maneira positiva	Oportunidade 007: assimilar conhecimentos relacionados a esse tipo de projeto		
	50%	Ameaça 011: atualização errônea no algoritmo da solução (mistagem entre códigos incorreta que pode levar a divergências de interpretação)	Ameaça 012: atraso na entrega de possíveis funcionalidades extras solicitadas de última hora	Ameaça 013: integrante do projeto se ausentar por motivos pessoais	Ameaça 014: dados ausentes ou enviesados	Ameaça 015: distribuição inadequada do projeto	Oportunidade 002: cliente dar continuidade ao mínimo produto viável desenvolvido	Oportunidade 003: expandir a rede de contatos a partir das pessoas envolvidas no projeto	Oportunidade 004: adição do projeto ao portfólio		
	30%	Ameaça 006: backlog com informações desatualizadas e uso de metodologias ainda pouco exploradas	Ameaça 007: construção de um modelo preditivo não entendível	Ameaça 008: consumo de documentação ambíguo	Ameaça 009: definição de histórias de usuário inapropriadas	Ameaça 010: interpretação incorreta ou enviesada dos dados	Oportunidade 001: captar novas propostas e projetos do cliente em razão do resultado do projeto				
	10%	Ameaça 001: definição de preferências de design e experiência do usuário para solução de inteligência artificial inadequadas	Ameaça 002: desenvolvimento de códigos de aprendizado de máquina não padronizados	Ameaça 003: construção de uma solução com baixa aderência do público por desconfiança	Ameaça 004: demora no processamento e operações de dados pelos servidores disponíveis	Ameaça 005: servidores de processamento de dados indisponíveis para uso					
		Muito baixo	Baixo	Moderado	Alto	Muito alto	Muito alto	Alto	Moderado	Baixo	Muito Baixo
Impacto											

Anexo 3 - Matriz de Riscos.

4.1.6. Personas

Persona 1



Danilo Cavalcanti

Idade: 43 anos
Ocupação: médico oncologista
Cidade: Ribeirão Preto
Educação: ensino superior

Biografia

"De família humilde, conquistou a tão sonhada vaga no curso de medicina. Sempre muito dedicado, formou-se em segundo lugar geral na Universidade de São Paulo, sendo, também, orador de sua turma. Acredita que a medicina é mais que apenas uma ciência: é prática do amor pelo próximo e da empatia. Especializado em oncologia, trata seus pacientes os distraíndo de suas doenças. Para ele, acreditar na cura é o primeiro passo para que ela se materialize. Recomenda a prática de lazer e de dias agradáveis. Devido ao falecimento de sua mãe por câncer de mama, tem um grande interesse na área, se interessando pelo uso de novas tecnologias e a execução de pesquisas inovadoras na área. Em alguns momentos, sonha na possibilidade de recomendar tratamentos mais efetivos para seus pacientes diante suas condições de saúde. Acredita que com a evolução dos computadores e de modelos matemáticos e estatísticos refinados, isso se torne possível algum dia."

Personality

Introvertido Extrovertido

Analtico Criativo

Ocupado Ocioso

Bagunçado Organizado

Independente Equipe em primeiro lugar

Interesses

Oncologia	Medicina solidária	Saúde e boa forma
Alimentação equilibrada	Pesquisas científicas	Doutorado
Sistema Único de Saúde	Turismo	Tecnologia e suas aplicações
Enfermagem	Matemática	

Influências

Medicina regenerativa	Atenção primária à saúde	Dr. Drauzio Varella
Faculdade de Medicina da USP	Ministério da Saúde	Conselho Federal de Medicina
Sistema Único de Saúde (SUS)	Hospital das Clínicas (HC-FM/USP)	Conselho Nacional de Saúde
Instituto Nacional de Câncer (INCA)	Fundação Oswaldo Cruz (Fiocruz)	

Metas

Disseminar a prevenção do câncer	Conectar a medicina com a tecnologia	Entender a evolução do câncer de mama
Ler prontuários com objetivos claros	Ter algum tempo livre para ficar ocioso	

Necessidades e expectativas

Esquematizar consultas de maneira eficiente	Criar um calendário singular de atendimento	Priorizar pacientes em diferentes níveis
Oferecer um diagnóstico mais assertivo	Confirmar o prognóstico prestado	Perceber padrões de evolução da doença


Motivações

Tratar e curar seus pacientes	Promover o acesso à saúde de qualidade	Reestabelecer a crença na saúde pública
-------------------------------	--	---

Dores e frustrações

Maua priorização de pacientes	Fila gigantesca de pessoas para atender	Indicação de tratamentos agressivos
Ineficiência em estudar paciente por paciente		

Persona 2



Lucimara Santos

Idade: 55 anos

Profissão: caixa de supermercado

Cidade: Ribeirão Preto

Educação: ensino fundamental

Biografia

"Nascida no interior do estado, em uma região rural, mudou-se para a cidade grande aos 25 anos. Começou a trabalhar desde cedo, pois, infelizmente, sua realidade não oferecia tantas oportunidades assim. Mãe de três filhos, sendo dois meninos e uma menina, sempre fala do seu desejo em vê-los formados e independentes. A coisa que mais ama é ter a família reunida em sua casa nos domingos, ao redor de uma boa e farta mesa de comida. O câncer de mama virou a sua vida de cabeça para baixo. Além da doença, sofre com a possibilidade de perder seu emprego em decorrência das suas faltas para ir ao hospital. A sua saúde, apesar de boa, até então, vem se deteriorando devido a todas as preocupações advindas do seu diagnóstico. Tem fé que um bom médico poderá encontrar o melhor tratamento para si, evitando ser agressivo demais com o seu corpo e entendendo todos os processos que tangem a sua enfermidade. Acredita que superará essa fase e servirá de exemplo para que outras mulheres não desistam no meio do caminho."

Personality

Introversa Extroversa

Análise Criativa

Ocupada Ociosa

Bagunçada Organizada

Independente Equipe em primeiro lugar

Interesses

Receitas	Comidas de domingo	Salgados de festa
Unhas	Cabelo	Dicas domésticas
Menopausa	Família	Vida de mãe
Estudo dos filhos	Casa organizada	Saúde pública

Influências

Publicações do Facebook	Pesquisas na Internet	Conhecimentos populares
Comerciais de televisão	Rádio matutina	Revistas de beleza e saúde
Mães blogueiras	Enfermeiras e médicos	Curandeiras
Redes de apoio à mulher com câncer		

Metas

Curar seu câncer de mama	Ser exemplo para seus filhos	Não desistir independentemente da doença
Voltar a estudar	Ajudar outras mulheres na prevenção do câncer	

Necessidades e expectativas

Ir menos ou mais ao consultório médico	Encontrar um tratamento eficiente e efetivo	Ter um atendimento personalizado
Receber um prognóstico assertivo	Entender sua doença e possível evolução	Receber um parecer geral da sua saúde

Motivações

Ver seus filhos formados	Mudar de profissão	Propagar bons hábitos
Informar sobre saúde e prevenção	Promover saúde pública de qualidade	Propor bem-estar mesmo na doença

Dores e frustrações

Tratamento sem respostas mensuráveis	Enorme demora no atendimento do hospital	Tratamento muito agressivo
Sem perspectiva da evolução da doença	Muitas idas sem propósito ao hospital	Impessoalidade do médico
Sentimento de estar sozinha		

4.1.7. Jornadas do Usuário

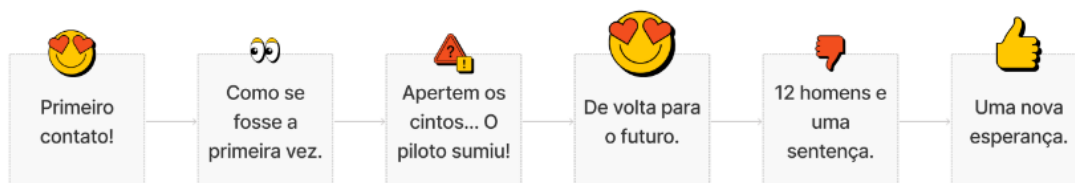


Danilo Cavalcanti



Cenário: médico oncologista testa, com a devida autorização, em um consultório, nova solução de inteligência artificial para a predição da variabilidade da evolução do câncer de mama em paciente mulher.

Expectativa: médico oncologista recebe, sem dificuldades, como resposta, em diferentes graus e escalas, a classificação de risco da paciente quanto a sua taxa de sobrevida estimada.



Oportunidades:

1. Facilitar o uso do sistema construindo telas com poucos elementos.
2. Utilizar cores semânticas para aprimorar a experiência do usuário.
3. Elaborar formulários em etapas, respeitando a visibilidade do status do sistema.
4. Criar carregamentos fluidos e diminuir tempo de resposta do algoritmo.
5. Exibir resultados de maneira simples e de fácil entendimento.
6. Subdividir em quartis a expectativa de sobrevida do paciente para melhor divisão.
7. Receber feedbacks sobre a precisão do algoritmo segundo a opinião do usuário.
8. Permitir a impressão do resultado.

Responsabilidades:

1. Ao time de análise de dados: garantir uma boa assertividade do algoritmo por meio da atenuação de possíveis vieses (inerentes a existência da base de dados capturada em ambiente hospitalar de terceiro nível de complexidade).

4.2. Compreensão dos Dados

Os dados utilizados para a construção do modelo preditivo aqui disposto são oriundos, em sua maioria, dos prontuários eletrônicos de pacientes do Hospital das Clínicas da Faculdade de Medicina da Universidade de São Paulo (HC/FM-USP). Esses dados representam mulheres diagnosticadas com câncer de mama em diferentes estágios e quatro subtipos: Luminal-A, Luminal-B, HER-2 e TNBC (triplo negativo).

Os bancos de dados disponibilizados encontram-se em diversos formatos, dos quais se sobressaem o CSV (valores separados por vírgula) e XLSX (planilha padrão do excel). O principal dos bancos, possui 61684 tuplas e 104 colunas, com registros de cerca de 3769 pacientes únicos. São, a princípio, para a solução desenvolvida, conteúdos relevantes dessa base:

- Escolaridade e nível de educação;
- Raça;
- Histórico de gravidez e quantidade de partos e abortos;
- Histórico de amamentação e tempo de amamentação;
- Menarca;
- Índice de massa corpórea;
- Tempo de reposição hormonal;
- Histórico de câncer na família; e
- Tempo de sobrevida calculado.

Devido a natureza dispersa dos dados, por meio dos identificadores únicos de cada paciente, visando a construção de um fluxo de trabalho mais sólido e consistente, uma mesclagem viabilizada a partir de soluções automáticas de limpeza e tratamento de dados se faz necessária.

É entendível, também, o caráter enviesado das informações adquiridas, uma vez que o Hospital das Clínicas da Faculdade de Medicina da Universidade de São Paulo (HC/FM-USP) atua como agente médico-hospitalar de nível terciário de complexidade, recebendo pacientes que necessitam de tratamentos e terapias especializadas.

Como os dados fornecidos possuem problemas quanto a sua qualidade, cobertura e diversidade, tomam-se alguns riscos e contingências quanto a construção da solução, incluindo o problema de sobreajuste, que emerge-se como sendo a baixa capacidade de generalização do algoritmo devido a um super ajuste ao conjunto de dados de treinamento por falta ou excesso de informações.

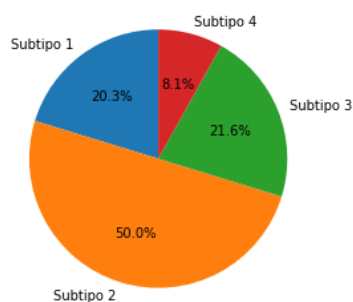
Tratando da existência de subconjuntos de dados, uma vez que o tamanho original da base de dados impossibilita a sua utilização completa em todas as etapas da definição do modelo, evidenciam-se alguns parâmetros principais, por ordem de prioridade, quais utilizados para uma

análise de primeira instância:

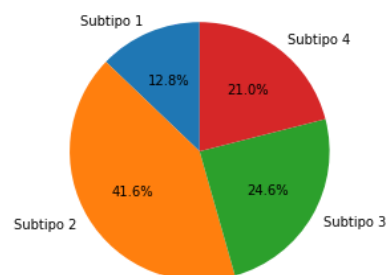
1. Subtipo;
2. Estadio;
3. Histórico de reposição hormonal;
4. Histórico de amamentação;
5. Histórico de gravidez; e
6. Resultados de hemogramas.

Como estudos iniciais, destacam-se os seguintes gráficos obtidos a partir de um algoritmo de construção estatística qualquer:

Mulheres que nunca ficaram grávidas e seus subtipos de câncer



Mulheres que já ficaram grávidas e seus subtipos de câncer



Anexo 7 - Gráficos - Ocorrência de tumores por subtipo.

Por fim, após o entendimento da solução e seu objetivo, define-se o atributo alvo (saída esperada) do modelo preditivo como sendo o índice de sobrevivência geral do paciente, de natureza binária, inicialmente dividido em baixo e alto, podendo, num futuro próximo, ser atualizado para gradientes escalares de quartis, possibilitando uma melhor acurácia e resposta.

4.3. Preparação dos Dados

4.3.1 Introdução

O presente tópico tem por objetivo viabilizar a criação de uma solução tecnológica capaz de analisar, filtrar e classificar dados de pacientes com câncer a fim de identificar a variabilidade da evolução da doença, bem como as possíveis respostas a tratamentos já implantados, para, assim, detectar padrões que indiquem aos profissionais de saúde uma possível estimativa de risco e grau de sobrevida de um paciente. As sessões seguintes são de caráter instrutivo, e resumem, passo a passo, como o tratamento dos dados, para processamento, foi realizado até então.

4.3.1.1 Configurações iniciais

Inicialmente, faz-se necessário, importar as bibliotecas necessárias à manipulação do banco de dados, tal qual:

```
# Importando as bibliotecas necessárias à manipulação do banco de dados.
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Definindo a quantidade máxima de colunas como indeterminada.
pd.set_option('display.max_columns', None)
```

Anexo 8 - Colab - Importação de bibliotecas.

Para que este notebook funcione corretamente, é necessário, após isso, fazer o upload do banco de dados nos arquivos do caderno. Observação: atente-se para que o arquivo esteja no formato .XLSX (EXCEL). Assim que fizer isso, digite o nome do arquivo na célula seguinte e a execute.

```
● Caminho do arquivo de banco de dados
file_path: "/BDIP.xlsx"
Mostrar código
```

Anexo 9 - Colab - Upload de banco de dados.

A fim de eliminar, logo de início, todas as colunas que não possuem ao menos um dado, executa-se:

```
# Limpando todas as colunas que possuem todos os seus valores vazios.
table.dropna(how='all', axis=1, inplace=True)
```

Anexo 10 - Colab - Remoção de colunas totalmente vazias.

E, com objetivo de validar se tudo ocorreu bem, exibe-se a tabela por meio do código abaixo:

```
# Imprimindo os quatro primeiros registros da tabela.
table.head()
```

Anexo 11 - Colab - Visualização primária da tabela.

4.3.2. Manipulação dos dados

4.3.2.1 Premissas de priorização, escolha de colunas de maior relevância e etiquetamento

Nome da coluna	Etiquetamento	Conteúdo
record_id	patient_id	número de identificação
dob	date_of_birth	data de nascimento
race	race	raça
pregnancy_history	has_children	histórico de gravidez (se teve filhos ou não)
partos	number_of_births	quantidade de partos concebidos
abortion	abortion	histórico abortivo (se sofreu abortos ou não)
breast_feeding	has_breastfed	histórico de amamentação (se já amamentou ou não)
breast_feeding_time	breastfed_time	período de tempo pelo qual amamentou (em meses)
menarche	menarche_age	menarca (idade, em anos, da primeira menstruação)
period	menopause_status	status da menopausa
bmi	body_mass_index	IMC medido (índice de massa corporal)
hormone_therapy	hormone_therapy_status	histórico de terapia hormonal (se fez terapia de reposição hormonal ou não)
tempo_rep_homo	hormone_therapy_time	tempo de terapia de reposição hormonal (em anos)
antec_fam_cancer_mama	has_breast_cancer_family_history	histórico de câncer de mama na família (se tem parentes com câncer ou não)
tobaco	tobaco	histórico de fumo (se fez o uso de tabaco ou não)
alcohol	alcohol	histórico de bebidas alcoólicas (se fez uso de bebida alcoólica ou não)
benign	is_benign	tipo do tumor (se é benigno ou maligno)
primary_digagnosis	primary_diagnostic	diagnóstico primário (tipo histológico)
diff_tubular	tubular_differentiation	diferenciação tubular ou grau arquitetural
mitotic_index	mitotic_index	índice mitótico (medida da rapidez com que as células cancerígenas crescem e se dividem).
histological_grade	histological_grade	grau histológico (o quão diferente a arquitetura do tecido tumoral está em relação ao tecido mamário normal)
grau_hist	histological_grade_based_on_nottingham	grau histológico baseado em Nottingham
tumor_subtype	tumor_subtype	subtipo tumoral
progesteron_percent	progesteron_percentage	receptor de progesterona (quantificação em %)
date_last_fu	date_of_last_patient_information	data da última informação sobre o paciente
follow_up_days_recidive	days_since_the_last_tumor	tempo desde o diagnóstico até a primeira recidiva
ultinfo	last_known_patient_status	última informação do paciente
recidive	recidive_status	histórico de recidiva (se sofreu recidiva ou não)
dtrecidiva	recidive_date	data de recidiva (se houver)
follow_up_days_recidive	days_until_recidive	tempo desde o diagnóstico até a primeira recidiva (em dias)
data_2	exam_date	data da consulta
weight	weight	peso
height	height	altura

Anexo 12 - Colab - Tabela de relações entre nome da coluna, etiquetamento e conteúdo.

Com o intuito de facilitar a construção de um modelo preditivo com assertividade relevante, algumas colunas de dados foram priorizadas em detrimento de outras. A premissa utilizada para a escolha dessas colunas baseia-se em dois principais motivos:

1. A relevância conhecida dos dados (o quanto as informações presentes nas colunas são importantes para a evolução da doença); e
2. A expressividade dos dados (quantas tuplas foram, para determinada coluna, preenchidas de maneira satisfatória).

Com base nisso, e com o objetivo de promover a compreensão total dos dados, foram realizados tratamentos de etiquetamento e dicionarização das colunas. Por etiquetamento entende-se modificações e ajustes nos nomes das colunas. Dito isso, exibe-se abaixo a relação das features adotadas e seus produtos:

Para cada coluna e feature selecionada, seguem as justificativas com base no que se sabe até então, respeitando as premissas supracitadas:

patient_id:

é necessário distinguir uma paciente de outra;

date_of_birth:

pode ser útil para identificar padrões relativos à idade;

race:

pode ser útil para distinguir uma possível reação diferente à doença;

has_children:

pode ser útil para entender a influência ou não da gravidez sobre o câncer;

number_of_births:

é necessário para avaliar o impacto da quantidade de partos e a progressão da doença;

abortion:

é necessário para avaliar o impacto entre abortos e a progressão do câncer;

has_breastfed:

pode ser útil para entender o impacto da amamentação e a progressão da doença;

breastfed_time:

é necessário para entender a influência do tempo de amamentação sobre o desenvolvimento do câncer;

menarche_age:

é necessário para entender a influência da idade da primeira menstruação sobre o desenvolvimento da doença;

menopause_status:

pode ser útil para entender se existe alguma relação entre o status da menopausa e o desenvolvimento do câncer;

body_mass_index:

pode ser útil para entender se existe alguma relação entre o índice de massa corporal e o desenvolvimento da doença;

hormone_therapy_status:

é necessário para entender a influência da terapia de reposição hormonal sobre o desenvolvimento do câncer;

hormone_therapy_time:

pode ser útil para entender se existe alguma relação entre o tempo de terapia de reposição hormonal e o desenvolvimento da doença;

has_breast_cancer_family_history:

pode ser útil para entender o impacto genético sobre o desenvolvimento do câncer;

tobaco:

é necessário para entender a influência do consumo de tabaco sobre o desenvolvimento da doença;

alcohol:

é necessário para entender a influência do consumo de álcool sobre o desenvolvimento do câncer;

is_benign:

é necessário para entender a correlação entre pessoas com tumores benignos ou malignos e o desenvolvimento da doença;

primary_diagnostic:

é necessário para entender a correlação entre o primeiro diagnóstico dado e o desenvolvimento do câncer;

tubular_differentiation:

é necessário para entender características biomorfológicas do câncer;

mitotic_index:

é necessário para entender características biomorfológicas do câncer;

histological_grade:

é necessário para entender características biomorfológicas do câncer;

histological_grade_based_on_nottingham:

é necessário para entender características biomorfológicas do câncer;

tumor_subtype:

é necessário para entender a correlação existente entre o subtipo do câncer a evolução da doença;

progesteron_percent:

é necessário para entender a correlação entre os índices de receptores de progesterona no corpo e a evolução do câncer;

date_of_last_patient_information:

pode ser útil para entender a validade da última informação do paciente conhecida;

days_since_the_last_tumor:

pode ser útil para estabelecer a correlação entre o diagnóstico e status do câncer com uma possível recidiva;

last_known_patient_status:

é necessário para caracterizar o último estado conhecido do paciente;

recidive_status:

é necessário para estabelecer a correlação entre uma possível recidiva e a evolução da doença;

recidive_date:

é necessário para estabelecer um período médio de tempo até uma possível recidiva;

days_until_recidive:

é necessário para estabelecer um período médio de tempo até uma possível recidiva;

exam_date:

é necessário para criar correlações baseadas em tempo da evolução do câncer;

weight:

pode ser útil para estabelecer possíveis correlações entre o peso e a evolução da doença;

height:

pode ser útil para estabelecer possíveis correlações entre a altura e a evolução do câncer;

4.3.2.2 Selecionando apenas as colunas desejadas

A princípio, precisávamos remover colunas que estavam com poucos ou nenhum dado preenchido, assim, após uma série de análises e discussões entre os integrantes, com os devidos comandos, separamos do banco as colunas relevantes e as renomeamos. Com isso, obtivemos uma melhor visualização dos dados que seriam utilizados nas próximas etapas. Dadas as colunas expostas no tópico anterior, cria-se um conjunto de dados com apenas as features desejadas.

```
# Sobrescrevendo a tabela com apenas as colunas desejadas.
table = table[['record_id', 'dob',
'race', 'pregnancy_history', 'partos', 'abortion', 'breast_feeding', 'breast_feedi
ng_time', 'menarche', 'period', 'bmi', 'hormone_therapy', 'tempo_rep_hormo', 'antec
_fam_cancer_mama', 'tobaco', 'alcohol', 'benign', 'primary_diganosis', 'diff_tubul
ar', 'mitotic_index', 'histological_grade', 'grau_hist', 'tumor_subtype', 'progest
eron_perct', 'date_last_fu', 'follow_up_days', 'ultinfo', 'recidive', 'dtrecidiva'
, 'follow_up_days_recidive', 'data_2', 'weight', 'height']].copy()

# Renomeando as colunas de acordo com o etiquetamento feito.
table = table.rename(columns={'record_id': 'patient_id',
'dob': 'date_of_birth',
'race': 'race',
'pregnancy_history': 'has_children',
'partos': 'number_of_births',
'abortion': 'abortion',
'breast_feeding': 'has_breastfed',
'breast_feeding_time': 'breastfed_time',
'menarche': 'menarche_age',
'period': 'menopause_status',
'bmi': 'body_mass_index',
'hormone_therapy': 'hormone_therapy_status',
'tempo_rep_hormo': 'hormone_therapy_time',
'histological_grade': 'histological_grade',
'tumor_subtype': 'tumor_subtype',
'recidive': 'ricidive_status',
'dtrecidiva': 'recidive_date',
'data_2': 'exam_date',
'weight': 'weight',
'height': 'height'})
```

4.3.2.3 Criando uma lista com pacientes únicos

Dado o escopo da aplicação, faz-se necessário, visando realizar manipulações futuras, por meio de números de identificações (ID's), obter uma lista com todos os pacientes únicos.

```
# Criando um vetor com todos os números de identificação únicos de cada
paciente.
unique_patient_ids = table['patient_id'].unique();

# Imprimindo a lista gerada.
unique_patient_ids
```

Anexo 14 - Colab - Lista de pacientes únicos

```
# Criando uma coluna para armazenar a idade em meses.
table['age_when_cancer_was_discovered_in_months'] = np.nan

# Realizando um loop para cada paciente único existente.
for id in unique_patient_ids:

    # Criando uma tabela temporária para apenas um paciente.
    id_table = table.loc[table["patient_id"] == id]

    # Pegando o valor da data de nascimento e excluindo valores nulos.
    date_of_birth_id = id_table.date_of_birth.values
    date_of_birth_id = [x for x in date_of_birth_id if pd.isnull(x) == False]

    # Pegando as datas de exames e excluindo valores nulos.
    data_id = id_table.exam_date.values
    data_id = [x for x in data_id if pd.isnull(x) == False]

    if data_id != [] and date_of_birth_id != []:

        # Pegando apenas um valor da lista para a data de nascimento
        date_of_birth_id = date_of_birth_id[0]

        # Pegando a primeira data de consulta
        min_date = min(data_id)
```

Anexo 15 - Colab - Instanciando idade em meses - Parte 1

```
# Achando o número de dias entre a data de nascimento e a primeira consulta
date_delta = min_date - date_of_birth_id
date_delta = date_delta / np.timedelta64(1, 'D')

# Associando o paciente à sua idade quando o câncer foi descoberto em anos.
table.loc[table['patient_id'] == id, 'age_when_cancer_was_discovered_in_months'] =
int(date_delta/30)

# Criando uma coluna chamada time_with_tumor_delta
table['time_with_tumor_delta'] = np.nan

for id in unique_patient_ids:

    # Criando uma tabela temporária para apenas um paciente
    id_table = table.loc[table['patient_id'] == id]

    # Pegando as datas de exame e tirando valores nulos
    data_id = id_table.exam_date.values
    data_id = [x for x in data_id if pd.isnull(x) == False]

    if data_id != []:

        # Pegando a primeira data de consulta
        min_date = min(data_id)

        # Pegando a data de óbito
        date_last_fu = id_table.date_of_last_patient_information.values
        date_last_fu = [x for x in date_last_fu if pd.isnull(x) == False]

        if date_last_fu != []:

            max_date = max(date_last_fu)

            # Achando o número de dias entre a primeira consulta e a data de óbito
            date_delta = max_date - min_date
            date_delta = date_delta / np.timedelta64(1, 'D')

            # Colocando em todas as linhas do paciente
            userindex = table.index[table['patient_id'] == id]

            for index in userindex:

                table.loc[userindex, ["time_with_tumor_delta"]] = date_delta
```

4.3.2.4 Criação de colunas relevantes e seu preenchimento

Acreditamos que seria relevante para a aplicação a idade de descoberta do câncer do paciente e o tempo que ele teve de se submeter a algum tratamento:

```
# Criando uma coluna para armazenar a idade em meses.
table['age_when_cancer_was_discovered_in_months'] = np.nan

# Realizando um loop para cada paciente único existente.
for id in unique_patient_ids:

    # Criando uma tabela temporária para apenas um paciente.
    id_table = table.loc[table["patient_id"] == id]

    # Pegando o valor da data de nascimento e excluindo valores nulos.
    date_of_birth_id = id_table.date_of_birth.values
    date_of_birth_id = [x for x in date_of_birth_id if pd.isnull(x) == False]

    # Pegando as datas de exames e excluindo valores nulos.
    data_id = id_table.exam_date.values
    data_id = [x for x in data_id if pd.isnull(x) == False]

    if data_id != [] and date_of_birth_id != []:

        # Pegando apenas um valor da lista para a data de nascimento
        date_of_birth_id = date_of_birth_id[0]

        # Pegando a primeira data de consulta
        min_date = min(data_id)

        # Achando o número de dias entre a data de nascimento e a primeira consulta
        date_delta = min_date - date_of_birth_id
        date_delta = date_delta / np.timedelta64(1, 'D')

        # Associando o paciente à sua idade quando o câncer foi descoberto em anos.
        table.loc[table['patient_id'] == id, 'age_when_cancer_was_discovered_in_months'] =
int(date_delta/30)
```



```
# Criando uma coluna chamada time_with_tumor_delta
table['time_with_tumor_delta'] = np.nan

for id in unique_patient_ids:

    # Criando uma tabela temporária para apenas um paciente
    id_table = table.loc[table['patient_id'] == id]

    # Pegando as datas de exame e tirando valores nulos
    data_id = id_table.exam_date.values
    data_id = [x for x in data_id if pd.isnull(x) == False]

    if data_id != []:

        # Pegando a primeira data de consulta
        min_date = min(data_id)

        # Pegando a data de óbito
        date_last_fu = id_table.date_of_last_patient_information.values
        date_last_fu = [x for x in date_last_fu if pd.isnull(x) == False]

        if date_last_fu != []:

            max_date = max(date_last_fu)

            # Achando o número de dias entre a primeira consulta e a data de óbito
            date_delta = max_date - min_date
            date_delta = date_delta / np.timedelta64(1, 'D')

            # Colocando em todas as linhas do paciente
            userindex = table.index[table['patient_id'] == id]

            for index in userindex:

                table.loc[userindex, ["time_with_tumor_delta"]] = date_delta
```

4.3.2.5 Trabalhando dados dispersos em um período de tempo de avaliação do paciente

A fim de efetivar e facilitar o uso dos dados na aplicação, resumimos todos os respectivos dias de análise e seus resultados relevantes de cada paciente, em uma única linha, para tal serão utilizados os conceitos de média, moda e mediana. Com isso, entendemos que conseguimos manter as relações relevantes do banco, em conjunto com uma maior facilidade de fornecê-las aos modelos.

```
# Colunas alvo para a minificação através de média.
columns = ['body_mass_index', 'weight', 'height']
# Realizando um loop para cada paciente.
for id in unique_patient_ids:
    # Criando uma tabela temporária para apenas um paciente.
    id_table = table.loc[table['patient_id'] == id]
    # Coletando os index referentes a aquele paciente no banco.
    userindex = table.index[table['patient_id'] == id]
    # Começando um looping para cada coluna selecionada.
    id_table.replace([np.inf, -np.inf], np.nan, inplace=True)
    for column in columns:
        sum = 0
        count = 0
        # Criando uma lista com todos os valores daquela coluna, excluindo nulos, nans, infs, e zeros.
        values_list = id_table[column].dropna().to_list()
        clean_values_list = [x for x in values_list if pd.isnull(x) == False or x != 0]
        np.array(clean_values_list, dtype=float)
        # Looping para soma de valores e cálculo de média
        for value in values_list:
            value = float(value)
            sum += value
            count += 1
        if values_list != []:
            mean_value = (sum/count)
            mean_format = "{:.2f}".format(mean_value)
            # Inserção dos valores encontrados na tabela
            table.loc[userindex[0], [column]] = float(mean_format)
        else:
            table.loc[userindex[0], [column]] = 0
```

Anexo 18 - Colab - Criando e preenchendo colunas importantes

Com essa lógica, conseguimos alcançar nosso objetivo de, para cada paciente, reduzir seus dados recorrentes a um único, considerando os valores todos os que foram previamente preenchidos no banco.

4.3.2.6 Definindo às funções necessárias para a manipulação corrente

Nessa parte, funções foram desenvolvidas para verificar se os valores das colunas são constantes ou variáveis (“are_all_values_constants()”, que recebe um array de valores não nulos, que são previamente segregados pela função “is_null()”). Segue o código com as funções detentoras das lógicas supracitadas:

```
# Instanciando uma função para verificar se um valor é nulo ou não.
def is_null(value):

    if(pd.isnull(value) == True):

        # Retornando verdadeiro se o valor é nulo.
        return True;

    else:

        # Retornando falso se o valor não é nulo.
        return False;

# Instanciando uma função para verificar se todos os valores de um vetor são constantes.
def are_all_values_constants(values):

    # Partindo do pressuposto que todos os valores são iguais.
    are_all_values_constants = True;

    # Para cada posição no vetor de valores, realizando um teste de hipótese com fim de negar que todos os
    # valores são constantes.
    for index in range(len(values)):

        # Se o índice for 0, continuamos com o loop.
        if(index == 0):

            continue;

        else:

            # Se não, verificando se o valor do índice atual é diferente do índice anterior.
            if(values[index-1] != values[index]):

                # Caso verdadeiro, negando a proposição inicial.
                are_all_values_constants = False;

                # Parando o loop uma vez que a proposição foi negada.
                break;

    # Retornando se os valores são constantes ou não.
    return are_all_values_constants;
```

4.3.2.6.2 Aplicação das funções criadas em todas as colunas, para cada paciente respectivamente

Depois de consolidarmos as lógicas acima, segue o código necessário para além de utilizá-las devidamente, adaptá-las às condições do banco disponibilizado. Caso os valores não-nulos de uma determinada coluna sejam constantes, o valor é replicado para a primeira linha do paciente em questão:

```
'''
Partindo do pressuposto que já possuímos acesso à
lista de pacientes com os seus números de identificação
únicos, captam-se as tuplas de todos os pacientes de maneira individual, tal qual:
'''
for column in list(table.columns.values):
    for patient in unique_patient_ids:
        # Instanciando um contador.
        i = 0
        # Criando uma tabela com somente os dados do paciente corrente.
        unique_patient_rows_table = table[table['patient_id'] == patient]
        # Selecionando, inicialmente, todos os valores da coluna de primary_diagnostic, incluindo os nulos.
        all_values = unique_patient_rows_table[column].values
        # Instanciando uma lista de valores não nulos.
        not_null_values = [];
        for value in all_values:
            if(is_null(value)):
                continue;
            else:
                not_null_values.append(value);
        for value in not_null_values:
            if type(value) == str:
                value = value.lower()
                if value == 'sim':
                    not_null_values[i] = 1
                    i = i + 1
                elif value in ['não', 'nao', 'neg', 'inconclusivo', ]:
                    not_null_values[i] = 0
                    i = i + 1
                else:
                    not_null_values[i] = 0
                    i = i + 1
            elif type(value) == float:
                not_null_values[i] = float("{:.2f}".format(value))
                i = i + 1
            elif isinstance(value, np.floating) == True:
                not_null_values[i] = np.around(value, 2)
                i = i + 1
        # Se todos os valores não nulos forem constantes, aplicando o seguinte algoritmo...
        if(are_all_values_constants(not_null_values)):
            found_value = None;
            if(len(not_null_values) == 0):
                # Definindo o valor para a primeira linha do paciente como 0, uma vez que não fora encontrado nenhum valor.
                found_value = 0;
            else:
                # Definindo o valor para a primeira linha do paciente como o primeiro valor constante encontrado.
                found_value = not_null_values[0];
            table.loc[(table.index[table['patient_id'] == patient])[0], column] = found_value;
```

4.3.2.7 Criando uma tabela com apenas a primeira linha de cada paciente

Aqui definimos como “final_table” a tabela final e delimitamos seu conteúdo:

```
# Instanciando a tabela final.
final_table = pd.DataFrame()

# Realizando um loop para cada paciente
for patient in unique_patient_ids:

    # Criando uma tabela com somente os dados do paciente corrente.
    unique_patient_rows_table = table[table['patient_id'] == patient]

    # Definindo o conteúdo da tabela final.
    final_table = final_table.append(unique_patient_rows_table.iloc[0], True)

# Exibindo a tabela final.
display(final_table)
```

Anexo 21 - Colab - Criação de uma tabela com somente uma linha para cada paciente.

4.3.7.1 Removendo colunas irrelevantes ou que poderiam causar problemas no treinamento do modelo

Primeiramente foram removidas as colunas que continham datas, já que estas são inúteis por seu formato ao modelo:

```
final_table.drop(columns=['date_of_birth',
'date_of_last_patient_information', 'recidive_date', 'exam_date'],
inplace=True)
```

Anexo 22 - Colab - Remoção de colunas irrelevantes ou de difícil manipulação pelo algoritmo.

```
final_table.fillna(0, inplace=True)
```

Anexo 23 - Colab - Remoção de valores nulos.

Depois colunas que são irrelevantes à modelagem ou que poderiam causar vieses:

```
final_table.drop(columns=['height', 'last_known_patient_status', 'race', 'weight'], inplace=True)
```

Anexo 24- Colab - Remoção de colunas que poderiam ocasionar em possíveis vieses.

4.3.2.7.2 Criando, definindo e classificando a coluna do “overall_survive”

Tendo em vista o objetivo da aplicação, decidimos criar a coluna que deveria ser predita. Para tal, nos baseamos no maior eo menor tempo de vida após a descoberta do câncer dos pacientes disponíveis, e a partir disso definimos que os pacientes seriam classificados entre 4 possibilidades, cada uma sendo um quartis de tempo em ordem crescente, logo 1 seriam pacientes com menor tempo de sobrevida e 4 com maior.

```
# Instanciando uma variável com somente a coluna de dias desde o último tumor.
days_since_the_last_tumor = final_table.days_since_the_last_tumor.values
# Instanciando uma nova coluna na tabela final.
final_table['overall_survive'] = np.nan
# Estabelecendo o menor dia de tempo de sobrevida.
lower_day = min(days_since_the_last_tumor)
# Estabelecendo o maior dia de tempo de sobrevida.
max_day = max(days_since_the_last_tumor)
# Estabelecendo o quartil.
quartile = (max_day - lower_day)/4
# Estabelecendo o teto máximo do primeiro quartil.
first_quartile = lower_day + quartile
# Estabelecendo o teto máximo do segundo quartil.
second_quartile = lower_day + (2*quartile)
# Estabelecendo o teto máximo do terceiro quartil.
third_quartile = lower_day + (3*quartile)
# Instanciando um índice para associar os dados da coluna com um determinado paciente.
i = 0
# Para cada dia presente na coluna de dias desde o último tumor, realizando um loop.
for day in days_since_the_last_tumor:
    # Definindo o quartil com base no dia corrente.
    if day == 0:
        final_table.loc[i, 'overall_survive'] = -1
    if day <= first_quartile:
        final_table.loc[i, 'overall_survive'] = 1
    elif day <= second_quartile:
        final_table.loc[i, 'overall_survive'] = 2
    elif day <= third_quartile:
        final_table.loc[i, 'overall_survive'] = 3
    elif day > third_quartile:
        final_table.loc[i, 'overall_survive'] = 4
    else:
        final_table.loc[i, 'overall_survive'] = -1
    # Próximo dia.
    i += 1
```

Anexo 25 - Colab - Criando a coluna de atributo alvo.

Por fim, depois de aplicar a lógica para a criação da coluna “overall_survive” pudemos remover a propriedade de dias desde o último tumor (que já havia sido utilizada) e dias até a recidiva, algo que evitaria o modelo de dar muita relevância à segunda, uma vez que, na maioria dos casos, é um fator definitivo para a situação do paciente. Portanto, faz-se necessário sua retirada, a fim de analisar casos sem esse viés.

```
final_table.drop(columns=['days_since_the_last_tumor', 'days_until_recidive'],  
inplace=True)
```

Anexo 26 - Colab - Dropando colunas redundantes.

4.4. Modelagem

Para a aplicação do aprendizado de máquina, utilizamos os seguintes métodos: Árvore de Decisão, KNN, Redes Neurais e SVM. Estes algoritmos são considerados os melhores no quesito aprendizagem supervisionada, oferecendo precisão, estabilidade e mais facilidade na interpretação, que são atributos que prezamos na nossa aplicação. Para avaliação, definimos primeiramente a matriz de confusão, para visualização geral da apresentação dos dados dispersos após a tentativa de predição e em sequência calcularemos a acurácia, precisão, erro quadrático e recall de cada modelo, a fim de considerarmos melhor nossos resultados, já que seria difícil uma precisão muito alta no contexto do projeto.

A fim de melhorar as predições das informações coletadas, utilizamos a função `RamdomizedSearchCV`. Essa função faz uma quantidade específica de combinações aleatórias, de acordo com os hiperparâmetros definidos. Os hiperparâmetros definidos para os algoritmos foram pensados de forma estratégica para cada um dos modelos, de modo que prevenisse que existisse uma aprendizagem apenas com os dados mostrados, evitando overfitting e underfitting, tornando-o capaz de generalizar para outras situações possíveis. Além disso, também contribui para o aperfeiçoamento das métricas de avaliação para classificação. Fizemos o treinamento de todos os métodos citados acima, com e sem o uso da função, para mostrar na prática a melhoria deles.

A princípio instalamos e importamos as bibliotecas necessárias para a construção de todos os nossos modelos:

```
# Importando todas as bibliotecas necessárias ao treinamento dos modelos.
from sklearn.svm import SVC
from sklearn.tree import plot_tree
from sklearn.tree import export_graphviz
from sklearn.pipeline import make_pipeline
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import plot_confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import accuracy_score, mean_squared_error, precision_score, recall_score

# Limpando o console de saída.
clear_output()
```


4.4.1 Árvore de decisão

A árvore de decisão funciona para variáveis categóricas e contínuas de entrada e de saída. Nela, repartem-se os dados em grupos cada vez menores e mais específicos, até atingirem um certo lugar para ser rotulado. De início, importamos as bibliotecas necessárias e criamos um dataframe somente com os dados de entrada (todas as colunas) e outro apenas com o atributo alvo ("overall_survive"). Dividimos os dados entre treino e teste e definimos o tamanho dos dados de teste como 30%. Pode-se citar, ainda, o fato de termos não só instanciado e ajustado o modelo, como também associado o resultado dos testes à variável "y_pred".

```
# Criando um dataframe com somente os dados de entrada.
x = final_table.loc[:, final_table.columns!='overall_survive']

# Criando um dataframe com somente o atributo alvo.
y = final_table.loc[:, final_table.columns=='overall_survive']

# Dividindo os dados entre treino e teste e definindo o tamanho dos dados de teste como 30%.
x_train, x_test, y_train, y_test = train_test_split(x, y, stratify=y, test_size=0.3,
random_state=42)

# Instanciando o modelo.
model_no_tuning = DecisionTreeClassifier()

# Treinando o modelo de busca
model_no_tuning.fit(x_train, y_train)

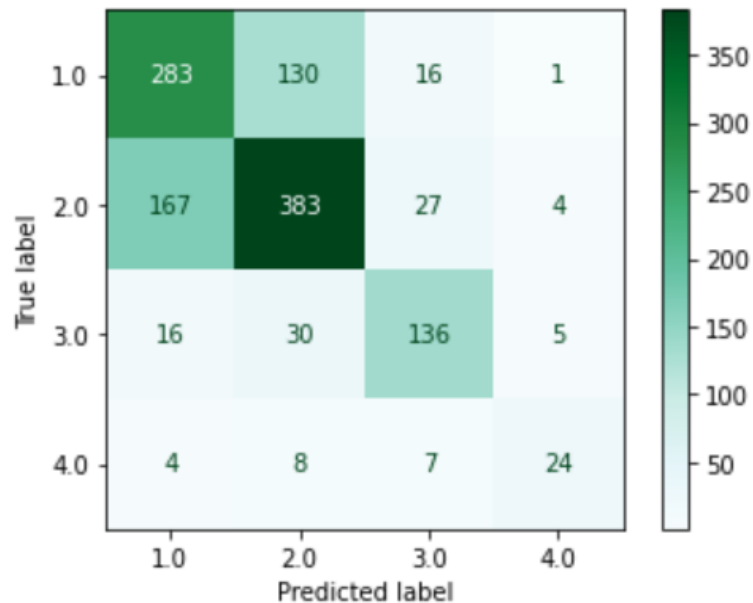
# Associando resultado.
y_pred_no_tuning = model_no_tuning.predict(x_test)
```

Anexo 28 - Colab - Treinamento de modelo sem tunagem de hiperparâmetros.

Para a avaliação do nosso modelo de classificação, utilizamos a Matriz de Confusão, uma tabela que mostra as frequências de classificação para cada classe do modelo:

```
# Plotando a matriz de confusão.
print("Segue a Matriz de Confusão: ")
_ = plot_confusion_matrix(model_no_tuning, x_test, y_test, cmap='BuGn')
```

Anexo 29 - Colab - Construção da matriz de confusão.



Anexo 30 - Colab - Matriz de Confusão sem tunagem de hiperparâmetros

```
# Definindo os parâmetros para o randomSearch
params = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'max_depth': range(1,10),
    'max_features': ['auto', 'sqrt', 'log2'],
    'random_state': range(1,50)
}
# Instanciando o modelo.
modelDT = DecisionTreeClassifier()
# Instanciando o modelo de busca
searchDT = RandomizedSearchCV(
    estimator = modelDT,
    param_distributions = params
)
# Treinando o modelo de busca
modelTraining = searchDT.fit(x_train, y_train)
# Associando resultado.
y_pred = modelTraining.predict(x_test)
# Limpando o console de saída.
clear_output()
# Mostrando os melhores parâmetros
modelTraining.best_params_
```

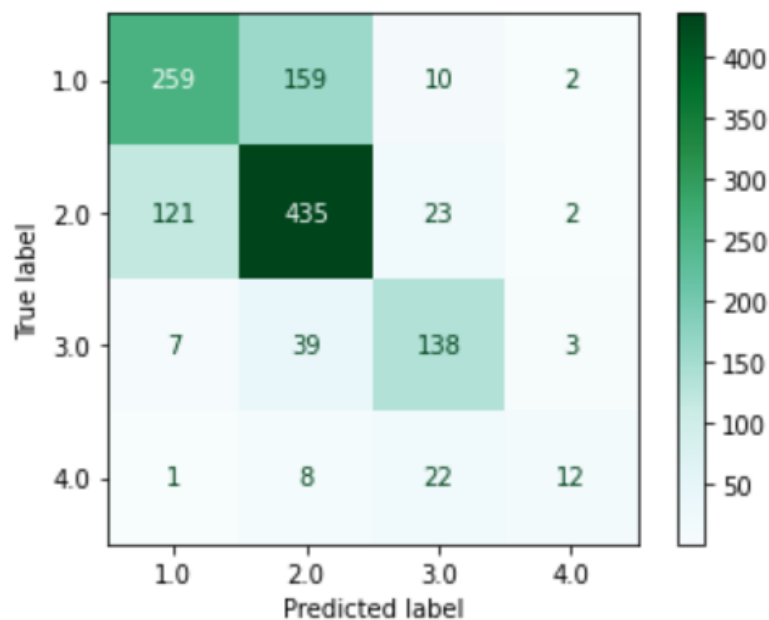
Anexo 31 - Colab - Criação de um novo modelo utilizando Random Search

```
{
  'random_state': 29,
  'max_features': 'log2',
  'max_depth': 7,
  'criterion': 'gini'
}
```

Anexo 32 - Colab - Output do anexo anterior.

```
# Plotando a matriz de confusão.
print("Segue a Matriz de Confusão: ")
_ = plot_confusion_matrix(model, x_test, y_test, cmap='BuGn')
```

Anexo 33 - Colab - Plotando a matriz de confusão



Anexo 34 - Colab - Matriz de confusão com tunagem de hiperparâmetros

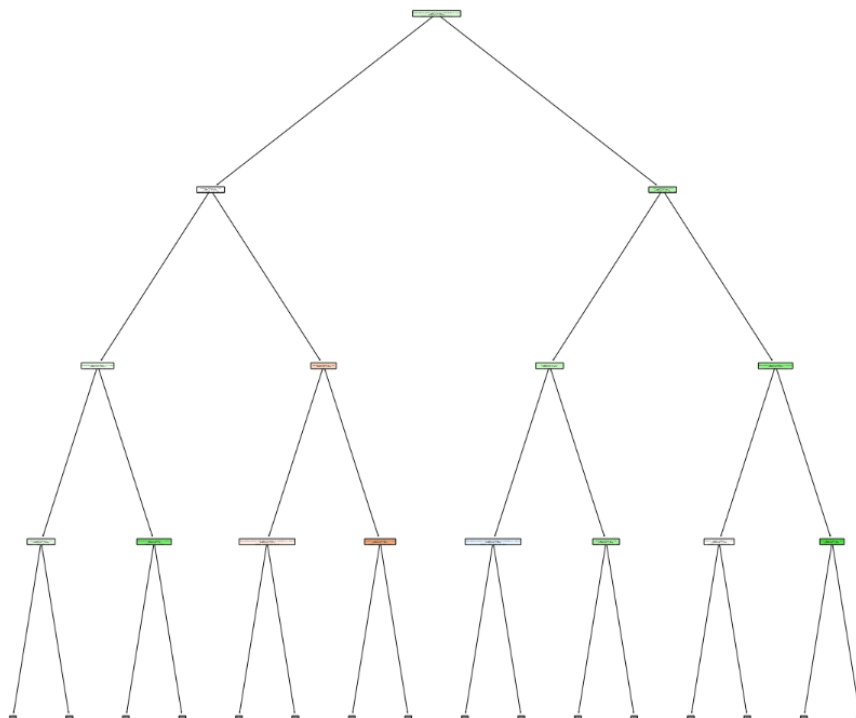
```
# Importando a biblioteca necessária.
from sklearn.tree import plot_tree

# Ajustando as configurações da figura.
fig, ax = plt.subplots(figsize=(20, 20))

# Gerando a imagem.
_ = plot_tree(model.best_estimator_,
              feature_names=x.columns,
              class_names=None,
              filled=True, rounded=True, ax=ax,
              max_depth = 3)

# Salvando a figura gerada.
fig.savefig('output.pdf')
```

Anexo 35 - Colab - Criação da árvore de decisão



Anexo 36 - Colab - Árvore de decisão

4.4.3 KNN

O algoritmo KNN (K-Nearest Neighbor) é um dos algoritmos mais utilizados em Machine Learning, o destaque do modelo é a possibilidade de utilização do mesmo tanto para classificação quanto para regressão. Usamos para determinar o rótulo de classificação de uma amostra baseado nas amostras vizinhas advindas de um conjunto de treinamento.

Para que o algoritmo consiga dividir o dataset com base nos dados fornecidos e prever a característica desejada para um item, o modelo deve seguir 3 passos essenciais:

- Calcular a distância entre o item a ser classificado e todos os outros itens do dataset de treino
- Identificar os K itens mais próximos do item a ser classificado.
- Calcular a moda dentre as características (característica mais presente entre os K itens). Essa será a classificação final dada para o item com característica desconhecida.

Separando a tabela para treino e qual será predita, em conjunto com o código para treinar o modelo:

```
# Criando um dataframe com somente os dados de entrada.
xKNN = final_table.loc[:, final_table.columns!='overall_survive']
# Criando um dataframe com somente o atributo alvo.
yKNN = final_table.loc[:, final_table.columns=='overall_survive']
```

Anexo 37 - Colab - separação de tabela de treino

```
# Dividindo os dados entre treino e teste e definindo o tamanho dos dados de teste como 30%.
x_train_KNN, x_test_KNN, y_train_KNN, y_test_KNN = train_test_split(xKNN, yKNN, test_size=0.3,
random_state=52)

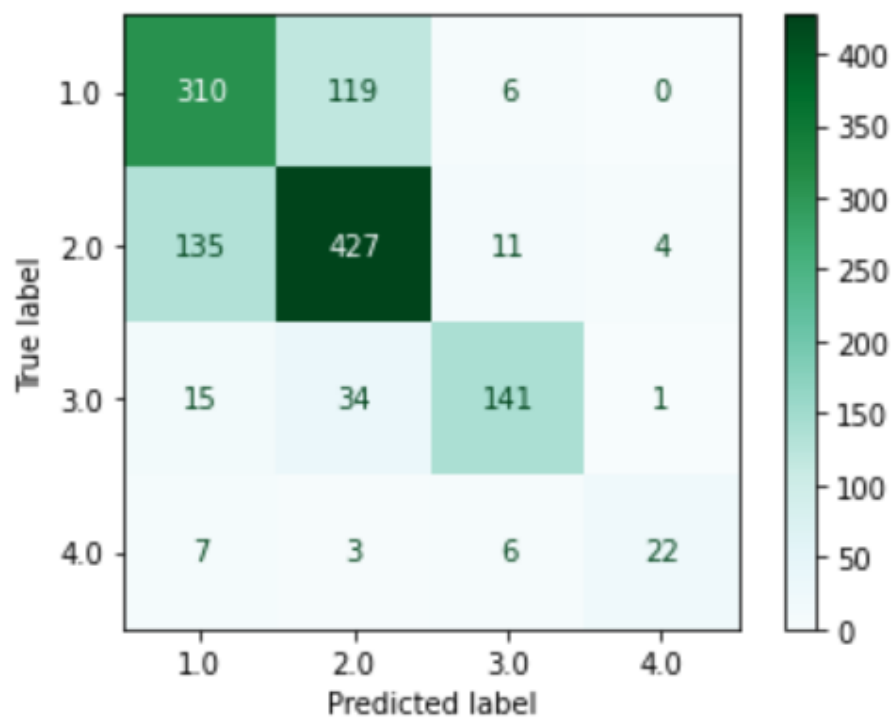
# Instanciando o modelo.
kNN = KNeighborsClassifier()
# Treinando o modelo de busca
kNN.fit(x_train_KNN, y_train_KNN)
# Associando resultado.
y_pred_KNN = kNN.predict(x_test_KNN)
# Limpando o console de saída.
clear_output()
```

Anexo 38 - Colab - construção do modelo sem tunagem de hiperparâmetros

```
# Imprimindo mensagem na tela.
print("Segue a Matriz de Confusão: ")

# Plotando a matriz de confusão.
_ = plot_confusion_matrix(kNN, x_test_KNN, y_test_KNN, cmap='BuGn')
```

Anexo 39 - Colab - plotando a matriz de confusão .



Anexo 40 - Colab - matriz de confusão sem tunagem de hiperparâmetros.

Para a melhoria do modelo, construímos ele novamente, provendo testes para encontrar os melhores parâmetros:

```
# Definindo os parâmetros para o randomSearch
params = {
    'n_neighbors': range(5,35),
    'weights': ['uniform', 'distance'],
}

# Instanciando o modelo de busca

search = RandomizedSearchCV(
    estimator = kNN,
    param_distributions = params
)

# Treinando o modelo de busca
kNN_model1 = search.fit(x_train_KNN, y_train_KNN)

clear_output()

# Associando resultado.
y_pred_KNN = kNN_model1.predict(x_test_KNN)

# Mostrando os melhores parâmetros
kNN_model1.best_params_
```

Anexo 41 - Colab - adição de melhores parâmetros.

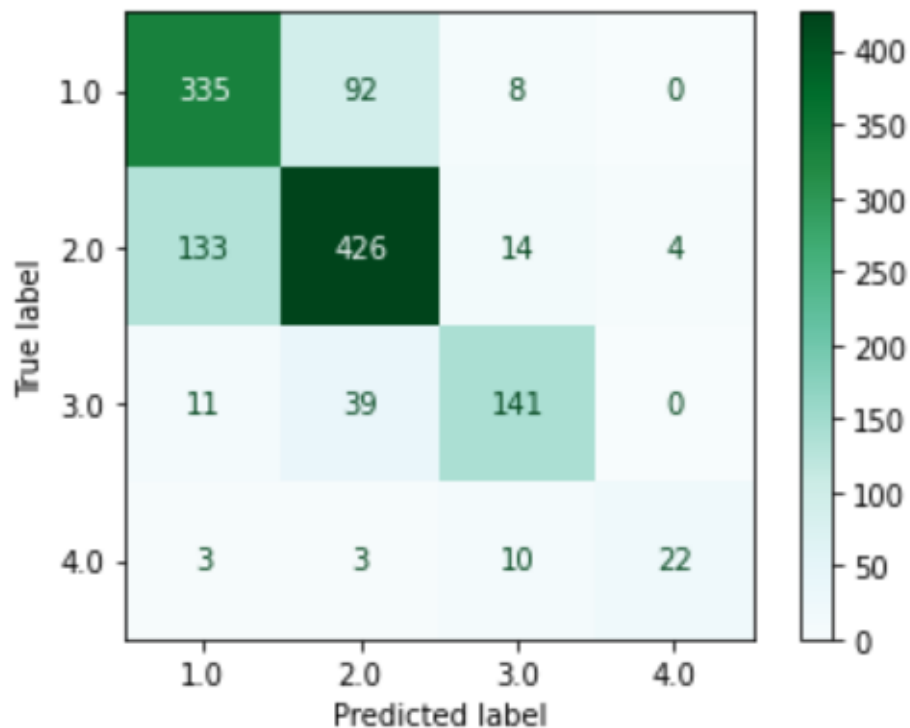
```
{'weights': 'uniform', 'n_neighbors': 23}
```

Anexo 42 - Colab - output do anexo anterior.

Para a avaliação do modelo com a adição dos hiperparâmetros, utilizamos novamente a Matriz de confusão:

```
# Plotando a matriz de confusão.
print("Segue a Matriz de Confusão: ")
_ = plot_confusion_matrix(kNN_model1, x_test_KNN, y_test_KNN, cmap='BuGn')
```

Anexo 43 - Colab - colando a matriz de confusão



Anexo 44 - Colab - Matriz de confusão com tunagem de hiperparâmetros

4.4.4 Redes Neurais

Redes neurais é um método de inteligência artificial que ensina computadores a processar dados de uma forma inspirada pelo cérebro humano. É um tipo de processo de machine learning, chamado aprendizado profundo, que usa nós ou neurônios interconectados em uma estrutura em camadas, semelhante ao cérebro humano.

```
# Criando um dataframe com somente os dados de entrada.
x_neural_network = final_table.loc[:, final_table.columns!='overall_survive']

# Criando um dataframe com somente o atributo alvo.
y_neural_network = final_table.loc[:, final_table.columns=='overall_survive']
```

Anexo 45 - Colab - Criação de data frame

Para a construção do modelo, definimos o tamanho dos dados de teste como 30% e o random_state como 52:

```
# Dividindo os dados entre treino e teste e definindo o tamanho dos dados de teste
como 30%.
x_train_NN, x_test_NN, y_train_NN, y_test_NN = train_test_split(x_neural_network,
y_neural_network, test_size=0.3, random_state=52)

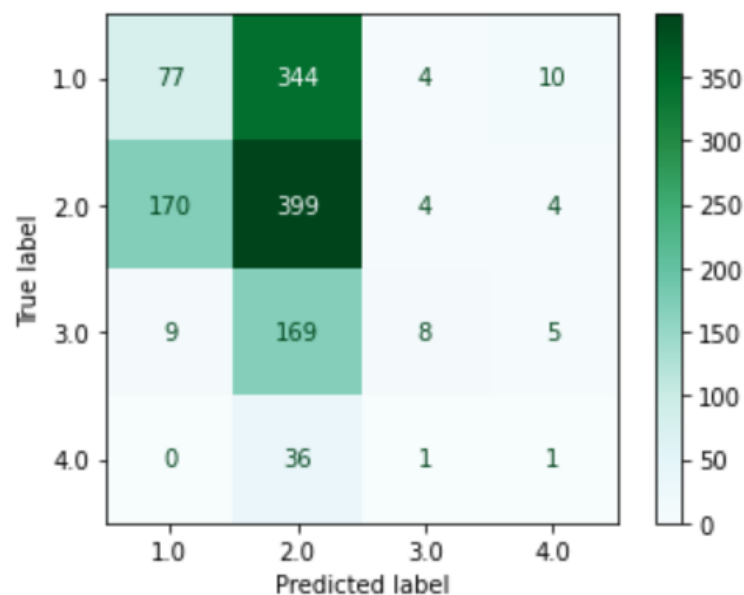
# Instanciando o modelo.
NN = MLPClassifier()

# Treinando o modelo de busca
NN_model = NN.fit(x_train_NN, y_train_NN)

# Associando resultado.
y_pred_NN = NN_model.predict(x_test_NN)

# Limpando o console de saída.
clear_output()
```

Anexo 46 - Colab - definição dos dados de teste.



Anexo 47 - Colab - matriz de confusão sem tunagem de hiperparâmetros.

Com o objetivo de aprimorar a qualidade dos resultados, construímos o modelo com os melhores parâmetros:

```
# Colocando parâmetros
params = {
    "hidden_layer_sizes": range(2, 15, 10),
    'max_iter': range(350, 1000, 200)
}

# Instanciando o modelo de busca
search = RandomizedSearchCV(
    estimator = NN,
    param_distributions = params
)

# Treinando o modelo de busca
NN_model1 = search.fit(x_train_NN, y_train_NN)

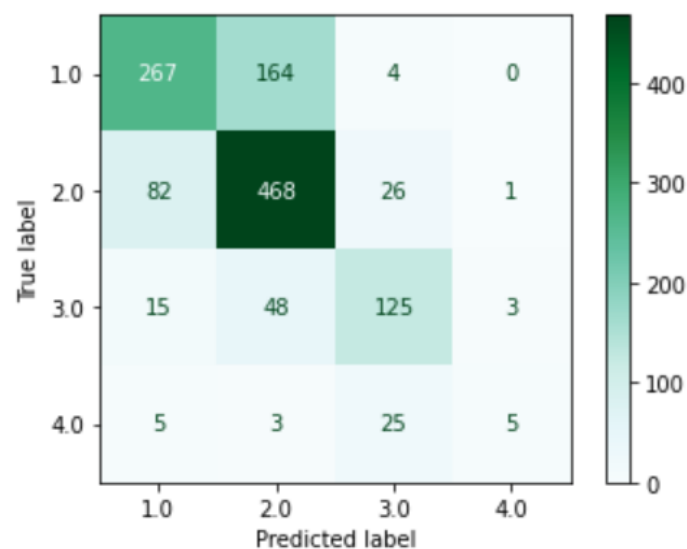
# Associando resultado.
y_pred_NN = NN_model1.predict(x_test_NN)
```

Anexo 48 - Colab - criação de um novo modelo utilizando Random Search.

Avaliamos novamente o modelo, usando a Matriz de Confusão:

```
# Matriz de Confusão do modelo
NNMatrix = plot_confusion_matrix(NN_model1, x_test_NN, y_test_NN, cmap='BuGn')
```

Anexo 49 - Colab - Plotando a matriz de confusão.



Anexo 50 - Colab - matriz de confusão com tunagem de hiperparâmetros.

4.4.5 SVM

O SVM é um algoritmo que busca uma linha de separação entre duas classes distintas, analisando os dois pontos, um de cada grupo, mais próximos da outra classe. Isso significa que o SVM escolhe a reta entre dois grupos que se distanciam mais de cada um.

```
# Criando um dataframe com somente os dados de entrada.
x_SVC = final_table.loc[:, final_table.columns!='overall_survive']

# Criando um dataframe com somente o atributo alvo.
y_SVC = final_table.loc[:, final_table.columns=='overall_survive']
```

Anexo 51 - Colab - criação de data frame

```
# Dividindo os dados entre treino e teste e definindo o tamanho dos dados de teste
como 30%.
x_train_SVC, x_test_SVC, y_train_SVC, y_test_SVC = train_test_split(x_SVC, y_SVC,
test_size=0.3, random_state=42)

clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))

clf.fit(x_train_SVC, y_train_SVC)

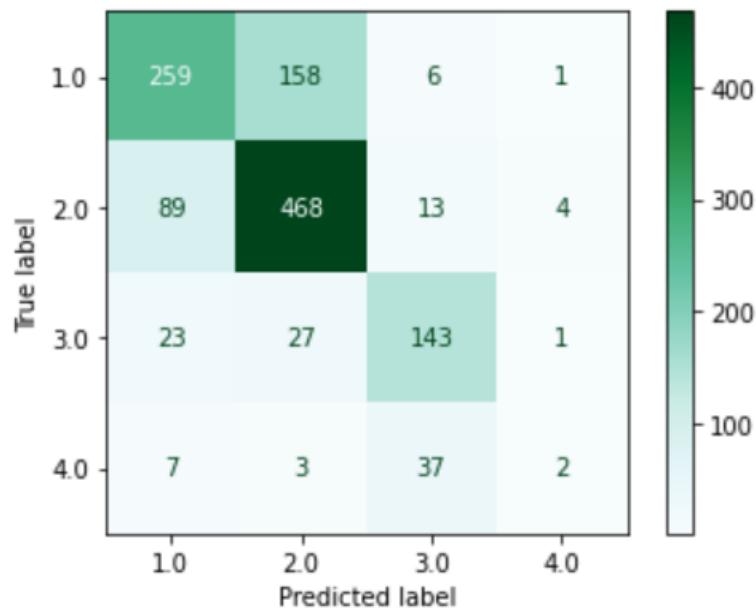
y_pred_SVC = clf.predict(x_test_SVC)

# Limpando o console de saída.
clear_output()
```

Anexo 52 - Colab -divisão e definição dos dados

```
SVM_Matrix = plot_confusion_matrix(clf, x_test_SVC, y_test_SVC, cmap='BuGn')
```

Anexo 53 - Colab - inserindo a Matriz de Confusão



Anexo 54 - Colab - matriz de confusão sem uso de tunagem de hiperparâmetros.

Buscando o aprimoramento, construímos o modelo, agora utilizando os melhores parâmetros:

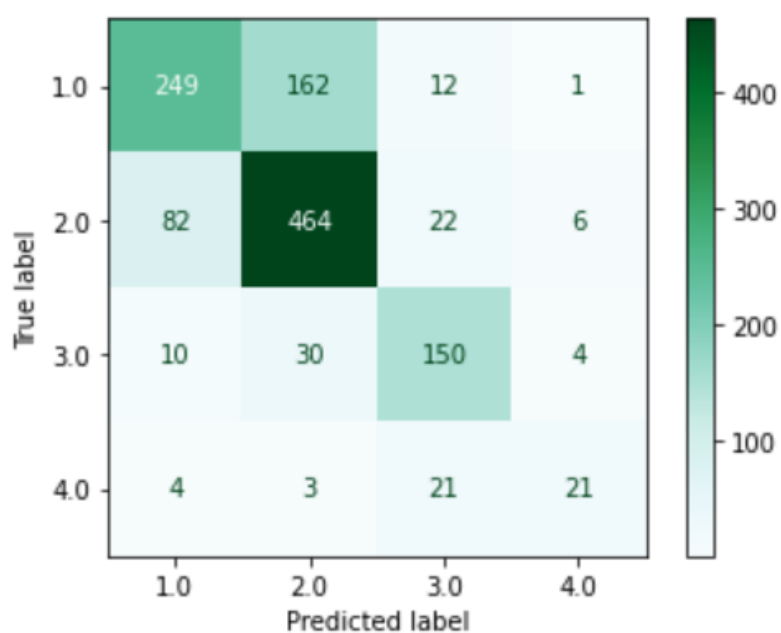
```
params = {
    'svc_C': [0.1, 1, 10, 100, 1000],
    'svc_gamma': [1, 0.1, 0.01, 0.001, 0.0001],
    'svc_random_state': [None, 0, 42],
    'svc_probability': [True, False],
    'svc_kernel': ['linear', 'rbf', 'poly']}

# Instanciando o modelo de busca
search = RandomizedSearchCV(
    estimator = clf,
    param_distributions = params
)

SVM_model = search.fit(x_train_SVC, y_train_SVC)
y_pred_SVC = SVM_model.predict(x_test_SVC)

# Imprimindo parâmetros
print(SVM_model.get_params())
```

Anexo 55 - Colab - implementando hiperparâmetros no modelo.



Anexo 57 - Colab - matriz de confusão do modelo SVM com tunagem de hiperparâmetros.

4.5. Avaliação

Com o intuito de avaliar o poder de predição dos nossos modelos (árvore de decisão, kNN, redes neurais e SVM), escolhemos comparar os resultados dos dados de treinamento com os de teste, considerando que a coluna que seria predita é "overall_survive". Para a avaliação, utilizamos as seguintes métricas:

Acurácia:

Indica uma performance geral do modelo. Dentre todas as classificações, quantas o modelo classificou corretamente.

Precisão:

Dentre todas as classificações de modelo Positivo que o modelo fez, quantas estão corretas.

Erro quadrático:

Medida de como o retorno de um fundo se afasta do retorno do benchmark.

Recall:

Porcentagem de dados classificados como positivos comparado com a quantidade real de positivos que existem em nossa amostra.

4.5.1 Árvore de Decisão

```
# Verificando a pontuação de classificação de precisão.
print("Acurácia: {:.2f}%".format(accuracy_score(y_test, y_pred) * 100))
print("Precisão: {:.2f}%".format(precision_score(y_test, y_pred,
average='macro') * 100))
print("Erro Quadrático: {:.2f}%".format(mean_squared_error(y_test, y_pred) *
100))
print("Recall: {:.2f}%".format(recall_score(y_test, y_pred, average='macro')
* 100))
```

Anexo 58 - Colab - Impressão das métricas do modelo.

```
Acurácia: 67.28%
Precisão: 69.17%
Erro Quadrático: 46.98%
Recall: 65.47%
```

Anexo 59 - Colab - resultados sem uso da tunagem de hiperparâmetros.

```
Acurácia: 71.56%
Precisão: 75.17%
Erro Quadrático: 40.21%
Recall: 68.17%
```

Anexo 60 - Colab - resultado com uso da tunagem de hiperparâmetros.

Um algoritmo foi criado para definir os atributos de maior relevância, algo importante para avaliarmos as condições transmitidas ao modelo. Para isso, uma lista foi instanciada para pegar esses itens e ordená-los de forma decrescente de acordo com a relevância calculada para cada coluna especificamente, o que nos permitiu melhor entendimento do poder de cada dado analisado.

```
def get_relevance(table, model):

    relevance_list = []

    for feature, importance in zip(final_table.columns,
model.best_estimator_.feature_importances_):
        relevance_list.append({'feature': feature, 'importance':importance*100})

    sorted_relevance_list = sorted(relevance_list, key=lambda d: d['importance'], reverse=True)

    for i in sorted_relevance_list:
        print('{:}: {:.2f}%'.format(i.get('feature'), i.get('importance'))))

get_relevance(final_table, modelTraining)
```

Anexo 61 - Colab - definindo os atributos de maior relevância

```
time_with_tumor_delta: 71.54%
menarche_age: 11.18%
age_when_cancer_was_discovered_in_months: 6.89%
histological_grade_based_on_nottingham: 5.73%
ricidive_status: 1.01%
body_mass_index: 0.77%
tobaco: 0.56%
progesteron_percentage: 0.47%
primary_diagnostic: 0.43%
hormone_therapy_time: 0.43%
breastfed_time: 0.37%
mitotic_index: 0.22%
has_breast_cancer_family_history: 0.17%
has_children: 0.15%
histological_grade: 0.05%
number_of_births: 0.00%
abortion: 0.00%
has_breastfed: 0.00%
menopause_status: 0.00%
hormone_therapy_status: 0.00%
alcohol: 0.00%
is_benign: 0.00%
tubular_differentiation: 0.00%
tumor_subtype: 0.00%
```

Anexo 62 - Colab - Output , features com porcentagem relevância

4.5.2 KNN

Verificação e cálculo das métricas do modelo.

```
# Verificando a pontuação de classificação de precisão.
print("Acurácia: {:.2f}%".format(accuracy_score(y_test_KNN, y_pred_KNN) *
100))
print("Precisão: {:.2f}%".format(precision_score(y_test_KNN, y_pred_KNN,
average='macro') * 100))
print("Erro Quadrático: {:.2f}%".format(mean_squared_error(y_test_KNN,
y_pred_KNN) * 100))
print("Recall: {:.2f}%".format(recall_score(y_test_KNN, y_pred_KNN,
average='macro') * 100))

# Plotando a matriz de confusão.
print("Segue a Matriz de Confusão: ")
_ = plot_confusion_matrix(knn_model1, x_test_KNN, y_test_KNN, cmap='BuGn')
```

Anexo 63 - Colab - checagem e cálculo das métricas do modelo

```
Acurácia: 72.52%
Precisão: 76.77%
Erro Quadrático: 38.76%
Recall: 69.25%
```

Anexo 64 - Colab - resultados sem o uso de tunagem de hiperparâmetros

```
Acurácia: 74.70%
Precisão: 78.09%
Erro Quadrático: 33.52%
Recall: 70.87%
```

Anexo 65 - Colab - resultados com o uso de tunagem de hiperparâmetros

4.5.3 Redes neurais:

Verificação e cálculo das métricas do modelo:

```
# Verificando a pontuação de classificação de precisão.
print("Acurácia: {:.2f}%".format(accuracy_score(y_test_SVC,
y_pred_SVC_no_tuning) * 100))
print("Precisão: {:.2f}%".format(precision_score(y_test_SVC,
y_pred_SVC_no_tuning, average='macro') * 100))
print("Erro Quadrático: {:.2f}%".format(mean_squared_error(y_test_SVC,
y_pred_SVC_no_tuning) * 100))
print("Recall: {:.2f}%".format(recall_score(y_test_SVC, y_pred_SVC_no_tuning,
average='macro') * 100))

SVM_Matrix = plot_confusion_matrix(clf, x_test_SVC, y_test_SVC, cmap='BuGn')
```

Anexo 66 - Colab - checagem e cálculo das métricas do modelo.

```
Acurácia: 61.64%
Precisão: 53.51%
Erro Quadrático: 48.19%
Recall: 57.10%
```

Anexo 67 - Colab - resultados sem o uso de tunagem de hiperparâmetros.

```
Acurácia: 64.87%
Precisão: 63.74%
Erro Quadrático: 42.63%
Recall: 51.98%
```

Anexo 68 - Colab - Resultados com o uso de tunagem de hiperparâmetros.

4.5.4 SVM

Verificação e cálculo das métricas do modelo:

```
# Verificando a pontuação de classificação de precisão.
print("Acurácia: {:.2f}%".format(accuracy_score(y_test_SVC,
y_pred_SVC_no_tuning) * 100))
print("Precisão: {:.2f}%".format(precision_score(y_test_SVC,
y_pred_SVC_no_tuning, average='macro') * 100))
print("Erro Quadrático: {:.2f}%".format(mean_squared_error(y_test_SVC,
y_pred_SVC_no_tuning) * 100))
print("Recall: {:.2f}%".format(recall_score(y_test_SVC, y_pred_SVC_no_tuning,
average='macro') * 100))

SVM_Matrix = plot_confusion_matrix(clf, x_test_SVC, y_test_SVC, cmap='BuGn')
```

Anexo 69 - Colab - checagem e cálculo das métricas do modelo.

```
Acurácia: 70.27%
Precisão: 59.18%
Erro Quadrático: 43.59%
Recall: 55.10%
```

Anexo 70 - Colab - resultados sem o uso de tunagem de hiperparâmetros.

```
Acurácia: 71.46%
Precisão: 60.95%
Erro Quadrático: 42.40%
Recall: 57.04%
```

Anexo 71 - Colab - resultados com uso de tunagem de hiperparâmetros.

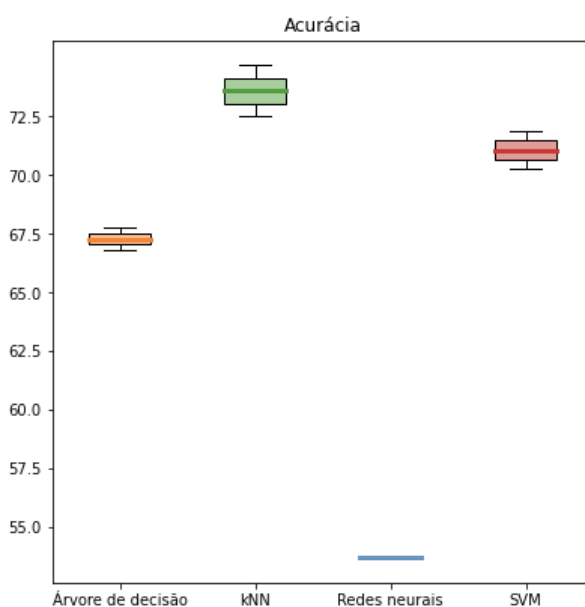
Com a análise da relevância de cada coluna e acurácia dos modelos em si, pudemos concluir que além de termos escolhidos bons modelos preditivos para nosso contexto, pois apresentaram acurácia acima de 64% em todos os modelos treinados, também conseguimos avaliar e concluir que os atributos relevantes para a área da saúde também foram relevantes para os modelos e refletiram em seus resultados, algo que era extremamente requisitado nessa aplicação. Portanto com o devido rebuscamento e refinamento desses dados e seu tratamento, podemos obter resultados cada vez melhores e precisos para a medicina preditiva.

4.6 Comparação de Modelos

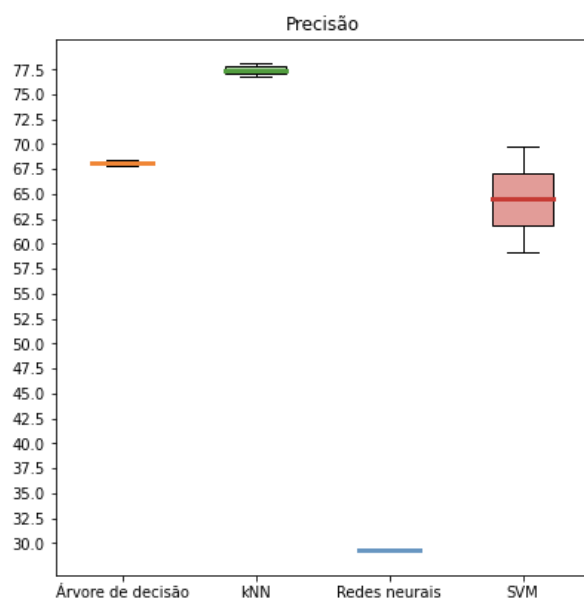
O modelo que mais se destacou dentre todos os testados (Árvore Binária, KNN, Redes Neurais e SVM) foi o KNN, que é um dos mais simplistas quando falamos de machine learning e oferece uma fácil compreensão dos resultados gerados. Como métrica de avaliação, utilizamos acurácia, precisão, erro quadrático e recall, e o K-nearest neighbors se sobressaiu dentre todas.

	Árvore Binária	kNN*	NN	SVM
Acurácia	71,56%	74,70%	64,87%	71,46%
Precisão	75,17%	78,09%	63,74%	60,95%
Erro quadrático	40,21%	33,52%	42,63%	42,40%
Recall	59,16%	70,87%	32,54%	69,15%

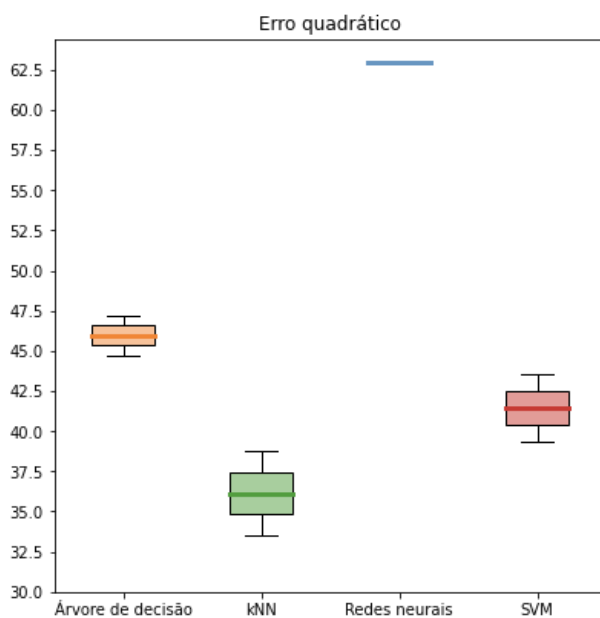
Para deixar ainda mais visível, fizemos gráficos de comparações entre os modelos, utilizando as métricas de avaliação.



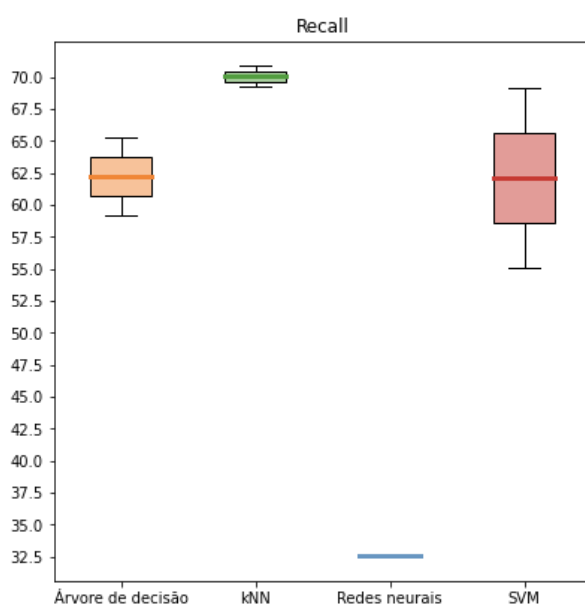
Anexo 72 - Colab - gráfico de comparação da Acurácia entre modelos



Anexo 73 - Colab - gráfico de comparação da Precisão entre modelos

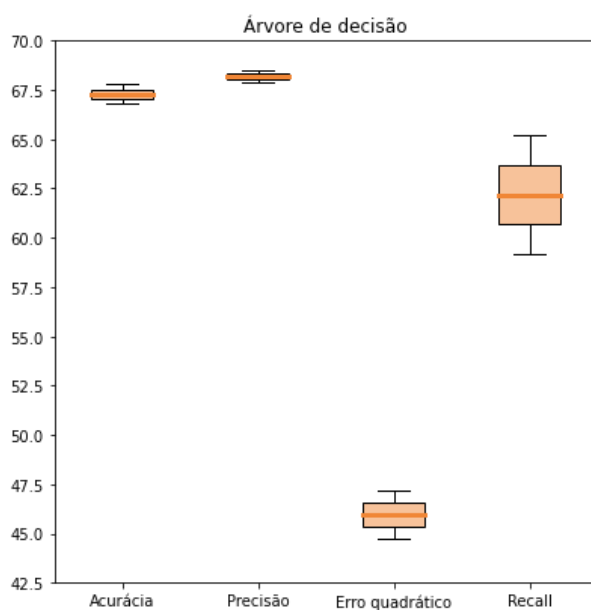


Anexo 74 - Colab - gráfico de comparação de Erro Quadrático entre modelos.

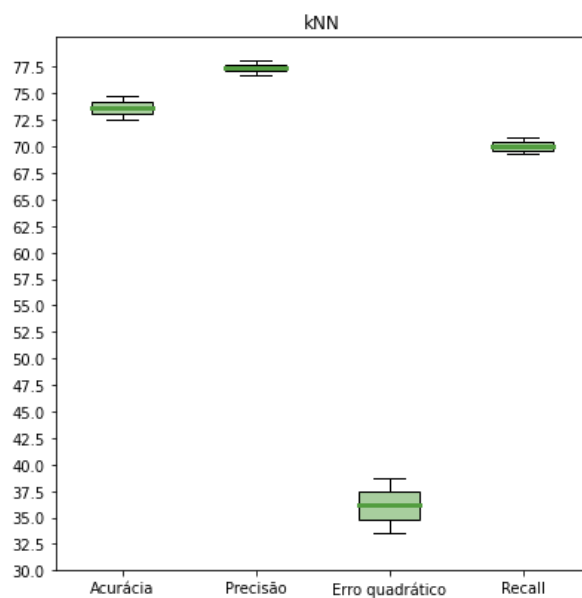


Anexo 75 - Colab - gráfico de comparação de Recall entre modelos

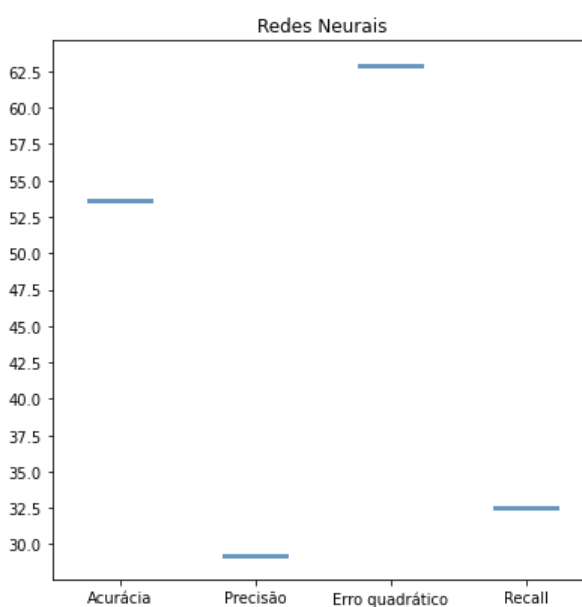
Além disso, há também gráficos de cada modelo específico, tendo como base os parâmetros avaliativos definidos:



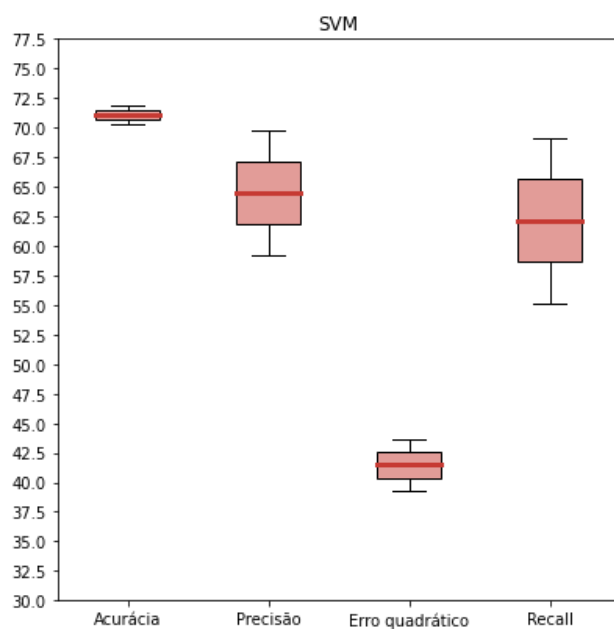
Anexo 76 - Colab - gráfico resumo de resultados obtidos do modelo de Árvore de decisão.



Anexo 77 - Colab - gráfico resumo de resultados obtidos do modelo KNN.



Anexo 78 - Colab - gráfico resumo de resultados obtidos do modelo de Redes Neurais.



Anexo 79 - Colab - gráfico resumo de resultados obtidos do modelo SVM.

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

Nesta seção você deve incluir as principais referências de seu projeto, para que seu parceiro possa consultar caso ele se interessar em aprofundar.

Utilize a norma ABNT NBR 6023 para regras específicas de referências. Um exemplo de referência de livro:

SOBRENOME, Nome. **Título do livro**: subtítulo do livro. Edição. Cidade de publicação: Nome da editora, Ano de publicação.

Anexos

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