



**Modelo preditivo para estimar a
sobrevida das pacientes:**

Hospital de Medicina USP



Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
07/08/2022	Larissa Carvalho	1.1.1	Análise SWOT - 4.1.2
07/08/2022	João Lucas	1.1.2	Value proposition canvas- 4.1.4
07/08/2022	Luiz Granville	1.1.3	5 forças de Porter - 4.1.1.2.
07/08/2022	Camila Anacleto	1.1.4	Personas - 4.1.6
07/08/2022	André Luís	1.1.5	Contexto da Indústria - 4.1.1.1
08/08/2022	Ueliton Rocha	1.1.6	Jornada do Usuário - 4.1.7
08/08/2022	Camila, André, Larissa, Luiz, João, Ueliton	1.1.7	Planejamento Geral da Solução - 4.1.3
08/08/2022	Larissa, Camila, André, Luiz, João, Ueliton	1.1.8	Matriz de Riscos - 4.1.5
12/08/2022	Larissa Carvalho	1.1.9	Proposta de Solução - 2.2 Compreensão dos Dados - 4.2
12/08/2022	Luiz Granville	1.1.10	Objetivos - 2.1 Compreensão dos Dados - 4.2
13/08/2022	Camila Anacleto	1.1.11	Justificativa - 2.3 Compreensão dos Dados - 4.2
13/08/2022	André Luís Lessa	1.1.12	Compreensão dos Dados - 4.2
13/08/2022	Ueliton Rocha	1.1.13	Introdução - 1
24/08/2022	João Lucas	1.2.1	Compreensão dos Dados - 4.3 - item A
24/08/2022	Larissa Carvalho, André Luís, Ueliton Rocha, Camila Anacleto e Luiz Granville	1.2.2	Compreensão dos Dados - 4.3 - itens B, C e D

			- Algoritmos escolhidos - 4.4 - Estratégia de avaliação - 4.4
11/09/2022	Camila Anacleto, Larissa Carvalho e Ueliton Rocha	1.3.1	- Resultados - 4.4 - Análise dos resultados - 4.5
20/09/2022	João Lucas	1.4.1	- Modelo de Naive Bayes - 4.2.2
20/09/2022	Larissa Carvalho	1.4.2	- Regressão Logística
20/09/2022	Luiz Granville	1.4.3	- Modelo Random Forest - 4.4
21/09/2022	André Lessa	1.4.4	- Modelo de K-Nearest Neighbors(KNN)
23/09/2022	Camila Anacleto	1.4.5	- Modelo Árvore de Decisão
23/09/2022	Ueliton Rocha e Larissa Carvalho	1.4.6	- Documentação Análise de Resultados
03/10/2022	João Lucas	1.5.1	- Atualização da seleção de features
04/10/2022	Larissa Carvalho	1.5.2	- Atualização dos métodos escolhidos
05/10/2022	Larissa Carvalho e Camila Anacleto	1.5.3	- Conclusão
05/10/2022	André Lessa	1.5.3	- Atualização do Modelo KNN
05/10/2022	André Lessa	1.5.3	- Metodologias - 3.0

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. As 5 forças de Porter	7
4.1.2. Análise SWOT	7
4.1.3. Planejamento Geral da Solução	7
4.1.4. Value Proposition Canvas	7
4.1.5. Matriz de Riscos	7
4.1.6. Personas	8
4.1.7. Jornadas do Usuário	8
4.2. Compreensão dos Dados	9
4.3. Preparação dos Dados	10
4.4. Modelagem	11
4.5. Avaliação	12
5. Conclusões e Recomendações	13
6. Referências	14
Anexos	15

1. Introdução

Nosso parceiro é o Instituto do Câncer do Estado de São Paulo, unidade do Hospital das Clínicas da Faculdade de Medicina da USP, é um dos maiores centros de oncologia da América Latina, com pilares em assistência, ensino e pesquisa, e atendimento 100% pela rede pública de saúde.

O problema a ser resolvido é que a evolução do câncer e sua resposta a tratamentos convencionais é muito variável.

2. Objetivos e Justificativa

2.1. Objetivos

Os objetivos gerais deste projeto são:

- Desenvolver um modelo preditivo para mitigar a razão entre os resultados de tratamentos de pacientes de mesmo subtipo através de uma referência estatística;
- Gestão mais produtiva e eficaz das vagas dos hospital bem como dos seus recursos;
- Aumentar o tempo de vida dos pacientes através de uma melhor escolha no tratamento tumoral.

Os objetivos específicos deste projeto são:

- Prognósticos mais rápidos e precisos;
- Aumento da pesquisa e desenvolvimento científico no HCFMUSP;
- Replicação do produto no SUS;
- Aumento da rotina de categorização e cadastramento de informações de pacientes bem como do aumento da noção de importância dos dados e da tecnologia na ajuda ao tratamento do câncer.

2.2. Proposta de Solução

Nossa solução trata-se de um modelo preditivo com inteligência artificial, que tem como funcionalidade predizer o tempo de sobrevida de um paciente com câncer de mama. Logo, a aplicação tem como o objetivo de encontrar padrões nos dados, para que assim possa ser possível predizer esse tempo de sobrevida. Com os dados que nos foram disponibilizados, será possível fazer análises para que encontre-se assim uma padronização, colocando-se assim, um fim no problema da variabilidade existente atual, fazendo com que seja possível um cálculo preciso e eficaz para que os diagnósticos dos pacientes sejam disponibilizados de uma maneira mais rápida e ágil.

2.3. Justificativa

Nossa proposta de solução tem como principal potencial o cruzamento de dados correlacionados, através de inteligência artificial, que apresentarão ao nosso cliente diversas informações sobre diferentes variáveis. Isto é, a solução ajudará a identificar diferentes previsões para que haja uma análise completa de cada paciente que a usará. Com isso, ao inserir os dados no nosso software, a solução identificará os padrões segundo a especificidade de cada paciente, e mostrará resultados estatísticos de forma objetiva e de fácil visualização. Como diferencial, mostraremos um score para cada paciente, que terá, entre outras métricas

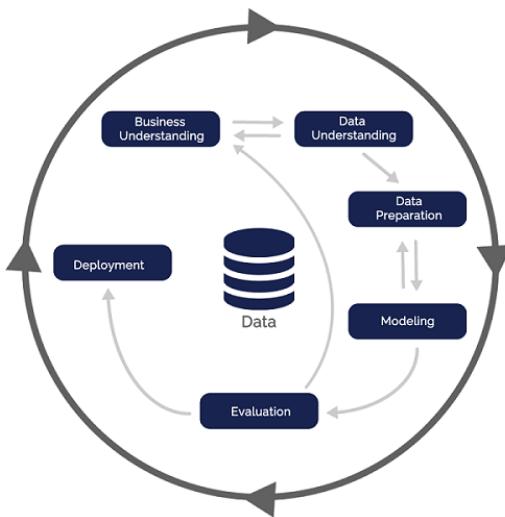
resultantes, o tempo de sobrevida. Portanto, teremos um resultado por legendas (de 0 á 25% de chance de sobrevida (com a cor vermelha), de 25% a 50% (com a cor laranja), de 50% a 75% (com a cor amarela) e, de 75% até 100% (com a cor verde). Desta forma, o médico e o gestor do hospital poderão fazer a análise dos resultados de maneira rápida e, além disso, verão todas as métricas em gráficos por trás do resultado, isto é, tudo que a IA considerou para formular tal resultado.

3. Metodologia

Nesta seção serão descritas as etapas metodológicas que foram utilizadas para o desenvolvimento do modelo preditivo.

3.1. CRISP-DM

Crisp-DM é uma sigla em inglês que significa Cross Industry Standard Process for Data Mining.



O objetivo dessa metodologia é desenvolver modelos a partir da análise de informações e dados de um negócio para prever futuras falhas e soluções. Essa metodologia tem seis etapas, são elas:

- 1- Entendimento do negócio.
- 2- Entendimento dos dados.
- 3- Preparação dos dados .
- 4- Modelagem.
- 5- Avaliação.
- 6- Implementação

Na fase inicial temos o entendimento do negócio e entendimento dos dados, sabermos porque estamos trabalhando e quais são os dados necessários para cumprirmos estes objetivos, após isso temos a preparação de dados, nas quais tratamos os dados, como por exemplo tratar os dados nulos da base enviada pelo cliente.

O próximo passo é a modelagem de dados, na qual iremos selecionar as melhores features e aplicamos os modelos preditivos realizando diversos testes, na avaliação comparamos os resultados de cada um, cujo dentro dos 5 executados, KNN, Naive Bayes, Regressão Logística, Árvore de Decisão e Random Forest, o modelo Random Forest foi o melhor modelo.

Já na última fase, que é a implementação, criamos um formulário no Google Colaboratory com a linguagem Python para que os clientes possam inserir os dados e obter o resultado do modelo.

3.2. Ferramentas

Para o desenvolvimento do modelo preditivo foram utilizadas diversas ferramentas, primeiro para executar o código, cujo é o Google Colaboratory, na qual utilizamos a linguagem Python para criação do modelo, também para armazenar o código, na qual utilizamos o Github.

Já para a criação da análise de negócio foi utilizado o Miro, no qual, montamos as cinco forças de Porter, análise SWOT, matriz de risco e o canva de proposta de valor.

3.3. Principais técnicas empregadas

Durante o módulo, utilizamos algoritmos para diferentes fins das etapas do CRISP. Para o tratamento dos dados nulos na preparação de dados, utilizamos alguns métodos para deixá-los funcionais e aumentar a qualidade e a precisão das análises finais. Sendo estes: mediana; moda; média; desvio padrão e primeiro registro;

Mediana: o peso das pacientes registradas na base de dados foi o único valor que tratamos com o uso da mediana. Utilizamos esse método pois as pacientes com câncer de mama perdem peso ao decorrer do processo do tratamento. Com a mediana, chegaremos mais próximos aos valores centrais, que fogem dos extremos tanto de seu peso normal (antes do tratamento) quanto do seu peso com a variação final pela interferência do tratamento.

Moda: utilizamos a moda para as colunas que estão mais relacionadas com a importância na frequência de um mesmo valor de dada situação.

Média: utilizamos a média para valores que não apresentam muita variação e são mais uniformes.

Desvio padrão: para esta categoria utilizamos a feature de IMC, cujo, os dados sem aplicar um cálculo matemático parece não estar uniforme, logo para encontrarmos dados mais consistentes utilizamos o desvio padrão para tornar uniformes.

Primeiro registro: utilizamos o primeiro registro para a maioria das colunas pois, como muitas pacientes apresentam mais de um subtipo, se faz necessário relacionar os dados com um único subtipo, que no caso será o primeiro a ter sido desenvolvido, uma vez que na nossa avaliação, o histórico de tratamento e desdobramentos deste, é maior, resultando em uma análise mais precisa.

Já para a modelagem de dados, primeiramente utilizamos as estratégias de selecionar as melhores features através do algoritmo **Select KBest** de forma univariada, ou seja, considerando as variáveis uma por uma, após isso utilizamos os modelos de machine learning para realizar a predição. Sendo estes: KNN, Árvore de Decisão, Regressão Logística, Random Forest e Naive Bayes.

KNN: Algoritmo de aprendizado de máquina supervisionado, este modelo assume que as coisas semelhantes existem nas proximidades, ou seja, quanto mais próximo as amostras de dados, mais similares elas são, calculando a distância entre os dados.

Árvores de decisão: Algoritmo de aprendizado de máquina supervisionado, ela utiliza decisões para criação do modelo preditivo, estabelecendo quais recursos e condições para separar para dividir essas decisões, sabendo quando parar.

Regressão Logística: Algoritmo de aprendizado de máquina supervisionado, sendo um modelo estatístico usado para determinar a probabilidade de um evento acontecer, mostrando as relações entre recursos, classificando de modo binário, ou seja, 0 ou 1.

Random Forest: Algoritmo de aprendizado de máquina supervisionado, sendo criado por várias árvores de decisão, combinando os modelos aumentando os resultados finais. O Random Forest também é utilizado para preenchimento de nulos, ou seja, predizer quais são nulos com base em uma grande quantidade de dados preenchidos.

Naive Bayes: Algoritmo de aprendizado de máquina supervisionado, utilizado para categorizar com base na frequência, desconsiderando a correlação entre as variáveis.

4. Desenvolvimento e Resultados

De maneira geral, você deve descrever nesta seção a aplicação dos métodos aprendidos e os resultados obtidos por seu grupo em seu projeto

4.1. Compreensão do Problema

4.1.1 Contexto da indústria

Na análise de indústria, busca-se entender quais são os concorrentes, os modelos de negócios e tendências da área, ajudando a identificar nossas vantagens competitivas em relação ao mercado e os riscos.

Foi identificado como possíveis concorrentes, hospitais e institutos de pesquisa que apresentam tecnologia avançada para desenvolvimento de novas ferramentas para a área de medicina, especialmente na parte de cancer de mama, e que são reconhecidos pela população, sendo o eles o Instituto Nacional de Câncer, o Hospital Israel Albert Einstein, a Agência Internacional de Pesquisa em Câncer (Iarc), Hospital Sírio-Libânes e A.C.Camargo Cancer Center.

Dos modelos de negócios, foram explorados diversos ramos, principalmente nas áreas de inovação e desenvolvimento de tecnologias, e prestação de serviço, que consideramos os principais a indústria farmacêutica de pesquisa e desenvolvimento, telemedicina, pesquisas de biotecnologia e prover atendimento de serviços hospitalares para a população.

A medicina é um ramo que está sempre sendo atualizado a fim de melhorar o atendimento e o tratamento de doenças, como o câncer de mama, logo foi realizado pesquisas pelo grupo para entender em qual direção que a medicina está se desenvolvendo, sendo identificadas a medicina personalizada, cuja é uma abordagem que usa o perfil genético do paciente e do tumor, o atendimento personalizado para determinados grupos etários específicos, análise genômica para definir o melhor tratamento a partir das mutações genéticas, terapia-alvo, imunoterapia e CAR-T Cells, que é uma outra forma de utilizar a imunoterapia.

Em suma, o mercado é grande e cheio de inovações, contudo, não foi identificado softwares que estimem o tempo de sobrevida para os médicos poderem entender a evolução do câncer de mama na paciente e na sua resposta ao tratamento, logo, nosso software será pioneiro.

4.1.1.2 As 5 forças de porter

Em uma análise minuciosa da indústria mapeamos as 5 forças de porter, um framework de análise, que proporciona uma visão mais imparcial da indústria, identificando os principais parâmetros de impacto do mercado em questão, nesse caso estamos falando da indústria hospitalar, a área da saúde vem sendo constantemente bombardeada por novas tecnologias. E por ser uma área longe da expertise dos desenvolvedores é importante ter cautelas nas afirmações abaixo.

- **Ameaça de novos entrantes:**

- Precisamos entender antes que o público atendido pelo setor hospitalar possui diversas classes, dividindo a população através do seu poder de compra em 5 grupos (A, B, C, D e E), podemos concluir que o público atendido no hospital universitário da USP se dá pelas últimas 3 classes (C, D e E), já que estamos tratando de um ambiente público, gratuito e acessível. Entretanto as classes A e B estão dispostas a frequentar hospitais mais bem equipados por se tratar de uma doença grave como o câncer. Feita essa constatação, entendemos que os principais entrantes seriam: Hospitais públicos brasileiros em geral e o próprio INCA (Instituto Nacional de Câncer), que além de atender o mesmo público são capazes de tomar a mesma iniciativa do nosso cliente.

- **Poder de barganha dos fornecedores:**

- Laboratórios de análise de exame - baixo poder de barganha.
 - Atualmente a indústria laboratorial é pulverizada o que torna a concorrência alta.
- Indústria farmacêutica - baixo poder de barganha.
 - Atualmente a indústria farmacêutica é pulverizada o que torna a concorrência alta.
- Indústria EMHO (equipamentos médicos, hospitalares e odontológicos), - alto poder de barganha.
 - Atualmente essa indústria é monopolizada e por isso podem decidir o preço de seus equipamentos.

- **Poder de barganha dos compradores:**

- Necessitam com urgência do serviço.
- Existe um mercado bem amplo de hospitais que oferecem tratamento para o câncer.
- Possuem um poder de barganha considerável, mas abrem mão do mesmo pela saúde.

- **Ameaça de produtos e serviços substitutos:**
- Empresas de telemedicina.
- Empresas de tecnologia que desenvolvem para a área de medicina, como a Onkos.
- Empresas farmacêuticas que desenvolvem medicamentos oncológicos, como a GSK.

- **Rivalidade entre concorrentes:**
- Institutos de pesquisas mais influentes pelo mundo, que possuem alta quantidade de dados e recursos tecnológicos.
- Se tratando dos possíveis entrantes, podemos entender que por serem de origem do estado, não são concorrentes diretos e por isso não estão interessados em uma concorrência e sim em se ajudarem por um objetivo único de prover saúde à população.

4.1.2. Análise SWOT

A Análise Swot tem como objetivo ter uma visão externa e interna do negócio. Desse modo, a matriz é organizada em quatro quadrantes que levam em conta o ambiente externo e interno, que são divididos em fatores internos controláveis e fatores externos incontroláveis. Essa forma de abordagem contribui para o fortalecimento dos pontos fortes e no amadurecimento dos pontos fracos, além de prevenir possíveis danos.

Matriz Swot: HOSPITAL DE MEDICINA USP

agendor

Forças

- O instituto é uma referência;
- Possui uma grande base de dados;
- Possui muitos recursos, como a tecnologia;
- Facilidade de contato com pacientes;
- Profissionais altamente capacitados;
- Referência em pesquisa;
- Acesso a grande banco de dados;
- Conhecimento sobre o assunto/ área da saúde;
- Serviço público;
- Profissionais capacitados.

Fraquezas

- Demora do resultado da estimativa;
- Falta de precisão nos resultados;
- Volatilidade dos resultados;
- Sobrecarga do sistema hospitalar;
- Estudantes em treinamento;
- Excesso de burocracia.

Oportunidades

- Possui um público muito grande para ser atendido;
- Indústria pouco explorada;
- Ganho de tempo no trabalho dos médicos, tendo-se assim maior agilidade nos diagnósticos.

Ameaças

- Hospitais que possuem mais recursos;
- Telemedicina;
- Perda de capital intelectual;
- Pessoas que discordam com tratamento tecnológico assistido;
- Tratamentos que diminuem a estadia do paciente;
- Falta de insumos hospitalares.

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)?

Os dados que baseiam nossa solução são originários dos registros de prontuários eletrônicos das pacientes com câncer de mama do Hospital das Clínicas da Universidade de São Paulo. Na planilha, foram disponibilizadas muitas variáveis como o subtipo do câncer, estadio, idade, peso, estilo de vida da mulher, histórico de gravidez, histórico de câncer na família e dados gerais sobre a saúde das pacientes.

b) Qual a solução proposta (pode ser um resumo do texto da seção 2.2)

A solução proposta para o cliente é criar uma inteligência artificial que estipula o tempo de sobrevida das pacientes utilizando a base de dados fornecidas pelo cliente, o resultado será baseado em pontuação utilizando cores, e essas irão representar porcentagem da taxa de sobrevida da paciente, se for menor que um terço dos casos será utilizado a cor vermelha, entre um terço e dois terços dos casos será utilizado a cor amarela, e acima de dois terços terá a cor verde.

c) Qual o tipo de tarefa (regressão ou classificação)

O tipo de tarefa que o nosso modelo preditivo realizará, será de classificação, uma vez que trata-se de um algoritmo supervisionado de machine learning usado para estimar a intervalos de sobrevida designados em classes, baseado em uma série de outros dados históricos. Sendo assim, com base no passado, pode-se prever o futuro. Ou seja, no nosso modelo preditivo, utilizaremos dados de entrada (preditores), já observados para prever uma resposta. Nesse caso, já foi disponibilizado uma grande base de dados de pacientes com câncer de mama, pelo nosso parceiro Faculdade de Medicina Usp, que são dados do prontuário eletrônico do paciente. Com referência nesses dados, será possível prever o tempo de sobrevida.

d) Como a solução proposta deverá ser utilizada?

O modelo preditivo será utilizado frequentemente pelo gestor do hospital, pois os tratamentos do SUS são padronizados, logo, entender quais são os pacientes que estão necessitando de maior assistência e assim, realizar uma estratégia para o remanejamento de vagas. Os médicos também irão ter um papel importante na utilização da inteligência artificial, inserindo na base de dados (estadio, subtipo, idade, obesidade) coletada através de exames na solução e a mesma irá realizar cálculos matemáticos para o resultado de tempo de sobrevida através de um score. Após o score ser fornecido, o médico irá utilizá-lo como base para a prescrição do tratamento, entendendo a situação atual do câncer do indivíduo e assim, identificando qual deve ser sua frequência de tratamento e visita no hospital.

e) Quais os benefícios trazidos pela solução proposta?

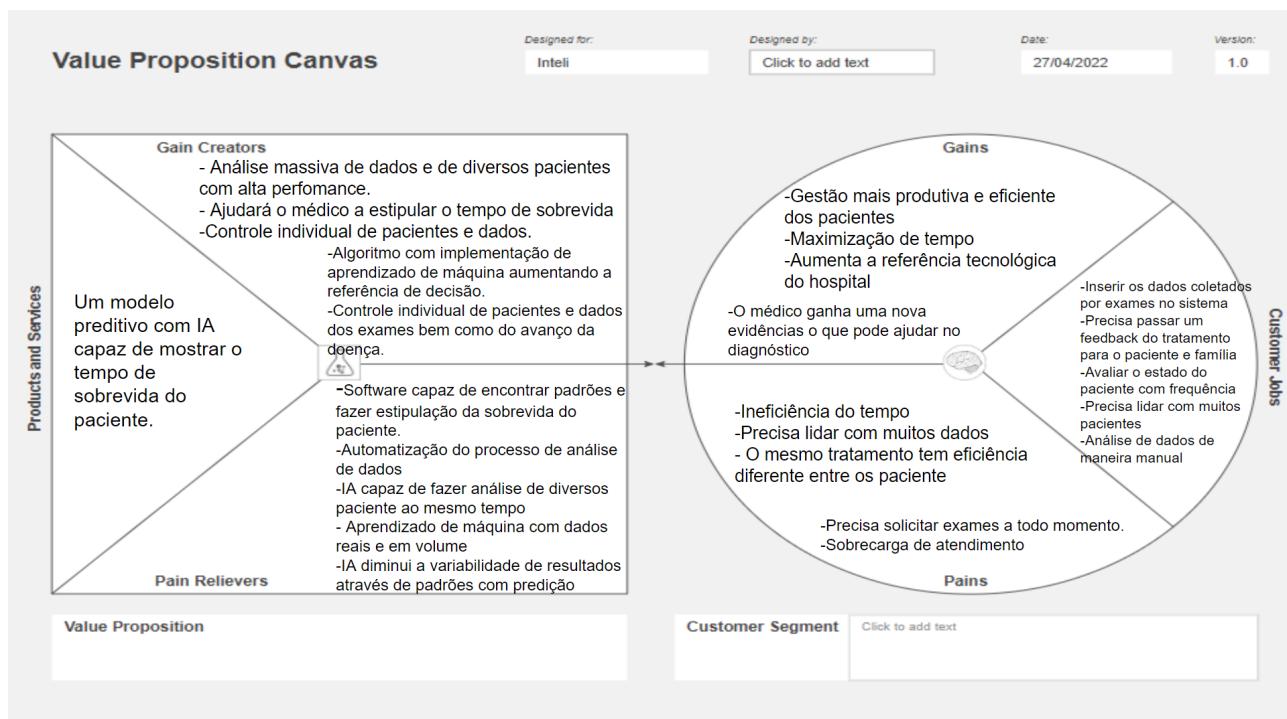
Uma vez que a base de dados é antiga (vem acompanhando pacientes desde 2008), é complicado ter uma previsão de sobrevida dos pacientes atuais, pelo fato de a observação e os dados serem antigos, com a solução do projeto proposto será possível estipular o tempo de sobrevida em forma de score, o que ajudará o médico a passar um feedback para o paciente e para sua família de uma forma mais rápida e ágil, sendo possível assim, os médicos atenderem cada vez mais esse público de pacientes. Além disso, o hospital ganha com eficiência e remanejamento dos recursos hospitalares.

f) Qual será o critério de sucesso e qual medida será utilizada para o avaliar?

Buscar uma alta taxa de assertividade utilizando como parâmetros o estadio, o subtípico, sua idade, obesidade entre outras variáveis menos estruturadas e menos coletadas. Com base nessas medidas será fornecido uma taxa que indica o atual tempo de sobrevida do paciente, quanto maior a taxa melhor, isto é, maior o tempo de sobrevida, desse modo buscamos diminuir a razão de diferença das respostas de tratamento entre pacientes do mesmo subtípico.

4.1.4. Value Proposition Canvas

O Canvas da proposta de valor serve para ajudar a criar e organizar produtos/serviços que se alinhem com o que seu cliente realmente valoriza e precisa.



4.1.5. Matriz de Riscos

		Ameaças			
Probabilidade	90%				Dados não tão bem estruturados Baixo conhecimento técnico das ferramentas
	70%				Pouco tempo para desenvolver Dados nulos
	50%				
	30%				Falta de assiduidade dos integrantes
	10%	Erros de Commit/Merge no GitHub	Ter que mapear novos parâmetros não identificados		Rompimento da integridade dos dados
	Muito Baixo	Baixo	Moderado	Alto	Muito Alto
Impacto					

Probabilidade	Oportunidades				
	90%	Experiência em trabalhar com processos de entregas	Desenvolver habilidades e conhecimento sobre um tema pertinente		
	70%		Aumentar a referência do hospital	Aumentar o networking	
	50%	O SUS pode dar continuidade no MVP	Aumentar o recurso de tecnologia do hospital		
	30%		Predizer novos parâmetros além da taxa de sobrevida		
	10%	Replicar o modelo preditivo para outras doenças cancerígenas			
	Muito Alto	Alto	Moderado	Baixo	Muito Baixo
Impacto					

4.1.6. Personas



Nome: William Silva.

Idade: 35 anos.

Gênero: Masculino.

Ocupação: Médico.

Resumo que define a persona: "Vivo pela vida!"

- Considerações biográficas e comportamentais:

- Nascido em Ribeirão Preto e formado na UNESP.
- Sua madrinha morreu de câncer de mama ainda quando William era criança, e isso o motivou a ser oncologista.
- William ama a natureza e por isso dá muito valor à vida e à família.
- William é extremamente empático e normalmente dá total respaldo e suporte a família do paciente que ele trata.

- Dores/motivações atuais com o problema:

- Gostaria de um software que me ajudasse na tomada de decisões.
- Gostaria de um software que analisasse os dados rapidamente, o que maximizaria meu tempo.
- Willian tem vários pacientes para tratar e por isso precisa constantemente diagnosticar o estado do câncer do paciente e isso toma muito do seu tempo.
- Gostaria de passar relatórios mais rápidos aos familiares.

- Objetivos/necessidades específicas em relação ao problema:

- William precisa de mais tempo para tratar seus pacientes.
- Obter um diagnóstico mais assertivo para um tratamento eficaz.
- Obter material e resultado de apoio sobre suas decisões.
- Indicar tratamentos menos invasivos e mais eficazes.



Nome: Fabiana Camargo .

Idade: 42 anos.

Gênero: Feminino .

Profissão: Fisioterapeuta .

Resumo que define a persona: “Apaixonada pela família e querendo vencer essa luta”.

- Considerações biográficas e comportamentais:

- Nascida no Rio de Janeiro e formada pela UFRJ.
- Casada.
- Tem duas filhas.
- Adora a natureza.
- Tem histórico de câncer na família.
- Acredita na medicina tecnológica.
- Desconfiada.

- Muito cuidadora.

- Dores/motivações atuais com o problema:

- Não tem um diagnóstico preciso atualmente.
- Acredita na tecnologia como solução.
- Gosta de se manter informada.
- Tem duas filhas para cuidar e precisa de um tratamento rápido.

- Objetivos/necessidades específicas em relação ao problema:

- Quer descobrir quanto tempo tem de vida.
- Quer um tratamento assertivo e eficaz.
- Deseja um diagnóstico completo e preciso.
- Deseja prolongar seu tempo de vida.
- Quer um tratamento menos invasivo.



Nome: Rogério Varelas.

Idade: 55 anos.

Gênero: Masculino.

Profissão: Gestor do Hospital.

Resumo que define a persona: "Trabalho com vidas, por isso prezo por cada uma delas".

- Considerações biográficas e comportamentais:

- Nascido no Paraná, é formado pela UFPR.
- Sua mãe era médica.
- Tem um filho e duas filhas.
- É gestor do hospital em que sua mãe trabalhava.
- Também é professor de uma faculdade de medicina.

- Dores/motivações atuais com o problema:

- Ter muitos pacientes nos leitos do hospital.

- Acredita que a tecnologia pode ajudar em suas decisões estratégicas no hospital.
 - Tem poucos médicos trabalhando para muitos pacientes.
- Deseja possuir mais recursos para o hospital.

- Objetivos/necessidades específicas em relação ao problema:

- Quer saber em qual momento alocar pacientes a depender de seu estado.
- Deseja um modelo que juntamente com o trabalho do médico traga melhores diagnósticos para os pacientes.
- Padronizar os dados com um método mais eficaz.
- Otimizar o tempo do trabalho dos médicos.
- Conseguir atender ainda mais um público de pacientes.
- Um sistema que prediz melhor o estágio de um paciente.

4.1.7. Jornadas do Usuário

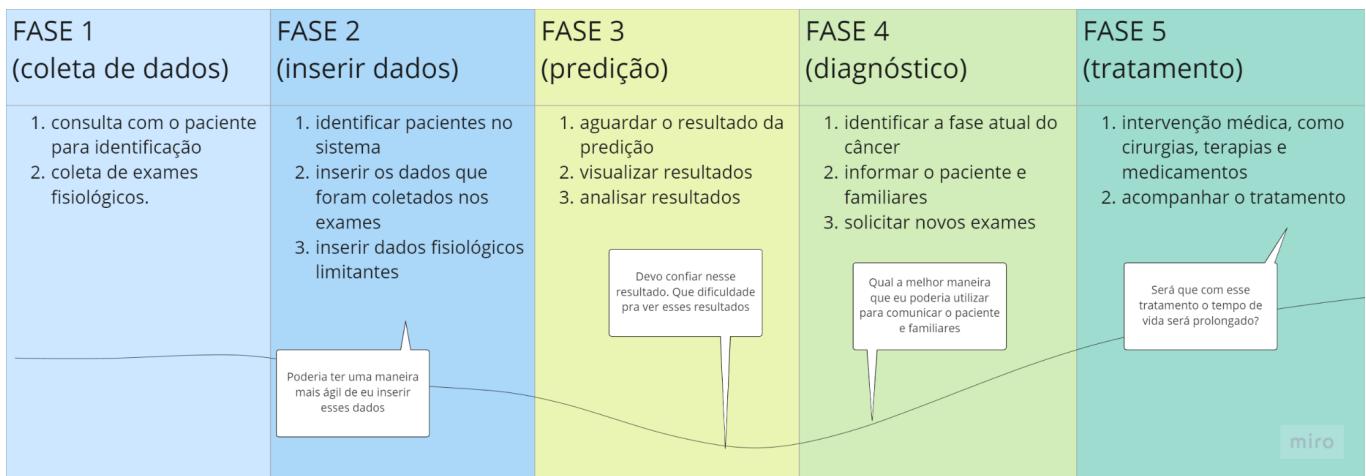
A jornada do usuário é uma forma de compreender todas as fases de interação que o cliente tem com o produto ou algum serviço, ou seja, quando se realiza o mapeamento detalhado de toda experiência possível que o usuário poderá ter durante a utilização do que está sendo proposto. Facilitando assim analisar todas as variáveis envolvidas.

Utilizamos nosso Persona William para nossa Jornada do usuário, conforme abaixo:

William Silva:

Cenário: William precisa designar tratamentos para pacientes com câncer baseados no estado do avanço da doença, além de apresentar o tempo de sobrevida para cada paciente.

Expectativas: William espera que com a IA ele possua mais tempo para tratar seus pacientes através de um diagnóstico mais preciso e um tratamento mais eficaz.



miro

Oportunidades: Existem as oportunidades de agilizar a inserção de dados na IA, aumentar a confiabilidade do resultado e facilitar a visualização dos resultados, criar um método de comunicação do resultado para o paciente e familiares, como informar a melhora do câncer com o tratamento.

Responsabilidades: A responsabilidade será atribuída para o médico, juntamente com a IA, uma vez que com ela, resultados serão gerados e avaliados pelo médico. Com isso, apresentará o diagnóstico de maneira assertiva e assim, indicará a melhor maneira de prosseguir com os tratamentos.

4.2. Compreensão dos Dados

1. Descreva os dados a serem utilizados (disponibilizados pelo cliente e outros se tiverem sido incluídos), detalhando a fonte, o formato (CSV, XLSX, banco de dados, etc.), o conteúdo e o tamanho.
 - a. Se houver mais de um conjunto de dados, descrição de como serão agregados/mesclados.

A base de dados que nos foi disponibilizada em formato de arquivo CSV contém informações do prontuário eletrônico do paciente. Ou seja, do documento preenchido com os dados do paciente de quando ele chega no hospital. Todos os pacientes que estão na base de dados, possuem ou possuíram o câncer de mama. A tabela possui mais de 8.000 dados de pacientes, mas o grande empecilho é o fato de conter muitos dados nulos, fazendo com que tenhamos que fazer análises desconsiderando-os, e seguindo com os dados que contém mais informações. A quantidade de colunas de tipos de dados é 96, desses 96 pretendemos fazer as melhores correlações para que a padronização ocorra.

Uma relação importante, que devemos nos basear em nossas análises, é os dados de subtipo, uma vez que existem quatro e podem ser correlacionados a outros dados. Como atualmente o foco é diminuir a razão de diferença das respostas de tratamento entre pacientes do mesmo subtipo, temos levado em consideração comparar dados que tenham relação e geram estatísticas relevantes para as nossas análises, bem como:

Subtipo - Últimas informações

Subtipo - Primeira menstruação - Terapia de reposição hormonal

Subtipo - Amamentou na primeira gestação?

Subtipo - Idade

Subtipo - Menopausa

Gráfico que contém a informação de quantos possuem histórico familiar de câncer?

Além dessas relações, podemos citar os dados mais relevantes da tabela, que os agrupamos em uma nova tabela, para analisar os dados preenchidos, sem nulos, que são os seguintes tipos de dados:

- Subtipo tumoral
- Já ficou grávida?
- IMC
- Amamentou na primeira gestação?
- Por quanto tempo amamentou?
- Idade da primeira menstruação?
- Fez terapia de reposição hormonal?
- Antecedentes de familiar com câncer de mama?
- Possui histórico familiar de câncer?
- Última informação do paciente?
- Tumor maligno ou benigno?
- Recidiva?

Esses foram os principais que queremos iniciar as análises. Partindo desses dados, será possível termos resultados estatísticos que servirão de base para o nosso projeto de modelo preditivo. Portanto, há muitos dados ainda para se analisar e se relacionar entre si.

b. Descrição dos riscos e contingências relacionados a esses dados (qualidade, cobertura/diversidade e acesso).

São muitos os riscos relacionados à base de dados fornecida. O primeiro, e mais importante, é a quantidade de células vazias (nulas) nas colunas. Por não apresentarem informações suficientes, corremos o risco de não entregar uma resposta que se aproxime da previsão esperada pela IA, pois para que a qualidade das respostas seja a mais adequada, precisamos de um conjunto completo com todas as informações preenchidas, não apenas de uma, mas todas as colunas de variáveis a serem analisadas. Dados faltantes também prejudicam a qualidade das diferentes métricas que poderíamos apresentar para o cliente, pois, com diferentes informações, podemos encontrar mais resultados que mostram padrões nas pacientes que compartilham um histórico de vivências/comportamentos similares. Além disso, outro risco que poderíamos correr com a base de dados é caso a precisão dos resultados aumentassem apenas considerando as pacientes que já foram a óbito, para que pudéssemos, de fato, analisar os casos do começo ao fim para uma previsão mais eficaz da solução. Esse seria um risco pois não temos à disposição muitos dados de pacientes que foram a óbito, em razão do câncer, do começo ao final do tratamento. Ademais, outro risco importante, dado o limitante de cédulas e colunas preenchidas, é o número reduzido de casos a serem analisados com as mesmas métricas, pois é difícil encontrar uma quantidade expressiva de dados com diferentes variáveis igualmente preenchidas. Dessa forma, podemos ter

como resultado valores que não necessariamente dizem respeito sobre um padrão.

- c. Se aplicável: descrição de como será selecionado o subconjunto para análises iniciais (quando o tamanho do conjunto de dados impossibilita a utilização do conjunto completo em todas as etapas da definição do modelo a ser usado).

Atualmente nossa população de dados é pouco confiável para que a mineração de dados seja feita com alta integridade, para isso iremos repopular a base através de estratégias algorítmicas de média, mediana, moda, desvio padrão e último registro, após isso iremos também revalidar os pacientes que trocaram de tumor durante seu tratamento e portanto devemos analisá-lo em uma outra instância sem descartar o estágio anterior.

Após a repopulação iremos utilizar todo o conjunto de dados para começarmos a identificar as variáveis preditoras através das análises descritas no item um deste capítulo.

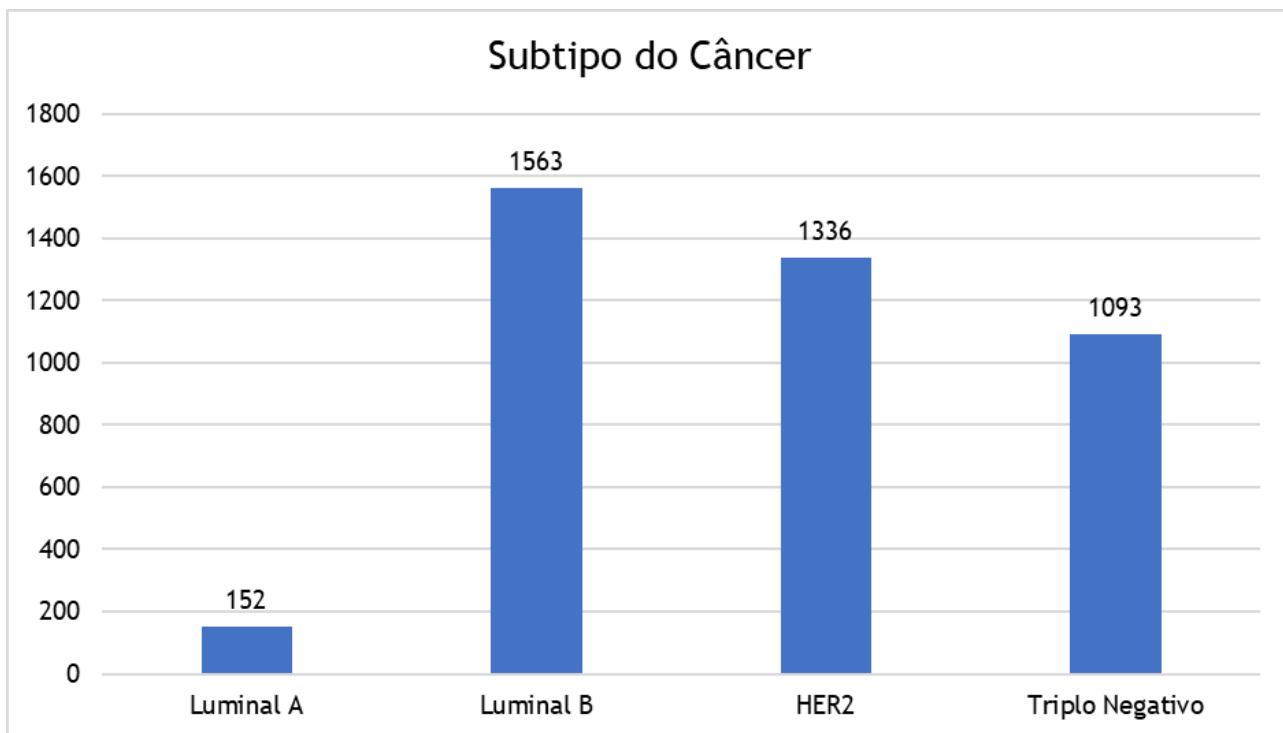
- d. Se houver: descrição das restrições de segurança.

Com base nas leis gerais de proteção de dados, a base de dados é composta por dados anonimizados, ou seja dados relativos a titular que não possa ser identificado, considerando a utilização de meios técnicos razoáveis e disponíveis na ocasião de seu tratamento, preenchendo requisitos de dados com relação à saúde pública, estes também não podem ser restaurados garantindo o sigilo dos titulares dos dados.

Apesar de não ter medidas específicas com os cuidados desses dados, é importante agir com cautela, mantendo a integridade desses dados para realização de estatísticas mais precisas, possibilitando um modelo de predição mais assertivo. É necessário cuidados com os vazamentos desses dados, já que são dados relacionados à saúde pública não podendo ser compartilhados fora do Inteli.

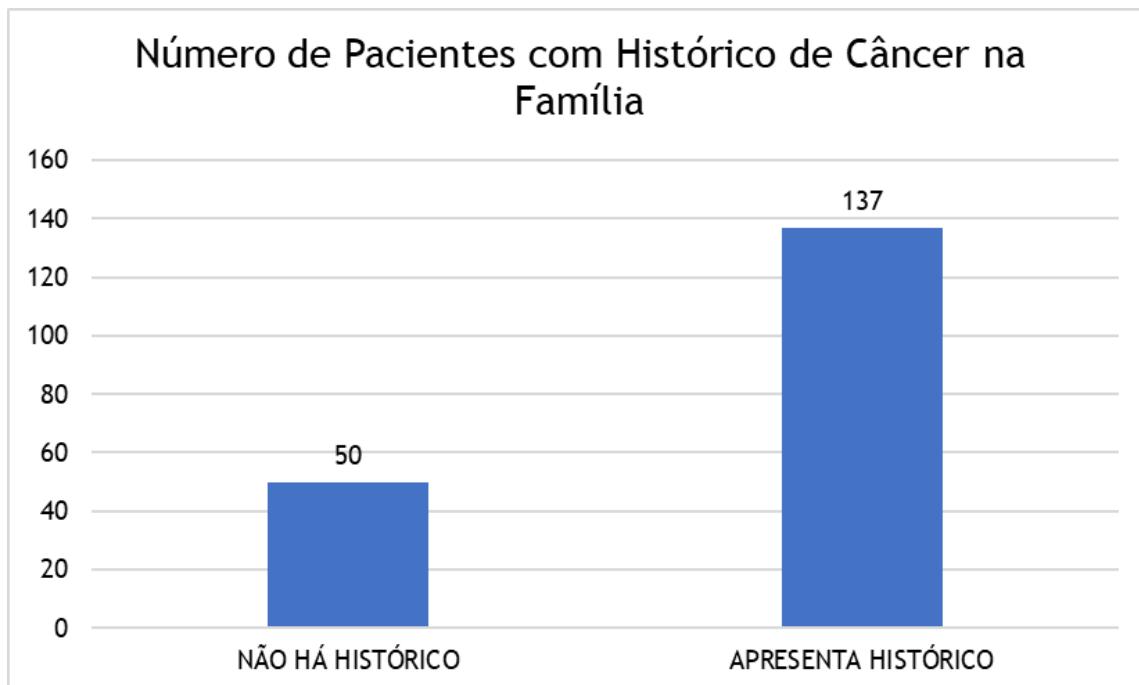
2. Descrição estatística básica dos dados, principalmente dos atributos de interesse, com inclusão de visualizações gráficas e como essas análises embasam suas hipóteses.

Gráfico 1:



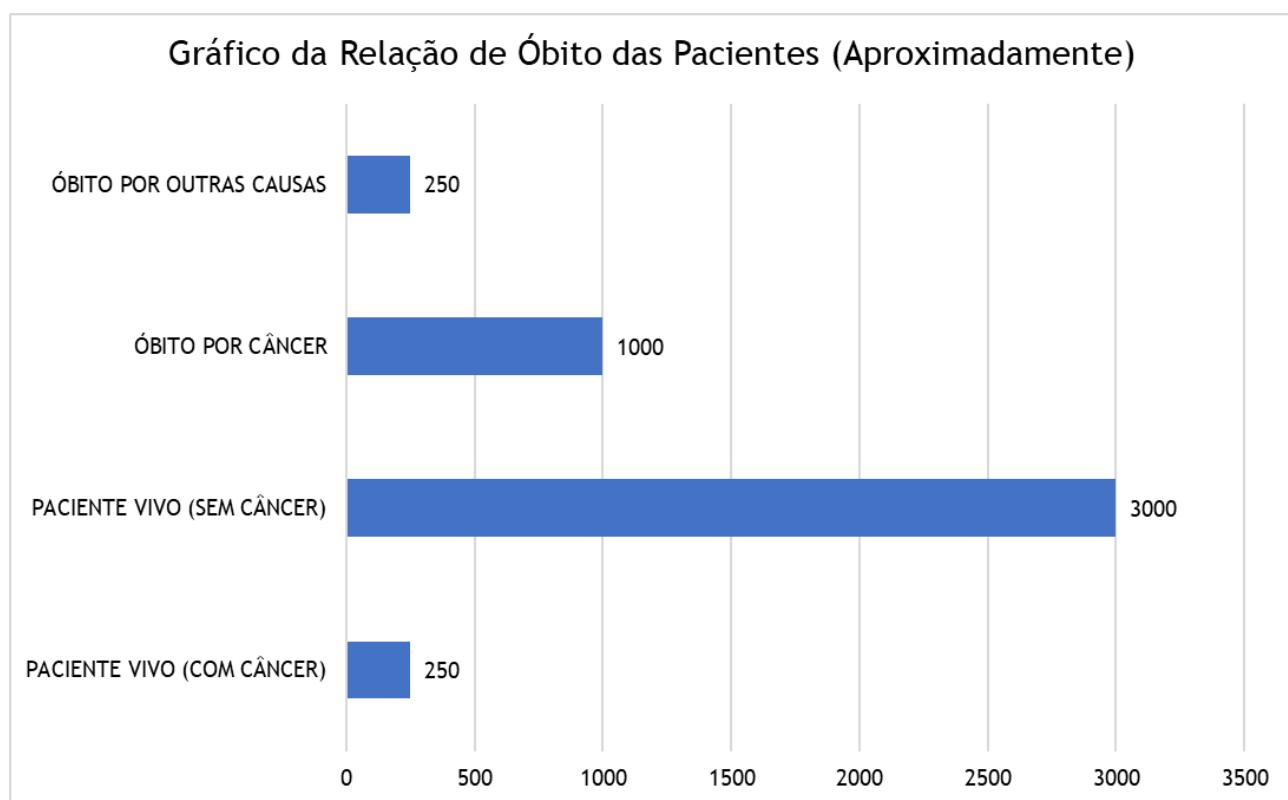
Nesse primeiro gráfico, nota-se uma relação entre subtipos de câncer, no caso são 4 tipos existentes, e de acordo com o gráfico, nota-se que o subtipo - Luminal B, é o mais recorrente nos pacientes que estão nas tabelas de dados. Portanto, esse gráfico serve como uma base de início para dar um direcionamento das nossas análises de dados, uma vez que podemos encontrar padrões que cada paciente possui para se enquadrar em cada subtipo desses.

Gráfico 2:



Já neste segundo gráfico, fizemos uma comparação entre quantas pessoas da base de dados possuem um histórico familiar de câncer de mama, e podemos ver o resultado de que a grande maioria possui, ficando evidente de que essa doença ocorre mais de maneira hereditária. Com isso, nos ajuda a encontrar padrões para que depois possamos comparar como cada paciente vai se encaixar de acordo com um histórico familiar ou não.

Gráfico 3:



Nesse terceiro e último gráfico, pode-se notar uma comparação entre as últimas informações de cada paciente, ou seja, em que estágio cada paciente com câncer de mama se encontra atualmente. Portanto, de acordo com o gráfico, a maioria está vivo sem câncer, o que é uma novidade para a gente, pois não era o que se esperava. O que realmente se esperava era que a quantidade de paciente vivo com câncer fosse maior, de acordo com a base de dados que nos foi disponibilizada. Logo, essa nossa hipótese foi quebrada após esse gráfico.

Em segundo lugar, vem a quantidade de óbito por câncer, com essa análise, é evidente que poderemos fazer grandes relações e comparar os motivos de óbito para ver o que levou cada

paciente a esse estado. Sendo assim, poderemos fazer grandes análises a partir dessas informações que nos foram dadas com esse gráfico e encontrar diversos padrões.

3. Descrição da predição desejada (“target”), identificando sua natureza (binária, contínua, etc.)

A variável target se trata de 3 classes que indicam a um conjunto de sobrevida do paciente, essas classes são divididas entre um terço e dois terços dos pesos dos valores de dias sobrevividos por um paciente, para que o modelo possa treinar com uma quantidade equivalente de casos para todas as classes. Esta target também foi descrita no item “b” do capítulo 4.1.3.

Atualmente essa variável foi realizada nas tuplas do conjunto de dados entre baixo nível de sobrevida e alto nível de sobrevida, para isso ela utilizou uma variável preditora , o tempo de sobrevida geral e com a mediana desta, comparou com o tempo de sobrevida geral do paciente e os classificou. Pretendemos analisar com mais calma o tempo de sobrevida geral do paciente, especialmente seus dados de frequência, para que assim criemos mais valores possíveis mantendo a mesma validade proposta pelo stakeholder no conjunto de dados.

4.3. Preparação dos Dados

Descreva as etapas realizadas para definir os dados e os atributos descritivos dos dados (“features”) a serem utilizados. Essa descrição deve ser feita de modo a garantir uma futura reprodução do processo por outras pessoas, e deve conter:

a) Descrição de quaisquer manipulações necessárias nos registros e suas respectivas features.

Nessa seção o objetivo foi encontrar valores dentro de colunas que estavam fora do padrão usual do modelo preditivo.

O primeiro passo realizado foi identificar colunas com valores em forma de string, para isso foi usado o código abaixo:

CÓDIGO:

```
data.dtypes
```

- Convertendo string para inteiro:

CÓDIGO:

```
for x in range(len(data_columns_string_to_numbers)):  
    data[data_columns_string_to_numbers[x]] = data[data_columns_string_to_numbers[x]].astype(str)  
    data[data_columns_string_to_numbers[x]] = data[data_columns_string_to_numbers[x]].str.lstrip('C')  
    data[data_columns_string_to_numbers[x]] = data[data_columns_string_to_numbers[x]].str.lstrip('nan')  
    data[data_columns_string_to_numbers[x]] = pd.to_numeric(data[data_columns_string_to_numbers[x]], downcast="integer")
```

Neste código todas as colunas foram primeiramente transformadas para string, pois nas linhas havia tanto valores em forma de string como de números. Após todos os valores passados para string, foram eliminados os erros de escrita, para assim passar todos os números escritos em forma de string para inteiros.

Observação: Havia valores em forma de string que representavam os mesmos valores de números.

CÓDIGO:

```
for x in range(len(data_columns_str_to_percentage)):  
    data[data_columns_str_to_percentage[x]] = data[data_columns_str_to_percentage[x]].astype(str)  
    for key, values in incorrect_values.items():  
        for specific_values in values:  
            data[data_columns_str_to_percentage[x]] = data[data_columns_str_to_percentage[x]].str.replace(especific_values, key)  
    data[data_columns_str_to_percentage[x]] = data[data_columns_str_to_percentage[x]].str.lstrip('nan')  
    data[data_columns_str_to_percentage[x]] = pd.to_numeric(data[data_columns_str_to_percentage[x]], downcast="integer")
```

Neste código todas as colunas foram primeiramente transformadas para string, pois nas linhas havia tanto valores em forma de string como de números. Após todos os valores passados para string, as palavras foram substituídas (função replace) por números, para assim passar todos os números escritos em forma de string para inteiros.

Observação: Havia valores em forma de string que representavam os mesmos valores de números.

CÓDIGO:

```
le = preprocessing.LabelEncoder()

default_columns = ["descmorfico","antec_fam_cancer_mama","desctopo"]

for x in default_columns:
    data[x] = le.fit_transform(data[x])
```

Neste código foi usado o método LabelEncoder para rotularizar as strings para números.

- Algumas colunas com valores de datas estavam escritas como string, para revertê-las para valores de data, foi usada o seguinte código:

CÓDIGO:

```
for x in range(len(data_columns_date)):
    data[data_columns_date[x]] = pd.to_datetime(data[data_columns_date[x]])

#Esse laço é capaz de transformar strings em forma de datas
```

As colunas que foram alteradas para valores de data são: “dob”, “date_last”, “date_last_fu”, “dtrecidiva” e “data_2”.

- Convertendo infinitos para NaN:

CÓDIGO:

```
for x in range(len(data_columns)):
    data[data_columns[x]].replace(np.inf, np.nan, inplace=True)
```

Anular os valores infinitos tem como objetivo evitar cálculos errados.

OBS: os dados que foram revertidos para NaN, também foram tratados e serão explicados no item C da seção 4.3.

b) Se aplicável, como deve ser feita a agregação de registros e/ou derivação de novos atributos.

Foi necessária a criação de uma nova coluna para assim, facilitar o processo de codificação. Logo, a partir da data de nascimento, foi possível com um algoritmo do colab, a criação de uma nova coluna na tabela. Com isso, a coluna que foi criada foi a da idade, a qual será de extrema importância para as análises.

A coluna criada tem valores das idades de cada ID. Para criá-la, a função DatetimeIndex, que atribui a data atual, é subtraída do valor da data de nascimento do ID, assim gerando a idade.

```

data['data_2_first']=data['data_2'].copy()
data['data_2_last']=data['data_2'].copy()

#Criação de colunas novas de data de consulta item b

default = ["hormone_therapy_tasy","tempo_rep_hormo_tasy","redcap_repeat_instrument","redcap_repeat_instance", "dtrecidiva", "data_2"]

for x in data.columns:
    if data[x].notna().sum() < 1:
        default.append(x)

default_data_1 =[ "dob", "escolari", "level_education", "race", "pregnancy_history", "partos", "abortion", "breast_feeding", "breast_feeding_time"]

#Vetor de colunas que usam método de primeiro valor e ultimo valor

default_data_2 =[ "weight", "height", "bmi", "tobaco_type", "mother_tumor", "sister_tumor_1", "daughter_tumor_1", "diff_tubular", "data_2_last"]

#Vetor de colunas que usam método ultimo valor, flutuacao, média, moda e devio padrão

default_data_3 = [ "weight", "height", "bmi", "tobaco_type", "mother_tumor", "sister_tumor_1", "daughter_tumor_1", "diff_tubular", "dob", "escolari"]

#Vetor de colunas que usam método primeiro valor, flutuacao, média, moda e devio padrão

for x in default:
    if x in default_data_1:
        default_data_1.remove(x)
    if x in default_data_2:
        default_data_2.remove(x)
    if x in default_data_3:
        default_data_3.remove(x)

#Lista de colunas padrões a serem excluidas

if default_columns == True:
    data = data.drop(columns = default)

data_1 = data.copy()
data_2 = data.copy()
data_3 = data.copy()

data_1 = data_1.drop(columns = default_data_1)
data_2 = data_2.drop(columns = default_data_2)
data_3 = data_3.drop(columns = default_data_3)

```

```
else:  
    del data[column]  
  
#Essa condição verifica se o usuário que usar as colunas padrões, caso verdadeiro, percorre  
#a lista de colunas a serem excluídas
```

c) Se aplicável, como devem ser removidos ou substituídos valores ausentes/em branco.

De acordo com a base de dados que nos foi disponibilizada, algumas colunas vieram com dados nulos, ou seja, impedem que possamos prosseguir para que tenhamos uma análise mais precisa. Pelo fato de estarmos construindo um modelo preditivo, a partir da aprendizagem de máquina, foi necessário tratarmos esses dados, fazendo com que os dados nulos se tornassem inexistentes, trazendo uma maior precisão para o modelo, uma vez que a saúde é um tema sensível.

Sendo assim, através de alguns métodos como: média, moda, mediana, desvio padrão e primeiro valor, criados com um algoritmo do colab, fizemos uma análise geral das colunas da base de dados que nos foi fornecida pelo nosso parceiro Instituto do Câncer/ Faculdade de Medicina USP, para que fosse possível decidirmos qual método seria o melhor para preencher os dados nulos de acordo com cada coluna. Com o algoritmo criado no colab ,é possível selecionarmos a coluna que desejamos e em seguida o método que escolhemos para aquela determinada coluna, pois há situações diferentes, e assim, preencher de acordo com o método que é mais próximo daquele tipo de dado.

Exclusão de colunas:

A remoção de dados foi feita com um algoritmo que selecionava uma coluna que estava totalmente vazia, ou seja, que não possuía nenhum tipo de dado, e com isso excluía totalmente essa coluna do nosso banco de dados, uma vez que não teria nenhuma relevância, pois todos os dados estavam nulos e não forneciam nenhum parâmetro para que pudéssemos analisar e escolher um método para preencher esses dados. Portanto, através de um algoritmo de exclusão de colunas, foram totalmente excluídas.

Além disso, precisamos dividir nosso *dataframe* em 2 *dataframes*, eles receberam sua cópia e dos mesmos foram excluídas as colunas referente ao conjunto de métodos de preenchimento de nulos de último e primeiro valor e o mesmo no outro *dataframe* com as colunas que seriam aplicadas os outros métodos.


```

#@title Algoritmo de preenchimento de nulos
default_columns= True #@param {type:"boolean"}

#markdown Informe o método de preenchimento de valores nulos
method = 'ultimo_Registro' #@param ["mediana", "moda", "media", "ultimo_Registro", "primeiro_Registro"]

#markdown Informe as coluna que você deseja implementar este método
column = 'level_education' #@param ["record_id", "redcap_repeat_instrument", "redcap_repeat_instance"]

columns = {
    "mediana": ["weight"],
    "moda": ["tobacco_type", "mother_tumor", "sister_tumor_1", "daughter_tumor_1", "diff_tubul"],
    "media": ["height"],
    "ultimo_Registro": [],
    "primeiro_Registro": ["dob", "escolari", "level_education", "race", "pregnancy_history"],
    "desvio_Padrao": ["bmi"]
}

```

Algoritmo de preenchimento de nulos

default_columns: 

Informe o método de preenchimento de valores nulos

method: 

Informe as coluna que você deseja implementar este método

column: 

- record_id
- redcap_repeat_instrument
- redcap_repeat_instance
- dob
- escolari
- level_education 
- race

Após a seleção da coluna que deseja aplicar o método, no algoritmo, 'columns' atualizará para a coluna selecionada.

Métodos de preenchimento:

Mediana:

```

def mediana():
    def func(x,column):
        try:
            x[column] = x[column].median()
        except IndexError:
            print("")
        return x

    return data.groupby(by=["record_id"]).apply(lambda x: func(x,column))

```

No método da mediana, foi utilizado uma função mediana(), essa função vai pegar a coluna e seus valores, e com isso preencher os dados nulos com a mediana. A função “median()” é um recurso da biblioteca pandas, é nela que será feito o cálculo da mediana.

Moda:

```
def moda(column):
    def func(x,column):
        try:
            y = x[column].dropna().to_numpy()
            x[column] = statistics.mode(y)
        except IndexError:
            pass
        except StatisticsError:
            pass
        return x

    return data_1.groupby(by=["record_id"]).apply(lambda x: func(x,column))
```

No método da moda, foi utilizado uma função `moda()`, essa função vai pegar a coluna e seus valores, e com isso preencher os dados nulos com a moda. O *Data Frame* sem nulos é transformado em um vetor e com isso é aplicado o método `mode()`, e armazenado na coluna do *Data Frame*.

Média:

```
def media():
    def func(x,col):
        try:
            x[column] = x[column].mean()
        except IndexError:
            print("")

    return x

return data.groupby(by=["record_id",]).apply(lambda x: func(x,column))
```

No método da média, foi utilizado uma função `media()`, essa função vai pegar a coluna e seus valores, e com isso preencher os dados nulos com a média. A função “`mean()`” é um recurso da biblioteca pandas, é nela que será feito o cálculo da média.

Primeiro registro:

```
def func_fregister(mini_data):
    lista_valores = []
    for c in mini_data.columns:
        index_m = mini_data[c].first_valid_index()
        if index_m != None:
            valor_nao_nulo = mini_data[c].loc[index_m]
            lista_valores.append(valor_nao_nulo)
        else:
            lista_valores.append(np.nan)

    return pd.DataFrame(data=[lista_valores], columns=mini_data.columns)

def primeiro_Registro():
    return data_2.groupby(by=['record_id']).apply(func_register)
```

No método do primeiro registro, foi utilizado uma função “primeiro_Registro()”, essa função chama a função “func_fregister()” passando o conjunto de linhas pelo “record_id”, a função passa percorrendo as colunas do *dataframe* procurando em cada linha um valor válido para a coluna e atribuindo em um vetor, no fim retorna um *dataframe* com as colunas e os valores correspondentes por um ID apenas.

```
if default_columns == True:
    for method, column in columns.items():
        for especif_column in column:
            data_1 = functions[f'{method}']()
    data_2 = primeiro_Registro()

else:
    data_1 = functions[f'{method}']()
```

Por fim, caso o usuário tenha selecionado as colunas padrões, um laço passará pelo dicionário de métodos e colunas executando o método na coluna respectiva e atribuindo o retorno para o data frame, em sequência o segundo dataframe, recebe o retorno do método de primeiro registro. Caso não selecione as colunas padrões, será executado o método selecionado para a coluna selecionada.

Desvio padrão:

```
def desvio_Padrão():
    def func(x,column):
        try:
            x[column] = x[column].std()
        except IndexError:
            print("")
        return x

    return data.groupby(by=["record_id"]).apply(lambda x: func(x,column))
```

No método do desvio padrão, foi utilizado uma função “desvio_Padrão”, essa função vai pegar a coluna e seus valores, e com isso preencher os dados nulos com o percentil do desvio padrão. A função “std()” é um recurso da biblioteca pandas, é nela que será feito o cálculo do desvio padrão.

Individualização de ID's:

Tem-se também um algoritmo de individualização de ID's, por conta de se ter um mesmo ID, replicado em várias linhas. Com essa individualização, foi possível que muitas dessas linhas, se tornassem apenas 1, deixando apenas 1 ID (um paciente) por linha na nossa tabela de banco de dados.

Algoritmo que foi criado para a individualização de IDs:

```
#@title Algoritmo de individualização de IDs
#@markdown Execute o algoritmo para individualizar sua base de dados

data_1.drop_duplicates(subset='record_id', inplace = True)

data_2.reset_index(drop=True, inplace=True)

data = data_1.merge(data_2, on="record_id")

del data['record_id']

#Essa célula executa a deleção de linhas duplicadas pelo valor id e consequentemente d
```

Algoritmo de individualização de IDs

Execute o algoritmo para individualizar sua base de dados

No algoritmo, primeiramente foi retirado as duplicatas no primeiro *dataframe* pela coluna “record_id”, o mesmo não aconteceu com o segundo dataframe, pois ele já vem com uma linha por ID. Após o mesmo o index do segundo *dataframe* foi resetado para que fosse possível a junção entre os *dataframes* e por fim deletado a coluna “record_id”, pois não será mais útil.

Criação de colunas:

Por conta de ser necessário a coluna de idade da paciente e essa coluna ser inexistente, foi necessária a criação de um algoritmo que fizesse a criação dessa coluna. Isso foi possível pois a data de nascimento da paciente nos foi fornecida, logo, o cálculo da idade com o algoritmo será realizado e uma nova coluna será adicionada à nossa tabela de dados.

Algoritmo que foi criado para a criação de novas colunas:

```
[14]
today = date.today()
ages = []
for x in range(len(data)):
    ages.append(today.year - ((today.month, today.day) < (pd.DatetimeIndex(data['dob'])[x].month, pd.DatetimeIndex(data['dob'])[x].day)))
pd.DataFrame(data).assign(age=ages)
```

Se ao decorrer do projeto for necessária mais alguma coluna, esse algoritmo poderá fazer a criação da mesma.

Link do colab:

https://colab.research.google.com/drive/1iWwr-PupPAMCBQCNtcpE_eyJwcJsKvG

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

Para o tratamento dos dados nulos, utilizamos alguns métodos para deixá-los funcionais e aumentar a qualidade e a precisão das análises finais. Sendo estes: mediana; moda; média; desvio padrão e primeiro registro;.

Mediana: o peso das pacientes registradas na base de dados foi o único valor que tratamos com o uso da mediana. Utilizamos esse método pois as pacientes com câncer de mama perdem peso ao decorrer do processo do tratamento. Com a mediana, chegaremos mais próximos aos valores centrais, que fogem dos extremos tanto de seu peso normal (antes do tratamento) quanto do seu peso com a variação final pela interferência do tratamento.

Moda: utilizamos a moda para as colunas que estão mais relacionadas com a importância na frequência de um mesmo valor de dada situação.

Média: utilizamos a média para valores que não apresentam muita variação e são mais uniformes.

Desvio padrão: para esta categoria utilizamos a feature de IMC, cujo, os dados sem aplicar um cálculo matemático parece não estar uniforme, logo para encontrarmos dados mais consistentes utilizamos o desvio padrão para tornar uniformes.

Primeiro registro: utilizamos o primeiro registro para a maioria das colunas pois, como muitas pacientes apresentam mais de um subtipo, se faz necessário relacionar os dados com um único

subtipo, que no caso será o primeiro a ter sido desenvolvido, uma vez que na nossa avaliação, o histórico de tratamento e desdobramentos deste, é maior, resultando em uma análise mais precisa.

4.4. Modelagem

Algoritmos que foram escolhidos como adequados ao problema e aos dados:

Seleção de features:

Para testar os modelos foram necessária selecionar as features, para isso foi usado o seguinte código:

```
[3] corrrmat = data.corr()
    top_corr_features = corrrmat.index
    #plota
    plt.figure(figsize=(35,35))
    g = sns.heatmap(data[top_corr_features].corr(),
                     annot=True,
                     cmap="Blues")

[4] X = data.drop('follow_up_days', axis = 1).copy()
    y = data['follow_up_days']

[5] f_classif = SelectKBest(score_func = f_classif, k=20)
    fit = f_classif.fit(X,y)

[6] features = fit.transform(X)

[7] print(features)

▶ cols = fit.get_support(indices=True)
```

Link do colab:

https://colab.research.google.com/drive/1X92ilnZf7_TxnhxiH8CQIXERqiMvb9D9

Ao testar os modelos com essas features selecionadas encontramos resultados muito bons, tão bons que resolvemos buscar o que estava dando uma acurácia tão alta, ao investigarmos achamos features que davam cola aos modelos o que influenciava diretamente no resultado final, chamadas de dataliqued. Para resolver esse problema apagamos todas as colunas com dataliqued.

Algoritmos que foram escolhidos para tratar o problema:

Para a resolução da nossa problemática a criação de algoritmos de classificação para nos ajudar a montar uma estrutura que, a partir das variáveis anteriormente selecionadas, trouxessem resultados que pudessem ter uma acurácia satisfatória para o nosso projeto. Os modelos de regressão, ainda que possam trazer bons resultados em outros projetos, não se encaixam na resolução da nossa solução, pois não trazem resultados classificatórios. Esta foi a principal razão para escolhermos algoritmos de classificação, como a árvore de decisão, KNN, Naive Bayes, Random Forest, regressões logísticas .

Processo padrão para fazer os modelos:

Para todos os modelos utilizados, eles contam com etapas padrões, são elas:

Dividir as colunas em variáveis X e Y:

```
# Divide as colunas em variaveis X e y
X = df.drop('follow_up_days', axis=1).copy()
y = df[['follow_up_days']].copy()
```

Nesse código a variável X representa as features selecionadas e a coluna “follow_up_days” é a variável target dos nossos modelos.

Dividir em dados de treino e teste:

```
#dividindo em dados de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Nesta etapa, dividimos as variáveis X em X_treino e X_teste e a variável Y em y_treino e y_teste. Foi definido como métrica padrão de todos os modelos, 80% do data frame como treino e 20% como teste.

Colocamos quatro classificações, sendo elas: 0,1, 2 e 3 para inserção na base de dados.

- Quando o tempo de vida da paciente segue de 0 a 950 dias (representado pelas colunas vermelhas do gráfico) significa um baixo tempo de sobrevida. Classificamos com o numeral 0 essa sequência.
- Quando o tempo de sobrevida segue de 951 a 2470 dias (representado pelas colunas amarelas do gráfico), significa um tempo médio de sobrevida. Classificamos com o numeral 1 essa sequência.
- Quando o tempo de vida da paciente segue de 2471 a 3610 dias (representado pelas colunas verde clara do gráfico) significa um alto tempo de sobrevida. Classificamos com o numeral 2 essa sequência.

- E, por fim, quando o tempo de vida da paciente segue de 3611 dias para cima (representado pelas colunas verde escuras do gráfico) significa um tempo de sobrevida muito alto.
- Classificamos com o numeral 3 essa sequência.

Abaixo uma imagem de como ficou o resultado dessa classificação (0, 1, 2 e 3) no algoritmo para cada paciente da nossa base de dados:

y_train	
	follow_up_classified
2323	1
1918	0
1577	2
1934	1
3860	1
...	...
1033	1
3264	1
1653	1
2607	1
2732	1

Foi criado um algoritmo de flutuação de peso, pois o cliente pediu para que ao invés de mediana na coluna de peso, utiliza-se a seguinte fórmula: $(n/n-1)$, isto é, utilizar o peso atual da paciente, dividindo-o pelo peso anterior, com isso a flutuação é obtida e pode ter como resultado um número menor que 1, ou maior que 1. Os valores obtidos pelo resultado desta divisão podem ser tanto negativos como positivos. Logo, a conclusão que pode-se tirar dessa fórmula, é que se for obtida uma flutuação positiva, é porque a paciente teve um ganho de peso, já se for um valor negativo, é porque a paciente teve perda de peso.

Portanto, no novo algoritmo criado foi utilizada a biblioteca de regressão linear. Com esse algoritmo, é possível coletar todos os valores da coluna de peso de cada paciente por ID, e com isso, fazer o cálculo da flutuação de peso da paciente.

Primeiramente, foi criada uma variável que recebe as colunas que serão utilizadas para o cálculo de regressão linear:

```
# Aqui está recebendo e armazenando os dados das colunas record_id e weight
dfweight = data[['record_id', 'weight']]

# Exclusão dos dados nulos
dfweight.dropna(inplace=True)

# Mostra a quantidade de linhas
dfweight.head(100)
```

Após isso, foi importada a biblioteca de regressão linear, para que assim, fosse possível o cálculo da flutuação de peso de cada paciente:

```
# Importando a biblioteca de regressão linear
from sklearn.linear_model import LinearRegression

# Função que vai realizar a flutuação
def flutuacao(data):
    X = np.array(range(1,len(data.weight)+1)) # Percorre os dados de treinamento
    y = data.weight # Dados da coluna weight, ou seja, são os valores alvos

    # Linha de código que vai aplicar a regressão linear nos valores alvos
    reg = LinearRegression().fit(X.reshape(-1, 1), y)
    return reg.coef_[0]

dfweight.groupby(by='record_id').apply(flutuacao)
# Aplicação da flutuação pela individualização de ID

record_id
54      -0.046090
302      0.258462
710      0.039396
752      -0.027273
1589     0.182719
        ...
82057     1.470000
82059     0.738571
82122     0.040000
```

Logo, nota-se que alguns “record_id” tiveram resultados positivos, ou seja, pacientes que tiveram um aumento de peso, e outros tiveram resultados negativos, ou seja, pacientes que tiveram perda de peso.

Modelo de Naive Bayes:

Estratégia de avaliação escolhida:

O algoritmo de Naive Bayes Gaussiano analisa as características de uma base de dados assumindo que as features são independentes entre si. Além disso, ele também assume que as variáveis features são todas igualmente importantes para o resultado. Em cenários em que isso não ocorre, essa técnica deixa de ser a opção ideal, por isso a baixa acurácia desse modelo no projeto, entretanto resolvemos testá-lo para ter um embasamento maior.

Nesse código o modelo é instanciado e treinado no método Naive Bayes:

```
nayve = GaussianNB()

#fit
nayve.fit(X_train, y_train)

#faz predições
predictions = nayve.predict(X_test)
```

Resultados preliminares obtidos:

Ao rodar o modelo 10 vezes a acurácia variou de 59% a 65% sem o uso de hiperparâmetros. Já com o uso de hiperparâmetros a acurácia variou de 58% a 67%.

A precisão do modelo variou de 49% a 54% na classe 0, de 54% a 68% na classe 1, de 65% a 70% na classe 2. Já com o uso de hiperparâmetros a precisão da classe 0 variou de 50% a 53% a classe 1 variou de 55% a 58%, a classe 2 variou de 65% a 68%.

Resultados obtidos sem o uso de hiperparâmetros:

```
#Avaliando sem o uso de hiperparâmetro
print('Classification metrics: \n', classification_report(y_test, predictions))

#Medindo a taxa de erro do modelo

mse = metrics.mean_squared_error(y_test, predictions)

print('MSE (Mean-Squared-Error): %s' %mse)

Classification metrics:
    precision    recall    f1-score   support
 0          0.49     0.54     0.51      183
 1          0.65     0.70     0.67      369
 2          0.69     0.57     0.62      244

    accuracy                           0.62      796
   macro avg       0.61     0.60     0.60      796
weighted avg       0.63     0.62     0.62      796

MSE (Mean-Squared-Error): 0.5753768844221105
```

Nesse código é mostrado a precisão, acurácia, recall e f1-score. Nota-se uma acurácia de 62%.

Resultados obtidos com o uso de hiperparâmetros:

```
#Avaliando com o uso de hiperparâmetro
print('Classification metrics: \n', classification_report(y_test,new_predictions))

#Medindo a taxa de erro do modelo

mse = metrics.mean_squared_error(y_test, new_predictions)

print('MSE (Mean-Squared-Error): %s' %mse)

Classification metrics:
    precision    recall    f1-score   support
 0          0.52     0.24     0.33      183
 1          0.57     0.79     0.66      369
 2          0.68     0.55     0.61      244

    accuracy                           0.59      796
   macro avg       0.59     0.53     0.53      796
weighted avg       0.59     0.59     0.57      796

MSE (Mean-Squared-Error): 0.5113065326633166
```

Nesse código é mostrado a precisão, acurácia, recall e f1-score.
Com o uso de hiperparâmetros, a acurácia diminuiu de 62% para 59%.

Configuração de hiperparâmetros do modelo:

Na estatística Bayesiana, um hiperparâmetro é um parâmetro de uma distribuição anterior; o termo é usado para distingui-los dos parâmetros do modelo para o sistema subjacente em análise.

Código para implementar os hiperparâmetros:

```
#hiperparametros
params_NB = {'var_smoothing': np.logspace(0,-9, num=100)}
gs_NB = GridSearchCV(estimator=nayve,
                      param_grid=params_NB,
                      cv=3,
                      verbose=1,
                      scoring='accuracy')
gs_NB.fit(X_train, y_train)

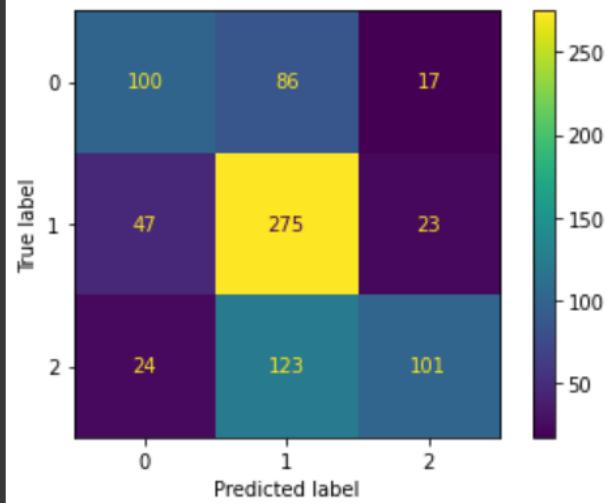
gs_NB.best_params_
Fitting 3 folds for each of 100 candidates, totalling 300 fits
{'var_smoothing': 0.1}
```

Nesse código o modelo é treinado, observe que a acurácia do modelo diminuiu após a implementação dos hiperparâmetros de 57% para 59% de acurácia.

Matriz de confusão:

```
#Exibindo a matriz de confusão dos resultados preditos
ConfusionMatrixDisplay(confusion_matrix(y_test, predictions)).plot()

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fc9116f4650>
```



Curva Roc:

```
#Construindo a base do Pipeline
pipe_rf = Pipeline([('nayve_tuned', GaussianNB(var_smoothing=gs_NB.best_params_['var_smoothing']))])

#Treinando o modelo com Pipeline
model = pipe_rf.fit(X_train, y_train)

#Calculando y_score
y_score = model.predict_proba(X_test)

#Declarando valores únicos de classes em ordem
n_binaries = y_test.unique()
n_binaries.sort()

#Binarizando a saída
y_test_bin = label_binarize(y_test, classes=n_binaries)
n_classes = y_test_bin.shape[1]
```

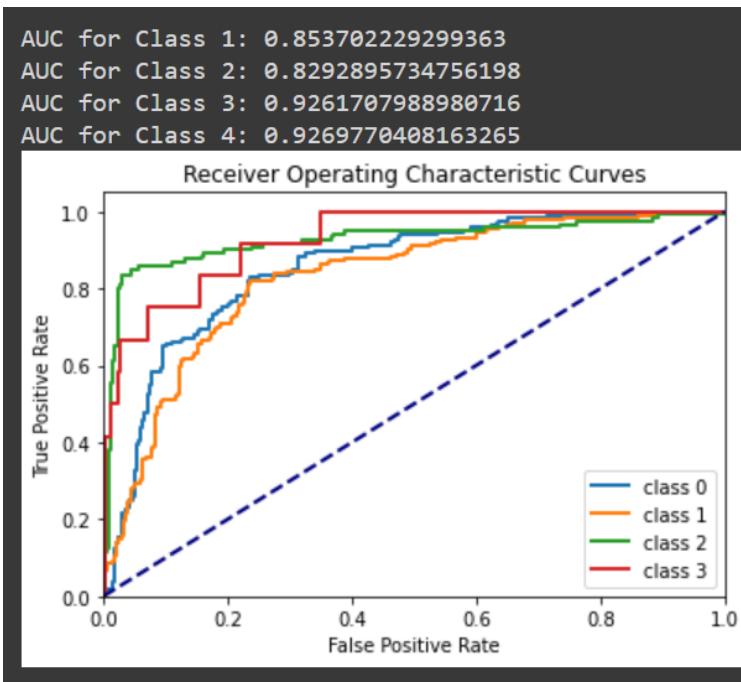
Nesse código são passados alguns parâmetros para a instância do modelo.

```
#Calculando e exibindo curva ROC das classes
fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i])
    plt.plot(fpr[i], tpr[i], lw=2, label='class {}'.format(i))
    print('AUC for Class {}: {}'.format(i+1, auc(fpr[i], tpr[i])))

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--' )
plt.legend(loc="lower right")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic Curves')
plt.show()
```

Esse é o código que exibe a curva Roc do modelo de Naive Bayes.



Esse é o resultado obtido da curva Roc.

Modelo de Regressão Logística:

Estratégia de avaliação escolhida:

A regressão logística é um modelo estatístico usado para determinar a probabilidade de um evento ocorrer. Esse modelo mostra a relação entre os recursos e, em seguida, calcula a probabilidade de um determinado resultado. A regressão logística é usada no aprendizado de máquina (ML) para ajudar a criar previsões precisas.

Além disso, a regressão logística é utilizada para problemas de classificação, os dados são passados para a máquina, a qual aprende com os dados históricos, para assim, realizar a avaliação.

Primeiramente foi feita uma amostragem dos dados, além disso, também foi definido o tamanho das amostras, que é 80% treino e 20% teste.

Amostragem dos dados

```
[ ] # Divide as colunas em variaveis X e y
X = df.drop('follow_up_days', axis=1).copy()
y = df[['follow_up_days']].copy()

# Definindo o tamanho das amostras
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
```

Resultados preliminares obtidos:

Na criação da máquina preditiva com regressão logística, foi realizada a criação do modelo e o seu treinamento, com isso, foi possível obter o resultado da acurácia nos dados de teste:

Criando máquina preditiva com logística

```
# Criação do modelo
modelo = LogisticRegression()

# Treinamento do modelo
modelo.fit(X_train, y_train)

y_predict = modelo.predict(X_test)

# Score do modelo nos dados de teste
result = modelo.score(X_test, y_test)
print("Acurácia nos dados de teste: %.3f%%" % (result * 100.0))

Acurácia nos dados de teste: 60.05%
```

A acurácia obtida com esse modelo de regressão logística foi de 60.05%

Avaliação da máquina sem hiperparâmetros:

Avaliando a máquina sem hiperparâmetros

```
[ ] #Avaliando a máquina

print('Classification metrics: \n', classification_report(y_test, y_predict))

#Medindo a taxa de erro do modelo
mse = metrics.mean_squared_error(y_test, y_predict)
print('MSE (Mean-Squared-Error): %s' %mse)

Classification metrics:
precision    recall   f1-score   support
          0       0.65      0.32      0.43      225
          1       0.54      0.80      0.64      327
          2       0.73      0.59      0.65      244

accuracy                           0.60      796
macro avg       0.64      0.57      0.58      796
weighted avg    0.63      0.60      0.59      796

MSE (Mean-Squared-Error): 0.49748743718592964
```

Nota-se uma acurácia de 60%, sem o uso de hiperparâmetros.

Avaliação da máquina com hiperparâmetros:

Avaliando a máquina com hiperparâmetros

```
[ ] # Avaliando a máquina com hiperparâmetros

print('Classification metrics: \n', classification_report(y_test, y_pred))

#Medindo a taxa de erro do modelo
mse = metrics.mean_squared_error(y_test, y_predict)
print('MSE (Mean-Squared-Error): %s' %mse)

Classification metrics:
precision    recall  f1-score   support
          0       0.70      0.46      0.55      225
          1       0.60      0.79      0.68      327
          2       0.71      0.63      0.67      244

accuracy                           0.65      796
macro avg       0.67      0.62      0.63      796
weighted avg    0.66      0.65      0.64      796

MSE (Mean-Squared-Error): 0.49748743718592964
```

Nesse código é possível observar que a acurácia do modelo aumentou após a implementação dos hiperparâmetros de 60% para 65% de acurácia.

Configuração de hiperparâmetros do modelo:

Para a configuração de hiperparâmetros foram utilizados os seguintes parâmetros:

```
from sklearn.model_selection import GridSearchCV

modelo = LogisticRegression()

# Cria o GridSearchCV

parameters = {'penalty':['l1', 'l2', 'elasticnet', 'none'],
              'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']}

modelGS = GridSearchCV(modelo, parameters)

# Treina os modelos e guarda na variável modelGS o melhor modelo
modelGS.fit(X_train, y_train)
modelGS.best_params_
```

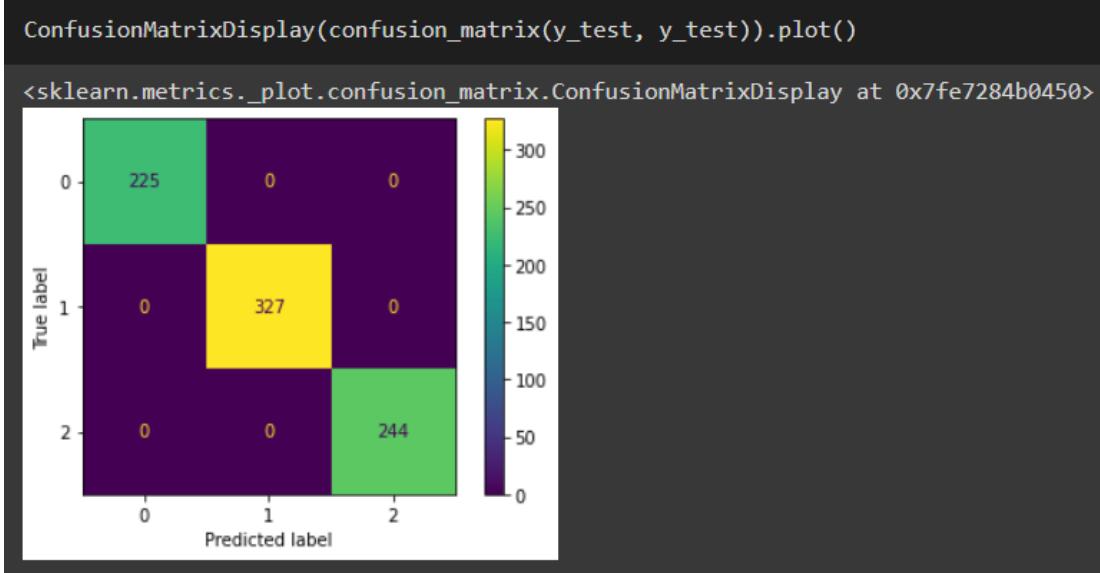
```
[12] # import métricas
from sklearn.metrics import recall_score, precision_score, f1_score

y_pred = reglog_best.predict(X_test)
print('Acc treino: ', reglog_best.score(X_train, y_train ))
print('Acc teste: ', reglog_best.score(X_test, y_test.squeeze() ))
print( 'Revocação: ', recall_score(y_test, y_pred, average='micro'))
print( 'Precisão: ', precision_score( y_test, y_pred, average='micro' ))
print( 'F1_score: ', f1_score(y_test, y_pred, average='micro' ))

Acc treino: 0.6510531279471864
Acc teste: 0.6482412060301508
Revocação: 0.6482412060301508
Precisão: 0.6482412060301508
F1_score: 0.6482412060301508
```

Portanto, é evidente que com o uso de hiperparâmetros, foi possível ter uma melhora no resultado da acurácia, atingindo um resultado de 65%.

Matriz de confusão:



Para uma melhor avaliação, foi gerado um gráfico de matriz de confusão, o qual evidenciou que o modelo mais acerta do que erra apresentando grandes taxas de acerto para a classe 1, um número de acerto mediano para as classes 0 e 2.

Modelo de K-Nearest Neighbors(KNN) :

Estratégia de avaliação escolhida:

O KNN é um algoritmo de aprendizado de máquinas supervisionado, sendo utilizado tanto para tarefas de classificação, este algoritmo busca classificar cada amostra de um conjunto de dados medindo a distância do vizinho mais próximo, ou seja, da amostra próxima. Como o modelo solicitado pelo cliente é trazer um resultado classificatório da predição, consideramos o KNN como uma boa alternativa para o modelo preditivo.

Primeiramente foi feita uma amostragem dos dados, além disso, também foi definido o tamanho das amostras, que é 80% treino e 20% teste.

```
[ ] #Definindo qual será a variável target  
X = data.drop(['follow_up_classified'], axis = 1)  
y = data['follow_up_classified']  
  
[ ] #Definição de treino e teste  
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=4)
```

Resultados preliminares obtidos:

Nesse código o modelo é instanciado e treinado no método KNN, tendo o hiperparâmetro definido de forma aleatória:

```
#Número de vizinhos  
k = 19  
  
#Treinando o modelo  
neigh = KNeighborsClassifier(n_neighbors = k)  
neigh.fit(X_train,y_train)  
Pred_y = neigh.predict(X_test)
```

O algoritmo preliminar obtém um resultado de 61% de acurácia, sendo gerado os seguintes indicadores:

```
#Avaliando sem o uso de hiperparâmetros
print('Classification metrics: \n', classification_report(y_test, Pred_y))

#Medindo a taxa de erro do modelo

mse = metrics.mean_squared_error(y_test, Pred_y)

print('MSE (Mean-Squared-Error): %s' %mse)

Classification metrics:
    precision    recall    f1-score   support
    0            0.56     0.53      0.54      229
    1            0.60     0.69      0.65      347
    2            0.67     0.55      0.60      220

    accuracy                           0.61      796
   macro avg             0.61     0.59      0.60      796
weighted avg            0.61     0.61      0.60      796

MSE (Mean-Squared-Error): 0.5829145728643216
```

Configuração de hiperparâmetros do modelo:

Para a configuração de hiperparâmetros foi utilizado o K número de vizinhos, para isso foi feito um algoritmo que testasse o K em um determinado intervalo de números e a partir desses resultados mostraria dois gráficos: o primeiro indicando qual a margem de erro mínima para aquele K e o segundo mostrando a acurácia do modelo para aquele K.

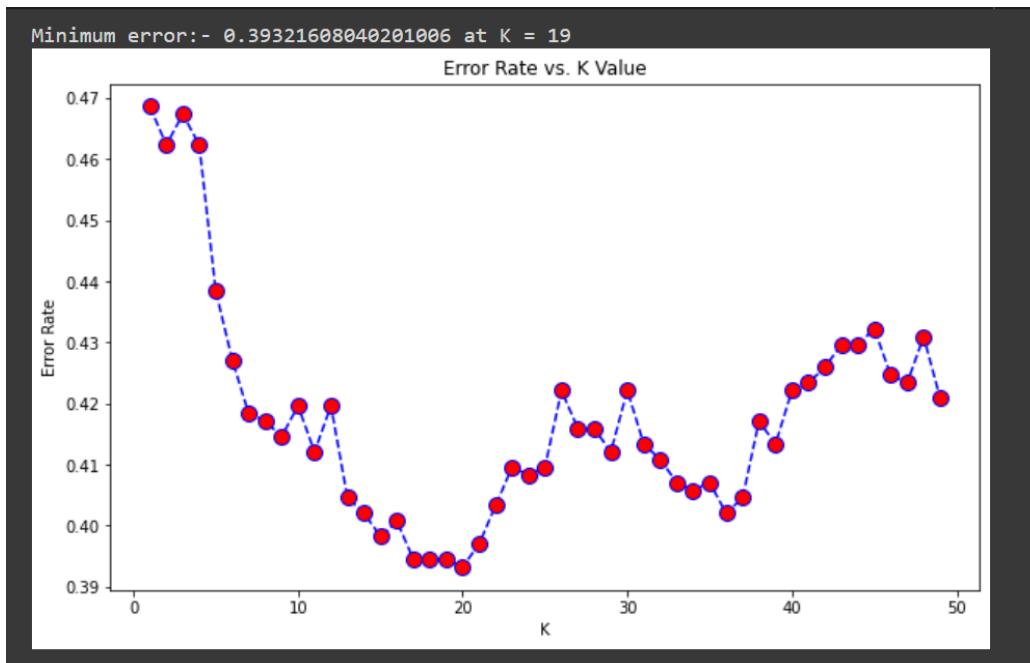
Para isso foi necessário criar um loop que percorre entre os intervalos selecionados e insere nesse loop o modelo KNN, no qual, cada vez que o loop conclui o K será um dos números do intervalo selecionado.

```
#Medir o erro de cada valor K

#Adicionar na variável error_rate o tanto de erro
error_rate = []
for i in range(1,50):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))

#Imprimir gráfico
plt.figure(figsize=(10,6))
plt.plot(range(1,50),error_rate,color='blue', linestyle='dashed',
         marker='o',markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
print("Minimum error:-",min(error_rate),"at K =",error_rate.index(min(error_rate)))
```

Ao medir o erro do intervalo de 1 a 50, o menor erro foi quando K era igual a 19, conforme foi aumentando o intervalo de números como 1 a 4000 a margem de erro sofria variações subindo e descendo até estabilizar a linha do gráfico, entretanto, para inserir esse intervalo todo não iria ser possível realizar a plotagem desse gráfico, também é importante ressaltar que mesmo com intervalos maiores, o K ideal continuou sendo 19.



Seguindo a mesma lógica do parágrafo descrito anteriormente, para esse código foi medido a acurácia conforme o loop é realizado trocando os valores de K, logo percebe-se que

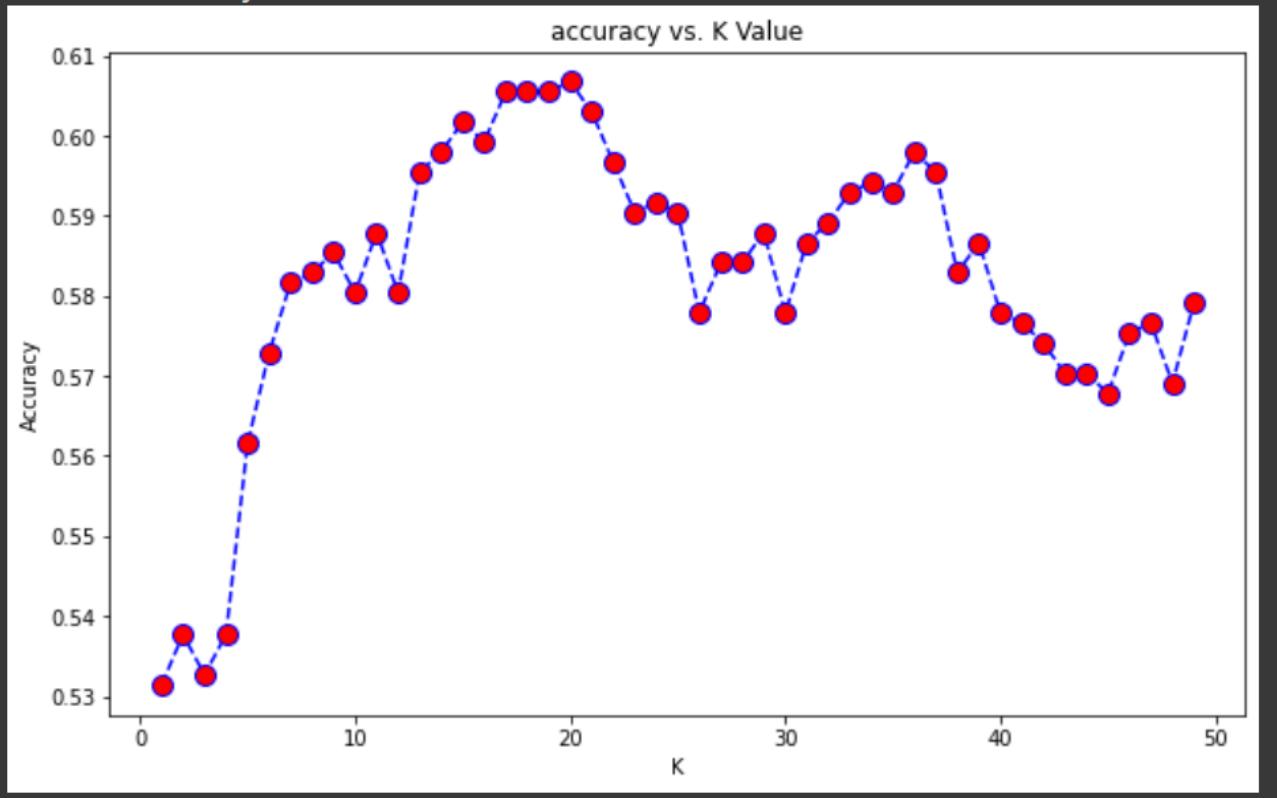
quanto maior o número de K, a partir de 19, menor será a acurácia do modelo, que irá se estabilizar.

```
#Medir a acurácia de cada valor K e adicionar na lista acc
acc = []

#Medir acurácia dos vizinhos de 1 a 50
for i in range(1,50):
    neigh = KNeighborsClassifier(n_neighbors = i).fit(X_train,y_train)
    yhat = neigh.predict(X_test)
    acc.append(metrics.accuracy_score(y_test, yhat))

#Imprimir gráfico
plt.figure(figsize=(10,6))
plt.plot(range(1,50),acc,color = 'blue',linestyle='dashed',
         marker='o',markerfacecolor='red', markersize=10)
plt.title('accuracy vs. K Value')
plt.xlabel('K')
plt.ylabel('Accuracy')
print("Maximum accuracy:-",max(acc),"at K =",acc.index(max(acc)))
```

Maximum accuracy:- 0.6067839195979899 at K = 19



Seguindo a mesma lógica do parágrafo descrito anteriormente, para esse código foi medido a acurácia conforme o loop é realizado trocando os valores de K, logo percebe-se que quanto maior o número de K, menor será a acurácia do modelo, que irá se estabilizar.

Também, foi utilizado o GridSearch para determinar quais são os melhores parâmetros do modelo KNN, sendo selecionado os seguintes parâmetros: Número de vizinhos próximos, peso, cujo, torna mais importante os mais próximos e menos importante os mais distantes, e métrica que será o cálculo realizado para medir as distâncias.

```

metric_params = {'n_neighbors': [15,16,17,18,19,20,21],
                 'weights': ['uniform','distance'],
                 'metric': ['euclidean','manhattan']}
}
gs = GridSearchCV(neigh,
                  metric_params,
                  verbose = 1,
                  cv = 100,
                  n_jobs = -1)

gsresult = gs.fit(X_train,y_train)

Pred_y = gs.predict(X_test)

```

Gerando o seguinte resultado:

```

#Avaliando com o uso de hiperparâmetros
print('Classification metrics: \n', classification_report(y_test, Pred_y))

#Medindo a taxa de erro do modelo

mse = metrics.mean_squared_error(y_test, Pred_y)

print('MSE (Mean-Squared-Error): %s' %mse)

Classification metrics:
      precision    recall  f1-score   support
          0       0.57      0.51      0.54      229
          1       0.59      0.71      0.64      347
          2       0.68      0.55      0.61      220

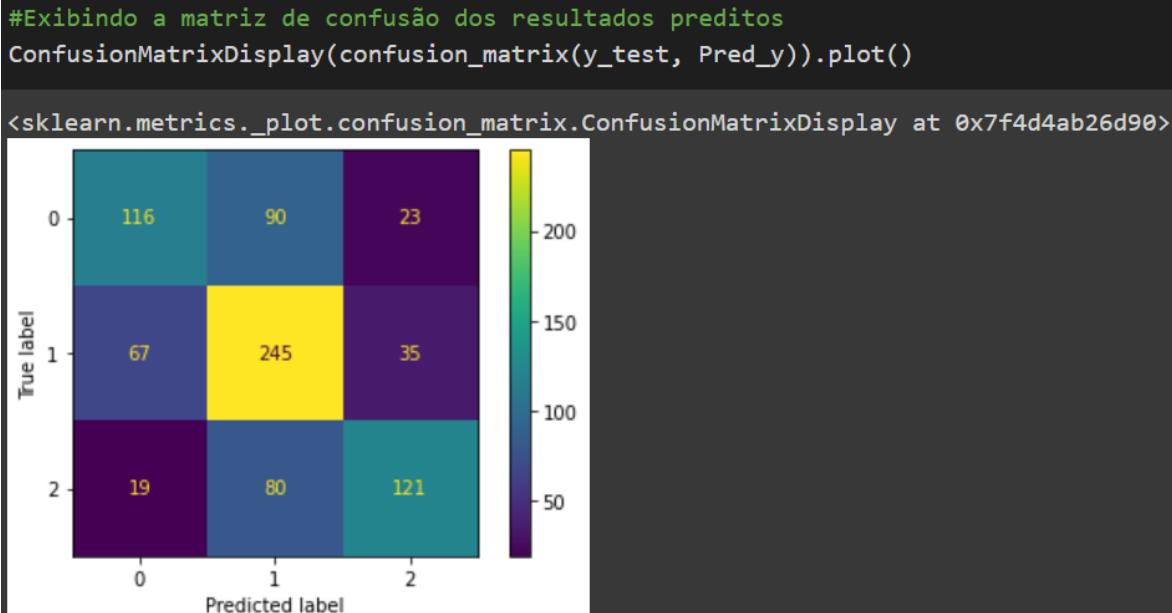
      accuracy                           0.61      796
     macro avg       0.61      0.59      0.60      796
weighted avg       0.61      0.61      0.60      796

MSE (Mean-Squared-Error): 0.5527638190954773

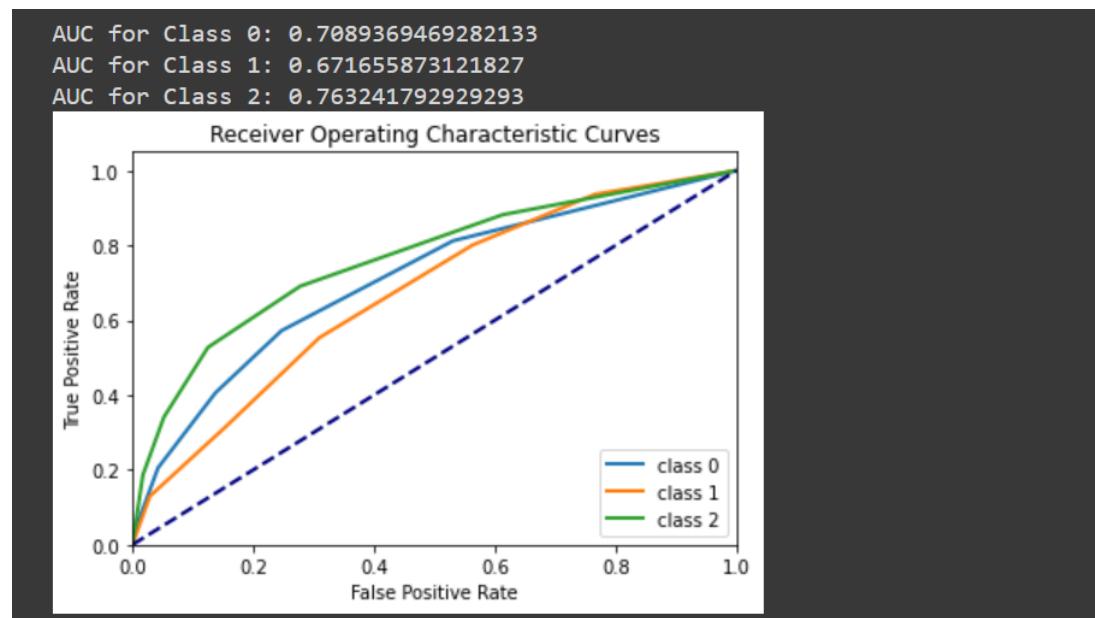
```

Sendo assim, nota-se que continua apresentando a mesma acurácia que antes sem os hiperparâmetros, entretanto a classificação 0 e 1 se tornaram mais precisas, enquanto a classe 2 diminuiu.

Para a avaliar melhor foi gerado dois gráficos que irá verificar a acurácia do modelo, sendo a matriz de risco que evidenciou que o modelo mais acerta do que erra apresentando grandes taxas de acerto para a classe 1, já para as classes 0 e 2 obteve um número de acerto mediano e para a classe 3 apresentou uma taxa baixa de acerto.



Já na curva ROC foi analisada o quanto as previsões estão corretas, mostrando a porcentagem de acerto da curva e não trabalhando com números absolutos, com isso percebemos que para classificação da classe 0 e 2 são bem assertivas, enquanto a classe 1 é menos assertivas, apresentando uma taxa mediana, cuja é de 67%, enquanto a 0 e 2 apresentam taxas muito boas sendo 70% e 76% respectivamente.



Modelo *Random Forest*:

Estratégia de avaliação escolhida:

Resumidamente, o algoritmo cria de forma aleatória várias Árvores de Decisão (*Decision Trees*) e combina o resultado de todas elas para chegar no resultado final. Por sua vez, árvores de decisão são estruturas de predição mais simples que, de maneira geral, criam uma estrutura parecida com uma árvore onde os ramos são os diferentes “caminhos” que o algoritmo toma para chegar no valor previsto.

Uma vantagem de utilizar o algoritmo *Random Forest* é que, além dele ser muito poderoso, ele retorna de maneira muito comprehensiva a importância atribuída para cada variável independente na hora de realizar as predições. Com isso, é possível medir o impacto de cada questão no resultado final. Entretanto, a desvantagem é que por ser um algoritmo de aprendizado supervisionado, é preciso que seja fornecida uma variável resposta, ou seja, uma única variável pode ser prevista.

O código abaixo instancia o modelo de árvore aleatória, treina o modelo e realiza predições pré hiper parametrização.

```
#Declarando modelo classificatório  
crf = RandomForestClassifier(random_state=32)  
  
#Treinando o modelo  
crf.fit(X_train, y_train)  
  
#Fazendo previsões  
predictions = crf.predict(X_test)
```

Resultados preliminares obtidos:

Ao rodar o modelo 10 vezes a acurácia variou de 65% a 72% sem o uso de hiperparâmetros. Já com o uso de hiperparâmetros a acurácia variou de 68% a 74%. Vamos observar os resultados abaixo.

A precisão do modelo variou de 76% a 78% na classe 0, na classe 1 de 65% a 70% , de 68% a 70% na classe 2. Já com o uso de hiperparâmetros a precisão do modelo variou de 77% a 82% na classe 0, de 65% a 69% na classe 1, de 65% a 70% na classe 2.

Resultados obtidos sem o uso de hiperparâmetros:

```
#Avaliando sem o uso de hiperparâmetros
print('Classification metrics: \n', classification_report(y_test, predictions))
```

```
Classification metrics:
  precision    recall  f1-score   support

    0       0.78      0.53      0.63      200
    1       0.69      0.81      0.74      351
    2       0.69      0.70      0.69      245

  accuracy                           0.70      796
  macro avg       0.72      0.68      0.69      796
weighted avg       0.71      0.70      0.70      796
```

Nesse código é mostrado a precisão, acurácia, *recall* e *f1-score*. Nota-se uma acurácia de 70%. Vemos que há uma baixa variação de precisão entre as classes 0, 1 e 2.

Resultados obtidos com o uso de hiperparâmetros:

```
#Avaliando com o uso de hiperparâmetros
print('Classification metrics: \n', classification_report(y_test, predictions))
```



```
Classification metrics:
  precision    recall  f1-score   support

    0       0.80      0.51      0.62      200
    1       0.67      0.82      0.74      351
    2       0.69      0.67      0.68      245

  accuracy                           0.70      796
  macro avg       0.72      0.67      0.68      796
weighted avg       0.71      0.70      0.69      796
```

Nesse código é mostrado a precisão, acurácia, *recall* e *f1-score*. Nota-se que a acurácia do modelo com hiperparâmetro é igual a do modelo sem o uso de hiperparâmetro que é de 70%. Entretanto, a precisão do modelo melhora ao usar os hiperparâmetros.

Configuração de hiperparâmetros do modelo:

Hiperparâmetros são configurações que não podem ser aprendidas a partir dos dados regulares que fornecemos ao algoritmo; eles são integrados ao algoritmo e cada algoritmo tem seu próprio conjunto predefinido de hiperparâmetros. Para este caso utilizamos o *RandomizedSearchCV*, pois ele é mais viável em relação custo de processamento no modelo *Random Forest*.

Código para implementar os hiperparâmetros:

```
# Números de árvore do modelo
n_estimators = [int(x) for x in np.linspace(start = 0, stop = 200, num = 100)]
# Número de funcionalidade para serem consideradas
max_features = ['auto', 'sqrt']
# Número máximo da profundidade da árvore
max_depth = list(range(0, 19))
# Número mínimo de exemplos requeridos para o split
min_samples_split = list(range(2, 20))
# Número mínimo de exemplos requeridos para a folha
min_samples_leaf = list(range(1, 20))
# Método de seleção para treinar cada árvore
bootstrap = [True, False]

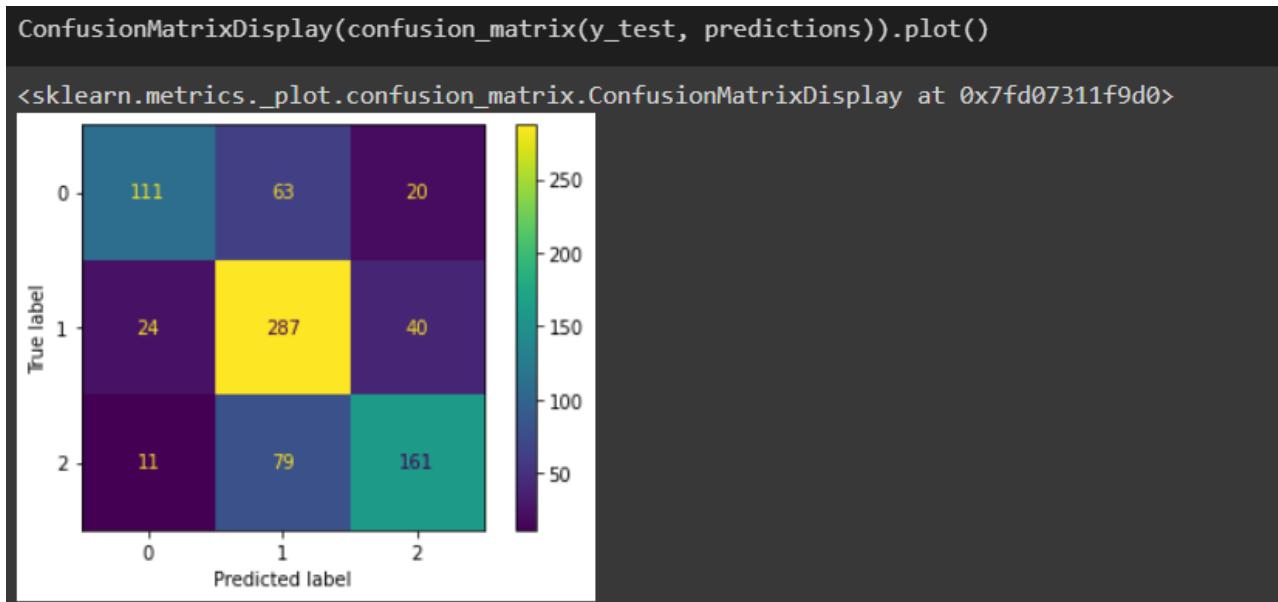
117] #Declarando parâmetros da busca randomizada

param_grid = {'n_estimators': n_estimators,
              'max_features': max_features,
              'max_depth': max_depth,
              'min_samples_split': min_samples_split,
              'min_samples_leaf': min_samples_leaf,
              'bootstrap': bootstrap}

118] #Declarando modelo com os parâmetros da busca randomizada
rf_RandomGrid = RandomizedSearchCV(estimator = crf, param_distributions = param_grid, cv = 100, verbose=0)
```

Nesse código o modelo é treinado, observe que a acurácia do modelo aumentou após a implementação dos hiperparâmetros de 87% para 88%, houve uma baixa variação e isso é constante, constatasse o baixo impacto da hiper parametrização do modelo *Random Forest*.

Matriz de confusão:



Curva Roc:

```
[123] #Construindo a base do Pipeline
    pipe_rf = Pipeline([('rf_RandomGrid', RandomForestClassifier(random_state=123))])

[124] #Treinando o modelo com Pipeline
    model = pipe_rf.fit(X_train, y_train)

[125] #Calculando y_score
    y_score = model.predict_proba(X_test)

▶ #Declarando valores únicos de classes em ordem
n_binaries = y_test.unique()
n_binaries.sort()

#Binarizando a saída
y_test_bin = label_binarize(y_test, classes=n_binaries)
n_classes = y_test_bin.shape[1]
```

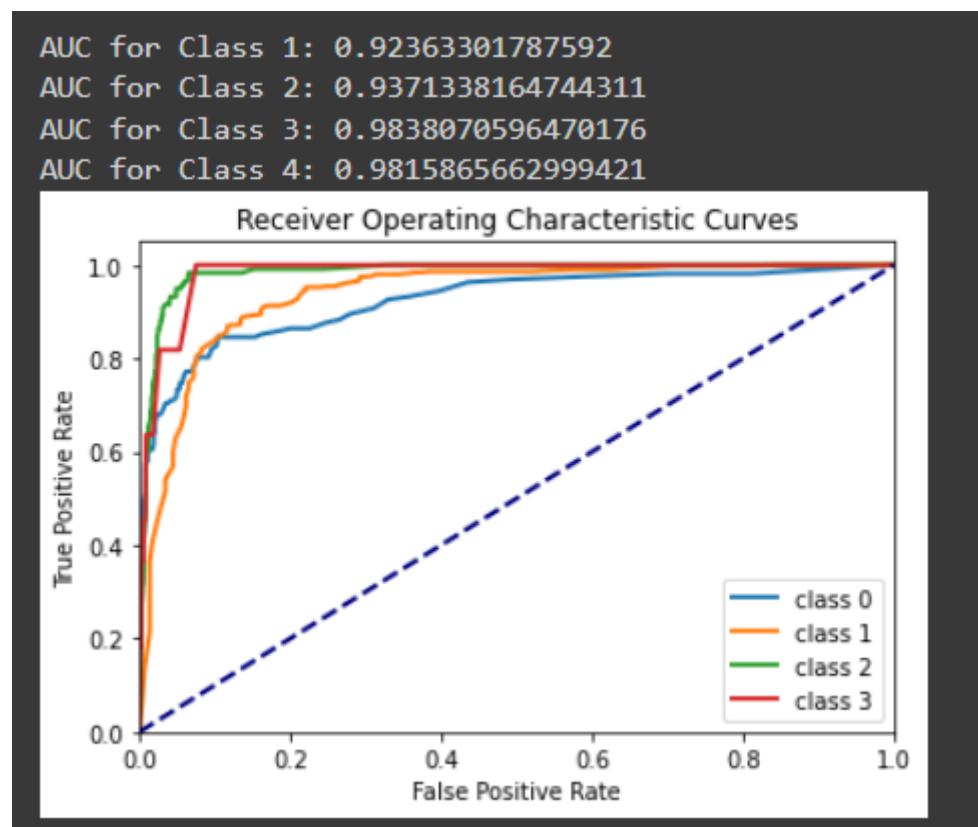
Nesse código são passados alguns parâmetros para a instância do modelo.

```
▶ #Calculando e exibindo curva ROC das classes
fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i])
    plt.plot(fpr[i], tpr[i], lw=2, label='class {}'.format(i))
    print('AUC for Class {}: {}'.format(i+1, auc(fpr[i], tpr[i])))

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.legend(loc="lower right")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic Curves')
plt.show()
```

Esse é o código que exibe a curva Roc do modelo de Naive Bayes.



Resultado obtido da curva Roc, observamos que as previsões da classe 0 e 1 são as mais prejudicadas pelo modelo.

Modelo Árvore de Decisão:

Estratégia de avaliação escolhida:

O algoritmo de árvore de decisão funciona a partir de um conjunto de escolhas provenientes do conjunto de dados e seu comportamento. De modo geral, obtemos o resultado a partir de uma coluna *target*, que será o primeiro nó principal a compor a árvore e, a partir desse dado, as ramificações serão criadas pelo próprio algoritmo que identifica quais outras sequências de dados são mais relacionais. Uma das razões mais importantes para usar esse modelo é a facilidade na interpretação dos resultados que ele fornece. Além disso, por ser um algoritmo também de classificação, se adequa ao objetivo da nossa solução, anteriormente proposta pelo cliente. Para tanto, instanciamos o Decision Tree e usamos 80% dos dados para treino e 20% para teste, conforme apresentado pelo código abaixo:

```
[6] # Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=123)

▶ # Create Decision Tree classifier object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifier
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

Resultados preliminares obtidos:

Sem utilizar os hiperparâmetros, aplicando o treino e teste, a precisão do modelo variou de 49% a 60% na classe 0, de 61% a 65% na classe 1 e de 60% a 65% na classe 2. O resultado da acurácia, sem a utilização dos hiperparâmetros, resultou em 60%, como apresentado no código abaixo:

```
#Avaliando sem o uso de hiperparâmetros
print('Classification metrics: \n', classification_report(y_test, y_pred))

#Medindo a taxa de erro do modelo

mse = metrics.mean_squared_error(y_test, y_pred)

print('MSE (Mean-Squared-Error): %s' %mse)

Classification metrics:
precision    recall  f1-score   support
          0       0.60      0.49      0.54     225
          1       0.61      0.65      0.63     327
          2       0.60      0.65      0.62     244

accuracy                           0.60      796
macro avg       0.60      0.60      0.60      796
weighted avg    0.60      0.60      0.60      796

MSE (Mean-Squared-Error): 0.6331658291457286
```

No entanto, já com a utilização de hiperparâmetros, aplicando o treino e teste, o resultado da acurácia foi de 60% para 81%, como apresentado no código abaixo:

```
#Avaliando com o uso de hiperparâmetros
print('Classification metrics: \n', classification_report(y_test, y_pred))

#Medindo a taxa de erro do modelo

mse = metrics.mean_squared_error(y_test, y_pred)

print('MSE (Mean-Squared-Error): %s' %mse)

Classification metrics:
      precision    recall  f1-score   support
          0       0.90      0.66      0.76     225
          1       0.75      0.92      0.83     327
          2       0.85      0.80      0.82     244

      accuracy                           0.81      796
     macro avg       0.83      0.79      0.80      796
weighted avg       0.82      0.81      0.81      796

MSE (Mean-Squared-Error): 0.2851758793969849
```

Configuração de hiperparâmetros do modelo:

Ao utilizar o algoritmo de árvore de decisão, um hiperparâmetro é a otimização pela definição da profundidade da árvore e, também, pela quantidade de ramificações que ela apresentará. Para essa otimização, utilizamos o *Grid Search*. O código abaixo mostra a implementação dos hiperparâmetros:

```
grid_search_cv = GridSearchCV(
    estimator = DecisionTreeClassifier(random_state = 37),
    param_grid = {
        'max_depth':range(1,50)
    }, # Testando comprimentos máximos de 1 a 50
    scoring='roc_auc',
    cv = 3)

# Realizando a otimização por GridSearch para os dados de cancer de mama:
grid_search_cv.fit(X,y)
#Vamos ver informações relevantes:
print('Melhor comprimento máximo: {}'.format(grid_search_cv.best_params_['max_depth']))
print('Desempenho AUC-ROC do melhor modelo: {}'.format(round(grid_search_cv.best_score_,3)))
print('Tempo para realizar a otimização:')
%timeit -n 1 -r 1 grid_search_cv.fit(X,y)
```

```

grid_search_cv = GridSearchCV(estimator = DecisionTreeClassifier(random_state = 37),
                             param_grid = {
                                 'max_depth':range(1,50)
                             }, # Testando comprimentos máximos de 1 a 50
                             scoring='f1_macro', cv = 3)

# Realizando a otimização por GridSearch para os dados de cancer de mama:
grid_search_cv.fit(X,y)
#Vamos ver informações relevantes:
print('Melhor comprimento máximo: {}'.format(grid_search_cv.best_params_['max_depth']))
print('Desempenho AUC-ROC do melhor modelo: {}'.format(round(grid_search_cv.best_score_,3)))
print('Tempo para realizar a otimização:')
%timeit -n 1 -r 1 grid_search_cv.fit(X,y)

Melhor comprimento máximo: 10
Desempenho AUC-ROC do melhor modelo: 0.589
Tempo para realizar a otimização:
3.77 s ± 0 ns per loop (mean ± std. dev. of 1 run, 1 loop each)

```

Matriz de confusão:

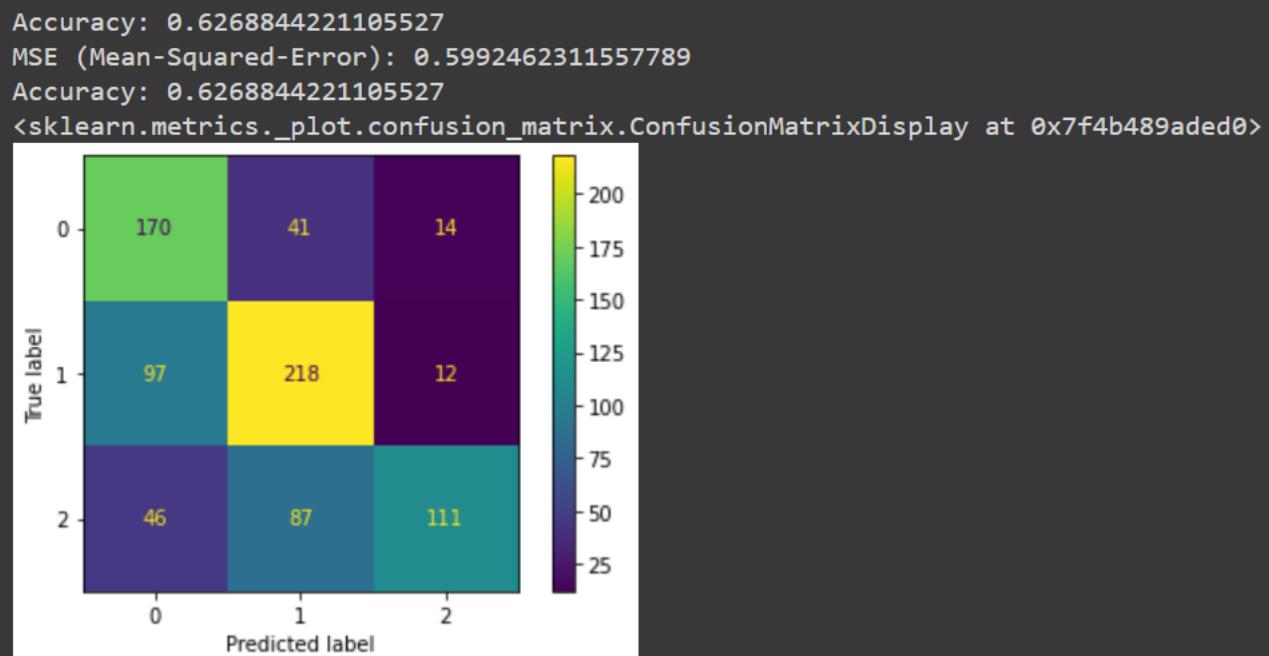
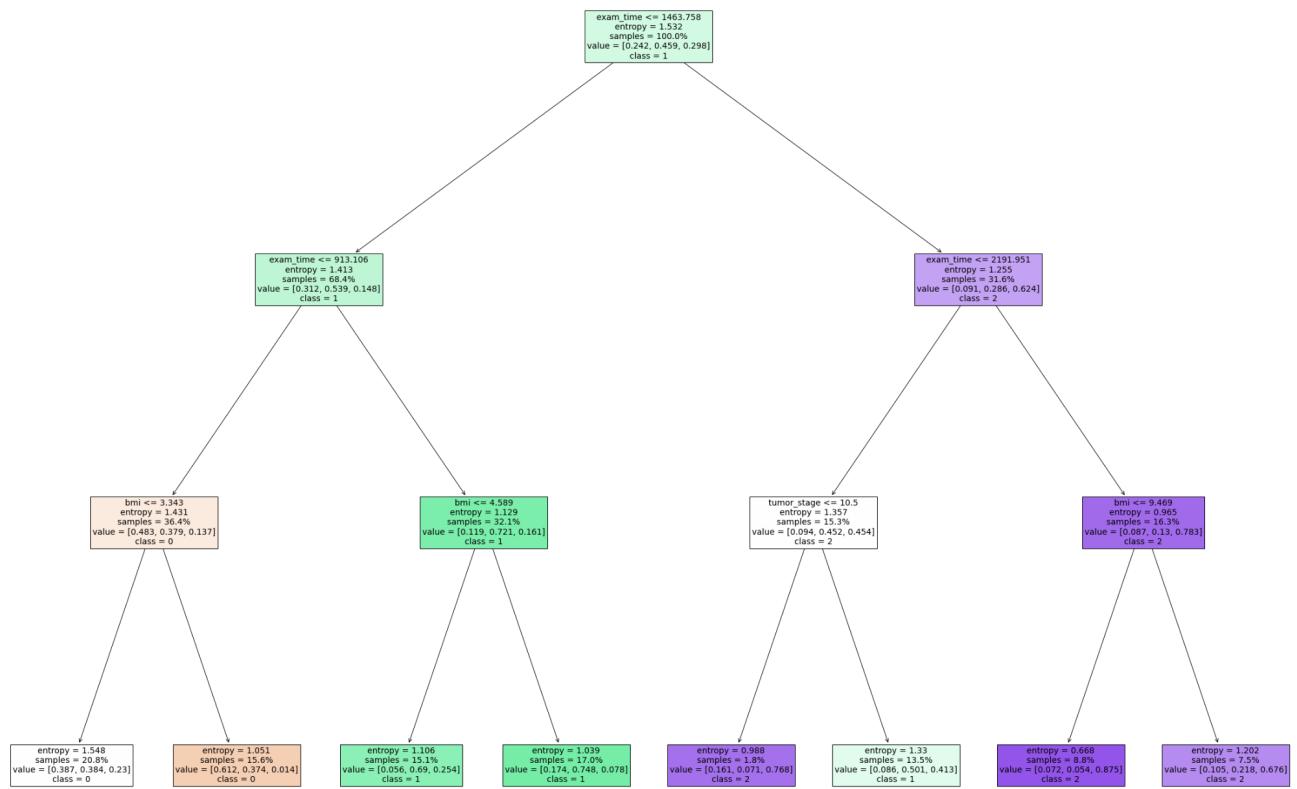


Imagen da árvore:



4.5. Avaliação

Análise dos resultados

Por ter sido utilizado diversos tipos de modelos, os resultados de acurácia obtidos foram diferentes, sendo assim alguns modelos puderam obter resultados melhores.

Logo, é possível fazer uma comparação desses modelos, analisando a acurácia, a precisão, o f1 e a taxa de erro.

Naive Bayes sem hiperparâmetros:

Classification metrics:				
	precision	recall	f1-score	support
0	0.49	0.42	0.45	179
1	0.59	0.71	0.65	376
2	0.63	0.49	0.55	241
accuracy			0.58	796
macro avg	0.57	0.54	0.55	796
weighted avg	0.58	0.58	0.58	796
MSE (Mean-Squared-Error):	0.5766331658291457			

Tem-se uma acurácia de 58%.

Naive Bayes com hiperparâmetros:

Classification metrics:				
	precision	recall	f1-score	support
0	0.54	0.38	0.45	179
1	0.58	0.75	0.65	376
2	0.63	0.49	0.55	241
accuracy			0.59	796
macro avg	0.59	0.54	0.55	796
weighted avg	0.59	0.59	0.58	796
MSE (Mean-Squared-Error):	0.5301507537688442			

Há um aumento da acurácia de 58% para 59%.

Regressão Logística sem hiperparâmetros:

Classification metrics:				
	precision	recall	f1-score	support
0	0.65	0.32	0.43	225
1	0.54	0.80	0.64	327
2	0.73	0.59	0.65	244
accuracy			0.60	796
macro avg	0.64	0.57	0.57	796
weighted avg	0.63	0.60	0.58	796
MSE (Mean-Squared-Error):	0.5025125628140703			

Tem-se uma acurácia de 60%.

Regressão Logística com hiperparâmetros:

Classification metrics:				
	precision	recall	f1-score	support
0	0.70	0.47	0.56	225
1	0.60	0.78	0.68	327
2	0.70	0.64	0.67	244
accuracy			0.65	796
macro avg	0.67	0.63	0.64	796
weighted avg	0.66	0.65	0.64	796
MSE (Mean-Squared-Error):	0.5025125628140703			

Há um aumento da acurácia de 60% para 65%.

Random Forest sem hiperparâmetros:

Classification metrics:				
	precision	recall	f1-score	support
0	0.74	0.57	0.65	194
1	0.67	0.81	0.74	351
2	0.74	0.65	0.69	251
accuracy			0.70	796
macro avg	0.72	0.68	0.69	796
weighted avg	0.71	0.70	0.70	796

Tem-se uma acurácia de 70%.

Random Forest com hiperparâmetros:

Classification metrics:				
	precision	recall	f1-score	support
0	0.76	0.57	0.65	194
1	0.67	0.82	0.74	351
2	0.73	0.64	0.68	251
accuracy			0.70	796
macro avg	0.72	0.68	0.69	796
weighted avg	0.71	0.70	0.70	796

A acurácia permanece 70%.

Árvore de Decisão sem hiperparâmetros:

Classification metrics:				
	precision	recall	f1-score	support
0	0.60	0.49	0.54	225
1	0.61	0.65	0.63	327
2	0.60	0.65	0.62	244
accuracy			0.60	796
macro avg	0.60	0.60	0.60	796
weighted avg	0.60	0.60	0.60	796
MSE (Mean-Squared-Error):	0.6331658291457286			

Tem-se uma acurácia de 60%.

Árvore de Decisão com hiperparâmetros:

Classification metrics:				
	precision	recall	f1-score	support
0	0.90	0.66	0.76	225
1	0.75	0.92	0.83	327
2	0.85	0.80	0.82	244
accuracy			0.81	796
macro avg	0.83	0.79	0.80	796
weighted avg	0.82	0.81	0.81	796
MSE (Mean-Squared-Error):	0.2851758793969849			

Há um aumento da acurácia de 60% para 81%.

KNN sem hiperparâmetros:

Classification metrics:				
	precision	recall	f1-score	support
0	0.56	0.53	0.54	229
1	0.60	0.69	0.65	347
2	0.67	0.55	0.60	220
accuracy			0.61	796
macro avg	0.61	0.59	0.60	796
weighted avg	0.61	0.61	0.60	796
MSE (Mean-Squared-Error):	0.5829145728643216			

Tem-se uma acurácia de 61%.

KNN com hiperparâmetros:

Classification metrics:				
	precision	recall	f1-score	support
0	0.58	0.54	0.56	229
1	0.60	0.71	0.65	347
2	0.68	0.55	0.61	220
accuracy			0.61	796
macro avg	0.62	0.60	0.61	796
weighted avg	0.62	0.61	0.61	796
MSE (Mean-Squared-Error):	0.5515075376884422			

A acurácia permaneceu 61%, porém a taxa de erro diminuiu de 58% para 55%.

Naive Bayes:

A qualidade dos resultados desse modelo é boa, está em quarto lugar entre os seis modelos apresentados, possuindo as seguintes métricas:

- Acurácia sem hiperparâmetro = 58%.
- Taxa de erro = 57%
- Acurácia com hiperparâmetro = 59%.
- Taxa de erro = 53%

O algoritmo de Naive Bayes Gaussiano analisa as características de uma base de dados assumindo que as features são independentes entre si. Além disso, ele também assume que as variáveis features são todas igualmente importantes para o resultado. Em cenários em que isso não ocorre, essa técnica deixa de ser a opção ideal, por isso a baixa acurácia desse modelo no projeto, entretanto resolvemos testá-lo para ter um embasamento maior.

Regressão Logística:

A qualidade dos resultados desse modelo é muito boa, está em terceiro lugar, juntamente com o KNN, entre os seis modelos apresentados, possuindo as seguintes métricas:

- Acurácia sem hiperparâmetro = 60%.
- Taxa de erro = 50%
- Acurácia com hiperparâmetro = 65%.
- Taxa de erro = 50%

O modelo *Regressão Logística* é adequado para este problema de predição, porque trata-se de um modelo que utiliza a classificação. Logo, esse modelo é muito indicado para prever o risco de se desenvolver uma determinada doença, por exemplo, baseado em características observadas de um paciente. Portanto, esse modelo pode ser utilizado no nosso projeto para fornecer a taxa do tempo de sobrevida de um paciente de forma adequada.

KNN:

A qualidade dos resultados desse modelo é muito boa, está em terceiro lugar, juntamente com o regressão logística, entre os seis modelos apresentados, possuindo as seguintes métricas:

- Acurácia sem hiperparâmetro = 61%
- Taxa de erro = 58%
- Acurácia com hiperparâmetro = 61%
- Taxa de erro = 55%

Esse modelo é adequado para esse tipo de projeto, pois o KNN é versátil, podendo ser usado tanto para classificação quanto para regressão, tendo como impedimento a alta dimensionalidade dos dados, entretanto apesar de estarmos trabalhando com uma grande quantidade de dados, o modelo atendeu a necessidade de classificação e teve um ótimo desempenho comparado aos outros modelo.

Random Forest:

A qualidade dos resultados desse modelo é ótima, é o que possui o melhor resultado de acurácia entre os seis modelos apresentados, ou seja, é o mais indicado para ser utilizado, possuindo as seguintes métricas:

- Acurácia sem hiperparâmetro = 70%
- Taxa de erro = 45%
- Acurácia com hiperparâmetro = 70%
- Taxa de erro = 45%

O modelo *Random Forest* é adequado para este problema de predição pois essencialmente estamos predizendo um parâmetro apenas, de tempo de sobrevida, e para essas predições este modelo é muito eficaz, dando a devida relevância aos parâmetros preditores, essencialmente pois é capaz de analisar a melhor árvore de decisão para esta predição.

Logo, o Random Forest é o melhor modelo de predição para o nosso problema, então foi criado um código para entender quais são as colunas que estavam sendo mais importantes para o modelo:

No código abaixo foi feito um cálculo de quanto tempo o algoritmo leva para determinar quais são as features mais importantes, apresentando uma alta performance.

```
[ ] #Calcular o tempo em que o código para descobrir as features mais importantes.
start_time = time.time()
#Define quais são as features mais importantes
importances = crf.feature_importances_
std = np.std([tree.feature_importances_ for tree in crf.estimators_], axis=0)
elapsed_time = time.time() - start_time

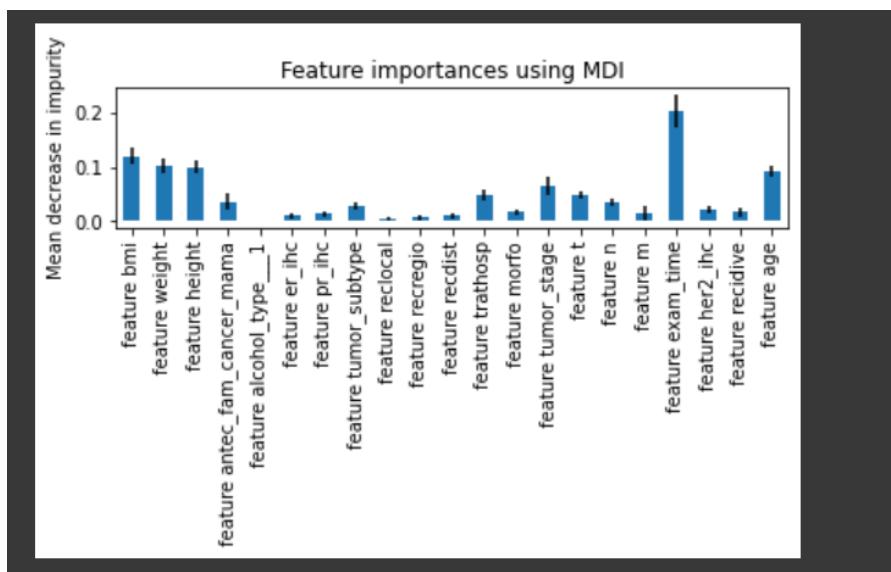
print(f"Elapsed time to compute the importances: {elapsed_time:.3f} seconds")

Elapsed time to compute the importances: 0.025 seconds
```

Já este código é para plotar o gráfico de barras que irá permitir melhor visualização da parte interessada.

```
#Plotar gafico de barras comparando as features
feature_names = [f"feature {i}" for i in X.columns]
forest_importances = pd.Series(importances, index= feature_names)

fig, ax = plt.subplots()
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title("Feature importances using MDI")
ax.set_ylabel("Mean decrease in impurity")
fig.tight_layout()
```



Árvore de Decisão:

- Acurácia sem hiperparâmetro = 60%.

- Taxa de erro = 63%.
- Acurácia com hiperparâmetro = 81%.
- Taxa de erro = 28.5%.

O modelo de árvore de decisão é adequado para esse tipo de projeto, pois permite que o resultado seja baseado nas melhores decisões relacionais dos dados fornecidos e, além disso, apresenta de forma clara e objetiva os resultados das classificações. Além disso, com os hiperparâmetros conseguimos ter controle da profundidade da árvore e suas ramificações, contribuindo para o aumento da acurácia nos resultados.

Análise comparativa

Os modelos que tiveram melhores resultados foram: Random Forest e Árvore de Decisão, pois ambos foram os que tiveram a menor taxa de erro em comparação aos demais modelos, portanto, foram os que obtiveram a maior acurácia. Sendo assim, pode-se concluir que são os melhores modelos a serem utilizados no projeto.

Comparando todos os modelos, estes foram os que apresentaram melhores resultados em suas métricas.

Métricas do Random Forest com hiperparâmetro:

Taxa de erro (MSE): 45%.

Precisão (PRE): quantifica a capacidade de precisão do modelo.

- Precisão (0) = 76%.
- Precisão (1) = 67%.
- Precisão (2) = 73%.

Recall (REC,revocação): quantifica a capacidade de correto reconhecimento do modelo.

- Recall (0) = 57%.
- Recall (1) = 82%.
- Recall (2) = 64%.

Medida-F (F1): trata-se da precisão e recall combinados em uma medida uniforme ponderada, é a média harmônica dos dois, ou seja, quadrado da média geométrica dividido pela média aritmética

- f1-score (0) = 65%.
- f1-score (1) = 74%.
- f1-score (2) = 68%.

Métricas da Árvore de Decisão com hiperparâmetro:

Taxa de erro (MSE): foi o modelo que obteve a menor taxa de erro: 13.5%.

Precisão (PRE): quantifica a capacidade de precisão do modelo.

- Precisão (0) = 90%.
- Precisão (1) = 75%.
- Precisão (2) = 85%.

Recall (REC,revocação): quantifica a capacidade de correto reconhecimento do modelo.

- Recall (0) = 66%.
- Recall (1) = 92%.
- Recall (2) = 80%.

Medida-F (F1): trata-se da precisão e recall combinados em uma medida uniforme ponderada, é a média harmônica dos dois, ou seja, quadrado da média geométrica dividido pela média aritmética

- f1-score (0) = 76%.
- f1-score (1) = 83%.
- f1-score (2) = 82%.

5. Conclusões e Recomendações

Com os modelos preditivos prontos, foi feita uma análise sobre cada modelo e o que cada um representa, sendo os seguintes modelos utilizados em nosso projeto: Random Forest, Árvore de Decisão, Regressão Logística, KNN e Naive Bayes.

Sendo assim, com a criação dos algoritmos desenvolvidos na plataforma Google Colaboratory (Colab), foi possível realizar uma análise profunda sobre os resultados de cada modelo. Portanto, cada modelo preditivo, foi desenvolvido com a mesma base de dados, a qual continha a seleção das features mais importantes, sendo as seguintes:

Após a exportação da base de dados para o colab e a conexão com o drive, foi feita a criação dos de cada modelo preditivo, e para desenvolver os algoritmos de cada tipo de modelo no colab, foi necessário a importação de bibliotecas do sklearn específicas para cada um.

Em todos os modelos foi utilizado a mesma margem de teste e treino, sendo 80% do data frame como treino e 20% como teste. Após essa definição em todos os modelos preditivos, foi desenvolvido o modelo classificatório para cada um, logo, cada modelo recebeu a sua biblioteca com as suas funcionalidades. E com isso, foi feito o treinamento do modelo em cada um, para que assim fosse possível realizar as previsões.

Na análise dos resultados de cada modelo, foi utilizada uma biblioteca do sklearn para mostrar as métricas de cada um, como: acurácia, precisão, recall e f1-score. Além disso, também foi utilizado um algoritmo que apresenta a taxa de erro de cada modelo. Primeiramente essas métricas foram apresentadas sem o uso de hiperparâmetros e depois com o uso. Portanto, foi possível notar que o resultado da acurácia em cada modelo teve uma melhora no resultado com o uso de hiperparâmetros.

Em síntese, pudemos concluir que os melhores modelos foram os que apresentaram um melhor resultado na acurácia, sendo os seguintes modelos: Árvore de Decisão (81%), Random Forest (71%) e Regressão Logística (65%). Em contrapartida os modelos que apresentaram menores resultados em sua acurácia, foram: KNN (61%) e Naive Bayes (59%).

A *acurácia* de um modelo, trata-se do percentual total de acertos do modelo.

Portanto, conclui-se que esses dois modelos preditivos possuem uma boa taxa de acurácia, tratando-se de modelos assertivos.

Com o classification report do SciKit-Learn é possível prover as três métricas de avaliação apresentadas na figura acima.

Precision: é a capacidade do modelo de não prever uma instância negativa como positiva (não cometer erro do tipo 1). Para todas as instâncias classificadas como positivas, qual é o percentual de acerto.

Recall: é a capacidade do modelo de encontrar todas as instâncias positivas. Para todas as instâncias que são de fato positivas, qual é o percentual de acerto.

A métrica *F1*, conjuga as duas anteriores como uma média harmônica entre ambas. Ela deve sempre ser priorizada para comparar modelos de classificação em relação à acurácia.

6. Referências

MATPLOTLIB. **Matplotlib 3.6.0 documentation**. Disponível em: <https://matplotlib.org/stable/>. Acesso em: 2 ago. 2022.

PANDAS. **User Guide**. Disponível em:
https://pandas.pydata.org/docs/user_guide/index.html#user-guide. Acesso em: 2 ago. 2022.

PYTHON. **12.1. pickle – Python object serialization**. Disponível em:
<https://python.readthedocs.io/en/stable/library/pickle.html>. Acesso em: 26 set. 2022.

SCIKIT LEARN. **1.11. Ensemble methods**. Disponível em:
<https://scikit-learn.org/stable/modules/ensemble.html#forest>. Acesso em: 8 set. 2022.

SCIKIT LEARN. **1.13. Feature selection**. Disponível em:
https://scikit-learn.org/stable/modules/feature_selection.html. Acesso em: 31 ago. 2022.

SCIKIT LEARN. 1.9. **Naive Bayes**. Disponível em:

https://scikit-learn.org/stable/modules/naive_bayes.html. Acesso em: 2 set. 2022.

SCIKIT LEARN. **Nearest Neighbors Classification**. Disponível em:

https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html#sphx-glr-auto-examples-neighbors-plot-classification-py. Acesso em: 22 set. 2022.

SCIKIT LEARN. **Sklearn.linear_model.LogisticRegression**. Disponível em:

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Acesso em: 4 set. 2022.

SCIKIT LEARN. **Sklearn.model_selection.GridSearchCV**. Disponível em:

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. Acesso em: 22 set. 2022.

SCIKIT LEARN. **Sklearn.model_selection.RandomizedSearchCV**. Disponível em:

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html. Acesso em: 22 set. 2022.

SCIKIT LEARN. **Sklearn.naive_bayes.GaussianNB**. Disponível em:

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html. Acesso em: 1 set. 2022.

SCIKIT LEARN. **Sklearn.neighbors.KNeighborsClassifier**. Disponível em:

<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>. Acesso em: 22 set. 2022.

SCIKIT LEARN. **Sklearn.tree.DecisionTreeClassifier**. Disponível em:

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>. Acesso em: 8 set. 2022.

SEABORN. **Seaborn: statistical data visualization**. Disponível em: <https://seaborn.pydata.org/>.

Acesso em: 11 ago. 2022.

SIGMOIDAL. **Como salvar seu modelo de Machine Learning**. Disponível em:

<https://sigmoidal.ai/como-salvar-seu-modelo-de-machine-learning/>. Acesso em: 5 out. 2022.

W3SCHOOL. **Pandas - Cleaning Data**. Disponível em:

https://www.w3schools.com/python/pandas/pandas_cleaning.asp. Acesso em: 11 ago. 2022.

YOUTUBE. **Aprenda Como Selecionar Features para seu Modelo de Machine Learning**.

Disponível em: <https://www.youtube.com/watch?v=4RGT2YRHRY>. Acesso em: 15 set. 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.