

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO
CÂMPUS CAMPINAS

CAIO AUGUSTO DE SOUZA MOTA

**AVALIAÇÃO DE ALGORITMOS DE APRENDIZADO DE MÁQUINA PARA
PREDIÇÃO DE MORTALIDADE NEONATAL UTILIZANDO DADOS DO DATASUS**

CAMPINAS

2021

CAIO AUGUSTO DE SOUZA MOTA

**AVALIAÇÃO DE ALGORITMOS DE APRENDIZADO DE MÁQUINA PARA
PREDIÇÃO DE MORTALIDADE NEONATAL UTILIZANDO DADOS DO DATASUS**

Trabalho de Conclusão de Curso apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia Câmpus Campinas.

Orientador: Prof. Me. Everton Josué da Silva.

CAMPINAS

2021

Dados Internacionais de Catalogação na Publicação

Solicitar a ficha catalográfica pelo sistema Pergamum, (Meu Pergamum, solicitações, ficha catalográfica) após as correções sugeridas pela banca.

Apresenta-se no verso da página de rosto

CAIO AUGUSTO DE SOUZA MOTA

**AVALIAÇÃO DE ALGORITMOS DE APRENDIZADO DE MÁQUINA PARA
PREDIÇÃO DE MORTALIDADE NEONATAL UTILIZANDO DADOS DO DATASUS**

Trabalho de Conclusão de Curso apresentado
como exigência parcial para obtenção do diploma
do Curso de Tecnologia em Análise e
Desenvolvimento de Sistemas do Instituto
Federal de Educação, Ciência e Tecnologia de
São Paulo Câmpus Campinas.

Aprovado pela banca examinadora em: ____ de ____ de ____.

BANCA EXAMINADORA

Prof. Me. Everton Josué da Silva (orientador)

IFSP Câmpus Campinas

Prof. Me. Carlos Eduardo Beluzo

IFSP Câmpus Campinas

Prof. Dr. Ricardo Barz Sovat

IFSP Câmpus Campinas

*Dedico este trabalho aos meus familiares,
colegas de classe, professores e servidores do Instituto
que colaboraram em minha jornada formativa.*

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelo dom da vida e pela oportunidade de concluir mais uma etapa de minha experiência acadêmica.

Agradeço a todos os professores e servidores do IFSP Câmpus Campinas, que contribuíram direta e indiretamente para a conclusão deste trabalho.

Agradeço também à minha família, que deu todo o apoio necessário para que eu chegasse até aqui.

Agradeço ao meu orientador que me auxiliou a solucionar as dificuldades encontradas no caminho.

*"Minha energia é o desafio, minha motivação é o impossível,
e é por isso que eu preciso
ser, à força e a esmo, inabalável.".*

Augusto Branco

RESUMO

A mortalidade infantil pode ser usada para analisar níveis de pobreza e socioeconômicos, assim como medir qualidade de saúde e tecnologia médica disponível em uma população. O aprendizado de máquina é uma área da inteligência artificial que propõe métodos de análise de dados que automatizam a construção de modelos analíticos com base em reconhecimento de padrões, baseado no conceito de que sistemas podem aprender com dados, identificando padrões e tomando decisões com o mínimo de intervenção humana. O objetivo deste trabalho é testar e analisar dois tipos diferentes de algoritmos de aprendizado de máquina, utilizando a base de dados do SIM e SINASC do Brasil, do período de 2016 até 2018, para gerar modelos para predição de mortalidade neonatal. Os métodos utilizados foram Árvore de Decisão e de Regressão Logística e como métricas para avaliar os modelos resultantes foram utilizadas a AUC, Curva ROC e a Matriz de Confusão. Como resultado, apesar de terem alcançado valores de AUC 0.93, ambos modelos de previsões acertaram muitas previsões de vivos cerca de 671.000 e erraram muitas previsões de mortos cerca de 2.600, principalmente devido ao fato de a base estar desbalanceada.

Palavras-chave: Mortalidade Neonatal, Predição, Aprendizado de Máquina.

ABSTRACT

Infant mortality can be used to analyze poverty and socioeconomic levels, as well as measure the quality of health and medical technology available in a population. Machine learning is an area of artificial intelligence that proposes data analysis methods that automate the construction of analytical models based on pattern recognition, based on the concept that systems can learn from data, identifying patterns and making decisions with a minimum of human intervention. The objective of this work is to test and analyze two different types of machine learning algorithms, using the SIM and SINASC do Brasil database, from 2016 to 2018, to generate models for predicting neonatal mortality. The methods used were Decision Tree and Logistic Regression and as metrics to evaluate the resulting models, AUC, ROC Curve and Confusion Matrix were used. As a result, despite achieving values of AUC 0.93, both prediction models correct many live birth predictions around 671.000 and miss many stillbirth predictions around 2600, mainly due to the fact that the base is unbalanced.

Keywords: Neonatal Mortality, Prediction, Machine Learning.

LISTA DE FIGURAS

Figura 1 – Exemplo de Árvore de Decisão.....	19
Figura 2 – Exemplo de Diagrama de Regressão Logística.....	22
Figura 3 – Exemplo de Curva ROC.....	25
Figura 4 – Distribuição da base de dados considerando se o recém-nascido viveu ou morreu durante o período neonatal.....	29
Figura 5 – BoxPlot das features Peso ao nascer e Idade da mãe.....	35
Figura 6 – BoxPlot da feature semana de gestação.....	36
Figura 7 – Histograma de distribuição do peso entre as classes.....	36
Figura 8 – Gráfico de barras da distribuição da variável Escolaridade da mãe.....	37
Figura 9 – Gráfico de barras da distribuição da variável Número de nascidos vivos.....	37
Figura 10 – Gráfico de barras da distribuição da variável Pontuação Apgar de 1 minuto.....	38
Figura 11 – Gráfico de barras da distribuição da variável Pontuação Apgar de 5 minutos.....	38
Figura 12 – Gráfico de barras da distribuição da variável Presença de malformação congênita.....	39
Figura 13 – Gráfico de barras da distribuição da variável Número de gestações.....	39
Figura 14 – Gráfico de barras da distribuição da variável Assistência ao parto.....	40
Figura 15 – Gráfico de densidade de peso entre as classes.....	41
Figura 16 – Gráfico de correlação.....	42
Figura 17 – Distribuição da idade da mãe por raça.....	43
Figura 18 – Distribuição de peso ao nascer por raça.....	43
Figura 19 – Distribuição de semanas de gestação por raça.....	44
Figura 20 – Distribuição de peso ao nascer por estado civil.....	44
Figura 21 – Distribuição de semanas de gestação por estado civil.....	45
Figura 22 – Distribuição da idade da mãe por estado civil.....	45
Figura 23 – Distribuição da idade da mãe por escolaridade da mãe.....	46
Figura 24 – Distribuição de peso ao nascer por escolaridade da mãe.....	46
Figura 25 – Distribuição de semanas de gestação por escolaridade da mãe.....	47
Figura 26 – Separação dos DataFrames de entrada e rótulo.....	48

Figura 27 – Curva ROC modelo de Árvore de decisão.....	49
Figura 28 – Gráfico da importância das features.....	50
Figura 29 – Árvore de decisão montada a partir do algoritmo.....	51
Figura 30 – Curva ROC modelo regressão logística usando particionamento 90/10.....	54
Figura 31 – Curva ROC modelo regressão logística usando K-folds cross-validation...	56
Figura 32 – Resultado da função RFECV Base Brasil.....	58
Figura 33 – Resultado da função GridSearchCV.....	58
Figura 34 – Curva ROC modelo regressão logística usando GridSearchCV e RFECV.....	60

LISTA DE TABELAS

Tabela 1 – Exemplo da separação <i>K-fold cross-validation</i>	23
Tabela 2 – Modelo de Matriz de confusão.....	24
Tabela 3 – Colunas da base de dados e suas descrições.....	27 e 28
Tabela 4 – Resultados do modelo de árvore de decisão.....	48
Tabela 5 – Resultados do modelo de árvore de decisão utilizando K-folds cross-validation.....	52
Tabela 6 – Matriz de confusão do modelo de árvore de decisão utilizando K-folds cross-validation.....	52
Tabela 7 – Resultados do modelo de regressão logística usando particionamento 90/10.....	53
Tabela 8 – Matriz de confusão do modelo de regressão logística usando particionamento 90/10.....	54
Tabela 9 – Resultados do modelo de regressão logística usando K-folds cross-validation.....	55
Tabela 10 – Matriz de confusão do modelo de regressão logística usando K-folds cross-validation.....	57
Tabela 11 –Resultados do modelo de regressão logística usando GridSearchCV e RFECV.....	59
Tabela 12 – Matriz de confusão do modelo de regressão logística usando GridSearchCV e RFECV.....	61

LISTA DE SIGLAS

SIM	Sistema de Informação sobre Mortalidade
SINASC	Sistema de Informações sobre Nascidos Vivos
TMI	Taxa de Mortalidade Infantil
TMN	Taxa de Mortalidade Neonatal
MN	Mortalidade Neonatal
ML	Machine Learning
DNV	Declaração de Nascido Vivo
VN	Verdadeiro Negativo
FN	Falso Negativo
VP	Verdadeiro Positivo
FP	Falso Positivo
ROC	Curva Característica de Operação do Receptor
NaN	Não é um Número
RFE	Eliminação Recursiva de <i>Feature</i>
RFECV	Eliminação Recursiva de <i>Feature</i> com Validação Cruzada

SUMÁRIO

1 INTRODUÇÃO	14
2 JUSTIFICATIVA	15
3 OBJETIVOS	16
3.1 Objetivo Geral	16
3.2 Objetivos Específicos	16
4 FUNDAMENTAÇÃO TEÓRICA	17
4.1 Mortalidade Infantil e Neonatal	17
4.2 Métodos de Aprendizado de MÁQUINA	17
4.2.1 Árvore de Decisão	18
4.2.2 Regressão Logística	20
4.2.3 Treino, teste e validação do modelo de aprendizado de máquina	22
4.2.4 Métricas para avaliação de modelos	23
5 METODOLOGIA	25
5.1 Tecnologias e Ferramentas	25
5.2 Base de dados	26
5.3 Preparação da base de dados	29
5.4 Desenvolvimento dos modelos de aprendizado de máquina	29
5.5 Análise dos resultados	31
5.6 Disponibilização da base de dados e códigos fontes	31
6 RESULTADOS	32
6.1 Análise Exploratória	32
6.1.1 Apresentação dos Dados	32
6.1.1.1 Características demográficas e socioeconômicas maternas	32
6.1.1.2 Variáveis obstétricas maternas	33
6.1.1.3 Variáveis de histórico de gravidez	33
6.1.1.4 Variáveis relacionadas ao recém-nascido	33

6.1.2 Análise das variáveis	34
6.1.2.1 Variáveis contínuas	34
6.1.2.2 Variáveis categoricas	36
6.1.2.3 Analise bi-variada	40
6.1.2.4 Distribuição por raça	42
6.1.2.5 Distribuição por estado civil	44
6.1.2.6 Distribuição por escolaridade da mãe	46
6.2 Árvore de Decisão	47
6.2.1 Modelo de Árvore de Decisão utilizando particionamento 90/10	48
6.2.2 Modelo de Árvore de Decisão utilizando K-folds cross-validation	51
6.3 Regressão Logística	52
6.3.1 Modelo de Regressão Logística utilizando particionamento 90/10	53
6.3.2 Modelo de Regressão Logística Utilizando K-folds cross-validation	55
6.3.3 Modelo de Regressão Logística utilizando GridSearchCV e RFECV	57
7 CONCLUSÃO	62
REFERÊNCIAS	63

1 INTRODUÇÃO

A taxa de mortalidade infantil (TMI) é uma importante medida de saúde em uma população e é um grande problema no mundo todo. A TMI pode ser usada para analisar níveis de pobreza e socioeconômicos, assim como medir qualidade de saúde e tecnologia médica disponível. Esta taxa é dividida em duas categorias: neonatal e pós-neonatal. É categorizada neonatal quando o óbito ocorre nos 28 primeiros dias de vida após o pós-parto, e o pós-neonatal é quando o óbito ocorre entre 29 e 364 dias de vida (BELUZO et al., 2020).

Cerca de 46% das mortes no mundo acontecem entre os menores de cinco anos e grande parte está concentrada nos primeiros dias de vida (LANSKY, 2014). A diminuição dessas taxas é muito importante no mundo todo, refletindo nos indicadores de saúde pública e desenvolvimento do país.

Os fatores associados à mortalidade são profundamente influenciados pelas características biológicas maternas e neonatais, pelas condições sociais e pelos cuidados prestados pelos serviços de saúde (E. FRANÇA; S. LANSKY, 2009; R.M.D. NASCIMENTO, 2012). Em grande parte, o diagnóstico é altamente dependente da experiência adquirida pelo profissional que o realiza. Apesar de indiscutivelmente necessário, não é perfeito, principalmente pelo fato de ser dependente de fatores humanos, por este motivo os profissionais sempre tiveram recursos tecnológicos para auxiliar nessa tarefa (BELUZO et al., 2020). Segundo estudos de 2010 no Brasil, calcula-se que aproximadamente 70% dos óbitos infantis ocorridos poderiam ter sido evitados por ações de saúde e que 60% dos óbitos neonatais ocorreram por situações que poderiam ser evitadas (BARRETO; SOUZA; CHAPMAN, 2015).

No presente trabalho foram utilizados dois algoritmos de aprendizado de máquina para criar modelos de predição de mortalidade neonatal. Além disso, foi realizada também uma análise exploratória da base de dados. Para este trabalho foram utilizadas duas fontes de dados: Sistema de Informação sobre Mortalidade (SIM) e Sistema de Informações sobre Nascidos Vivos (SINASC) do Brasil inteiro.

2 JUSTIFICATIVA

Com o avanço da tecnologia, podemos notar que ela está cada vez mais presente no nosso dia a dia e nos auxilia em diversas tarefas do cotidiano, e vem sendo aplicada em diversas áreas, dentre elas a saúde.

A mortalidade infantil é uma preocupação mundial na saúde pública, a ONU (Organizações das Nações Unidas) definiu a redução da mortalidade infantil como uma meta para o desenvolvimento global. A mortalidade neonatal é responsável por aproximadamente 60% da TMI nos países em desenvolvimento (A.K. SINGHA, 2016). Existem diversos fatores que podem contribuir com a redução de óbitos neonatais, como por exemplo acompanhamento médico à gestante no período de gravidez, acompanhamento médico ao recém-nascido nos primeiros dias de vida e a disponibilização deste serviço com qualidade e proficiência (WHO, 2009).

A diminuição dessa taxa é importante e de interesse para o mundo todo, e utilizar da tecnologia para auxiliar nessa diminuição é uma proposta funcional, e inovadora para a realidade brasileira (MOTA et al., 2019). Havendo disponibilidade de tecnologia para auxiliar nas decisões dos diagnósticos, os profissionais da saúde podem focar nos cuidados do recém-nascido com risco de vir a óbito, fornecendo tratamentos melhores.

Segundo o site Multiedro (2019), um grande problema que os médicos enfrentam ao realizar o diagnóstico, é analisar um grande volume de dados. Para a área médica obter diagnósticos rápidos e precisos pode salvar vidas, e a utilização de *machine learning* para realizar esses processos é importante, com a ML os dados podem ser processados em questões de segundos e resultar em um diagnóstico mais rápido, além disso utilizando bons modelos ML os diagnósticos serão mais precisos.

Diversos trabalhos vêm sendo desenvolvidos neste contexto. No trabalho realizado por BELUZO et al. (2020a), os autores utilizaram uma base de dados com informações da cidade de São Paulo para criação de modelos de aprendizado de máquina para predição de mortalidade neonatal. Este recorte foi utilizado também no presente trabalho para fins de exercício e definição de etapas da implementação de código. Na conclusão os autores discutem os fatores que tiveram mais influência nas previsões, e na análise feita pelos autores, as consultas de pré-natal são muito importantes para a redução da mortalidade infantil, uma vez que algumas características muito importantes, como a malformação congênita, podem ser detectadas ao longo das consultas de pré-natal.

Em um outro estudo realizado por BELUZO et al. (2020a), foi realizada uma análise exploratória da base de dados *SPNeoDeath*, onde e podemos observar detalhadamente cada análise das colunas da tabela e diversos gráficos feito para um melhor entendimento dos dados.

Outro trabalho que foi realizado utilizando a base de dados com dados somente de São Paulo foi o de PELISSARI (2021), nesse trabalho é realizado uma análise exploratória na base de dados de São Paulo e é também analisado três algoritmos de aprendizado de máquina (*Logistic Regression*, *Random Forest Classifier* e *XGBoost*), nesse trabalho a autora conclui que os modelos de *Logistic Regression* e *XGBoost* foram os modelos que apresentaram os melhores desempenhos preditivos, e também mostra os problemas de estar utilizando uma base de dados desbalanceada.

O problema da mortalidade neonatal não é recente, e o mundo todo desenvolve iniciativas para lidar com ele. A ONU definiu como meta a redução da mortalidade infantil para o desenvolvimento global. Diversos fatores podem ajudar na diminuição da taxa de mortalidade neonatal como acompanhamento médico antes e após parto, tanto com a mãe quanto com o recém-nascido. Um serviço de qualidade e constante para os casos com maior risco de óbito pode salvar diversos recém-nascidos. Neste sentido, o desenvolvimento de ferramentas para a predição de mortalidade neonatal pode colaborar com esta iniciativa.

3 OBJETIVOS

3.1 OBJETIVO GERAL

O presente trabalho tem como objetivo testar e analisar os resultados da aplicação dos aprendizagem de máquina supervisionado Árvore de Decisão e Regressão Logística, utilizando dados do SIM e do SINASC, no período de 2016 até 2018, a fim de gerar modelos de predição de risco morte neonatal.

3.2 OBJETIVOS ESPECÍFICOS

- Realizar análise exploratória da base dados a ser utilizada na construção dos modelos;
- Implementar modelos de predição utilizando algoritmos Árvore de Decisão e Regressão Logística;
- Avaliar resultados e propor novas abordagens.

4 FUNDAMENTAÇÃO TEÓRICA

4.1 MORTALIDADE INFANTIL E NEONATAL

Como dito na monografia PELISSARI (2021), a TMI consiste no número de crianças que foram a óbito antes de concluir um ano de vida dividido por 1000 crianças nascidas vivas no tempo de um ano. A TMI pode ser usada para analisar níveis de pobreza e socioeconômicos e qualidade da saúde pública de um país (apud BELUZO et al., 2020).

Mencionado em PELISSARI (2021), a TMN é uma das duas categorias da TMI, é categorizado mortalidade neonatal quando o óbito ocorre do nascimento da criança até os 28 primeiros dias de vida. A mortalidade neonatal pode ser subdividida em mortalidade neonatal precoce que é quando o óbito ocorre entre 0 e 6 dias de vida, e o neonatal tardio quando o óbito ocorre entre 7 e 28 dias de vida (apud E. FRANÇA e S. LANSKY, 2009).

Segundo Lansky et al. (2014), a taxa de mortalidade neonatal é o principal componente da TMI desde a década de 1990 no Brasil e vem se mantendo em níveis elevados, com taxa de 11,2 óbitos por mil nascidos vivos em 2010 (apud Maranhão et al., 2012). Também é dito em Lansky et al. (2014), que a TMI do Brasil em 2011 foi 15,3 por mil nascidos vivos, alcançando a meta 4 dos objetivos de desenvolvimento do milênio, compromisso dos governos integrantes das Nações Unidas de melhorar a saúde infantil e reduzir em 2/3 a mortalidade infantil entre 1990 e 2015. (apud Maranhão et al., 2012; apud Murray et al., 2015). Segundo Lansky et al. (2014) o principal componente da TMI em 2009 é a mortalidade neonatal precoce, e grande parte das mortes infantis acontece nas primeiras 24 horas (25%), indicando uma relação estreita com a atenção ao parto e nascimento (apud E. FRANÇA e S. LANSKY, 2009).

4.2 MÉTODOS DE APRENDIZADO DE MÁQUINA

A utilização de dados para a resolução de problemas, de modo a se identificar padrões ocultos e posteriormente auxiliar na tomada de decisões é definida como aprendizado de máquina (BRESAN, 2018 apud Brink et al. 2013).

O uso de técnicas de Aprendizado de Máquina se mostra altamente eficiente na resolução de tarefas que se apresentem difíceis de se resolver a partir de programas escritos por humanos (BRESAN, 2018 apud GOODFELLOW et al., 2016), bem como também

possibilita uma maior compreensão sobre os funcionamentos dos princípios que compõem a inteligência humana.

Neste trabalho serão implementados modelos utilizando os algoritmos de Árvore de Decisão e Regressão Logística, os quais serão apresentados a seguir.

4.2.1 Árvore de Decisão

Segundo Batista e Filho (2019), os modelos preditivos baseados em árvores são geralmente utilizados para tarefas de classificação, embora também possam ser utilizados para tarefas de regressão. Considerado um dos mais populares algoritmos de predição, o algoritmo de árvore de decisão proporciona uma grande facilidade de interpretação.

As árvores de decisão utilizam a estratégia “dividir-e-conquistar”, na qual as árvores são construídas utilizando-se apenas alguns atributos. A árvore de decisão é uma das técnicas por meio da qual um problema complexo é decomposto em subproblemas mais simples. Recursivamente, a mesma estratégia é aplicada a cada subproblema (SILVA et al, 2008).

A árvore de decisões tem uma estrutura hierárquica onde cada nó da árvore representa uma decisão em uma das colunas do conjunto de dados, cada ramo da árvore representa uma tomada de decisão (BATISTA; FILHO, 2019).

O algoritmo segue algumas decisões para montar a árvore, o atributo mais importante é utilizado como nó raiz e será o primeiro nó, já os atributos menos importantes são mostrados nos ramos da árvore. Cada atributo é verificado para conferir se é possível gerar uma árvore menor e se é possível fazer uma classificação melhor, e assim é escolhido o nó que produz nós filhos (SILVA et al, 2008).

Existem diferentes tipos de algoritmo para a construção da árvore, como por exemplo, ID3 (*Iterative Dichotomiser*), C4.5 e CART (*Classification and Regression Tree*). O algoritmo ID3 escolhe os nós da árvore por meio da métrica do ganho de informação. Já o CART faz uso da equação de *Gini* (BATISTA; FILHO, 2019 apud KUHN; JOHNSON, 2013).

O algoritmo utilizado neste trabalho usará o cálculo da entropia e ganho de informação, para decidir qual atributo será usado no nó, a entropia é uma medida da desorganização dos sistemas, maior é a incerteza do sistema, quanto maior a entropia maior está a desorganização, com a árvore de decisão a ideia é ir organizando o sistema e ir separando as decisões da melhor forma para que a entropia diminui e o ganho de informação

aumente, para que a decisão do modelo fique mais assertiva. O cálculo da entropia utiliza a seguinte equação (ÁRVORE, 2020):

$$Entropia = \sum_i - P_i \log_2 P_i$$

Nesta equação o i representa possíveis classificações (rótulos), e o P é a probabilidade de cada uma. A ideia da árvore então é verificar qual é a entropia para cada uma das variáveis da base de dados, e verificar qual é a melhor organização de cada uma das variáveis. E isso é realizado através da técnica chamada de ganho de informação, essa técnica utiliza a equação a seguir (ÁRVORE, 2020):

$$ganho = Entropia(pai) - \Sigma peso(filhos) * Entropia(filhos)$$

Então o ganho de informação é calculado pela entropia do nó pai menos a soma do peso vezes a entropia dos nós filhos. Esse cálculo é realizado para todas as variáveis e a variável que tiver maior ganho de informação é utilizado para a decisão do nó. Esse processo é realizado recursivamente e a árvore vai sendo montada com base nesses cálculos (ÁRVORE, 2020).

Na Figura 1 temos a representação do início de uma árvore de decisão, no caso em questão o autor estava classificando exames positivos e negativos para o vírus SARS-CoV-2, ele utilizou uma base de dados que contém quase 6 mil exames realizados, contendo campos como o resultado do exame, idade do paciente, níveis de hemoglobina entre outras informações.

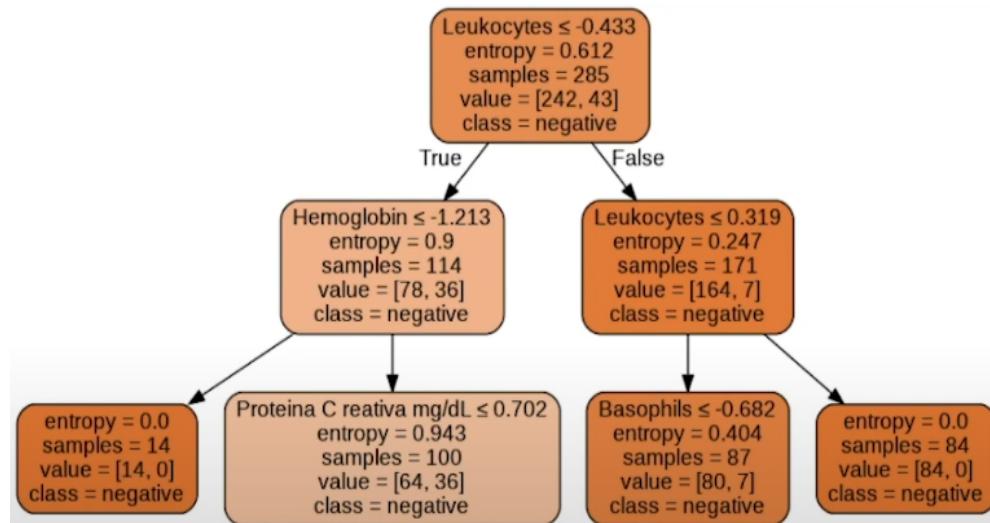


Figura 1. Exemplo de Árvore de Decisão. (Fonte: ÁRVORE 2020).

Pode-se observar que para o nó raiz foi utilizado a variável *Leukocytes*, logo foi o que apresentou a melhor organização para ser o nó raiz, e após ele temos os ramos da árvore. Cada retângulo da árvore é uma decisão do modelo e esses retângulos têm atributos importante como, a coluna da base de dados que será onde vai ser realizado a decisão, então pegando o nó raiz, pacientes que tiverem com os *Leukocytes* para menos que -0.43 seguiram o caminho para a esquerda (*true*), já os pacientes que tiverem mais seguiram o caminho da direita (*false*). A entropia calculada do nó também é apresentada e a classificação do nó também, então se o resultado parar neste nó, a classificação que está nele será a classificação final, e temos o *samples* que é o número de amostras que se enquadram no na decisão. E esse processo de verificação do modelo vai se repetindo até não ter mais como dividir em subproblemas ou até chegar na profundidade máxima.

Uma característica importante da árvore de decisão é que ela é um modelo *WhiteBox*, ou seja, é possível visualizar a árvore montada e as decisões que o modelo terá que tomar na árvore. Na Figura 1 pode-se ver uma árvore de decisões montada, cada retângulo da árvore é uma decisão que o modelo terá que tomar, conforme o modelo toma as decisões ele vai seguindo um caminho até chegar ao nó folha, nó folha é o nó que não contém nó filho, não tem mais ramos para baixo, chegando ao nó folha o modelo tem a classificação final.

4.2.2 Regressão Logística

O modelo de regressão logística estabelece uma relação entre a probabilidade de ocorrência da variável de interesse e as variáveis de entrado do modelo, sendo utilizada para tarefas de classificação, em que a variável de interesse é categórica, neste caso, a variável de interesse assume valores 0 ou 1, onde 1 é geralmente utilizado para indicar a ocorrência do evento de interesse (Batista e Filho, 2019).

De acordo com Mesquita (2014), a técnica de regressão logística, desenvolvida no século XIX, obteve maior visibilidade após 1950 ficando então mais conhecida. Ficou ainda mais difundida a partir dos trabalhos de Cox & Snell (1989) e Hosmer & Lemeshow (2000). Caracteriza-se por descrever a relação entre uma variável dependente qualitativa binária, associada a um conjunto de variáveis independentes qualitativas ou métricas.

Esta técnica inicialmente foi utilizada na área médica, porém a eficiência viabilizou sua implementação nas mais diversas áreas. Mesquita (2014) diz que o termo regressão

logístico, tem sua origem na transformação usada com a variável dependente, que permite calcular diretamente a probabilidade da ocorrência do fenômeno em estudo.

Segundo Santos (2018), para estimar a probabilidade, é utilizado uma técnica chamada distribuição binomial para modelar a variável resposta do conjunto de treinamento, que tem como parâmetro a probabilidade de ocorrência de uma classe específica. Uma característica importante desse modelo é que a probabilidade estimada deve estar limitada ao intervalo de 0 a 1. O algoritmo de Regressão Logística gera um modelo de classificação binária (0 e 1). O modelo utiliza a Regressão Linear como base, a equação linear é (REGRESSÃO, 2020):

$$y = m \cdot x + b$$

O “m” da equação é o coeficiente angular da reta, e o “b” é o intercepto. Então, aplicando a equação linear obtemos um valor contínuo como resultado, após obter esse valor a Regressão Logística tem que realizar a classificação do resultado, então é aplicado uma função chamada sigmoide (REGRESSÃO, 2020):

$$\sigma = \frac{1}{1+e^{-y}}$$

Essa função chamada de sigmoide, ou função logística, é responsável por “achatar” o resultado da regressão linear, calculando a probabilidade de o resultado pertencer a classe 1, classificando assim o resultado da função linear em 0 ou 1.

Segundo Batista e Filho (2019), o modelo de regressão logística que tem como objetivo realizar uma classificação binária de uma variável de interesse irá predizer a probabilidade de pertencer à classe positiva. Tem-se, portanto, que \hat{y} é dado por:

$$\hat{y} = \{ 0, \text{ se } Prob(y = 1 | X) < 0,5 \}$$

ou

$$\hat{y} = \{ 1, \text{ se } Prob(y = 1 | X) \geq 0,5 \}$$

Então a decisão do modelo de regressão logística está relacionada à escolha de um ponto de corte para $p(x)$. Caso o ponto de corte escolhido for $p(x)=0,5$, os resultados que forem $p(x)>0,5$ serão classificados como 1 e os resultados $p(x)<0,5$ serão classificados como 0 (SANTOS, 2018).

A Figura 2 ilustra um exemplo de como é o diagrama de Regressão Logística. Pelo diagrama então pode-se observar que o modelo possui uma representação gráfica em formato de “S”, essa seria a curva logística. As previsões do modelo sempre ficaram na linha 1 e 0 do

eixo y, pois é o intervalo estabelecido pelo algoritmo, então a classificação sempre será nesse intervalo.

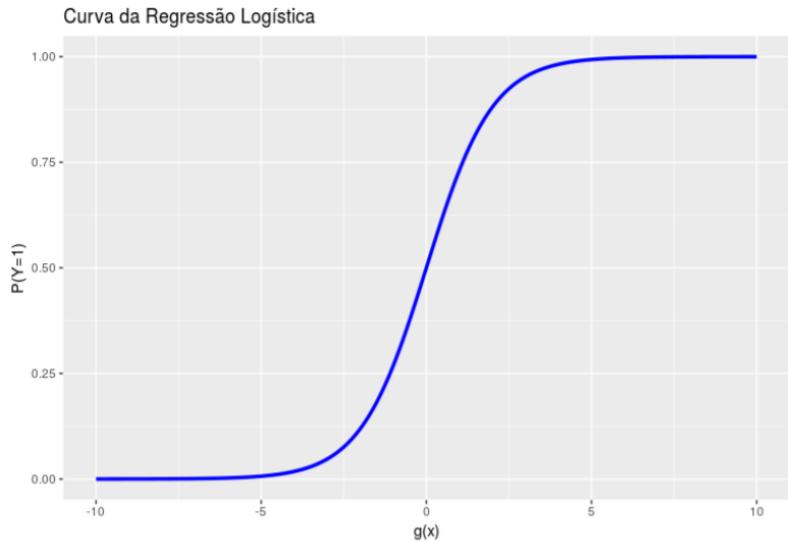


Figura 2. Exemplo de Diagrama de Regressão Logística. (Fonte: Batista e Filho 2019).

4.2.3 Treino, teste e validação do modelo de aprendizado de máquina

A criação de um modelo de aprendizado de máquina é realizada em duas etapas: treinamento do modelo, que é realizado a partir dos dados fornecidos para o algoritmo, e etapa de teste, onde os modelos recebem dados novos e sem o rótulo, para que seja avaliado a performance do modelo (PELISSARI, 2021).

Para o treinamento dos modelos foram utilizadas duas abordagens distintas, a primeira onde a base foi particionada entre conjunto de dados para teste e conjunto de dados para treino em uma porcentagem de 90% para treino e 10% para teste, porém dessa forma é muito comum ocorrer problema de sobreajuste ou *overfitting*. Nesta situação, o que pode ocorrer é o modelo não aprender com os dados, e apenas “decorar” os dados recebidos e quando é realizado o teste o modelo consegue predizer apenas situações parecidas, não sendo capaz de identificar padrões com diferenças mínimas.

Para evitar este problema, foi realizada então uma segunda abordagem utilizando a técnica de validação cruzada ou *K-folds cross-validation*. Nessa técnica a base de dados é dividida em K subconjuntos, e então são realizadas várias rodadas de treinamento e teste

alternando os subconjuntos utilizados para teste e treino, e o resultado do modelo baseia-se na média da acurácia observada em cada rodada (PELISSARI, 2021).

Pode-se observar na Tabela 1 uma ilustração da técnica *K-fold cross-validation*, onde temos um exemplo onde a base é dividida em 5 subconjuntos e cada subconjunto tem a parte de teste diferente dos outros subconjuntos, assim o modelo vai receber valores novos de teste e treino todas as vezes que executar um novo subconjunto.

Tabela 1. Exemplo da separação *K-fold cross-validation*. (Fonte: Adaptado de PELISSARI, 2021).

Predição 1	Predição 2	Predição 3	Predição 4	Predição 5
Teste	Treino	Treino	Treino	Treino
Treino	Teste	Treino	Treino	Treino
Treino	Treino	Teste	Treino	Treino
Treino	Treino	Treino	Teste	Treino
Treino	Treino	Treino	Treino	Teste

4.2.4 Métricas para avaliação de modelos

Para fins de avaliação, neste trabalho foram utilizadas duas métricas para avaliar o desempenho dos modelos: a Matriz de Confusão e a AUC (Area under *Receiver Operating Characteristic Curve - ROC*). Na matriz de confusão pode-se observar o comportamento do modelo em relação a cada uma das classes a serem preditas pelo modelo, utilizando variáveis binárias (positivo: 1; e negativo: 0), para apresentar uma matriz que faz uma comparação entre as previsões do modelo e os valores corretos do conjunto de dados (PELISSARI, 2021). Na Tabela 2 temos um exemplo de uma matriz de confusão, a qual consiste em duas colunas e duas linhas, e os valores são atribuídos nos quadrantes com os seguintes resultados (PELISSARI, 2021):

- quando o valor real do conjunto de dados é 0, e a previsão do modelo também classificou como 0, então temos um **Verdadeiro Negativo (VN)**;
- quando o valor real é 1, e o modelo classificado como 0, temos um **Falso Negativo (FN)**;

- quando o valor real é 0, e o modelo classificado como 1, então o resultado se enquadra no quadrante de **Falso Positivo (FP)**;
- e por fim o quadrante **Verdadeiro Positivo (VP)**, que são os resultados que tem o valor real 1, e o modelo classificado como 1.

Tabela 2. *Modelo de Matriz de confusão. (Fonte: Adaptado de PELISSARI, 2021).*

		Valor Preditivo	
		Negativo (0)	Positivo (1)
Classe Real	Negativo (0)	<i>Verdadeiro Negativo</i>	<i>Falso Positivo</i>
	Positivo (1)	<i>Falso Negativo</i>	<i>Verdadeiro Positivo</i>

A segunda métrica de avaliação utilizada foi a **Curva ROC** que permite avaliar o desempenho de um modelo com relação às previsões efetuadas. A curva ROC é um gráfico de Sensibilidade (taxa de verdadeiros positivos) versus taxa de falsos positivos, a representação da curva ROC permite evidenciar os valores para os quais existe otimização da Sensibilidade em função da Especificidade (CABRAL, 2013).

Na Figura 3 temos um exemplo de uma curva ROC. A linha traçada na cor preta é uma linha de referência, ela representa hipoteticamente a curva ROC de um classificador puramente aleatório. Caso a linha laranja, que representa o resultado do modelo que está sendo avaliado, ficasse igual a linha tracejada preta, indicaria que o modelo avaliado estaria predizendo aleatoriamente os resultados.

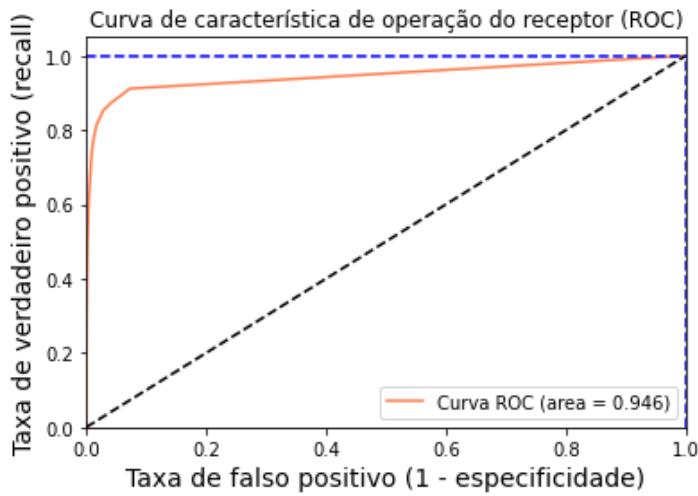


Figura 3: Exemplo de Curva ROC. (Fonte: Elaboração Própria).

Por uma análise de inspeção visual, quanto mais próximo a linha laranja estiver do valor 1 no eixo Y, melhor está a capacidade do modelo para diferenciar as classes. O valor da AUC representa a capacidade do modelo de diferenciar as classes, quanto maior o valor do AUC, melhor é a predição realizada pelo modelo, uma vez que, os valores resultantes iguais a 0 são realmente 0, e os iguais a 1 são de fato 1 (PELISSARI, 2021).

5 METODOLOGIA

O desenvolvimento deste trabalho foi realizado em três etapas: (1) Pré-processamento e Análise Exploratória do conjunto de dados, onde foram aplicadas técnicas comuns para preparar o conjunto de dados como tratamento de valores nulos e seleção de variáveis de interesse; (2) Implementação de modelos de aprendizado de máquina supervisionado para predição das mortes neonatais, utilizando os algoritmos de árvore de decisão e regressão logística; e (3) Análise de Resultados, que será realizada uma análise do desempenho do modelo.

5.1 TECNOLOGIAS E FERRAMENTAS

Neste trabalho vamos utilizar a linguagem de programação *Python* (PYTHON, 2021) pela sua simplicidade de codificação e alto poder de processamento. Foi utilizado também a

plataforma *Google Colab Research* (Colab, 2021), para o desenvolvimento dos algoritmos e treinamento dos modelos. Além disso, foi utilizado também o *Jupyter Notebook* (Jupyter, 2021), instalado localmente em computador pessoal, pois o *Google Colab Research* possui limitação de recursos de memória e processamento, então as duas plataformas foram usadas para realização deste trabalho. As principais bibliotecas utilizadas foram: *numpy* (NumPy, 2021), e *pandas* (Pandas, 2021), para a manusear os dados, *matplotlib* (Matplotlib, 2021), e *seaborn* (Seaborn, 2021), para a plotagem dos gráficos, e por fim a biblioteca *sklearn* (Scikit-Learn, 2021), que é a biblioteca responsável pelos algoritmos de aprendizado de máquina.

5.2 BASE DE DADOS

As bases de dados a serem utilizadas serão o SIM e SINASC, que são as duas principais fontes de informações sobre nascimentos e óbitos no Brasil. O SINASC é alimentado com base na Declaração de Nascido Vivo (Declaração de Nascido Vivo - DNV) (BELUZO et al., 2020b apud Oliveira et al., 2015). O SIM tem como objetivo principal apoiar a coleta, armazenamento e processo de gestão de registros de óbitos no Brasil (BELUZO et al., 2020b apud Morais et al., 2017), e foi usado para rotular o óbito registrado no SIM, utilizando o campo DNV como chave de associação. O SIM será utilizado para rotular os registros do SINASC, pois o SIM coletas informações sobre mortalidade e é usado como base para o cálculo de estatísticas vitais, como a taxa de mortalidade neonatal e o SINASC reúne informações sobre dados demográficos e epidemiológicos do bebê, da mãe, do pré-natal e do parto (BELUZO et al., 2020b).

A Tabela 3 apresenta as variáveis da base de dados. Essa base tem variáveis que contêm valores quantitativos e categóricos. A coluna ‘num_live_births’ por exemplo contém o número de nascidos vivos anteriores da mãe e é composta por valores quantitativos contínuos, e a coluna ‘tp_pregnancy’ utiliza valores nominais categóricos que indicam o tipo de gravidez.

Tabela 3. Colunas da base de dados e suas descrições. (Fonte: Adaptado de BELUZO et al., 2020b).

Nome da coluna	Descrição	Domínio de dados
Variáveis demográficas e socioeconômicas		
maternal_age	Idade da mãe	Quantitativo Contínuo (inteiro)
tp_maternal_race	Raça / cor da pele da mãe	<i>Nominal categórico (inteiro)</i> 1 - branco; 2 - Preto; 3 - amarelo; 4 - Pele morena; 5 - Indígena.
tp_marital_status	Estado civil da mãe	<i>Nominal categórico (inteiro)</i> 1 - Único; 2 - Casado; 3 - Viúva; 4 - Separados / divorciados judicialmente; 5 - Casamento por união estável; 9 - Ignorado.
tp_maternal_schooling	Anos de escolaridade da mãe	<i>Nominal categórico (inteiro)</i> 1 - nenhum; 2 - de 1 a 3 anos; 3 - de 4 a 7 anos; 4 - de 8 a 11 anos; 5 - 12 e mais; 9 - Ignorado.
Variáveis obstétricas maternas		
num_live_births	Número de nascidos vivos	<i>Quantitativo Contínuo (inteiro)</i>
num_fetal_losses	Número de perdas fetais	<i>Quantitativo Contínuo (inteiro)</i>
num_previous_gestations	Número de gestações anteriores	<i>Quantitativo Contínuo (inteiro)</i>
num_normal_labors	Número de partos normais (trabalhos de parto)	<i>Quantitativo Contínuo (inteiro)</i>
num_cesarean_labors	Número de partos cesáreos (partos)	<i>Quantitativo Contínuo (inteiro)</i>
tp_pregnancy	Tipo de gravidez	<i>Nominal categórico (inteiro)</i> 1 - Singleton; 2 - Twin; 3 - Trigêmeo ou mais; 9 - Ignorado.
Variáveis relacionadas a cuidados anteriores		
tp_labor	Tipo de parto infantil (tipo de parto)	<i>Categórico Nominal (inteiro)</i> 1 - Vaginal; 2 - Cesariana;
num_prenatal_appointments	Número de consultas de	<i>Quantitativo Contínuo (inteiro)</i>

	pré-natal	
tp_robson_group	Classificação do grupo Robson	<i>Ordinal categórico (inteiro)</i>
Variáveis relacionadas ao recém-nascido		
tp_newborn_presentation	Tipo de apresentação de recém-nascido	<i>Categórico Nominal (inteiro) 1 - Cefálico; 2 - Pélvico ou culatra; 3 - Transversal; 9 - Ignorado.</i>
has_congenital_malformation	Presença de malformação congênita	<i>Nominal categórico (inteiro) 1 - Sim; 2 - Não; 9 - Ignorado</i>
newborn_weight	Peso ao nascer em gramas	<i>Quantitativo Contínuo (inteiro)</i>
cd_apgar1	Pontuação de Apgar de 1 minuto	<i>Ordinal categórico (inteiro)</i>
cd_apgar5	Pontuação de Apgar de 5 minuto	<i>Ordinal categórico (inteiro)</i>
gestaional_week	Semana de gestação	<i>Quantitativo Contínuo (inteiro)</i>
tp_childbirth_care	Assistência ao parto	<i>Categórico Nominal (inteiro) 1 - Doutor; 2 - enfermeira ou obstetra; 3 - Parteira; 4 - outros; 9 - Ignorado.</i>
was_cesarean_before_labor	Foi cesáreo antes do parto.	
was_labor_induced	Foi induzido ao trabalho de parto.	
is_neonatal_death	Morte antes de 28 dias (rótulo)	<i>Nominal categórico (inteiro) 0 - sobrevivente; 1 - morto.</i>

Neste trabalho foi utilizado dados do período de 2014 à 2016 devido sua melhor qualidade. Este recorte possui 6.719.190 registros dos quais 99.4% são amostras de recém-nascidos vivos e 0.6% são amostras onde os recém-nascidos vieram a óbito conforme pode ser observado na Figura 4. Com essas informações consegue-se observar que a base de dados é desbalanceada.

Porcentagem de amostras de Nascidos Vivos e Nascidos Mortos

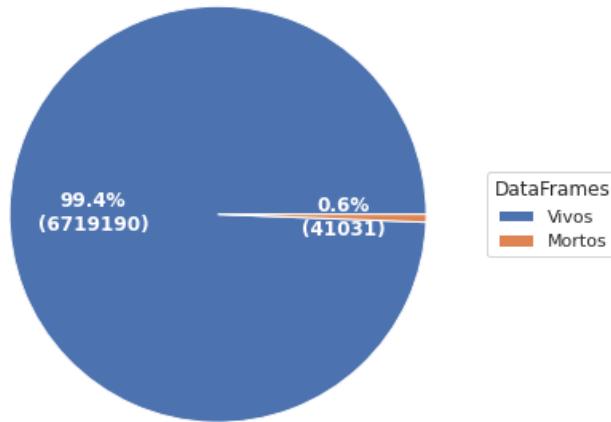


Figura 4. Distribuição da base de dados considerando se o recém-nascido viveu ou morreu durante o período neonatal. (Fonte: Elaboração Própria).

5.3 PREPARAÇÃO DA BASE DE DADOS

Nessa etapa foi realizada a preparação da base de dados para o treinamento dos modelos que consiste na limpeza, transformação e preparação dos dados a serem utilizados na análise exploratória e na criação dos modelos preditivos. A base de dados pode conter informações desnecessárias para o modelo que pode acabar atrapalhando as previsões, algumas colunas podem ser retiradas ou seus valores podem ser modificados, por exemplo valores reais são modificados para inteiros.

5.4 DESENVOLVIMENTO DOS MODELOS DE APRENDIZADO DE MÁQUINA

Como já dito anteriormente na seção 4 os algoritmos de aprendizado de máquina escolhidos para esse trabalho foram o de Regressão Logística e o de Árvore de Decisão. Foram escolhidos para este trabalho pois ambos são algoritmos de classificação supervisionados, atendendo os critérios necessários para a previsão de mortalidade neonatal. Além disso, são utilizados com frequência na comunidade acadêmica para problemas similares. A Árvore de Decisão por exemplo é um ótimo modelo para analisar as decisões que estão sendo feitas pelo modelo, uma vez que esse algoritmo tem a característica de ser um *WhiteBox*, ou seja, é possível ver o que o modelo aprendeu e o que ele está decidindo.

Para o algoritmo de Regressão Logística foi realizado três experimentos , no primeiro experimento foi utilizado o particionamento 90/10 onde foi usado 90% da base para o treinamento do modelo e 10% para os testes; no segundo experimento foi utilizado o *K-folds cross-validation* em um novo treinamento para conferirmos se o modelo estava sofrendo *overfitting* (sobreajuste); e para o terceiro experimento, com a intenção de melhorar as taxas preditivas do modelo, foi utilizado o método de treinamento *K-folds cross-validation* juntamente o método RFECV (Eliminação Recursiva de Feature com Validação Cruzada), o qual executa método RFE (Eliminação Recursiva de *Feature*), que tem como ideia selecionar *features* considerando recursivamente conjuntos cada vez menores de *features*.

No método RFE, inicialmente o estimador é treinado no conjunto inicial de *features* e a importância de cada *feature* é obtida por meio de um atributo "*coef*" ou por meio de um atributo "*feature_importances*". Em seguida, as *features* menos importantes são removidas do conjunto atual de *features*. Esse procedimento é repetido recursivamente no conjunto removido até que o número desejado de *features* a serem selecionados seja finalmente alcançado. Já o RFECV executa a RFE em uma iteração de validação cruzada para encontrar o número ideal ou o melhor número de *features*. Por fim, para completar a metodologia de aplicação de métodos de aprendizado de máquina, foi utilizado ainda o método *GridSearchCV*, essa função permite que seja realizado testes alterando a combinação de parâmetros do algoritmo em questão na criação dos nossos modelos (MNASSRI, 2020).

Já para o algoritmo de Árvore de Decisão foi realizado dois experimentos, no primeiro foi utilizado o particionamento 90/10 onde foi usado 90% da base para o treinamento do modelo e 10% para os testes, no segundo experimento foi utilizado o *K-folds cross-validation* em um novo treinamento para conferirmos se o modelo estava sofrendo *overfitting* (sobreajuste).

O algoritmo de Regressão Logística e análise exploratória foi baseado em códigos disponível na plataforma web *Kaggle* (MNASSRI, 2020), no exemplo do *Kaggle* o autor faz os códigos para predizer mortes no navio Titanic, para o trabalho de mortalidade neonatal os códigos foram alterados para realizar predição de mortes neonatais. Já no algoritmo de Árvore de Decisão foi baseado em uma vídeo aula do professor Diogo Cortiz (ÁRVORE..., 2020), nessa vídeo aula o professor explica sobre os algoritmos de Árvore de Decisão e depois implementa o um exemplo de código, o qual foi alterado para realizar as predições de mortalidade neonatal.

5.5 ANÁLISE DOS RESULTADOS

Para a análise de resultados foram utilizadas duas métricas: a matriz de confusão e a curva ROC. Com elas pode-se realizar uma avaliação do modelo, e assim foi possível analisar os valores de acurácia, precisão e *recall* do modelo, métricas que permitem avaliar o desempenho do modelo. Com a acurácia é possível calcular a proximidade entre o valor obtido na predição dos modelos e os valores esperados. Com a precisão é possível analisar as amostras que o modelo classificou como 0 eram realmente 0 na classe real, e com o *recall* é possível analisar as amostras que o modelo conseguiu reconhecer como a classe correta.

5.6 DISPONIBILIZAÇÃO DA BASE DE DADOS E CÓDIGOS FONTES

A base de dados está em processo de disponibilização na plataforma *Synapse*, e pode ser acessada diretamente pela url <https://www.synapse.org/#!Synapse:syn25575811>, mas pode ser solicitada ao autor. Os códigos fontes estão disponibilizados no GitHub e podem ser acessados pela url

https://github.com/CaioSMota/TCC_CaioMota_09-06-2021_TADS_ModalidadeNeonatal.

6 RESULTADOS

Os mesmos experimentos mostrados abaixo foram aplicados para uma base de dados que contém somente dados de São Paulo, a base SPNeoDeath (BELUZO et al.,2020b). Essa base é um recorte da base do Brasil todo em contém cerca de 1.400.000 amostras, esse recorte da base também é bem desbalanceado. Os resultados obtidos neste experimento usando o recorte de São Paulo foram bem parecidos com os experimentos da base do Brasil todo, e os códigos do experimento de árvore de decisão utilizando a base de São Paulo podem ser visualizados no apêndice A. E os experimentos de regressão logística utilizando a base de São Paulo podem ser visualizados no apêndice B.

6.1 ANÁLISE EXPLORATÓRIA

6.1.1 Apresentação dos Dados

Como dito anteriormente na seção 5.2 “Base de Dados” deste trabalho, a base de dados é formada pelas informações do SIM e do SINASC contém 6.760.222 amostras e 29 colunas, porém serão utilizadas somente 23 colunas sendo elas as colunas descritas na Tabela 3. Na seção 5.2 também tem a Figura 4 que mostra que a base dados é desbalanceada tendo 99.4% das amostras de recém-nascidos vivos e 0.6% das amostras onde os recém-nascidos vieram a óbito no período neonatal. O código completo deste experimento está disponível no apêndice C, e pode ser obtido no GitHub do projeto.

6.1.1. Características demográficas e socioeconômicas maternas

O apêndice D contém uma tabela que possui as estatísticas summarizadas das variáveis demográficas e socioeconômicas maternas. Nesta tabela podemos observar por exemplo a coluna “*maternal_age*” que contém summarização dos dados da variável a idade da mãe. Podemos observar que a idade média das mães é de 26.4 anos, e os dados possuem uma distribuição entre 8 e 55 anos, com uma mediana de 26 anos. Esta tabela também apresenta as colunas: “*tp_maternal_schooling*” que mostra a categoria de anos de escolaridade da mãe, “*tp_marital_status*” que mostra o estado civil da mãe em dados categóricos e a coluna “*tp_maternal_race*” que mostra a raça da mãe também em dados categóricos.

6.1.1.2 Variáveis obstétricas maternas

No apêndice E está disponível uma tabela que possui as estatísticas summarizadas das variáveis obstétricas maternas. Nesta tabela por exemplo temos a coluna “*num_live_births*”, essa coluna mostra o número de nascidos vivos anteriores. Esta variável apresenta média de 0,94 nascidos vivos por mãe, com mediana 1, o número mínimo de 0 e máximo de 10 nascidos vivos. Podemos observar também a coluna, “*num_fetal_losses*”, que indica o número de perdas fetais anteriores, com média de 0,21, mediana 0, mínimo de 0 e máximo de 5. Esses valores indicam que poucas mães tiveram perdas fetais antes da gravidez atual. Ainda nesta tabela estão disponíveis as estatísticas da coluna “*num_previous_gestations*” que mostra o número de gestações anteriores da mãe, com média de 1,14, mediana 1, número mínimo 0 e o máximo 10. Essa tabela também contém as colunas: “*num_normal_labors*” que é o número de partos normais da mãe, “*num Cesarean labor*” que mostra o número de partos cesáreos da mãe e por fim mostra a coluna “*tp_pregnancy*” que é o tipo da gravidez.

6.1.1.3 Variáveis de histórico de gravidez

No apêndice F temos uma tabela que possui as estatísticas summarizadas das variáveis do histórico de gravidez. Nesta tabela temos as colunas “*num_prenatal_appointments*” que mostra o número de consultas de pré-natal, com média de 7,8, mediana 8, e com variação entre 0 e 40 consultas. A tabela também contém as colunas: “*tp_labor*” que é o tipo de parto, e a coluna “*tp_robson_group*” que mostra estatísticas para variável classificação do grupo Robson.

6.1.1.4 Variáveis relacionadas ao recém-nascido

No apêndice G temos uma tabela que possui as estatísticas summarizadas das variáveis relacionadas ao recém-nascido. Nesta tabela por exemplo podemos observar a coluna “*newborn_weight*” que armazena o peso ao nascer dos recém-nascidos em gramas, a média dos dados dessa coluna é 3.187,3, a mediana é 3.215, a variação dos dados é de no mínimo 0 e

no máximo 6.000 gramas. Também podemos observar a coluna “*gestacional_week*” que mostra o número de semanas de gestação, a média é 38,4, a mediana é 39, a variação dos dados é de no mínimo 15 e no máximo 45 semanas. Além dessas colunas a tabela também contém as colunas: “*cd_apgar1*” que mostra a pontuação de Apgar de 1 minuto, “*cd_apgar5*” que mostra a pontuação de Apgar de 5 minuto, “*has_congenital_malformation*” que mostra se o recém-nascido contém a presença de alguma malformação, “*tp_newborn_presentation*” que mostra o tipo de apresentação de recém-nascido, “*tp_labor*” que mostra o tipo de parto, “*was_cesarean_before_labor*” que mostra se o recém-nascido foi cesáreo antes do parto, “*was_labor_induced*” que mostra se o recém-nascido foi induzido ao trabalho de parto, “*tp_childbirth_care*” que mostra se ouve assistência ao parto, e por fim temos a coluna “*is_neonatal_death*” que mostra se o recém-nascido veio a óbito ou não.

6.1.2 Análise das variáveis

Nesta seção serão mostrados a parte de análise de variáveis com diferentes tipos de gráficos. Os gráficos foram baseados na análise exploratória do artigo BELUZO et al.(2020b), e foram adaptados para este trabalho

6.1.2.1 Variáveis contínuas

Na Figura 5 temos dois gráficos *boxplot*. No *boxplot* à esquerda temos a distribuição da variável peso ao nascer e nele é possível observar duas caixas onde a caixa azul são os vivos e o laranja são os mortos. Nas duas caixas mostradas no gráfico podemos observar pequenos círculos nas pontas, esses círculos são *outliers* (dados fora da curva), então para a caixa de vivos os *outliers* são recém-nascidos com pesos acima de 4.500 gramas ou abaixo de 2.000 gramas, e para a caixa de mortos os outliers são os recém-nascidos com mais de 5.500 gramas. Para os vivos temos a mediana de aproximadamente 3.200 gramas e para os mortos a mediana é 1.300 gramas. Na caixa azul de vivos podemos ver que o peso de recém-nascidos que sobreviveram está aproximadamente entre 2.700 a 3.600 gramas, e para a caixa laranja de mortos o peso de recém-nascidos que vieram a óbito estão aproximadamente entre 700 a 2.500 gramas. Então o gráfico mostra para nós que os recém-nascidos que têm maior risco de vir a óbito tem pesos menores que 2.000 gramas.

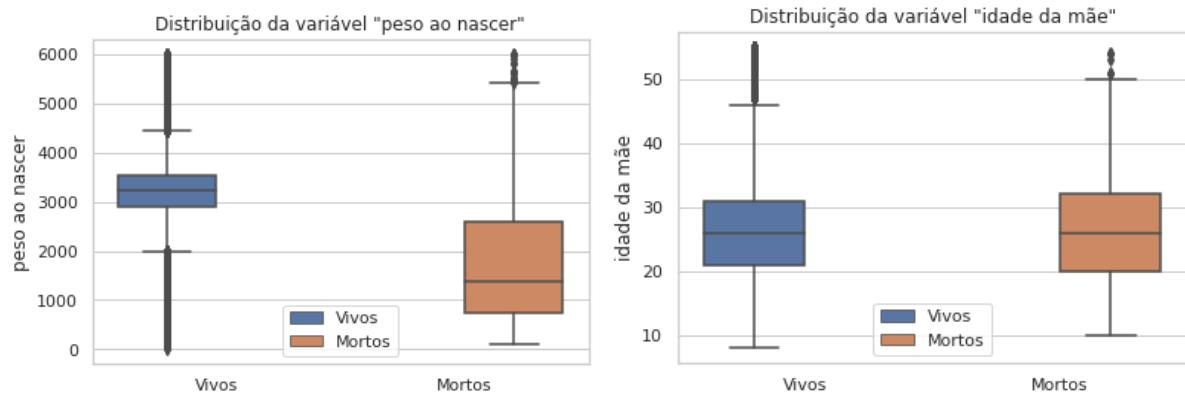


Figura 5. BoxPlot das features Peso ao nascer e Idade da mãe. (Fonte: Elaboração Própria).

O gráfico à direita da Figura 5 tem um *boxplot* da distribuição da variável idade da mãe. Nesse boxplot então podemos ver que os *outliers* da caixa de vivos são mães com idade acima de 46 anos aproximadamente, para os mortos os *outliers* são mães com idade acima de 50 anos. Para os vivos temos a mediana de aproximadamente 26 anos e para os mortos a mediana é de 26 anos. Nos vivos a idade das mães está aproximadamente entre 21 a 32 anos, e para os mortos a idade das mães está aproximadamente entre 20 a 34 anos. É importante ressaltar que o gráfico mostra as duas caixas bem parecidas, então de acordo com os dados não contém diferença significativa entre vivos e mortos usando a idade da mãe para distribuir.

Na Figura 6 temos um *boxplot* da distribuição da variável semanas de gestação. Nesse gráfico podemos ver que os *outliers* da caixa de vivos são gestações com mais de 44 semanas aproximadamente, e gestações com menos de 35 semanas. Para os vivos temos a mediana de aproximadamente 39 semanas e para os mortos a mediana é de 32 semanas. Nos vivos as semanas de gestação estão aproximadamente entre 36 a 40 semanas, e para os mortos a semanas de gestação estão aproximadamente entre 26 a 36 semanas. Os dados mostraram ainda que para as gestações que têm menos de 36 semanas os recém-nascidos têm maior risco de vir a óbito.

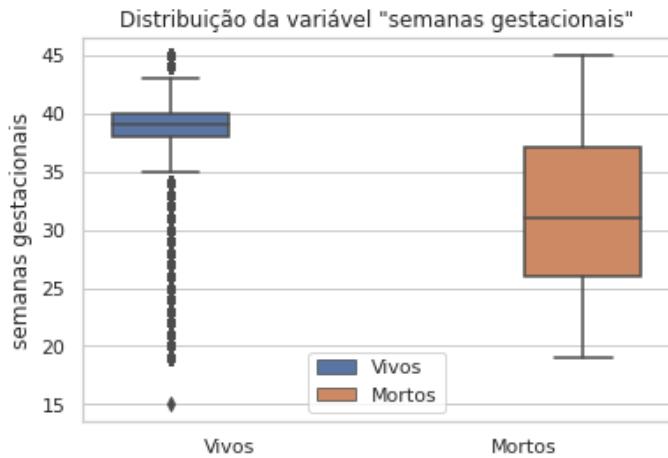


Figura 6. BoxPlot da feature semana de gestação. (Fonte: Elaboração Própria).

Na Figura 7 temos um histograma da distribuição de peso ao nascer por classe, podemos observar no gráfico que grande parte dos vivos (linha azul) estão distribuídos entre 2.000 e 4.000 gramas, a área entre esses valores tem muitos dados de vivos, já entre os valores 0 e 2.000 gramas temos uma concentração dos dados de mortos.

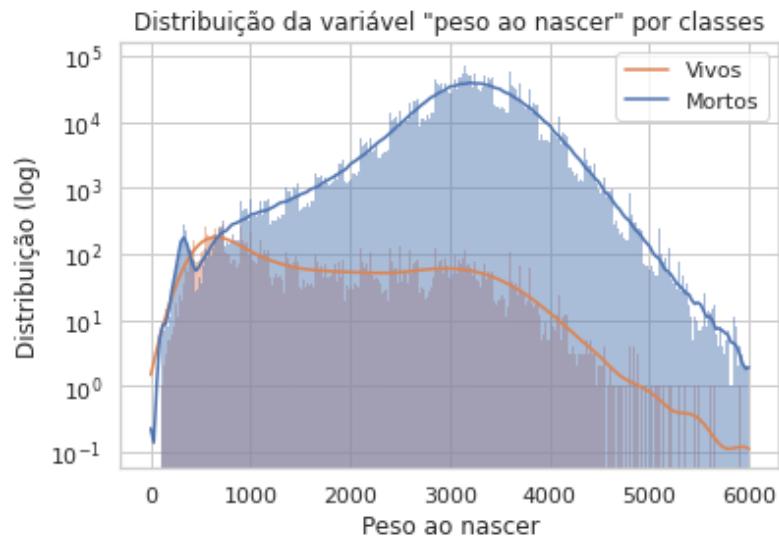


Figura 7. Histograma de distribuição do peso entre as classes. (Fonte: Elaboração Própria)

6.1.2.2 Variáveis categoricas

Observando a Figura 8 podemos ver um gráfico de barras da distribuição da variável escolaridade da mãe, este gráfico mostra a contagem de registros por escolaridade da mãe,

esse gráfico mostra para nós que a maior parte das mães estão na categoria 4 que representa de 8 a 11 anos de escolaridade.

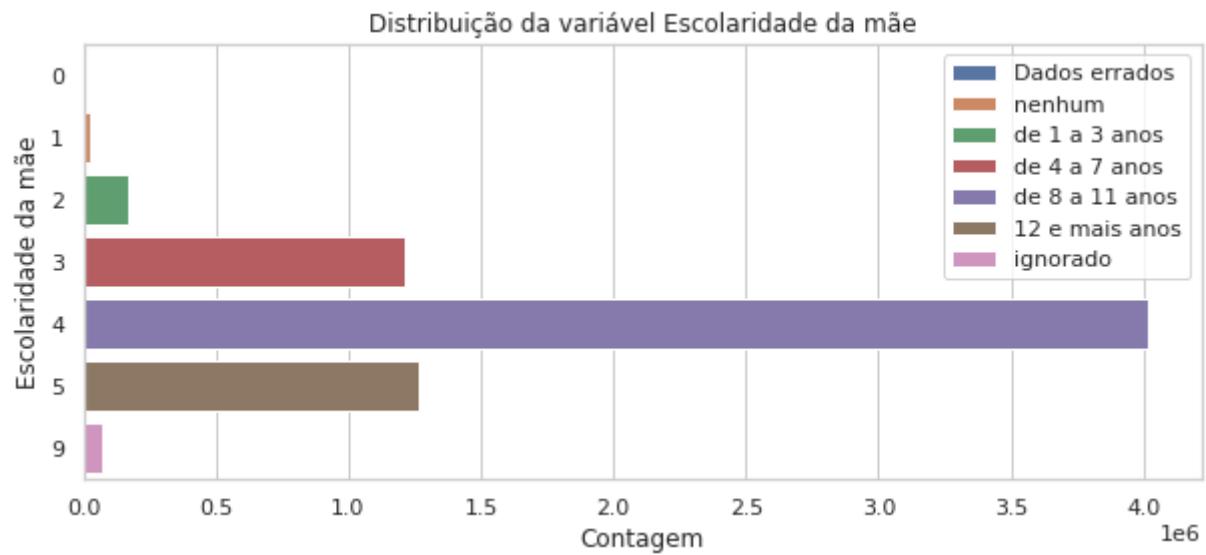


Figura 8. Gráfico de barras da distribuição da variável Escolaridade da mãe. (Fonte: Elaboração Própria).

Observando a Figura 9 podemos ver um gráfico de barras da distribuição da variável número de vivos, este gráfico mostra a contagem de registros por número de vivos, esse gráfico mostra para nós que a grande maioria das mães está tendo o primeiro filho ou o segundo filho.

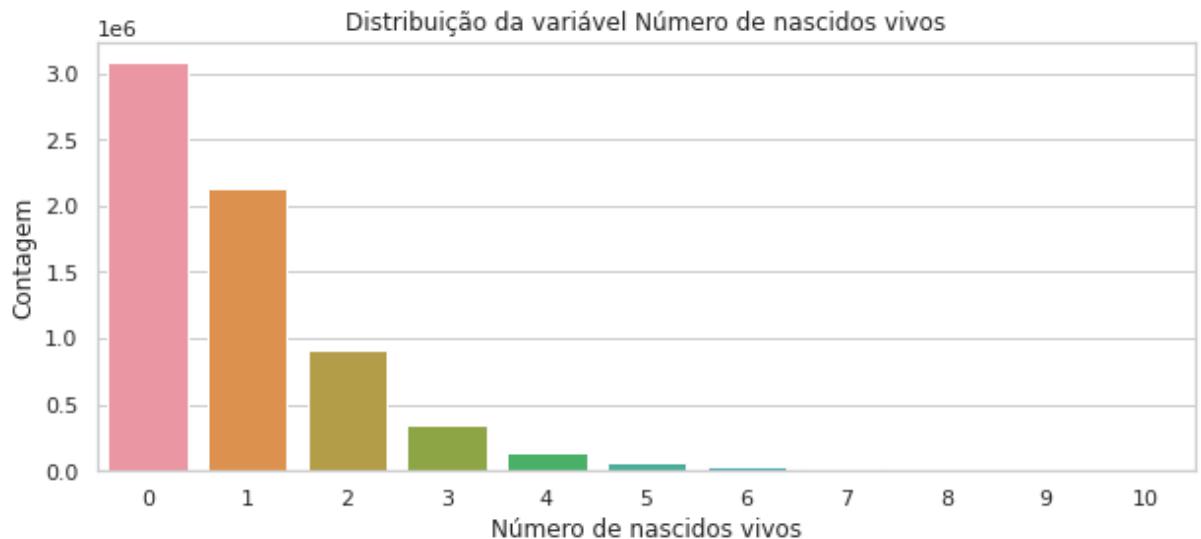


Figura 9. Gráfico de barras da distribuição da variável Número de nascidos vivos. (Fonte: Elaboração Própria).

Observando a Figura 10 podemos ver um gráfico de barras da distribuição da variável pontuação Apgar de 1 minuto, este gráfico mostra a contagem de registros por pontuação

Apgar de 1 minuto, esse gráfico mostra para nós que a grande maioria dos dados possuem apgar de 8 ou 9, esse alto valor de apgar é por conta da base ser desbalanceada e a maior parte dos dados são de recém-nascidos que sobreviveram.

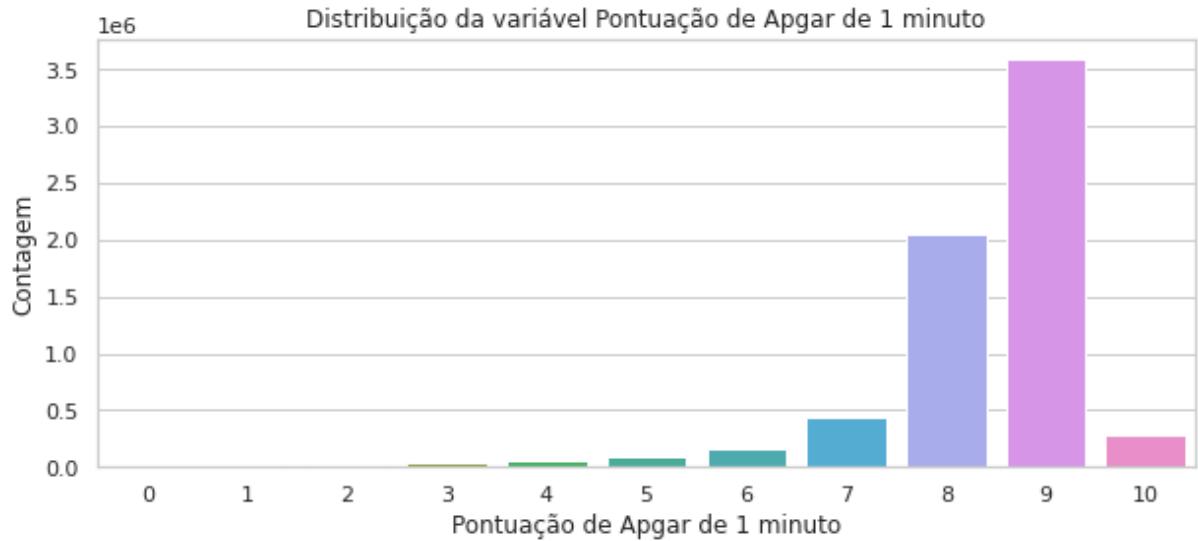


Figura 10. Gráfico de barras da distribuição da variável Pontuação de Apgar de 1 minuto.
(Fonte: Elaboração Própria).

Observando a Figura 11 podemos ver um gráfico de barras da distribuição da variável pontuação Apgar de 5 minuto, este gráfico mostra a contagem de registros por pontuação Apgar de 5 minuto, esse gráfico mostra para nós que a grande maioria dos dados possuem apgar de 9 ou 10, esse alto valor de apgar é por conta da base ser desbalanceada e a maior parte dos dados são de recém-nascidos que sobreviveram.

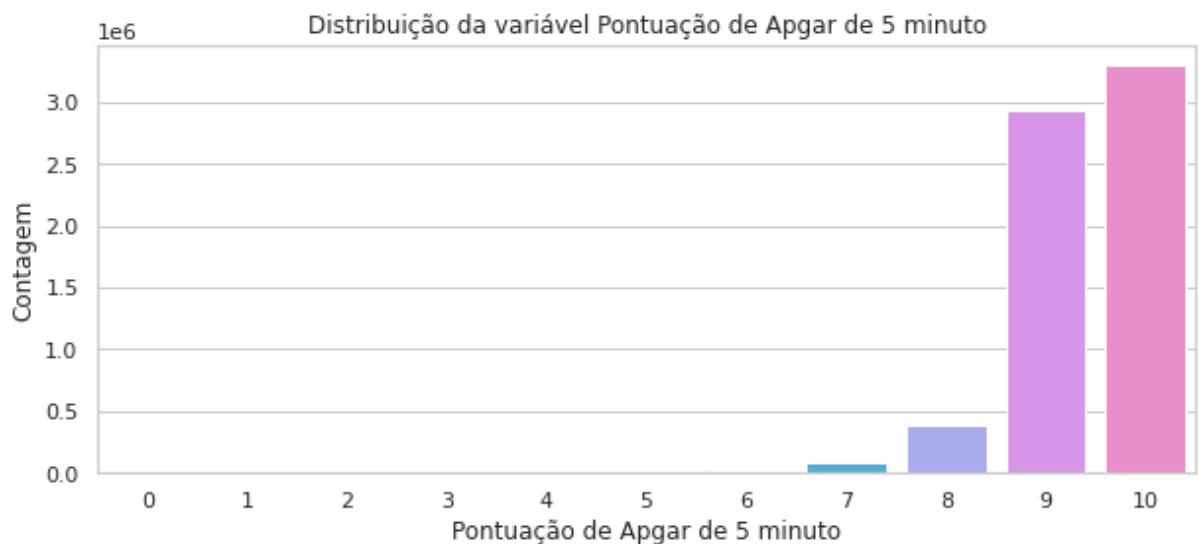


Figura 11. Gráfico de barras da distribuição da variável Pontuação Apgar de 5 minutos.
(Fonte: Elaboração Própria).

Observando a Figura 12 podemos ver um gráfico de barras da distribuição da variável presença de malformação congênita, este gráfico mostra a contagem de registros por presença de malformação congênita, esse gráfico mostra para nós que a grande maioria dos dados estão na categoria 2 que mostra que o recém-nascido não possui malformação, esse grande volume para a categoria 2 é causada pelo desbalanceamento da base.

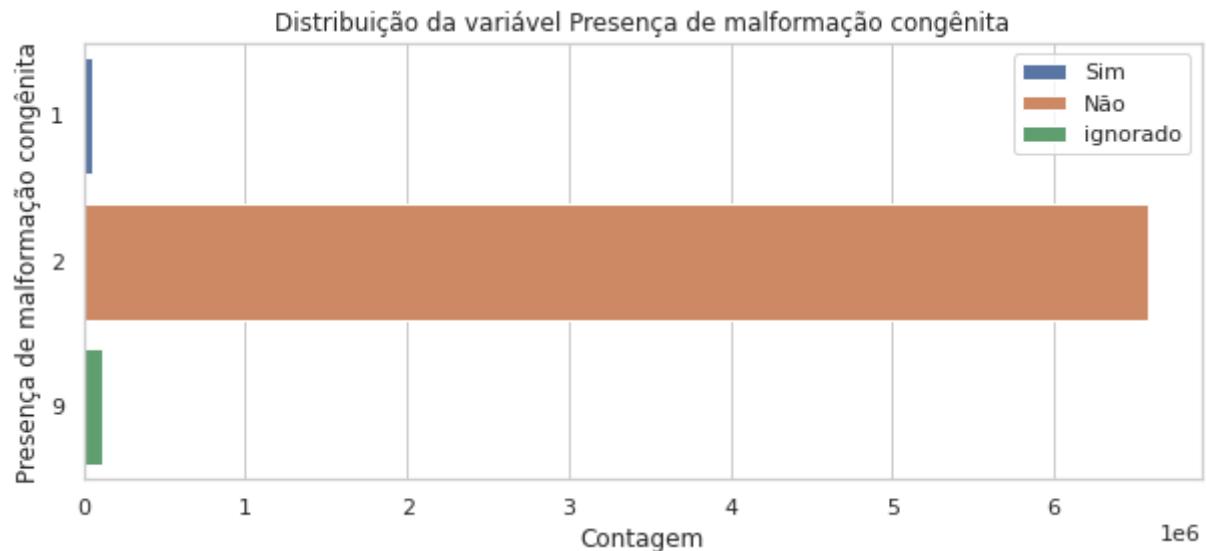


Figura 12. Gráfico de barras da distribuição da variável Presença de malformação congênita. (Fonte: Elaboração Própria).

Observando a Figura 13 podemos ver um gráfico de barras da distribuição da variável número de gestações, este gráfico mostra a contagem de registros por número de gestações, esse gráfico mostra para nós que a grande maioria das mães está tendo o primeiro filho ou o segundo filho, da mesma forma mostrada na Figura 9.

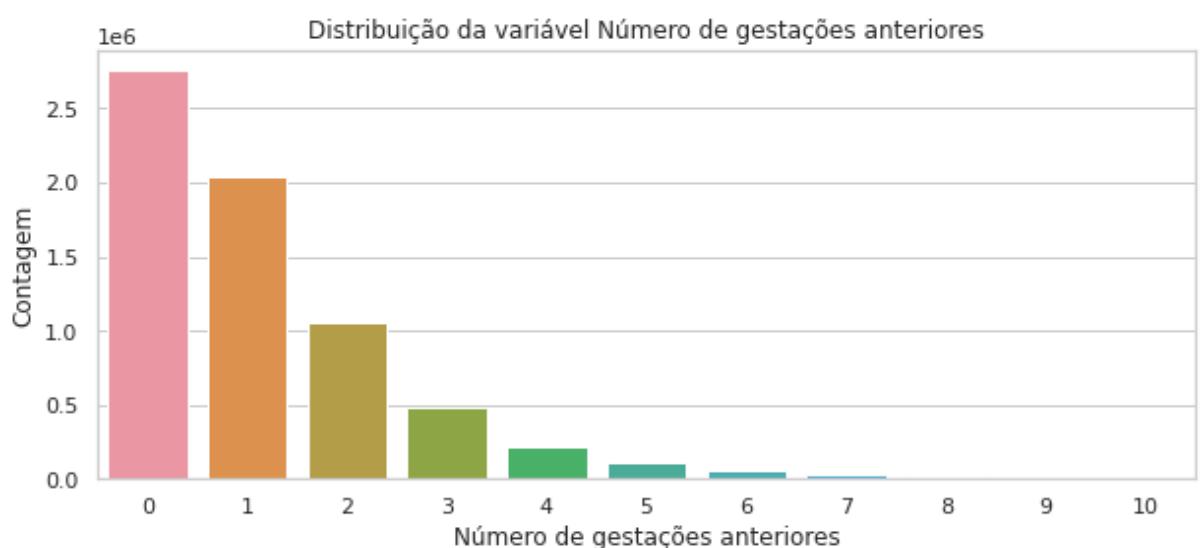


Figura 13. Gráfico de barras da distribuição da variável Número de gestações. (Fonte: Elaboração Própria).

Observando a Figura 14 podemos ver um gráfico de barras da distribuição da variável assistência ao parto, este gráfico mostra a contagem de registros por assistência ao parto, esse gráfico mostra para nós que a grande maioria dos dados mostram que o parto teve assistência de uma Enfermeira ou obstetra ou essa informação foi ignorada na hora do registro.

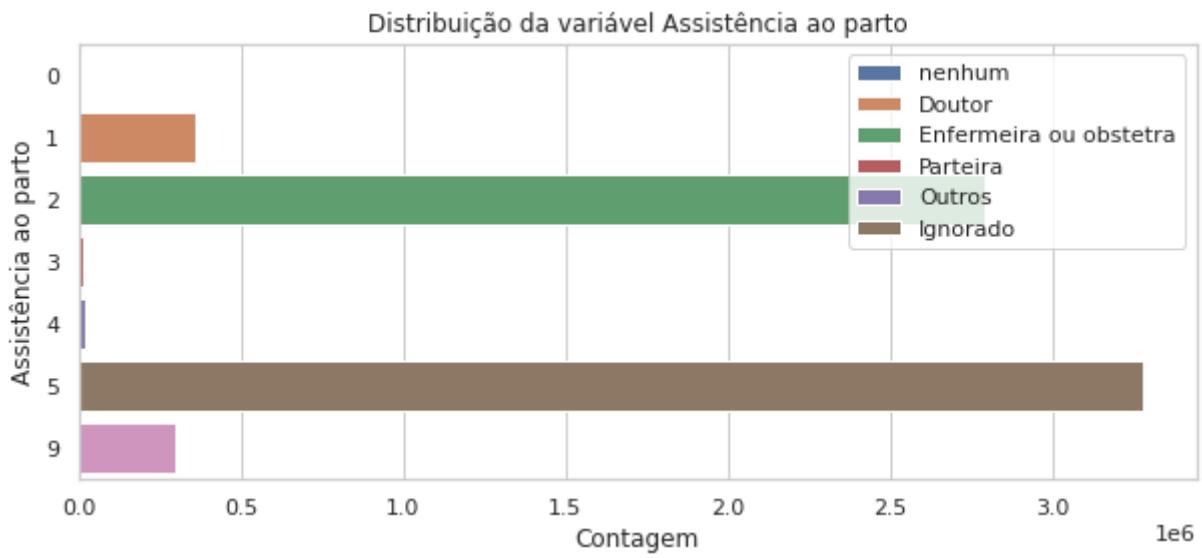


Figura 14. Gráfico de barras da distribuição da variável Assistência ao parto. (Fonte: Elaboração Própria).

6.1.2.3 Analise bi-variada

Na Figura 15 podemos observar a importância da *feature* peso ao nascer. No gráfico é possível observar claramente a diferença de pesos entre vivos e mortos, recém-nascidos com peso acima de 2.000 gramas sobreviveram, já os recém-nascidos com menos de 2.000 gramas vieram a óbito.

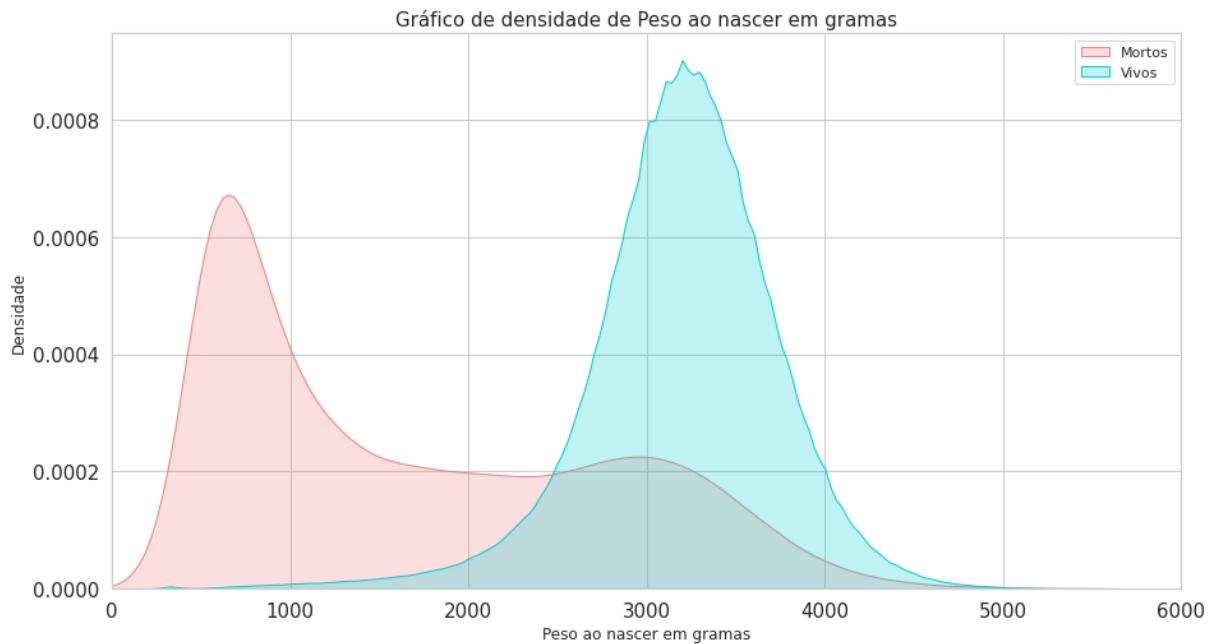


Figura 15. Gráfico de densidade de peso entre as classes. (Fonte: Elaboração Própria).

Na Figura 16 podemos observar um gráfico de correlação entre algumas *features* de valores contínuos, algumas dessas features possuem correlações positivas, como pontuação apgar de 1 minuto e pontuação apgar de 5 minutos, essas colunas possuem uma correlação de 0,7, então nascidos que recebem um valor alto de apgar de 1 minuto tendem a receber um valor alto no apgar de 5 minutos. O gráfico também mostra outras features com correlações positivas como, número de partos normais e número de gestações anteriores que contém uma correlação de 0,81, número de gestações anteriores com idade da mãe que tem uma correção de 0,39, número de partos de cesáreos com idade da mãe que possui correlação de 0,24, número de partos normais com idade da mãe com a correlação de 0,26, semanas de gestação com peso ao nascer em gramas com correção de 0,52, e as apgar de 1 e 5 minutos com semanas de gestação. E grande parte das *features*, possuem correlações baixas então elas são inversamente correlacionadas pelo que o gráfico mostra, como por exemplo número de partos normais com número de consultas pré-natais possuem uma correção de -0,17, e número de partos cesáreos com número de partos normais que contém uma correlação de -0,15. Então a correlação dessas *features* estão bem baixas tirando as correlações acima de 0,7.

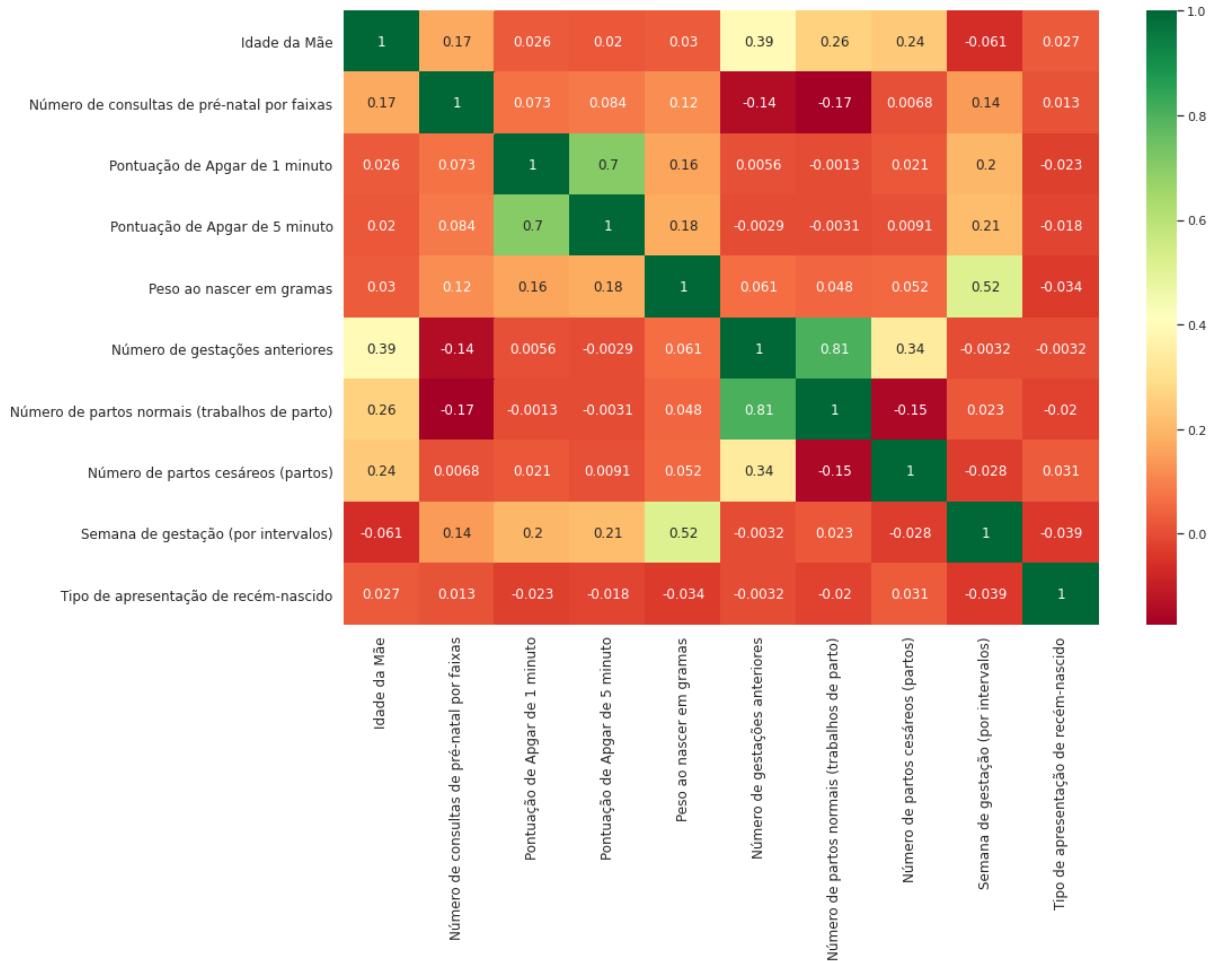


Figura 16. Gráfico de correlação. (Fonte: Elaboração Própria)

6.1.2.4 Distribuição por raça

Na Figura 17 podemos ver um gráfico de boxplot mostrando a distribuição da idade da mãe por raça, e podemos observar que a raça 5 (indígena), é o boxplot que contem a menor variação de idade, já as raças 1 (branco) e 3 (amarelo) são as caixas que possuem maior variação com a média em torno de 20 e 30 anos.

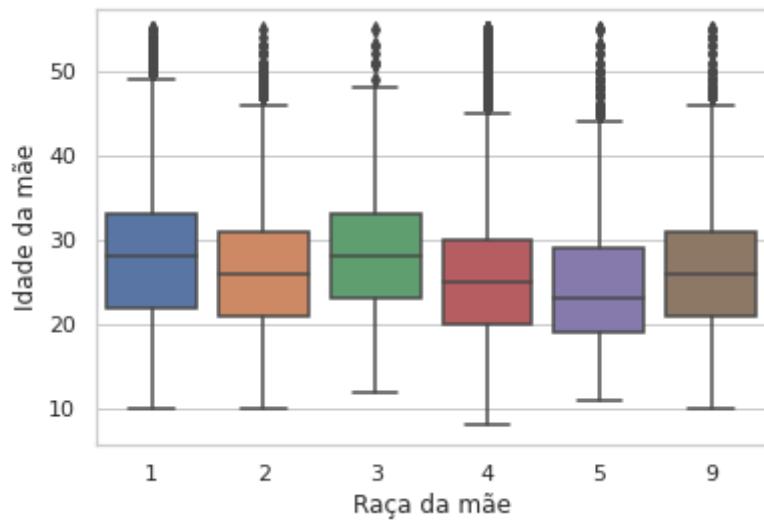


Figura 17. Distribuição da idade da mãe por raça. (Fonte: Elaboração Própria).

No boxplot mostrado na Figura 18 podemos observar a distribuição de peso ao nascer por raça, e podemos ver que não tem diferença significativa entre as caixas, todas são praticamente iguais. Então pouca variação é observada entre as raças para peso ao nascer.

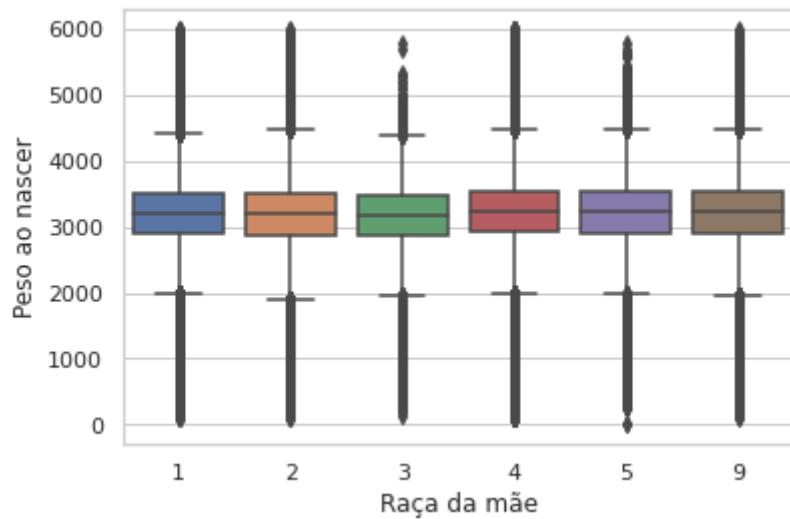


Figura 18. Distribuição de peso ao nascer por raça. (Fonte: Elaboração Própria).

Na Figura 19 temos o boxplot da a distribuição de semanas de gestação por raça, e praticamente todas as caixas são iguais, somente a caixa da raça 1 (branco) possui menor variação que as outras.

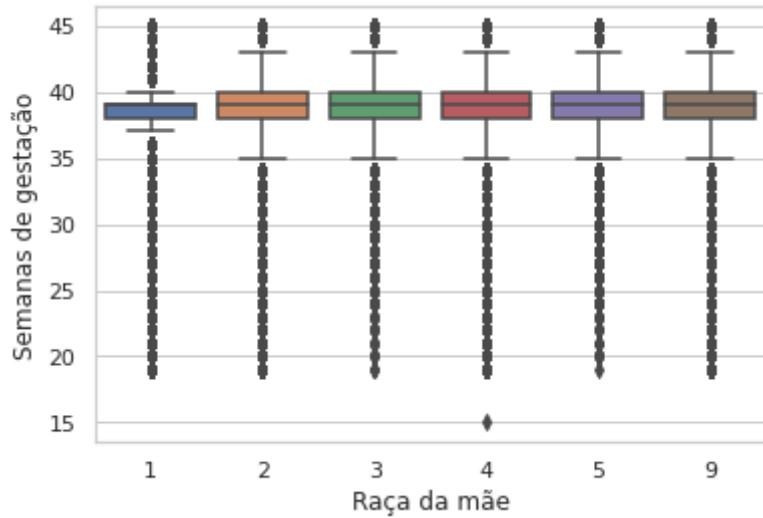


Figura 19. Distribuição de semanas de gestação por raça. (Fonte: Elaboração Própria).

6.1.2.5 Distribuição por estado civil

A Figura 20 mostra o boxplot da distribuição de peso ao nascer por estado civil, e podemos ver que não tem diferença significativa entre as caixas, todas são praticamente iguais. Então pouca variação é observada entre os estados civis para peso ao nascer.

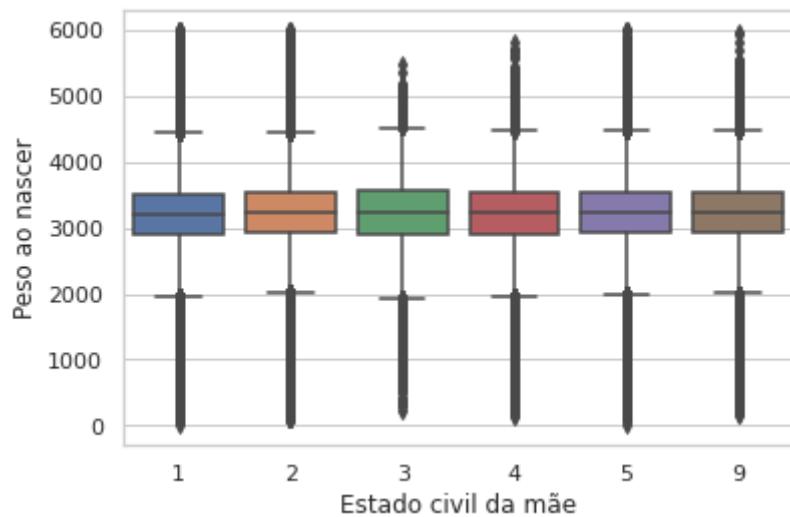


Figura 20. Distribuição de peso ao nascer por estado civil. (Fonte: Elaboração Própria).

A Figura 21 mostra o boxplot da distribuição de semanas de gestação por estado civil, e podemos ver que não tem diferença significativa entre as caixas, todas são

praticamente iguais. Porém os estados civis 2 (Casado) e 4 (Separados/divorciados judicialmente), contém variações menores.

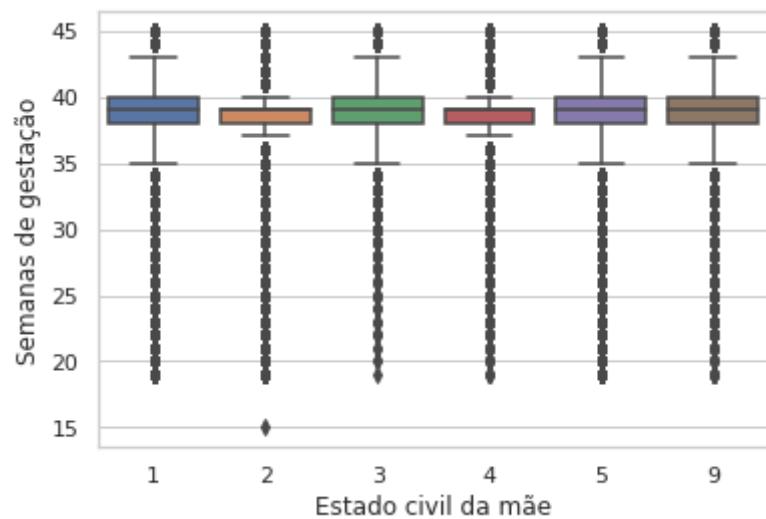


Figura 21. Distribuição de semanas de gestação por estado civil. (Fonte: Elaboração Própria).

Na Figura 22 podemos ver um gráfico de boxplot mostrando a distribuição da idade da mãe por estado civil, e podemos observar que o estado civil 3 (Viúva), possui a maior variação, já o estado civil 4 (Separados/divorciados judicialmente) é a caixa que possuem menor variação

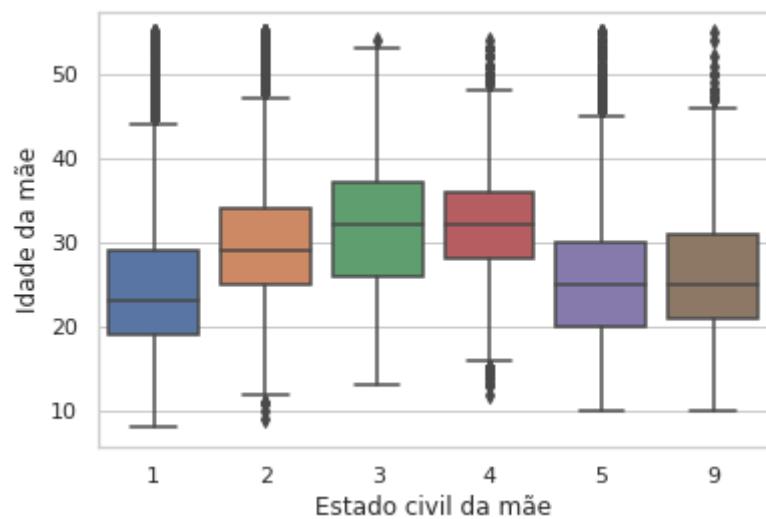


Figura 22. Distribuição da idade da mãe por estado civil. (Fonte: Elaboração Própria).

6.1.2.6 Distribuição por escolaridade da mãe

Na Figura 23 podemos ver um gráfico de boxplot mostrando a distribuição da idade da mãe por escolaridade da mãe, e podemos observar que a escolaridade 2 (1 a 3 anos) e 3 (4 a 7 anos), possui as maiores variações, já escolaridade 5 (12 ou mais anos) é a caixa que possuem menor variação

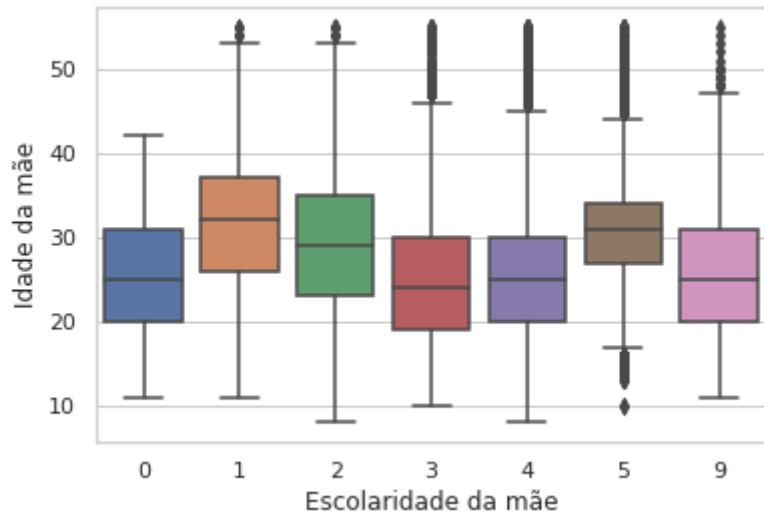


Figura 23. Distribuição da idade da mãe por escolaridade da mãe. (Fonte: Elaboração Própria).

A Figura 24 mostra o boxplot da a distribuição de peso ao nascer por escolaridade da mãe, e podemos ver que que não tem diferença significativa entre as caixas, todas são praticamente iguais. Então, pouca variação é observada entre a escolaridade da mãe para peso ao nascer.

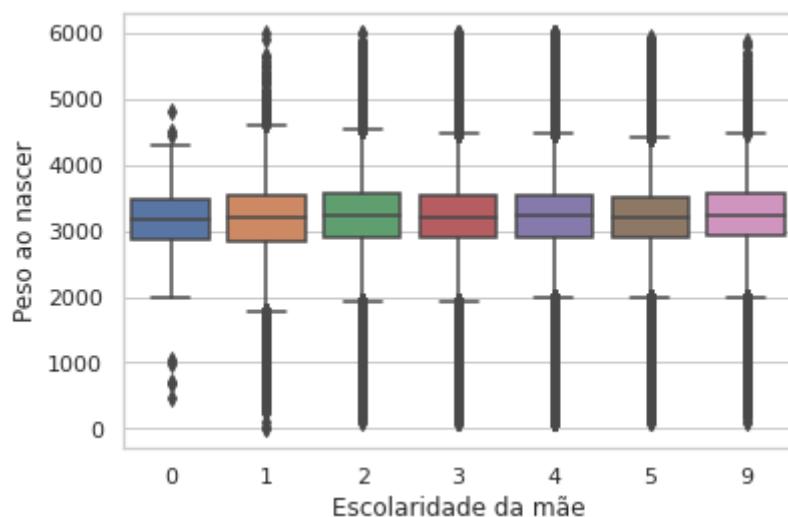


Figura 24. Distribuição de peso ao nascer por escolaridade da mãe. (Fonte: Elaboração Própria).

A Figura 25 mostra o boxplot da distribuição de semanas de gestação por escolaridade da mãe, e podemos ver que não tem diferença significativa entre as caixas, todas são praticamente iguais. Porém a escolaridade 5 (12 ou mais anos) contém variação menor que as outras.

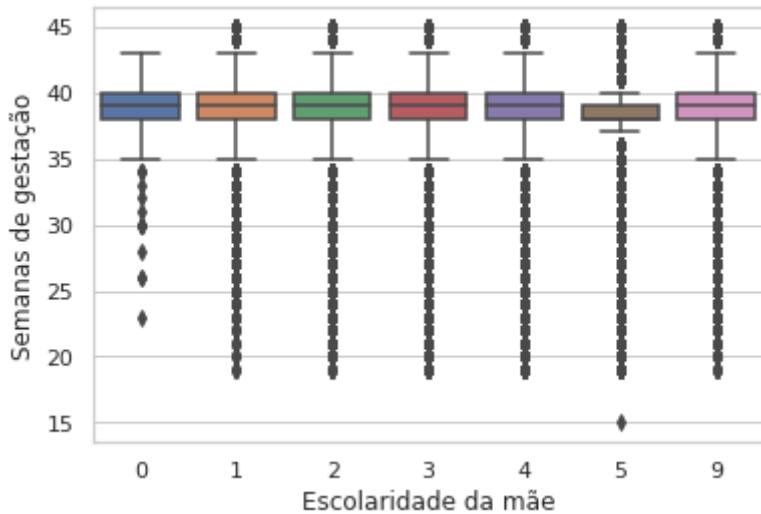


Figura 25. Distribuição de semanas de gestação por escolaridade da mãe. (Fonte: Elaboração Própria).

6.2 ÁRVORE DE DECISÃO

A Figura 26 mostra um trecho do algoritmo utilizado, nesse trecho é realizado a divisão do *DataFrame* das *features* de entrada , e o *DataFrame* que contém o rótulo.Neste caso o rótulo será o *DataFrame* “target”, esse Dataframe contém a feature “is_neonatal_death”, essa *feature* informa se o recém nascido veio a óbito ou não, sendo o foco da predição que desejamos realizar essa *feature* entra como rótulo para o modelo, já no *Dataframe* “nome_features” temos as *features* que vão servir de entrada para o modelo, é a partir dessas *features* que o modelo tentará encontrar padrões para predizer o risco de óbito do recém nascido. O código completo deste experimento pode ser visualizado no apêndice H, ou também no link do github exibido na “5.6 ACESSO A BASE DE DADOS E CÓDIGOS”.

```
[ ] # "Features" do modelo
nome_features = ['maternal_age', 'tp_maternal_schooling', 'tp_marital_status',
                  'tp_maternal_race', 'num_live_births', 'num_fetal_losses',
                  'num_previous_gestations', 'num_normal_labors', 'num Cesarean_labor',
                  'tp_pregnancy', 'newborn_weight', 'gestational_week',
                  'cd_apgar1', 'cd_apgar5', 'has_congenital_malformation',
                  'tp_newborn_presentation', 'num_prenatal_appointments', 'tp_labor',
                  'was_cesarean_before_labor', 'was_labor_induced', 'tp_childbirth_care',
                  'tp_robson_group']

target = ['is_neonatal_death']

# Array de features
X = df[nome_features].values

Y = df[target].values
```

Figura 26. Separação dos DataFrames de entrada e rótulo. (Fonte: Elaboração Própria).

6.2.1 Modelo de Árvore de Decisão utilizando particionamento 90/10

O algoritmo de árvore de decisão foi treinado utilizando duas formas diferentes: usando as partições 90% para treino e 10% para os testes e utilizando o método *K-folds cross-validation*. E para os experimentos iniciais foi utilizado o particionamento 90/10.

Tabela 4. Resultados do modelo de árvore de decisão. (Fonte: Elaboração Própria)

	Precision	Recall	F1-Score	Support
Vivo(0)	1,00	1,00	1,00	671.807
Morto(1)	0,71	0,29	0,41	4.216

Analizando a Tabela 4 pode-se ver os seguintes resultados, analisando a *precision* vemos que para a classe 0 a precisão é de 1.00, ou seja, 100% então todos as amostras que o modelo classificou como 0 eram realmente 0 na classe real, já para a classe 1 o modelo classificou 71% que realmente pertenciam a classe 1, então cerca de 29% das classificações que o modelo colocou como 1 estão incorretas. Analisando o *recall* é possível ver que o modelo conseguiu identificar 100% das classificações 0, o modelo conseguiu reconhecer muito bem as amostras classificadas como 0, já para as amostras classificadas como 1 o modelo reconheceu apenas 29% das amostras , o modelo deixou passar muitas amostras da classe 1 e não classificou como 1. Então o modelo treinado não está identificando muito bem a classe 1 que seria dos recém nascidos que poderiam vir a óbito, já para a classe de vivos a

classe 0 o modelo está identificando muito bem e classificando de forma correta. A tabela também mostra os valores de *F1-Score* para cada classe, esse resultado é formado pela soma da *Precision* e do *Recall*.

Esse modelo teve uma acurácia de 0,99 que é uma acurácia bem alta, porém somente pela acurácia não é possível dizer que o modelo está excelente pelo motivo da base que está sendo usada é altamente desbalanceada, então para a classe 0 que é a de vivos temos muito mais dados que para a classe de mortos, e o modelo está acertando bastante a classe de vivos logo a acurácia fica alta por esses acertos, enquanto para a classe de mortos o modelo não está acertando tanto.

A Figura 27 mostra a curva ROC do modelo, pode-se observar que a AUC da curva ROC ficou em 0,92, a AUC mostra que o modelo está acertando bastante as previsões, pois quanto mais perto de 1 melhor está no nosso modelo, mas no caso desse modelo vimos acima que as previsões para a classe de mortos(1) está ruim, o modelo está acertando poucas classificações como 1. Então a AUC assim como a acurácia está alta porque a base de dados utilizada é altamente desbalanceada tendo muito mais amostras de vivos(1), e como o modelo está bom para essa classificação e está acertando bastante os valores de AUC e acurácia ficam altos .

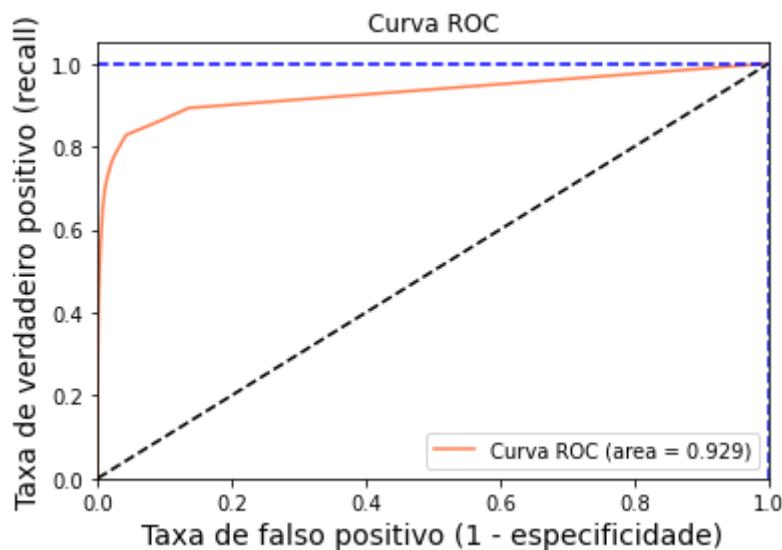


Figura 27. Curva ROC modelo de Árvore de decisão. (Fonte: Elaboração Própria).

A Figura 28 mostra a importância de cada feature para o modelo sendo assim a feature 10 - Peso ao nascer, a mais importante para o modelo, pois dela o modelo conseguiu tirar mais informações, seguido das *features*, 13 - Apgar dos 5 primeiros minutos, 11 - Semanas de Gestação, 14 - Presença de malformação 12 - Apgar do 1 primeiro minutos. Então essas são as *features* que foram mais decisivas para a montagem da árvore e para as previsões do modelo.



Figura 28. Gráfico da importância das features. (Fonte: Elaboração Própria).

Na Figura 29 temos uma imagem reduzida da árvore de decisão que foi montada com o treinamento utilizando 5 como profundidade máxima para a árvore, a imagem da árvore completa pode ser visualizada no apêndice I. É possível ver que as *features* marcadas como importante para o modelo apareceu diversas vezes na árvore, e a *feature* peso ao nascer foi escolhida para ser o nó raiz da árvore, de todas as features ela foi a que teve a melhor entropia para ser o nó raiz, e depois aparece mais vezes para outras decisões em outros níveis da árvore e isso faz com que essa *feature* se torne a mais importante para o modelo, também podemos ver que a *feature* ‘Apgar do 1 primeiro minuto’ mesmo sendo a menos importante para o modelo ela aparece somente no nó de número 42 isso mostra que a entropia e o ganho de informação dessa *feature* nas outras decisões não foram boas fazendo ela ser a feature com menos importância utilizada no modelo.

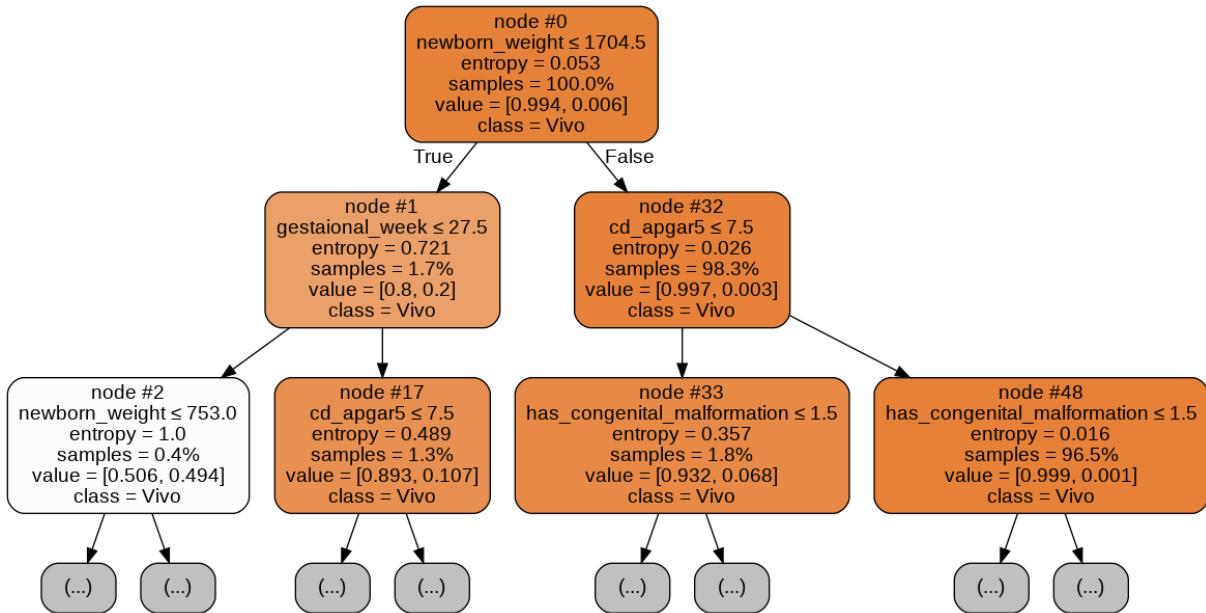


Figura 29. Árvore de decisão montada a partir do algoritmo. (Fonte: Elaboração Própria).

6.2.2 Modelo de Árvore de Decisão utilizando K-folds cross-validation

Após realizar esse experimento com a partição 90/10, foi realizado outro experimento com esse mesmo algoritmo porém agora utilizando o método *K-folds cross-validation* para o treinamento. O *K-folds cross-validation* foi configurado para separar a base de dados em 10 partes iguais, e assim o algoritmo foi treinado novamente gerando um novo modelo. O método de *K-folds cross-validation* é utilizado para conferir se o modelo não está sofrendo problemas de *overfitting*, é importante garantir que o modelo está aprendendo com os dados e não está somente decorando.

Na Tabela 5 temos então o resultados desse modelo, pode-se observar que os resultados para a classificação de vivos(0) não teve alteração, o modelo continua tendo 100% na precisão e no *recall*, mas na classificação de mortos(1) o modelo teve uma diminuição na precisão que agora está em 65% e também teve uma diminuição no *recall* que agora está em 23%, porém essa diminuição é justificada porque para o teste deste modelo foi utilizado a base completa e não somente a partição de teste que contém 10% da base total. Logo os resultados não tiveram uma alteração drástica e tiveram um comportamento bem semelhante, incluindo a acurácia que se manteve em 0,99 e a AUC que teve uma diminuição pequena também e ficou com 0,89.

Tabela 5. Resultados do modelo de árvore de decisão utilizando *K-folds cross-validation*.
(Fonte: Elaboração Própria)

	Precision	Recall	F1-Score	Support
Vivo(0)	1,00	1.00	1,00	6.719.191
Morto(1)	0,65	0,23	0,34	41.031

Na Tabela 6 é possível ver a matriz de confusão do modelo que mostra o número exato de erros e acertos do modelo, pode-se observar que para a classe de vivos o modelo classificou corretamente 6.714.117 amostras da base de dados completa, o modelo classificou como 0 as amostras que nos rótulos realmente eram 0, errou apenas 5.074 amostras. Já para a classe de mortos o modelo acertou somente 9.552 amostras, classificou como 1 as amostras que no rótulo eram 1 também, e errou 31.479. Isso mostra que o modelo está muito desbalanceado, pois está errado muito mais do que acertando as previsões de mortos, sendo isso um reflexo da base de dados desbalanceada também. E na tabela também mostra os valores de *F1-Score* para cada classe, esse resultado é formado pela soma da *Precision* e do *Recall*.

Tabela 6. Matriz de confusão do modelo de árvore de decisão utilizando *K-folds cross-validation*. (Fonte: Elaboração Própria)

		Preditivo	
		Vivos (0)	Mortos (1)
Classe Real	Vivos (0)	6.714.117	5.074
	Mortos (1)	31.479	9.552

6.3 REGRESSÃO LOGÍSTICA

Para o algoritmo de Regressão Logística também foi utilizado métodos diferentes para o treinamento, O algoritmo foi treinado utilizando três formas diferentes: usando as partições 90% para treino e 10% para os testes, utilizando o método *K-folds cross-validation* e foi treinada utilizando o método RFECV que seleciona as melhores *features* para o modelo, juntamente com o método GridSearchCV. O código completo deste experimento pode ser

visualizado no apêndice J, ou também no link do github exibido na “5.6 ACESSO A BASE DE DADOS E CÓDIGOS”.

6.3.1 Modelo de Regressão Logística utilizando particionamento 90/10

Para o primeiro experimento do algoritmo de regressão logística foi utilizado o particionamento 90/10 para o treinamento do algoritmo, sendo 90% da base de dados para realizar o treinamento do modelo e 10% da base para realizar os testes.

Na Tabela 7 temos os resultados do modelo de regressão logística utilizando o método de particionamento 90/10, analisando a *precision* vemos que para a classe 0 a precisão é de 1,00, ou seja, 100% então todos as amostras que o modelo classificou como 0 eram realmente 0 na classe real, já para a classe 1 o modelo classificou 65% que realmente pertenciam a classe 1, então cerca de 35% das classificações que o modelo colocou como 1 estão incorretas. Analisando o *recall* é possível ver que o modelo conseguiu identificar 100% das classificações 0, o modelo conseguiu reconhecer muito bem as amostras classificadas como 0, já para as amostras classificadas como 1 o modelo reconheceu apenas 24% das amostras , o modelo deixou passar muitas amostras da classe 1 e não classificou como 1. Então o modelo treinado não está identificando muito bem as amostras da classe 1 que seria dos recém-nascidos que poderiam vir a óbito, já para a classe de vivos a classe 0 o modelo está identificando muito bem e classificando de forma correta.E na tabela também mostra os valores de *F1-Score* para cada classe, esse resultado é formado pela soma da *Precision* e do *Recall*.

Tabela 7. Resultados do modelo de regressão logística usando particionamento 90/10.

(Fonte: Elaboração Própria)

	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Support</i>
Vivo(0)	1,00	1,00	1,00	671.885
Morto(1)	0,65	0,24	0,35	4.138

A Figura 30 abaixo mostra a curva ROC do modelo de regressão logística usando o particionamento 90/10, analisando a curva ROC podemos ver que a AUC da curva ficou em 0,93 e a acurácia do modelo ficou 0,99, mostrando que o modelo está acertando bastante as

predições, porém esses números para o nosso modelo não mostra que ele está ótimo, com a análise da Tabela 7 acima podemos observar que o modelo está acertando muitas predições da classe de vivos e poucas predições da classe de mortos, como a base de dados é desbalanceada, como a maior quantidade de dados está na classe de vivos e o modelo está certo quase todos dessa classe a acurácia e a AUC ficam altas por esses acertos. Logo é preciso analisar mais resultados além da acurácia e da AUC do modelo.

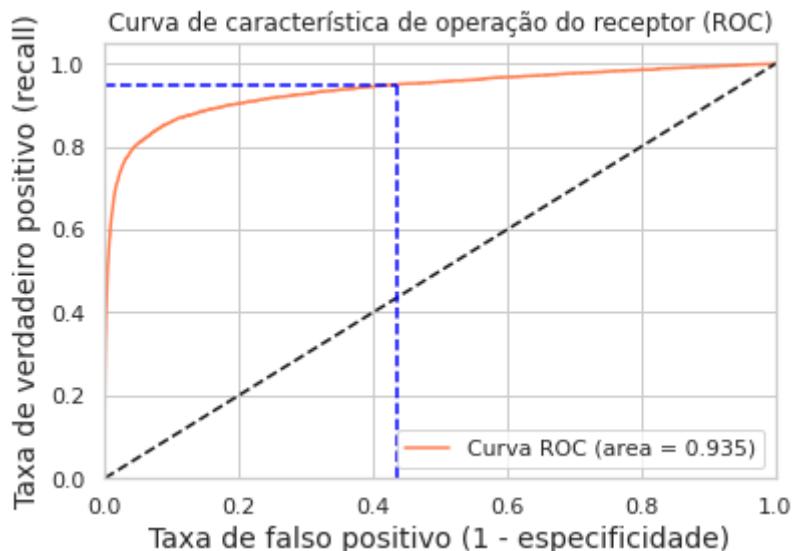


Figura 30. Curva ROC modelo regressão logística usando particionamento 90/10. (Fonte: Elaboração Própria).

Na Tabela 8 podemos analisar os número exatos que o modelo acertou de cada classe, como é mostrado na matriz de confusão o modelo acertou 671.435 da classe de vivos e acertou apenas 915 da classe de mortos, e errou mais que o triplo da classe de mortos. Então as predições do modelo estão muito desbalanceadas, para a classe de vivos o modelo está predizendo bem, porém para as classes de mortos o modelo está errando muitas predições.

Tabela 8. Matriz de confusão do modelo de regressão logística usando particionamento 90/10. (Fonte: Elaboração Própria)

		Predito	
		Vivos (0)	Mortos (1)
Classe Real	Vivos (0)	671.435	504
	Mortos (1)	3.169	915

6.3.2 Modelo de Regressão Logística Utilizando K-folds cross-validation

Para o segundo experimento do algoritmo de regressão logística foi utilizado o método *K-folds cross-validation* para o treinamento do modelo com a intenção de conferir e confirmar que o modelo não esteja tendo problemas com overfitting e realmente esteja aprendendo com os dados que está sendo usado para o treinamento. O método *K-folds cross-validation* foi configurado para realizar uma separação de 10 partes iguais.

Na Tabela 9 podemos observar os resultados do modelo, e os resultados foram parecidos com o experimento anterior usando o particionamento 90/10, única diferença nos resultado é que no experimento anterior a precisão da classe de mortos ficou em 65% e no caso desse experimento usando o *K-folds cross-validation* a precisão ficou em 64%, mas os valores de recall e F1-Score se mantiveram, assim como todos os resultados da classe de mortos se mantiveram em 100%. A execução desse experimento mostra que o modelo realmente está aprendendo com os dados e não está apenas decorando, retirando o risco do modelo estar com *overfitting*. Mesmo sem alterações no resultado é importante saber que o modelo não está com *overfitting* e está conseguindo aprender e encontrar padrões nos dados.

Tabela 9. Resultados do modelo de regressão logística usando *K-folds cross-validation*.
(Fonte: Elaboração Própria)

	Precision	Recall	F1-Score	Support
Vivo(0)	1,00	1,00	1,00	6.719.191
Morto(1)	0,64	0,24	0,35	41.031

Já na Figura 31 temos a curva ROC do modelo, essa curva ROC mostra a AUC de todas as 10 vezes que o modelo foi treinado e testado usando as 10 partes diferentes do método de *K-folds cross-validation*, e como pode-se observar as AUCs foram bem parecidas para todas as vezes que foi realizado os testes, a AUC se manteve em 0,93 e 0,94, sendo a média de AUC 0,93 a mesma AUC do experimento anterior então esse resultado também não se alterou. A acurácia do modelo também se manteve em 0,99, afinal o modelo continua acertando muito a classe de vivos que é a classe predominante na base de dados. Já era esperado os resultados não sofrerem alterações grandes, pois a base de dados não sofreu nenhuma alteração, esse experimento foi somente para conferir se a modelo não estava

sofrendo overfitting, e pelos resultados aparentemente a base não está com problemas de sobreajuste.

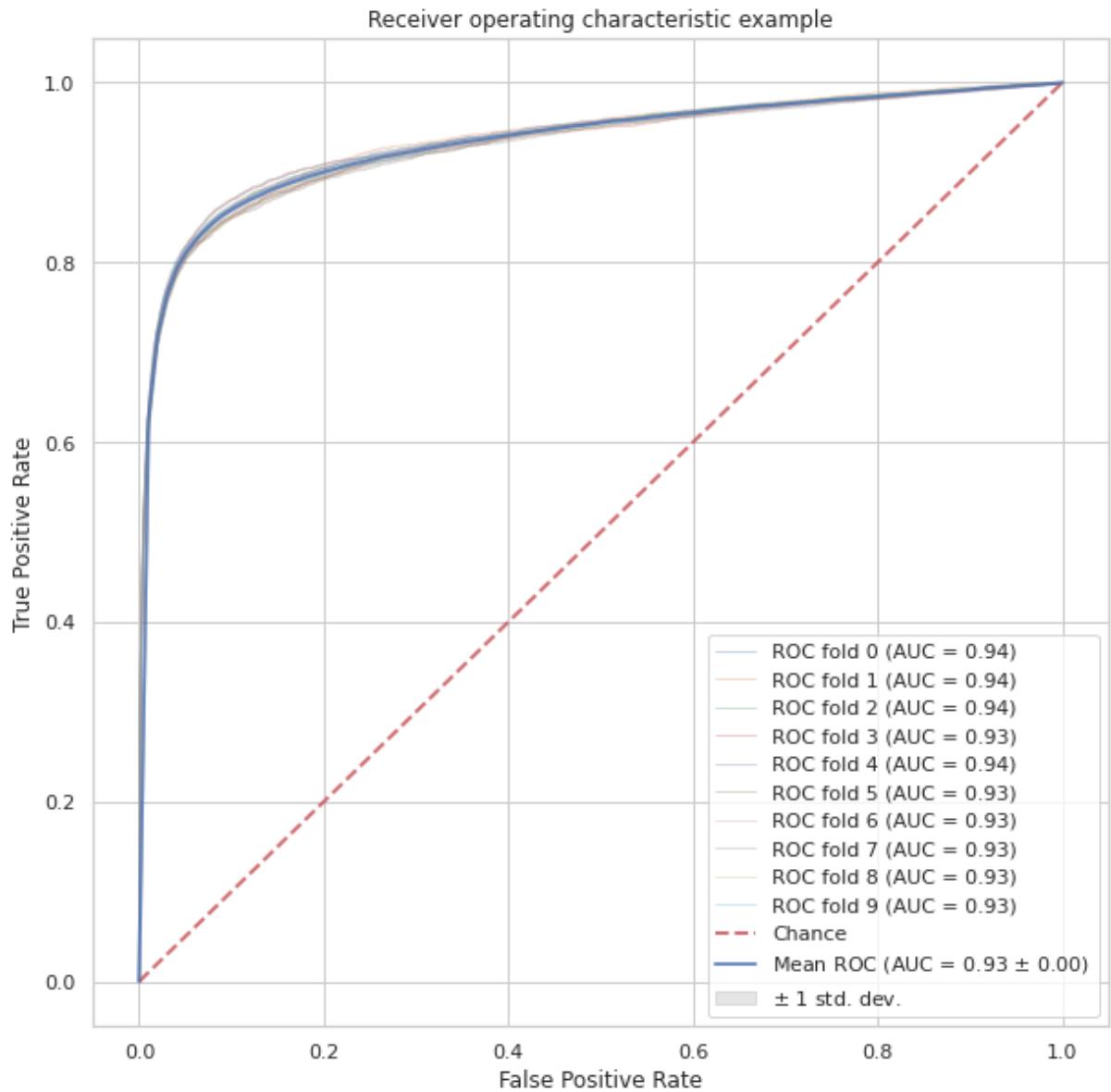


Figura 31. Curva ROC modelo regressão logística usando K-folds cross-validation. (Fonte: Elaboração Própria).

Na Tabela 10 temos a matriz de confusão do modelo, onde mostra que o modelo acertou 6.713.658 amostras da classe de vivos e errou apenas 5.533, já para a classe de mortos o modelo acertou somente 9.847 e novamente errou mais que o triplo errando 31.182 amostras. Como os resultados mostrados anteriormente não tiveram alteração era de se esperar que as predições corretas e incorretas do modelo também não sofressem alterações, o modelo continua acertando muito a classe de vivos e errando muito a classe de mortos, mantendo as predições bem desbalanceadas.

Tabela 10. Matriz de confusão do modelo de regressão logística usando *K-folds cross-validation*. (Fonte: Elaboração Própria)

		Predito	
		Vivos (0)	Mortos (1)
Classe Real	Vivos (0)	6.713.658	5.533
	Mortos (1)	31.182	9.847

6.3.3 Modelo de Regressão Logística utilizando *GridSearchCV* e *RFECV*

Para o terceiro e último experimento de regressão logística foi utilizado técnicas diferentes juntas para tentar melhorar as previsões do modelo, para esse experimento usamos o *K-folds cross-validation* para o particionamento da base de dados assim como foi utilizado no experimento acima, e também foi utilizado duas técnicas que ainda não foram usadas nos experimentos anteriores que são as funções *RFECV* e *GridSearchCV*. A função *RFECV* seleciona a quantidade ideal de *features* para ser usadas no treinamento do modelo e também mostra quais são as melhores *features* para essa previsão, então essa função ela utiliza a validação cruzada para ir treinando e testando N vezes o modelo, e sempre vai alterando a quantidade e as *features* utilizadas para o teste, então cada vez que a função testa os modelos ela grava a pontuação dos acertos e assim depois mostra o gráfico da Figura 32 e assim é possível ver quais são as *features* ideais para o modelo.

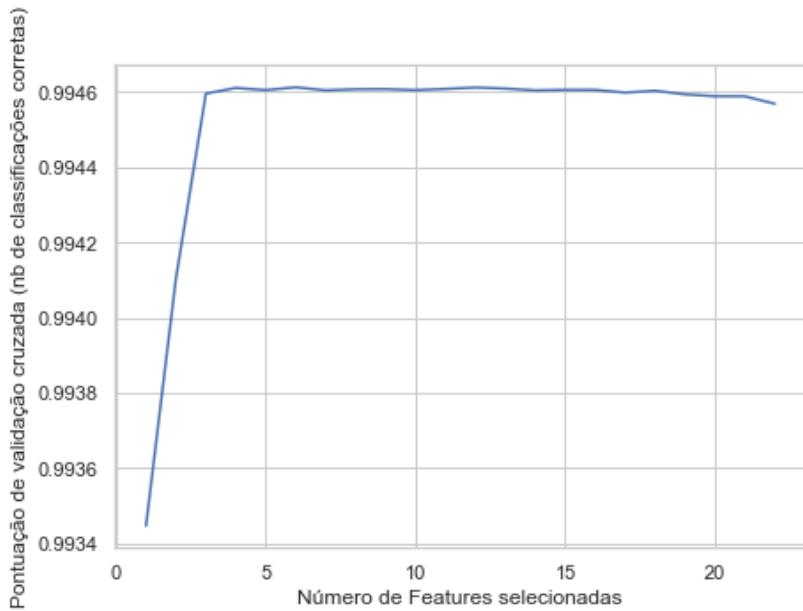


Figura 32. Resultado da função RFECV Base Brasil. (Fonte: Elaboração Própria).

Então os resultados mostrados na Figura 32 apresenta que o número ideal para treinar o modelo de *features* é 6, e as *features* sao: '*tp_maternal_schooling*', '*gestaional_week*', '*cd_apgar1*', '*cd_apgar5*', '*has_congenital_malformation*', '*tp_labor*'. Essas são as *features* que tiveram o melhor desempenho no RFECV, então para o treinamento do modelo desse experimento as *features* que vão ser utilizadas serão somente essas 6.

Após a execução da função de RFECV foi executado a função de *GridSearchCV*, o *GridSearchCV* é utilizado para descobrir quais são os melhores parâmetros para utilizar no algoritmo de regressão logística e criar os modelos, foi utilizado também a validação cruzada para testar qual seria os melhores parâmetros para melhorar o desempenho do modelo. Na Figura 33 pode-se ver que os parâmetros escolhidos pela função foram:

```
=====
Best params: LogisticRegression(C=1e-05, class_weight=None, dual=False, fit_intercept=True,
                                intercept_scaling=1, l1_ratio=None, max_iter=100,
                                multi_class='auto', n_jobs=None, penalty='l2',
                                random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                                warm_start=False)
Best params: {'C': 1e-05}
Best score: 0.9946170406794806
=====
```

Figura 33. Resultado da função GridSearchCV. (Fonte: Elaboração Própria).

Na Tabela 11 podemos observar os resultados do modelo, e os resultados foram parecidos com os experimentos anteriores, para a classe de mortos a precisão teve um aumento e foi para 69% e o *recall* diminuiu chegando em 20%, então utilizando as novas funções o modelo ganhou precisão e perdeu *recall*. Para a classe de vivos o modelo se manteve em 100% e não teve alterações para os resultados de precisão e *recall*. Mesmo com a utilização das funções de RFECV e *GridSearchCV* o modelo não teve uma melhora significativa para as previsões de mortos.

Tabela 11. Resultados do modelo de regressão logística usando *GridSearchCV* e *RFECV*.
(Fonte: Elaboração Própria)

	Precision	Recall	F1-Score	Support
Vivo(0)	1,00	1,00	1,00	6.719.191
Morto(1)	0,69	0,20	0,31	41.031

Na Figura 34 temos a curva ROC do modelo, essa curva ROC mostra a AUC de todas as 10 vezes que o modelo foi treinado e testado usando as 10 partes diferentes do método de *K-folds cross-validation*, e como pode-se observar as AUCs foram bem parecidas para todas as vezes que foi realizado os testes, a AUC se manteve em 0,92 e 0,93, sendo a média de AUC 0,93 a mesma AUC do experimento anterior então esse resultado também não se alterou. A acurácia do modelo também se manteve em 0,99, afinal o modelo continua acertando muito a classe de vivos que é a classe predominante na base de dados.

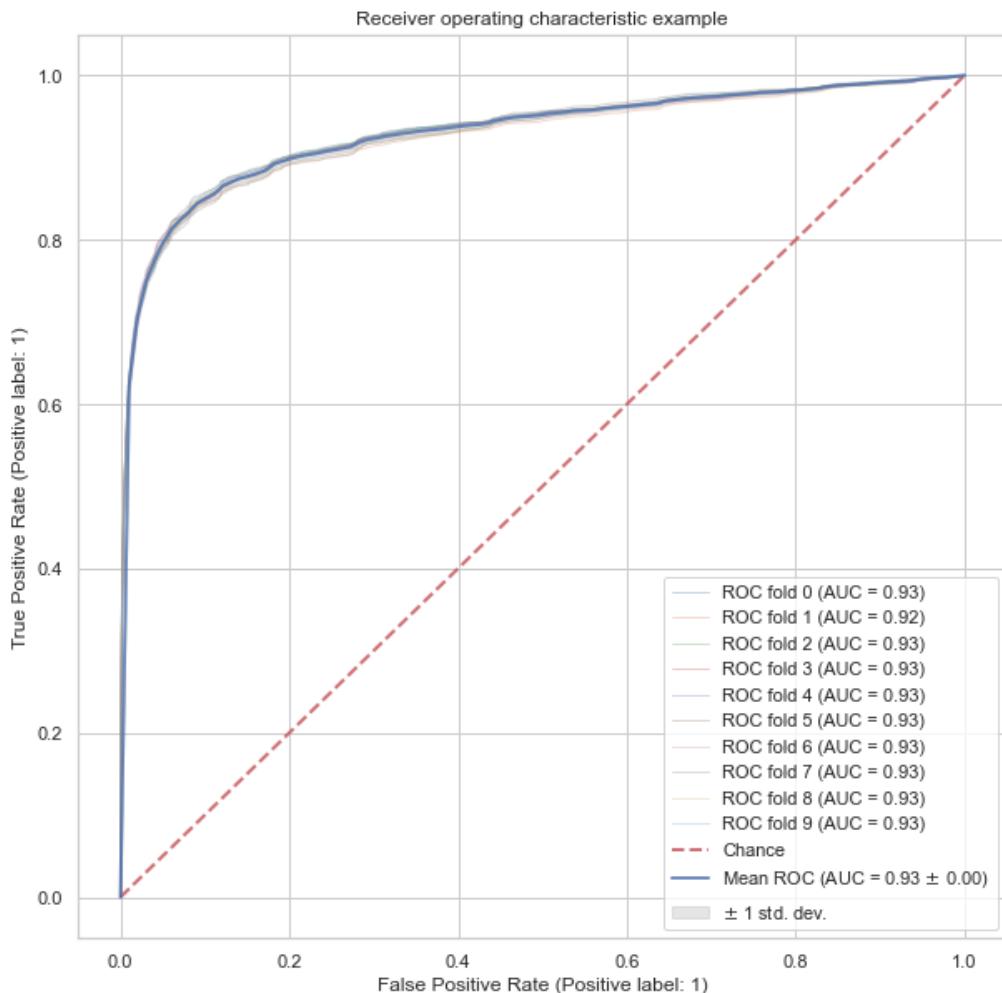


Figura 34. Curva ROC modelo regressão logística usando GridSearchCV e RFECV. (Fonte: Elaboração Própria).

Na Tabela 12 temos a matriz de confusão do modelo, onde mostra que o modelo acertou 6.715.535 amostras da classe de vivos e errou apenas 3.656, comparando com os experimentos anteriores de regressão logística as predições de vivos teve uma pequena piora e teve uma pequena diminuição nos acertos. Já para a classe de mortos, o modelo acertou 8.296 e errou 32.735 amostras, logo as predições de vivos tiveram uma piora e acertou um pouco menos do que os outros experimentos, e na classe de vivos teve uma melhora acertando mais predições, porém como o problema está nas predições de mortos e para essa classe teve uma piora pode-se dizer que a função de *GridSearchCV* não teve um ganho significativo nos resultados. Então como modelo final foi escolhido o modelo utilizando somente o *K-folds cross-validation*, pois foi o modelo que mostrou o melhor desempenho dos experimentos de regressão logística.

Tabela 12. Matriz de confusão do modelo de regressão logística usando GridSearchCV e RFECV. (Fonte: Elaboração Própria)

		Preditivo	
		Vivos (0)	Mortos (1)
Classe Real	Vivos (0)	6.715.535	3.656
	Mortos (1)	32.735	8.296

7 CONCLUSÃO

O principal objetivo deste trabalho foi analisar os resultados das previsões de mortalidade neonatal dos algoritmos de aprendizado de máquina usando uma base de dados com dados do Brasil inteiro. A partir destes resultados apresentados na seção anterior “Resultados”, é possível concluir que analisar somente a AUC e a acurácia do modelo não é o suficiente para garantir que o modelo está excelente, sendo assim temos que ter mais atenção a precisão, *recall* e as previsões mostradas na matriz de confusão.

Ambos os modelos ficaram desbalanceados nas previsões, os modelos acertaram mais previsões de vivos do que mortos, e as previsões corretas de mortos foram mais baixas do que as incorretas, para os modelos ficarem melhor precisam passar por algoritmos que possam balancear essas previsões, uma outra alternativa para melhorar o modelo é utilizar uma base de dados mais balanceada, pois essa base que foi utilizada é altamente desbalanceada.

Embora os dois modelos tenham tido resultados parecidos , o modelo de regressão logística utilizando somente o *K-folds cross-validation* se saiu melhor que os outros experimentos, o modelo criado na seção “6.3.2 Modelo de Regressão Logística Utilizando *K-folds cross-validation*” manteve a acurácia, AUC, precisão e recall dos demais modelos e acertou mais previsões, talvez com algoritmos para melhorar as previsões da classe de mortos, balanceando mais as previsões, esse modelo fique com ótimas taxas preditivas.

Além disso pode-se afirmar que o peso ao nascer do recém nascido é um fator muito importante para as decisões dos modelos, ambos os modelos usaram muito essa informação para as previsões, também podemos colocar, as colunas de presença de malformação e semana de gestação, como colunas que foram importantes para os modelos, essas causas podem ser evitadas se houver um acompanhamento médico com qualidade e eficiência.

REFERÊNCIAS

- ÁRVORE de Decisão - Aula 3. Gravação de Diogo Cortiz. [S. l.]: YouTube, 2020. Disponível em: <https://www.youtube.com/watch?v=ecYpXd4WREk&t=4089s>. Acesso em: 1 jul. 2020.
- BARRETO, J. O. M.; SOUZA, N. M.; CHAPMAN, E. **Síntese de evidências para políticas de saúde: reduzindo a mortalidade perinatal**. Ministério da Saúde, p. 1–44, 2015.
- BATISTA, André Filipe de Moraes; FILHO, Alexandre Dias Porto Chiavegatto. Machine Learning aplicado à Saúde. In: ZIVIANI, Artur; FERNANDES, Natalia Castro; SAADE, Débora Christina Muchaluat. **SBCAS 2019: 19 Simpósio Brasileiro de Computação Aplicada à Saúde**. [S. l.: s. n.], 2019. cap. Capítulo 1.
- BELUZO, Carlos Eduardo; SILVA, Everton; ALVES, Luciana Correia; BRESAN, Rodrigo Campos; ARRUDA, Natália Martins; SOVAT, Ricardo; CARVALHO, Tiago. **Towards neonatal mortality risk classification: A data-driven approach using neonatal, maternal, and social factors**, [s. l.], 23 out. 2020.
- BELUZO, Carlos Eduardo; SILVA, Everton; ALVES, Luciana Correia; BRESAN, Rodrigo Campos; ARRUDA, Natália Martins; SOVAT, Ricardo; CARVALHO, Tiago. **SPNeoDeath: A demographic and epidemiological dataset having infant, mother, prenatal care and childbirth data related to births and neonatal deaths in São Paulo city Brazil – 2012–2018**, [s. l.], 19 jul. 2020.
- BRESAN, Rodrigo. **Um método para a detecção de ataques de apresentação em sistemas de reconhecimento facial através de propriedades intrínsecas e Deep Learning**. Orientador: Me. Carlos Eduardo Beluzo. 2018. Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia Câmpus Campinas., [S. l.], 2018.
- CABRAL, Cleidy Isolete Silva. **Aplicação do Modelo de Regressão Logística num Estudo de Mercado**. Orientador: João José Ferreira Gomes. 2013. Projeto Mestrado em Matemática Aplicada à Economia e à Gestão (Mestrado) - Universidade de Lisboa Faculdade de Ciências Departamento de Estatística e Investigação Operacional, [S. l.], 2013. Disponível em: https://repositorio.ul.pt/bitstream/10451/10671/1/ulfc106455_tm_Cleidy_Cabral.pdf. Acesso em: 1 maio 2021.
- CAMPOS, Raphael. **Árvores de Decisão**: Então diga-me, como construo uma?. [S. l.], 28 nov. 2017. Disponível em: <https://medium.com/machine-learning-beyond-deep-learning/%C3%A1rvore-de-decis%C3%A3o-3f52f6420b69>. Acesso em: 23 jan. 2021.
- Colab**. 2021. Disponível em: <https://colab.research.google.com/>. Acesso em: 20 mar. 2021.
- COX, D.R.; SNELL, E.J. **Analysis of Binary Data**. London: Chapman & Hall, 2^a Edição, 1989. HOSMER, D.W.; LEMESHOW, S. Applied Logistic Regression. New York: John Wiley, 2^a Edição, 2000.
- E.França, S.Lansky. **Mortalidade infantil neonatal no brasil: situação, tendências e perspectivas** Rede Interagencial de Informações para Saúde - demografia e saúde:

contribuição para análise de situação e tendências Série Informe de Situação e Tendências(2009), pp.83-112.

FREIRE, Sergio Miranda. **Bioestatística Básica:** Regressão Linear. [S. l.], 20 out. 2020. Disponível em: http://www.lampada.uerj.br/arquivosdb/_book/bioestatisticaBasica.html. Acesso em: 23 jan. 2021.

Jupyter. 2021. Disponível em: <https://jupyter.org/>. Acesso em: 20 jun. 2021.

GOODFELLOW, I. et al. Deep learning. [S.l.]: MIT press Cambridge, 2016.

Pandas. 2021. Disponível em: <https://pandas.pydata.org/>. Acesso em: 20 jun. 2021.

Python. 2021. Disponível em: <https://www.python.org/>. Acesso em: 20 jun. 2021.

KUHN, M.; JOHNSON, K. Applied predictive modeling. [S.l.]: Springer, 2013. v. 26.

LANSKY, S. et al. **Pesquisa Nascer no Brasil: perfil da mortalidade neonatal e avaliação da assistência à gestante .** Cadernos de Saúde Pública, Scielo, v. 30, p. S192 – S207, 00 2014.

LANSKY, Sônia; FRICHE, Amélia Augusta de Lima; SILVA, Antônio Augusto Moura; CAMPOS, Deise; BITTENCOURT, Sonia Duarte de Azevedo; CARVALHO, Márcia Lazaro; FRIAS, Paulo Germano; CAVALCANTE, Rejane Silva; CUNHA, Antonio José Ledo Alves. **Pesquisa Nascer no Brasil: perfil da mortalidade neonatal e avaliação da assistência à gestante e ao recém-nascido.** [s. l.], Ago 2014. Disponivel em: <https://www.scielosp.org/article/csp/2014.v30suppl1/S192-S207/pt/>

Matplotlib. 2021. Disponível em: <https://matplotlib.org/>. Acesso em: 20 mar. 2021.

Maranhão AGK, Vasconcelos AMN, Trindade CM, Victora CG, Rabello Neto DL, Porto D,et al. **Mortalidade infantil no Brasil: tendências, componentes e causas de morte no período de 2000 a 2010 .** In: Departamento de Análise de Situação de Saúde, Secretaria de Vigilância em Saúde, Ministério da Saúde, organizador. Saúde Brasil 2011: uma análise da situação de saúde e a vigilância da saúde da mulher. v. 1. Brasília: Ministério da Saúde; 2012. p. 163-82.

MESQUITA, PAULO. UM MODELO DE REGRESSÃO LOGÍSTICA PARA AVALIAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO NO BRASIL. Orientador: Prof. RODRIGO TAVARES NOGUEIRA. 2014. Dissertação (Mestre em Engenharia de Produção) - Universidade Estadual do Norte Fluminense, [S. l.], 2014.

MNASSRI , Baligh. **Titanic: logistic regression with python.** [S. l.], 1 ago. 2020. Disponível em: <https://www.kaggle.com/mnassrib/titanic-logistic-regression-with-python>. Acesso em: 14 jan. 2021.

Morais, R.M.d., Costa, A.L: **Uma avaliação do sistema de informações sobre mortalidade.** Saúde em Debate 41, 101–117 (2017). Disponível em: <https://doi.org/10.1109/SIBGRAPI.2012.38>.

MOTA, Caio Augusto de Souza; BELUZO , Carlos Eduardo; TRABUCO, Lavinia Pedrosa; SOUZA , Adriano; ALVES, Luciana; CARVALHO, Tiago. **DESENVOLVIMENTO DE UMA PLATAFORMA WEB PARA CIÊNCIA DE DADOS APLICADA À SAÚDE MATERNO INFANTIL ,** [s. l.], 28 nov. 2019.

MULTIEDRO (Brasil). **Conheça as aplicações do machine learning na área da saúde.** [S. l.], 28 nov. 2019. Disponível em: <https://blog.multiedro.com.br/machine-learning-na-area-da-saude/#:~:text=O%20machine%20learning%20na%20%C3%A1rea,diagn%C3%B3sticos%20e%20tratamentos%20mais%20precisos.> Acesso em: 13 jun. 2021.

MUKHIYA,S.K.;AHMED,U. ***Hands-On Exploratory Data Analysis with Python: Perform EDA Techniques to Understand, Summarize, and Investigate Your Data.*** [S.l.]: PacktPublishingLtd,2020.ISBN978-1-78953-725-3.22,32

Murray CJ, Laakso T, Shibuya K, Hill K, Lopez AD. ***Can we achieve Millennium Development Goal 4? New analysis of country trends and forecasts of under-5 mortality to 2015.*** Lancet 2007; 370:1040-54.

NumPy. 2021. Disponível em:<https://numpy.org/>. Acesso em: 20 jun. 2021.

Oliveira, M.M.d., de Araújo, A.S.S.C., Santiago, D.G., Oliveira, J.a.C.G.d., Carvalho, M.D., de Lyra et al, R.N.D.: ***Evaluation of the national information system on live births in brazil , 2006-2010.*** Epidemiol. Serv. Saúde 24(4), 629–640 (2015).

PELISSARI, ANA CAROLINA CHEBEL. **MORTALIDADE NEONATAL DA CIDADE DE SÃO PAULO: UMA ABORDAGEM UTILIZANDO APRENDIZADO DE MÁQUINA SUPERVISIONADO.** 2021. Trabalho de Conclusão de Curso (Superior) - Instituto Federal de Educação, Ciência e Tecnologia Campus Campinas, [S. l.], 2021.

REGRESSÃO logística e binary cross entropy - Aula 7. Direção: Diogo Cortiz. [S. l.]: YouTube, 2020. Disponível em: <https://www.youtube.com/watch?v=3J-LBtHVsm4>. Acesso em: 21 jan. 2021.

Rmd Nascimento, AJM Leite, NMGSd Almeida, Pcd Almeida, Cfd Silva **Determinantes da mortalidade neonatal: estudo caso-controle em fortaleza, ceará, brasil Cad Saúde Pública,** 28(2012), pp.559 - 572.

SANTOS, Hellen Geremias. Machine learning e análise preditiva: considerações metodológicas: métodos lineares para classificação. In: SANTOS, Hellen Geremias. **Comparação da performance de algoritmos de machine learning para a análise preditiva em saúde pública e medicina.** 2018. Tese (Pós-Graduação em Epidemiologia) - Faculdade de Saúde Pública da Universidade de São Paulo, [S. l.], 2018.

Scikit-learn. 2021. Disponível em: <https://scikit-learn.org/stable/>. Acesso em: 20 jun. 2021.

Seaborn. 2021. Disponível em: <https://seaborn.pydata.org/>. Acesso em: 20 jun. 2021.

SILVA, Wesley; CORSO, Jansen; WELGACZ, Hanna; PEIXE, Julinês. **AVALIAÇÃO DA ESCOLHA DE UM FORNECEDOR SOB CONDIÇÃO DE RISCOS A PARTIR DO MÉTODO DE ÁRVORE DE DECISÃO.** ARTIGO – MÉTODOS QUANTITATIVOS, [s. l.], 12 ago. 2008.

WHO, W. H. O. ***Women and health: today's evidence, tomorrow's agenda.*** [S.I.]: UNWorld Health Organization, 2009. ISBN 9789241563857.

APÊNDICE A

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_arvoredecisao_spneodeath_2012_2018_final.py

```

1 # -*- coding: utf-8 -*-
2 """
3     TCC_CaioMota_ArvoreDecisao_SPNeoDeath_2012_2018_FINAL
4     .ipynb
5
6     Automatically generated by Colaboratory.
7
8     Original file is located at
9         https://colab.research.google.com/drive/
10        11WavsUi1N0M-HTOFKWiYbJUU92NadjRa
11
12    **Implementação de modelo utilizando algoritmo de
13    Árvore de Decisão**
14    <br>
15    <br>
16    **Autor**: Caio Augusto de Souza Mota (*caiomota802@gmail.com*)
17
18    <br>
19    <br>
20    *Codigo adaptado de Diogo Cortiz disponivel em: https
21        ://github.com/diogocortiz/Curso-IA-para-todos/tree/
22        master/ArvoreDecis%C3%A3o*
23
24    ---
```

Este código é parte do Trabalho de Conclusão de Curso apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo Câmpus Campinas.

1. Importação de Bibliotecas, carga de dados e funções

"""
Instalando o Synapse Client

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_arvoredecisao_spneodeath_2012_2018_final.py

```
28 ! pip install synapseclient
29
30 # Util
31 import itertools
32 import synapseclient as syna
33 from getpass import getpass
34
35 # Data
36 import numpy as np
37 import pandas as pd
38
39 from sklearn.tree import DecisionTreeClassifier,
    export_graphviz
40 from sklearn import preprocessing
41 from sklearn.model_selection import train_test_split
    , cross_val_score, cross_validate, cross_val_predict
42 from sklearn.externals.six import StringIO
43 from sklearn.metrics import accuracy_score,
    classification_report, precision_score, recall_score
    , confusion_matrix, precision_recall_curve, roc_curve
    , auc, log_loss
44
45 # Images
46 import matplotlib.pyplot as plt
47 import pydotplus
48 import matplotlib.image as mpimg
49 from IPython.display import Image
50
51 """## 1.1 Carregando base de dados disponível no
Synapse"""
52
53 # SPNeodeath
54 # Recuperando a base de dados do repositório de dados
    Synapse
55 syn = syna.Synapse()
56 syn.login(input('Synapse User: '), getpass('Passwd: '))
57
58 # Obtendo um ponteiro e baixando os dados
59 dataset = syn.get(entity='syn22240290') # ID do
dataset SPNeodeath
```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_arvoredecisao_spneodeath_2012_2018_final.py

```
60
61 df_ori = pd.read_csv(dataset.path)
62 df_ori.shape
63
64 # Amostra para testes rápidos
65 df = df_ori.sample(frac=1)
66
67 # REMOVENDO OS REGISTROS NOS QUAIS PELO MENOS UM
    CAMPO ESTÁ EM BRANCO (NAN)
68 df = df.dropna()
69
70 # Exibindo as 5 primeiras linhas
71 print(df.head(5))
72
73 count_row = df.shape[0] # Número de linhas
74 count_col = df.shape[1] # Número de colunas
75
76 print(count_row)
77 print(count_col)
78
79 # Distribuição entre registros de vivos (negativo =
    0) e mortos (positivo = 1)
80 print ('Total de registros negativos: ', df[df['
    neonatal_death'] == 0 ].shape[0])
81 print ('Total de registros positivos: ', df[df['
    neonatal_death'] == 1 ].shape[0])
82
83 """# 2. Aplicação de modelos de Machine Learning:
    Decision Tree
84
85 ---
86
87 Precisamos converter o Dataframe para um Array Numpy
    , que é o tipo de dados que iremos usar no
    treinamento.
88
89 Um com as features de entrada, e outro com os labels
    (etiquetas, rótulos do registro).
90 """
91
92 # "Features" do modelo
```

```
File - C:\Users\Caio Mota\Desktop\tcc_caiomota_arvoredecisao_spneodeath_2012_2018_final.py
93 nome_features = ['tp_birth_place', 'maternal_age', '
94     tp_marital_status',
95         'tp_maternal_education_years', '
96     num_live_births', 'num_fetal_losses',
97         'tp_pregnancy_duration', '
98     tp_pregnancy', 'tp_labor',
99         'tp_prenatal_appointments', '
100    cd_apgar1', 'cd_apgar5',
101        'newborn_weight', '
102    has_congenital_malformation', '
103    tp_maternal_skin_color',
104        'num_gestations', 'num_normal_labors'
105 , 'num Cesarean_labors',
106         'num_gestational_weeks', '
107     tp_presentation_newborn', 'tp_childbirth_assistance'
108 ,
109         'tp_fill_form_responsible', '
110     cd_robson_group']
111
112 target = ['neonatal_death']
113
114 # Array de features
115 X = df[nome_features].values
116
117 Y = df[target].values
118
119 """### Criando modelo usando divisão da base em 90
120 %/10% para treino e teste."""
121
122 # "Split" do dataset para Treino e Teste do modelo
123 # Usando treino com 90% dos dados e 10% dos dados
124 # para teste
125 xTreino, xTeste, yTreino, yTeste = train_test_split(
126     X, Y,
127     test_size=0.1,
128     shuffle=False,
129     random_state=7)
130
131
```

```
File - C:\Users\Caio Mota\Desktop\tcc_caiomota_arvoredecisao_spneodeath_2012_2018_final.py
118 # Instanciando algortimo
119 algortimo_arvore = DecisionTreeClassifier(criterion=
    'entropy', max_depth=5)
120
121 # "Treinamento"/Construção do modelo
122 modelo_90_10 = algortimo_arvore.fit(xTreino, yTreino
    )
123
124 """### Exibindo a árvore que o modelo montou
125
126 A árvore de decisão pode ser considerada um modelo
    White Box, ou seja, um modelo que podemos entender
    melhor o que ele aprendeu e como ele decide. Podemos
    mostrar a árvore para isso.
127 """
128
129 importances = modelo_90_10.feature_importances_
130 indices = np.argsort(importances)[::-1]
131 print("Importância de cada variável para o modelo:")
132
133 for f in range(X.shape[1]):
134     print("%d. feature %d (%f)" % (f + 1, indices[f],
        ), importances[indices[f]]))
135 f, ax = plt.subplots(figsize=(10, 5))
136 plt.title("Importância de cada variável para o
    modelo", fontsize = 14)
137 plt.bar(range(X.shape[1]), importances[indices],
138         color="b",
139         align="center")
140 plt.xticks(range(X.shape[1]), indices)
141 plt.xlim([-1, X.shape[1]])
142 plt.ylabel("Importância", fontsize = 12)
143 plt.xlabel("Features (índice)", fontsize = 12)
144
145 plt.savefig("importance_features_tree.png", dpi=300)
146
147 plt.show()
148
149
150 # 0 - 'maternal_age'
151 # 1 - 'tp_maternal_schooling'
```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_arvoredecisao_spneodeath_2012_2018_final.py

```

152 # 2 - 'tp_marital_status'
153 # 3 - 'tp_maternal_race'
154 # 4 - 'num_live_births'
155 # 5 - 'num_fetal_losses'
156 # 6 - 'num_previous_gestations'
157 # 7 - 'num_normal_labors'
158 # 8 - 'num Cesarean_labor'
159 # 9 - 'tp_pregnancy'
160 # 10 - 'newborn_weight'
161 # 11 - 'gestaional_week'
162 # 12 - 'cd_apgar1',
163 # 13 - 'cd_apgar5'
164 # 14 - 'hasCongenital_malformation'
165 # 15 - 'tp_newborn_presentation'
166 # 16 - 'num_prenatal_appointments'
167 # 17 - 'tp_labor'
168 # 18 - 'wasCesarean_before_labor'
169 # 19 - 'was_labor_induced'
170 # 20 - 'tp_childbirth_care'
171 # 21 - 'tp_robson_group'
172
173 # Montando imagem da árvore de decisão
174 dot_data = StringIO()
175 export_graphviz(modelo_90_10, out_file=dot_data,
    filled=True, feature_names=nome_features,
    class_names=['Vivo', 'Morto'],
    rounded=True, special_characters=True,
    node_ids=True, proportion=True,
    max_depth=5, rotate=True)
176
177 graph = pydotplus.graph_from_dot_data(dot_data.
   .getvalue())
178 Image(graph.create_png())
179 graph.write_png("arvore.png")
180 Image('arvore.png')
181
182
183
184 # Montando imagem da árvore de decisão
185 dot_data = StringIO()
186 export_graphviz(modelo_90_10, out_file=dot_data,
    filled=True, feature_names=nome_features,
    class_names=['Vivo', 'Morto'],
    
```

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_arvoredecisao_spneodeath_2012_2018_final.py
187 rounded=True, special_characters=True,
188                     node_ids=True, proportion=True,
189                     max_depth=2)
190 graph = pydotplus.graph_from_dot_data(dot_data.
191                                         getvalue())
192 Image(graph.create_png())
193 graph.write_png("arvore_short.png")
194 Image('arvore_short.png',)
195 """
196     #### Curva ROC
197 """
198
199 y_pred = modelo_90_10.predict(xTeste)
200 y_pred_proba = modelo_90_10.predict_proba(xTeste)
201 [fpr, tpr, thr] = roc_curve(yTeste, y_pred_proba)
202
203 print('Treino/Teste resultados para %s:' %
204       modelo_90_10.__class__.__name__)
205 print("Accuracy = %2.3f" % accuracy_score(yTeste,
206                                              y_pred))
207 print("Log_loss = %2.3f" % log_loss(yTeste,
208                                         y_pred_proba))
209 print("AUC = %2.3f" % auc(fpr, tpr))
210
211 idx = np.min(np.where(tpr > 0.95))
212
213 #plt.figure(figsize=(13, 13))
214 plt.plot(fpr, tpr, color='coral', label='Curva ROC (' +
215           'area = %0.3f)' % auc(fpr, tpr))
216 plt.plot([0, 1], [0, 1], 'k--')
217 plt.plot([0, fpr[idx]], [tpr[idx], tpr[idx]], 'k--',
218           color='blue')
219 plt.plot([fpr[idx], fpr[idx]], [0, tpr[idx]], 'k--',
220           color='blue')
221 plt.xlim([0.0, 1.0])
222 plt.ylim([0.0, 1.05])
223 plt.xlabel('Taxa de falso positivo (1 - especificidade)', fontsize=14)

```

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_arvoredecisao_spneodeath_2012_2018_final.py
218 plt.ylabel('Taxa de verdadeiro positivo (recall)', fontsize=14)
219 plt.title('Curva ROC')
220 plt.legend(loc="lower right")
221
222 plt.savefig("ROC.png", dpi=300)
223 print("\n*Nota: Usando um limite de %.3f " % thr[idx] +
224     "garante uma sensibilidade de %.3f " % tpr[idx] +
225     "\ne uma especificidade de %.3f" % (1-fpr[idx]) +
226     ", ou seja, uma taxa de falsos positivos de %.2f%%.\n" % (np.array(fpr[idx])*100))
227
228 plt.show()
229
230 """Métricas do modelo usando divisão 90/10
231
232 - PRECISÃO: DAS CLASSIFICAÇÕES QUE O MODELO FEZ PARA
    UMA DETERMINADA CLASSE
233
234 - RECALL: DOS POSSÍVEIS DATAPoints PERTECENTES A UMA
    DETERMINADA CLASSE
235 """
236
237 Y_predicoes = modelo_90_10.predict(xTeste)
238
239 print("ACURÁCIA DA ÁRVORE: ", accuracy_score(yTeste, Y_predicoes))
240 print(classification_report(yTeste, Y_predicoes))
241
242 """ # 3. Criando modelo usando validação cruzada"""
243
244 # k = 10 na divisão da base para treino e teste
245 algortimo_arvore = DecisionTreeClassifier(criterion='entropy', max_depth=3)
246 scoring = {'accuracy': 'accuracy', 'log_loss': 'neg_log_loss', 'auc': 'roc_auc'}
247
248 modelo2 = cross_validate(algortimo_arvore, X, Y, cv=

```

```
File - C:\Users\Caio Mota\Desktop\tcc_caiomota_arvoredecisao_spneodeath_2012_2018_final.py
248 10,
249                      scoring=list(scoring.values
250 ()�,
251                     return_train_score=False)
252 print('Resultado usando K-fold = 10 no cross-
253 validation:\n')
254 for sc in range(len(scoring)):
255     print("%s: %.3f (+/-.3f)" % (list(scoring.keys
256 ()[sc], -modelo2['test_%s' % list(scoring.values
257 ()[sc]]].mean()
258                                     if list(scoring.
259 values())[sc]=='neg_log_loss'
260                                     else modelo2['test_%s
261 ' % list(scoring.values())[sc]].mean(),
262                                     modelo2['test_%s' %
263                                         list(scoring.values())[sc]].std()))
264
265 """
266 Matriz de confusão
267 """
268
269 algortimo_arvore = DecisionTreeClassifier(criterion=
270     'entropy', max_depth=3)
271 modelo2_predicts = cross_val_predict(
272     algortimo_arvore, X, Y, cv=10)
273 cm_modelo2 = confusion_matrix(Y, modelo2_predicts)
274 print(cm_modelo2)
275
276 # generate report
277 print(classification_report(Y, modelo2_predicts))
```

APÊNDICE B

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_RegLogic_SRNeodeath_2012_2018_final.py

```

1 # -*- coding: utf-8 -*-
2 """TCC_Caiomota_RegLogic_SRNeodeath_2012_2018_FINAL.
3 ipynb
4 Automatically generated by Colaboratory.
5
6 Original file is located at
7     https://colab.research.google.com/drive/
8         1J4dedmeN3DC5sL0vvJ0hBmaJu0xch0c3
9 **Implementação de modelo utilizando algoritmo de
10    Regressão Logística**
11 <br>
12 **Autor**: Caio Augusto de Souza Mota (*caiomota802@gmail.com*)
13
14 Data: 09/06/2021
15
16 **Revisor**: Carlos Eduardo Beluzo (*cbeluzo@gmail.com*)
17 <br>
18 <br>
19 *Código adaptado de Baligh Mnassri disponível em:
20     https://www.kaggle.com/mnassrib/titanic-logistic-
21     regression-with-python/notebook*
22 Este código é parte do Trabalho de Conclusão de Curso
23     apresentado como exigência parcial para obtenção do
24     diploma do Curso de Tecnologia em Análise e
25     Desenvolvimento de Sistemas do Instituto Federal de
26     Educação, Ciência e Tecnologia de São Paulo Câmpus
27     Campinas.
28
29 # 1. Importação de Bibliotecas, carga de dados e
30     funções
31 """
32
33 # instalando o Synapse Client
34 ! pip install synapsecient

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_spneodeath_2012_2018_final.py

```
29
30 import os
31 import synapseclient as syna
32 from getpass import getpass
33
34 import numpy as np
35 from math import sqrt
36 import pandas as pd
37
38 from sklearn.linear_model import LogisticRegression
39 from sklearn.model_selection import train_test_split,
, cross_val_score, cross_validate, cross_val_predict
, GridSearchCV, StratifiedKFold
40 from sklearn.feature_selection import RFE, RFECV
41 from sklearn.metrics import roc_curve, plot_roc_curve
, accuracy_score, auc, log_loss, confusion_matrix,
classification_report
42
43 import matplotlib.pyplot as plt
44 plt.rc("font", size=14)
45
46 import seaborn as sns
47 sns.set(style="white") #white background style for
seaborn plots
48 sns.set(style="whitegrid", color_codes=True)
49
50 import warnings
51 warnings.simplefilter(action='ignore')
52
53 """#### Funções auxiliares"""
54
55 # Matrix de Confusão
56 def my_cm(p_cm, p_acc):
57     plt.figure(figsize=(3,3))
58     sns.heatmap(p_cm, annot=True, fmt=".0f", linewidths=.9, square=True, cmap='Blues_r')
59     plt.ylabel('Valores reais')
60     plt.xlabel('Valores preditos')
61     plt.title('Acurácia: %2.3f' % p_acc, size=12)
62
63 """## 1.1 Carregando base de dados disponível no
```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_spneodeath_2012_2018_final.py

```
63 Synapse"""
64
65 # SPNeodeath
66 # Recuperando a base de dados do repositório de
67 # dados Synapse
67 syn = syna.Synapse()
68 syn.login(input('Sybapce User: '), getpass('Passwd: '))
69
70 # Obtendo um ponteiro e baixando os dados
71 dataset = syn.get(entity='syn22240290') # ID do
dataset SPNeodeath
72
73 df_ori = pd.read_csv(dataset.path)
74
75 """## 1.2 Divisão da base em conjunto para treino e
teste do modelo de ML"""
76
77 df = df_ori.sample(frac=1)
78
79 """# REMOVENDO OS REGISTROS NOS QUAIS PELO MENOS UM
CAMPO ESTÁ EM BRANCO (NAN) """
80 df = df.dropna()
81
82 print("Shape:", df.shape, "\n")
83 df.head()
84
85 # Todas as "Features" da base de dados
86 features = ['tp_birth_place', 'maternal_age',
'tp_marital_status',
'          tp_maternal_education_years', '
num_live_births', 'num_fetal_losses',
88           tp_pregnancy_duration', '
tp_pregnancy', 'tp_labor',
89           tp_prenatal_appointments', '
cd_apgar1', 'cd_apgar5',
90           newborn_weight', '
has_congenital_malformation', '
tp_maternal_skin_color',
91           num_gestations', 'num_normal_labors'
, 'num Cesarean_labors',
```

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_spneodeath_2012_2018_final.py
92             'num_gestational_weeks', 'tp_presentation_newborn', 'tp_childbirth_assistance'
93             'tp_fill_form_responsible', 'cd_robson_group']
94
95 target = ['neonatal_death']
96
97 # Separação das features e do "target" para serem
# usados no modelo posteriormente
98 xArray = df[features]
99 yArray = df[target]
100
101 # "Split" do dataset para Treino e Teste do modelo
# usando 90% para o treino e 10% para teste
102 xTreino, xTeste, yTreino, yTeste = train_test_split(
    xArray, yArray, test_size=0.1 ,random_state=42)
103
104 print(xTreino.shape)
105 print(xTeste.shape)
106 print(yTreino.shape)
107 print(yTeste.shape)
108
109 """# 2. Aplicação de modelos de Machine Learning:
# Logistic Regression
110
111 ---
112
113 ## Usando particionamento 90/10
114 """
115
116 logreg = LogisticRegression()
117
118 logreg.fit(xTreino, yTreino)
119 y_pred = logreg.predict(xTeste)
120 y_pred_proba = logreg.predict_proba(xTeste)[:, 1]
121
122 [fpr, tpr, thr] = roc_curve(yTeste, y_pred_proba)
123
124 print('Treino/Teste resultados divididos %s:' %
    logreg.__class__._name__)

```

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_spneodeath_2012_2018_final.py
125 print("Accuracy is %2.3f" % accuracy_score(yTeste,
      y_pred))
126 print("Log_loss is %2.3f" % log_loss(yTeste,
      y_pred_proba))
127 print("AUC is %2.3f" % auc(fpr, tpr))
128
129 """#### Curva ROC"""
130
131 idx = np.min(np.where(tpr > 0.95))
132
133 plt.figure()
134 plt.plot(fpr, tpr, color='coral', label='Curva ROC (
    area = %0.3f)' % auc(fpr, tpr))
135 plt.plot([0, 1], [0, 1], 'k--')
136 plt.plot([0,fpr[idx]], [tpr[idx],tpr[idx]], 'k--',
    color='blue')
137 plt.plot([fpr[idx],fpr[idx]], [0,tpr[idx]], 'k--',
    color='blue')
138 plt.xlim([0.0, 1.0])
139 plt.ylim([0.0, 1.05])
140 plt.xlabel('Taxa de falso positivo (1 -
    especificidade)', fontsize=14)
141 plt.ylabel('Taxa de verdadeiro positivo (recall)',
    fontsize=14)
142 plt.title('Curva de característica de operação do
    receptor (ROC)')
143 plt.legend(loc="lower right")
144 plt.show()
145
146 plt.savefig('ROC_90_10.png', dpi=300)
147
148 print("Usando um limite de %.3f " % thr[idx] + "
    garante uma sensibilidade de %.3f " % tpr[idx] +
149     "e uma especificidade de %.3f" % (1-fpr[idx
    ]) +
150     ", ou seja, uma taxa de falsos positivos de %.2f%." %
    (np.array(fpr[idx])*100))
151
152 """#### Matrix de Confusão"""
153
154 acc = accuracy_score(yTeste, y_pred)

```

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_spneodeath_2012_2018_final.py
155 my_cm(confusion_matrix(yTeste, y_pred), acc)
156 plt.savefig('CM_90_10.png', dpi=300)
157
158 """#### Relatório de Classificação"""
159
160 print(classification_report(yTeste, y_pred))
161 report = classification_report(yTeste, y_pred,
162                                 output_dict=True)
162 round(pd.DataFrame(report).transpose(),2).to_csv('
163 ClassRep_90_10.csv')
164 """
165 ## Cross validation com k = 10
166 **No Trabalho incluir apenas o relatório de
167 classificação para comparar com o resultado sem
168 cross-validation, não precisar incluir matrix de
169 conf.**
170 """
171
172 # Regressão logística de validação cruzada de 10
173 # vezes
174 logreg = LogisticRegression()
175 scoring = {'accuracy': 'accuracy', 'log_loss': 'neg_log_loss', 'auc': 'roc_auc'}
176
177 modelo_KFold = cross_validate(logreg, xArray, yArray,
178                                cv=10,
179                                scoring=list(scoring.values
180                                ()),
181                                return_train_score=False)
182
183 print('K-fold cross-validation results:')
184 for sc in range(len(scoring)):
185     print("%s: %.3f (+/-.3f)" % (list(scoring.keys
186                               ())[sc], -modelo_KFold['test_%s' % list(scoring.
187                                values())[sc]].mean()
188                                if list(scoring.
189                                values())[sc]=='neg_log_loss'
190                                else modelo_KFold['
191                                test_%s' % list(scoring.values())[sc]].mean(),
192                                modelo_KFold['test_%s

```

```
File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_spneodeath_2012_2018_final.py
182 ' % list(scoring.values())[sc]].std()))
183
184 """#### CURVA ROC para CROSS VALIDATION"""
185
186 #https://scikit-learn.org/stable/auto_examples/
    model_selection/plot_roc_crossval.html
187 X = xArray.copy()
188 y = yArray.copy()
189
190 X.reset_index(inplace=True)
191 X.drop(columns={'index'}, inplace=True)
192
193 y.reset_index(inplace=True)
194 y.drop(columns={'index'}, inplace=True)
195
196 n_samples, n_features = X.shape
197
198 # ##########
# ##########
199 # Classification and ROC analysis
200
201 # Run classifier with cross-validation and plot ROC
    curves
202 cv = StratifiedKFold(n_splits=10)
203 classifier = LogisticRegression()
204
205 tprs = []
206 aucss = []
207 mean_fpr = np.linspace(0, 1, 100)
208
209 fig, ax = plt.subplots()
210 fig.set_figwidth(10)
211 fig.set_figheight(10)
212
213 for i, (train, test) in enumerate(cv.split(X, y)):
214     classifier.fit(X[X.index.isin(train)], y[y.index
        .isin(train)])
215     viz = plot_roc_curve(classifier, X[X.index.isin(
        test)], y[y.index.isin(test)],
216                         name='ROC fold {}'.format(i),
        ),
```

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_spneodeath_2012_2018_final.py
217                                         alpha=0.3, lw=1, ax=ax)
218     interp_tpr = np.interp(mean_fpr, viz.fpr, viz.
219     tpr)
220     interp_tpr[0] = 0.0
221     tprs.append(interp_tpr)
222     aucss.append(viz.roc_auc)
223
224 ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color=
225         'r',
226         label='Chance', alpha=.8)
227
228 mean_tpr = np.mean(tprs, axis=0)
229 mean_tpr[-1] = 1.0
230 mean_auc = auc(mean_fpr, mean_tpr)
231 std_auc = np.std(aucss)
232 ax.plot(mean_fpr, mean_tpr, color='b',
233         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' %
234             (mean_auc, std_auc),
235         lw=2, alpha=.8)
236
237 std_tpr = np.std(tprs, axis=0)
238 tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
239 tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
240 ax.fill_between(mean_fpr, tprs_lower, tprs_upper,
241     color='grey', alpha=.2,
242     label=r'$\pm$ 1 std. dev.')
243
244
245 ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],
246         title="Receiver operating characteristic
247             example")
248 ax.legend(loc="lower right")
249 plt.show()
250
251 plt.savefig('ROC_KFold_10.png', dpi=300)
252
253 """#### Matrix de Confusão"""
254
255 y_pred = cross_val_predict(logreg, xArray, yArray,
256     cv=10)
257 acc = modelo_KFold['test_accuracy'].mean()
258 my_cm(confusion_matrix(yArray, y_pred), acc)

```

```
File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_spneodeath_2012_2018_final.py
252 plt.savefig('CM_KFold.png', dpi=300)
253
254 """#### Relatório de Classificação"""
255
256 print(classification_report(yArray, y_pred))
257 report = classification_report(yArray, y_pred,
258 output_dict=True)
259 round(pd.DataFrame(report).transpose(),2).to_csv('
260 ClassRep_KFold.csv')
261
262 """## RFE (Eliminação recursiva de features)
263 A ideia da eliminação recursiva de feature (RFE) é
selecionar features considerando recursivamente
conjuntos cada vez menores de features, isso ocorre
da seguinte forma. Primeiro, o estimador é treinado
no conjunto inicial de features e a importância de
cada feature é obtida por meio de um atributo "coef_
" ou por meio de um atributo "feature_importances_".
Em seguida, os features menos importantes são
removidos do conjunto atual de features. Esse
procedimento é repetido recursivamente no conjunto
removido até que o número desejado de features a
serem selecionados seja finalmente alcançado.
264 """
265
266 """ Criando um modelo com número de features
267 otimizado usando RFECV
268 RFECV executa RFE em um loop de validação cruzada
para encontrar o número ideal ou o melhor número de
features. No código abaixo temos uma eliminação de
feature recursiva aplicada na regressão logística
com ajuste automático do número de features
selecionados com validação cruzada.
269
270 Como resultado o código trouxe 6 features que é o
271 numero ideia de features para o modelo e trouxe
também as melhores features sendo elas:
272 - 'tp_maternal_schooling'
273 - 'gestaional_week'
```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_spneodeath_2012_2018_final.py

```

272 - 'cd_apgar1'
273 - 'cd_apgar5'
274 - 'has_congenital_malformation'
275 - 'tp_labor'
276 """
277
278 # Crie o objeto RFE e calcule uma pontuação com
# validação cruzada.
279 # A pontuação de "precisão" é proporcional ao número
# de classificações corretas
280 rfecv = RFECV(estimator=LogisticRegression(), step=1
, cv=10, scoring='accuracy')
281 rfecv.fit(xArray, yArray)
282
283 print("Número ideal de Features: %d" % rfecv.
n_features_)
284 print('Features Selecionadas: %s' % list(xArray.
columns[rfecv.support_]))
285
286 # Número do lote das Features VS. pontuações de
# validação cruzada
287 plt.figure(figsize=(7,5))
288 plt.xlabel("Número de Features selecionadas")
289 plt.ylabel("Pontuação de validação cruzada (nb de
classificações corretas)")
290 plt.plot(range(1, len(rfecv.grid_scores_) + 1),
rfecv.grid_scores_)
291 plt.show()
292
293 """## Modelo Final: Usar apenas variáveis
selecionadas e aplicar o GridSearchCV
294
295 No RFECV vc esta esclhendo mehiores features de sua
base, no gridSearch vc esta escolhendo parametros do
algoritmo.
296 """
297
298 xArray_RFE = xArray[list(xArray.columns[rfecv.
support_])]
299 xArray_RFE.head()
300

```

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_spneodeath_2012_2018_final.py
301 param_grid = {'C': np.arange(1e-05, 3, 0.1)}
302 scoring = {'Accuracy': 'accuracy', 'AUC': 'roc_auc'
303             , 'Log_loss': 'neg_log_loss'}
304 gs = GridSearchCV(LogisticRegression(),
305                     return_train_score=True,
306                     param_grid=param_grid, scoring=
307                     scoring, cv=10, refit='Accuracy')
308 results = gs.cv_results_
309
310 print('='*20)
311 print("Best params: " + str(gs.best_estimator_))
312 print("Best params: " + str(gs.best_params_))
313 print('Best score:', gs.best_score_)
314 print('='*20)
315
316 plt.figure(figsize=(10, 5))
317 plt.title("GridSearchCV evaluating using multiple
318 scorers simultaneously", fontsize=16)
319 plt.xlabel("Inverse of regularization strength: C")
320 plt.ylabel("Score")
321 plt.grid()
322
323 ax = plt.axes()
324 ax.set_xlim(0, param_grid['C'].max())
325 ax.set_ylim(0.0, 1.2)
326
327 X_axis = np.array(results['param_C'].data, dtype=
328                     float)
329 for scorer, color in zip(list(scoring.keys()), ['g'
330             , 'k', 'b']):
331     for sample, style in (('train', '--'), ('test',
332             '-')):
333         sample_score_mean = -results['mean_%s_%s'
334             % (sample, scorer)] if scoring[scorer]=='_
335 neg_log_loss' else results['mean_%s_%s' % (sample,
336             scorer)]
```

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglógica_spneodeath_2012_2018_final.py
332         sample_score_std = results['std_%s_%s' % (
333             sample, scorer)]
334         ax.fill_between(X_axis, sample_score_mean -
335                         sample_score_std,
336                             sample_score_mean +
337                         sample_score_std,
338                         alpha=0.1 if sample == 'test'
339                         ' else 0, color=color)
340         ax.plot(X_axis, sample_score_mean, style,
341             color=color,
342             alpha=1 if sample == 'test' else 0.7
343             ,
344             label="%s (%s)" % (scorer, sample))
345
346         best_index = np.nonzero(results['rank_test_%s' %
347             scorer] == 1)[0][0]
348         best_score = -results['mean_test_%s' % scorer][
349             best_index] if scoring[scorer]=='neg_log_loss' else
350             results['mean_test_%s' % scorer][best_index]
351
352         ax.plot([X_axis[best_index], ] * 2, [0,
353             best_score],
354             linestyle='-.', color=color, marker='x'
355             , markeredgewidth=3, ms=8)
356
357         ax.annotate("%0.2f" % best_score,
358             (X_axis[best_index], best_score + 0.
359             005))
360
361 plt.legend(loc="best")
362 plt.grid('off')
363 plt.show()
364
365 """#### MODELO FINAL: Usando modelo sugerido pelo
366 Grid Search com variáveis selecionadas"""
367
368 logReg_GS = LogisticRegression(C=1e-05, class_weight
369             =None, dual=False, fit_intercept=True,
370                 intercept_scaling=1, l1_ratio=
371                 None, max_iter=100,
372                     multi_class='auto', n_jobs=None,

```

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_spneodeath_2012_2018_final.py
357 penalty='l2',
358                     random_state=None, solver='lbfgs'
359                     , tol=0.0001, verbose=0,
360                     warm_start=False)
361 scoring = {'accuracy': 'accuracy', 'log_loss': 'neg_log_loss', 'auc': 'roc_auc'}
362
363 modelo_Final = cross_validate(logReg_GS, xArray_RFE
364 , yArray, cv=10,
364                     scoring=list(scoring.values
365 ()},
365                     return_train_score=False)
366
367 print('K-fold cross-validation results:')
368 for sc in range(len(scoring)):
369     print("%s: %.3f (+/-%.3f)" % (list(scoring.keys
369 ())[sc], -modelo_Final['test_%s' % list(scoring.
369 values())[sc]].mean()
370                     if list(scoring.
370 values())[sc]=='neg_log_loss'
371                     else modelo_Final['
371 test_%s' % list(scoring.values())[sc]].mean(),
372                     modelo_Final['test_%s
372 ' % list(scoring.values())[sc]].std()))
373
374 """#### Curva ROC"""
375
376 #https://scikit-learn.org/stable/auto_examples/
376 model_selection/plot_roc_crossval.html
377 X = xArray_RFE.copy()
378 y = yArray.copy()
379
380 X.reset_index(inplace=True)
381 X.drop(columns={'index'}, inplace=True)
382
383 y.reset_index(inplace=True)
384 y.drop(columns={'index'}, inplace=True)
385
386 n_samples, n_features = X.shape
387

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_spneodeath_2012_2018_final.py

```

388 # ##########
389 # #####
389 # Classification and ROC analysis
390
391 # Run classifier with cross-validation and plot ROC
391 # curves
392 cv = StratifiedKFold(n_splits=10)
393 classifier = logReg_GS
394
395 tprs = []
396 aucs = []
397 mean_fpr = np.linspace(0, 1, 100)
398
399 fig, ax = plt.subplots()
400 fig.set_figwidth(10)
401 fig.set_figheight(10)
402
403 for i, (train, test) in enumerate(cv.split(X, y)):
404     classifier.fit(X[X.index.isin(train)], y[y.index
404 .isin(train)])
405     viz = plot_roc_curve(classifier, X[X.index.isin(
405 test)], y[y.index.isin(test)],
406                         name='ROC fold {}'.format(i
406 ),
407                         alpha=0.3, lw=1, ax=ax)
408     interp_tpr = np.interp(mean_fpr, viz.fpr, viz.
408 tpr)
409     interp_tpr[0] = 0.0
410     tprs.append(interp_tpr)
411     aucs.append(viz.roc_auc)
412
413 ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color=
413 'r',
414         label='Chance', alpha=.8)
415
416 mean_tpr = np.mean(tprs, axis=0)
417 mean_tpr[-1] = 1.0
418 mean_auc = auc(mean_fpr, mean_tpr)
419 std_auc = np.std(aucs)
420 ax.plot(mean_fpr, mean_tpr, color='b',
421         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)'

```

```
File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_spneodeath_2012_2018_final.py
421 % (mean_auc, std_auc),
422         lw=2, alpha=.8)
423
424 std_tpr = np.std(tprs, axis=0)
425 tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
426 tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
427 ax.fill_between(mean_fpr, tprs_lower, tprs_upper,
428                   color='grey', alpha=.2,
429                   label=r'$\pm$ 1 std. dev.')
430
431 ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],
432         title="Receiver operating characteristic
433             example")
434 ax.legend(loc="lower right")
435 plt.show()
436
437 """#### Matriz de confusão"""
438
439 y_pred = cross_val_predict(logReg_GS, xArray_RFE,
440     yArray, cv=10)
441 acc = modelo_Final['test_accuracy'].mean()
442 my_cm(confusion_matrix(yArray, y_pred), acc)
443 plt.savefig('CM_Final.png', dpi=300)
444
445 """#### Relatório de Classificação"""
446 print(classification_report(yArray, y_pred))
447 report = classification_report(yArray, y_pred,
448     output_dict=True)
449 round(pd.DataFrame(report).transpose(), 2).to_csv('
450     ClassRep_Final.csv')
```

APÊNDICE C

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py
1 # -*- coding: utf-8 -*-
2 """TCC_CaioMota_09/06/
3     2021_AnaliseExploratória_BaseBrasil_2016ate2018.ipynb
4
5     Automatically generated by Colaboratory.
6
7     Original file is located at
8         https://colab.research.google.com/drive/1oIT-
9             RG6DPcliZ2bLMvFicESEuH2DRB_P
10
11    **Implementação do algoritmo de Analise Exploratoria
12        **
13
14    <br>
15
16    **Autor**: Caio Augusto de Souza Mota (*caiomota802@gmail.com*)
17
18    <br>
19    <br>
20    *Codigo adaptado de Baligh Mnassri disponivel em:
21        https://www.kaggle.com/mnassrib/titanic-logistic-
22            regression-with-python/notebook*
23
24    ---
```

22 Este código é parte do Trabalho de Conclusão de Curso
apresentado como exigência parcial para obtenção do
diploma do Curso de Tecnologia em Análise e
Desenvolvimento de Sistemas do Instituto Federal de
Educação, Ciência e Tecnologia de São Paulo Câmpus
Campinas.

23

24 # 1. Importação de Bibliotecas, carga de dados e
funções

25 """

26

27 # instalando o Synapse Client

28 ! pip install synapseclient

```
File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py
29
30 import os
31 import synapseclient as syna
32 from getpass import getpass
33
34 import numpy as np
35 from math import sqrt
36 import pandas as pd
37
38
39 import matplotlib.pyplot as plt
40 plt.rc("font", size=14)
41
42 from matplotlib.ticker import PercentFormatter
43 import matplotlib.ticker as ticker
44
45 import seaborn as sns
46 sns.set(style="white") #white background style for
        seaborn plots
47 sns.set(style="whitegrid", color_codes=True)
48
49 """## 1.1 Carregando base de dados disponível no
        Synapse"""
50
51 # BRNeodeath
52 # Recuperando a base de dados do repositório de dados
        Synapse
53 syn = syna.Synapse()
54 syn.login(input('Sybapse User: '), getpass('Passwd: '))
55
56 # Obtendo um ponteiro e baixando os dados
57 dataset = syn.get(entity='syn25575811') # ID do
        dataset BRNeodeath
58
59 df_ori = pd.read_csv(dataset.path)
60 df_ori
61
62 df = df_ori.sample(frac=1).copy()
63 df.shape
64
```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py

```

65 """# 2. Etapa de pre-processamento de dados
66
67 Esta base já foi pre-processada, logo não precisa de
   tratamento de nulos. Apenas référenciaie ela dizendo
   que você recuperou ela do Synapse por link msm. Diz
   que ela não é pública, mas pode ser reconstruída a
   partir de dados públicos do SIM e SINASC.
68
69 O único campo com valores nulos é "death_date", que
   os registros dos vivos, logo o campo deve ser nulo.
   De qualquer forma liste as features que serão
   removidas.
70
71 birth_date          0
72 death_date          6719191
73 birth_year          0
74 uf                  0
75 id                  0
76
77 Não precisa colocar código no TCC, apenas descritivo
   .
78
79 ## Não incluir
80 """
81
82 # verificar os valores ausentes nos dados de treino
83 df.isnull().sum()
84
85 # Base final
86 features = ['maternal_age', 'tp_maternal_schooling',
   , 'tp_marital_status',
87           'tp_maternal_race', 'num_live_births',
   , 'num_fetal_losses',
88           'num_previous_gestations', ,
   num_normal_labors', 'num Cesarean_labor',
89           'tp_pregnancy', 'newborn_weight', ,
   gestaional_week',
90           'cd_apgar1', 'cd_apgar5', ,
   has_congenital_malformation',
91           'tp_newborn_presentation', ,
   num_prenatal_appointments', 'tp_labor',

```

```
File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py
92             'was_cesarean_before_labor', '
93             was_labor_induced', 'tp_childbirth_care',
93             'tp_robson_group','is_neonatal_death'
94         ]
94
95 df =  df[features]
96
97 # Exibindo o nome de todas as colunas
98 colunas = df.columns
99 colunas
100
101 # Exibindo os valores de cada coluna (domínio de
101   valores)
102 for col in colunas:
103     print(col, ":", df[col].dtype)
104     print(df[col].unique(), "\n")
105
106 df.is_neonatal_death.value_counts()
107
108 """## Incluir estas 4 tabelas como apêndice, e citar
108   no início da seção que estão disponíveis as tabelas
108   sumário da base nos apêndices XX, X1, X3 e X4.
109
110 ### Características demográficas e socioeconômicas
110   maternas
111 """
112
113 aux = df[['maternal_age', 'tp_maternal_schooling', '
113   tp_marital_status', 'tp_maternal_race']].describe()
114 aux.to_csv("tab1.csv")
115 aux
116
117 """### Variáveis obstétricas maternas"""
118
119 aux = df[['num_live_births','num_fetal_losses', '
119   num_previous_gestations',
120     'num_normal_labors', 'num_cesarean_labor', '
120   tp_pregnancy']].describe()
121 aux.to_csv("tab2.csv")
122 aux
123
```

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py
124 """### Variáveis de histórico de gravidez"""
125
126 aux = df[['num_prenatal_appointments', 'tp_labor', 'tp_robson_group']].describe()
127 aux.to_csv("tab3.csv")
128 aux
129
130 """### Variáveis relacionadas ao recém-nascido"""
131
132 aux = df[['newborn_weight', 'gestaional_week',
133             'cd_apgar1', 'cd_apgar5', 'has_congenital_malformation',
134             'tp_newborn_presentation', 'tp_labor'
135             ,
136             'was Cesarean before labor', 'was_labor_induced', 'tp_childbirth_care'
137             , 'is_neonatal_death']].describe()
138 aux
139
140 """# 3. Etápa de Análise Exploratória
141
142 No grafico de pizza abaixo podemos ver os valores
    que temos em nosso rotulo sendo 99.4% dos dados de
    nascidos vivos e 0.6% de nascidos mortos, isso
    mostra para nos o quanto a base de dados esta
    desbalanceada.
143 """
144
145 fig,ax = plt.subplots(figsize=(6,6), subplot_kw=dict(
    aspect="equal"))
146
147 topic = ['Nascidos Vivos', 'Nascidos Mortos']
148 labels = list(topic)
149
150 data = [df['is_neonatal_death'].value_counts()]
151
152 def func(pct, allvals):
153     absolute = int(pct/100.*np.sum(allvals))
154     return "{:.1f}%\n({:d})".format(pct, absolute)
155

```

```
File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py
156 wedges, texts, autotexts = ax.pie(data, autopct=lambda
157     pct: func(pct, data), textprops=dict(color="w"))
158 ax.legend(wedges, labels, title="DataFrames", loc=
159             "center left", bbox_to_anchor=(1, 0, 0.5, 1))
160 plt.setp(autotexts, size=12, weight="bold")
161
162 ax.set_title("Porcentagem de amostras de Nascidos
163 Vivos e Nascidos Mortos")
164 plt.show
165 """
166     <br>Nessa sessão plotamos graficos que nos ajuda
167     a visualizar a distribuição que temos nos dados,
168     vendo quais são as medias de cada variáveis.
169 """
170 # Função pra imprimir boxplot
171 def print_boxplot(x_, y_, h_, lbl):
172     legendas = ["Vivos", #0
173                 "mortos", #1
174                 ]
175     ax = sns.boxplot(x=x_, y=y_, hue=h_, data=df)
176     ax.set(xlabel='')
177     ax.set(ylabel=lbl)
178     ax.set_title('Distribuição da variável "%s"' % lbl
179 )
180     ax.set_xticklabels(['Vivos', 'mortos'])
181     h, l = ax.get_legend_handles_labels()
182     ax.legend(h, legendas)
183
184     ax.figure.savefig("y_.png")
185
186 print_boxplot('is_neonatal_death', 'newborn_weight')
```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py

```
186 , 'is_neonatal_death', 'peso ao nascer')
187
188 print_boxplot('is_neonatal_death', 'maternal_age', 'is_neonatal_death', 'idade da mãe')
189
190 print_boxplot('is_neonatal_death', 'gestaional_week', 'is_neonatal_death', 'semanas gestacionais')
191
192 legendas = ["Vivos", #0
193             "Mortos", #1
194             ]
195
196 ax = sns.histplot(data=df, x="newborn_weight", hue="is_neonatal_death", kde=True,
197                     )
198 ax.set_yscale('log')
199
200 ax.set(xlabel='Peso ao nascer')
201 ax.set(ylabel='Distribuição (log)')
202 ax.set_title('Distribuição da variável "%s" por classes' % "peso ao nascer")
203
204 ax.legend(legendas)
205
206 ax.figure.savefig("ditro.png")
207
208 """Nesse Grafico temos a distribuição dos dados da feature "anos de escolaridade da mãe", essa feature utiliza dados categoricos Nominais as categorias sao :
209 <br>1 - nenhum;
210 <br>2 - de 1 a 3 anos;
211 <br>3 - de 4 a 7 anos;
212 <br>4 - de 8 a 11 anos;
213 <br>5 - 12 e mais anos;
214 <br>9 - ignorado;
215 """
216
217 plt.figure(figsize=(10,4))
218 titulo = "Distribuição da variável Escolaridade da mãe"
```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py

```

219 label_Y = "Escolaridade da mãe" []
220 label_X = "Contagem" []
221 legendas = ["Dados errados",
222                 "nenhum", #1
223                 "de 1 a 3 anos", #2
224                 "de 4 a 7 anos", #3
225                 "de 8 a 11 anos", #4
226                 "12 e mais anos", #6
227                 "ignorado"] #9
228
229 ax = sns.countplot(y = 'tp_maternal_schooling', hue=
230                      'tp_maternal_schooling', dodge=False, data = df)
231 ax.set(xlabel=label_X)
232 ax.set(ylabel=label_Y)
233 ax.set_title(titulo)
234 ax.legend(loc='upper right', labels=legendas)
235 ax.figure.savefig("tp_maternal_schooling.png")
236
237 # media Semana de gestação(por intervalos)
238 print('A média de "Anos de escolaridade da mãe" é %.0f' %(df["tp_maternal_schooling"].mean(skipna=True)))
239 # mediana Semana de gestação(por intervalos)
240 print('A mediana de "Anos de escolaridade da mãe" é %.0f' %(df["tp_maternal_schooling"].median(skipna=True)))
241
242 print('Agrupamento dos anos de escolaridade da mãe:')
243
244 variável = 'num_live_births'
245 plt.figure(figsize=(10,4))
246 titulo = "Distribuição da variável Número de nascidos vivos"
247 label_X = "Número de nascidos vivos" []
248 label_Y = "Contagem" []
249
250 ax = sns.countplot(x = 'num_live_births', dodge=
251                      False, data = df)
252 ax.set(xlabel=label_X)

```

```
File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py
252 ax.set(ylabel=label_Y)
253 ax.set_title(titulo)
254 ax.figure.savefig("num_live_births.png")
255
256 # media Semana de gestação(por intervalos)
257 print('A média de "Número de nascidos vivos da mãe" é %.1f' %(df["num_live_births"].mean(skipna=True)))
258 # mediana Semana de gestação(por intervalos)
259 print('A mediana de "Número de nascidos vivos da mãe" é %.0f' %(df["num_live_births"].median(skipna=True)))
260
261 print('Agrupamento do número de nascidos vivos da mãe:')
262 print(df['num_live_births'].value_counts())
263
264 variavel = 'cd_apgar1'
265 plt.figure(figsize=(10,4))
266 titulo = "Distribuição da variável Pontuação de Apgar de 1 minuto"
267 label_X = "Pontuação de Apgar de 1 minuto"
268 label_Y = "Contagem"
269
270
271 ax = sns.countplot(x=variavel, dodge=False, data = df)
272 ax.set(xlabel=label_X)
273 ax.set(ylabel=label_Y)
274 ax.set_title(titulo)
275 ax.figure.savefig("graf.png")
276
277 # media Semana de gestação(por intervalos)
278 print('A média de "Pontuação de Apgar de 1 minuto" é %.0f' %(df["cd_apgar1"].mean(skipna=True)))
279 # mediana Semana de gestação(por intervalos)
280 print('A mediana de "Pontuação de Apgar de 1 minuto" é %.0f' %(df["cd_apgar1"].median(skipna=True)))
281
282 print('Agrupamento da pontuação de Apgar de 1 minuto :')
283 print(df['cd_apgar1'].value_counts())
```

```
File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py
284
285 variavel = 'cd_apgar5'
286 plt.figure(figsize=(10,4))
287 titulo = "Distribuição da variável Pontuação de
288 Apgar de 5 minuto"
289 label_X = "Pontuação de Apgar de 5 minuto"
290 label_Y = "Contagem"
291
292 ax = sns.countplot(x=variavel, dodge=False, data =
293 df)
294 ax.set(xlabel=label_X)
295 ax.set(ylabel=label_Y)
296 ax.set_title(titulo)
297 ax.figure.savefig("graf.png")
298
299 # media Semana de gestação(por intervalos)
300 print('A média de "Pontuação de Apgar de 5 minuto" é
301 %.0f' %(df["cd_apgar5"].mean(skipna=True)))
302 # mediana Semana de gestação(por intervalos)
303 print('A mediana de "Pontuação de Apgar de 5 minuto
304 " é %.0f' %(df["cd_apgar5"].median(skipna=True)))
305
306 print('Agrupamento da pontuação de Apgar de 5 minuto
307 :')
308 print(df['cd_apgar5'].value_counts())
309
310 variavel = 'has_congenital_malformation'
311 plt.figure(figsize=(10,4))
312 titulo = "Distribuição da variável Presença de
313 malformação congênita"
314 label_Y = "Presença de malformação congênita"
315 label_X = "Contagem"
316 legendas = ["Sim", #1
317             "Não", #2
318             "ignorado"] #9
319
320
321 ax = sns.countplot(y=variavel, hue=variavel, dodge=
322 False, data = df)
323 ax.set(xlabel=label_X)
324 ax.set(ylabel=label_Y)
325 ax.set_title(titulo)
```

```
File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py
318 ax.legend(loc='upper right', labels=legendas)
319 ax.figure.savefig("graf.png")
320
321 # media Semana de gestação(por intervalos)
322 print('A média de "Presença de malformação congênita"
       " é %.0f' %(df["has_congenital_malformation"].mean(
           skipna=True)))
323 # mediana Semana de gestação(por intervalos)
324 print('A mediana de "Presença de malformação
       congênita" é %.0f' %(df["has_congenital_malformation
       "].median(skipna=True)))
325
326 print('Agrupamento da Presença de malformação
       congênita:')
327 print(df['has_congenital_malformation'].value_counts(
      ()))
328
329 variavel = 'num_previous_gestations'
330 plt.figure(figsize=(10,4))
331 titulo = "Distribuição da variável Número de
       gestações anteriores"
332 label_X = "Número de gestações anteriores"
333 label_Y = "Contagem"
334
335 ax = sns.countplot(x=variavel, dodge=False, data =
       df)
336 ax.set(xlabel=label_X)
337 ax.set(ylabel=label_Y)
338 ax.set_title(titulo)
339 ax.figure.savefig("graf.png")
340
341 # media Semana de gestação(por intervalos)
342 print('A média de "Número de gestações anteriores" é
       %.0f' %(df["num_previous_gestations"].mean(skipna=
           True)))
343 # mediana Semana de gestação(por intervalos)
344 print('A mediana de "Número de gestações anteriores
       " é %.0f' %(df["num_previous_gestations"].median(
           skipna=True)))
345
346 print('Agrupamento do Número de gestações anteriores
```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py

```

346 da mãe:")
347 print(df['num_previous_gestations'].value_counts())
348
349 variavel = 'tp_childbirth_care'
350 plt.figure(figsize=(10,4))
351 titulo = "Distribuição da variável Assistência ao
352 label_Y = "Assistência ao parto"
353 label_X = "Contagem"
354 legendas = ["nenhum", #0
355             "Doutor", #1
356             "Enfermeira ou obstetra",#2
357             "Parteira", #3
358             "Outros",#4
359             "Ignorado"] #9
360
361 ax = sns.countplot(y=variavel, hue=variavel, dodge=
362                      False, data = df)
363 ax.set(xlabel=label_X)
364 ax.set(ylabel=label_Y)
365 ax.set_title(titulo)
366 ax.legend(loc='upper right', labels=legendas)
367 ax.figure.savefig("graf.png")
368
369 # media Semana de gestação(por intervalos)
370 print('A média de "Assistência ao parto" é %.0f' %(df["tp_childbirth_care"].mean(skipna=True)))
371 # mediana Semana de gestação(por intervalos)
372 print('A mediana de "Assistência ao parto" é %.0f' %(df["tp_childbirth_care"].median(skipna=True)))
373 print('Agrupamento do Assistência ao parto:')
374 print(df['tp_childbirth_care'].value_counts())
375
376 plt.figure(figsize=(15,8))
377 ax = sns.kdeplot(df["newborn_weight"][df.
378                     is_neonatal_death == 1], color="lightcoral", shade=
379                     True)
380 sns.kdeplot(df["newborn_weight"][df.
381                     is_neonatal_death == 0], color="darkturquoise",
382                     shade=True)

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py

```

379 plt.legend(['Mortos', 'Vivos'])
380 plt.title('Gráfico de densidade de Peso ao nascer em
            gramas', fontsize=15)
381 ax.set(xlabel='Peso ao nascer em gramas')
382 ax.set(ylabel='Densidade')
383 ax.tick_params(labelsize=15)
384 plt.xlim(0,6000)
385 plt.show()
386
387 X = df[['maternal_age',
388           'num_prenatal_appointments',
389           'cd_apgar1 ',
390           'cd_apgar5',
391           'newborn_weight',
392           'num_previous_gestations',
393           'num_normal_labors',
394           'num Cesarean_labor',
395           'gestaional_week',
396           'tp_newborn_presentation']]
397
398 X.rename(columns = {'maternal_age':'Idade da Mãe',
399                 'num_prenatal_appointments':'
Número de consultas de pré-natal por faixas'
400                 ,'cd_apgar1':'Pontuação de Apgar
de 1 minuto ',
401                 'cd_apgar5':'Pontuação de Apgar
de 5 minuto ',
402                 'newborn_weight':'Peso ao nascer
em gramas',
403
404                 'num_previous_gestations':'
Número de gestações anteriores',
405                 'num_normal_labors':'Número de
partos normais (trabalhos de parto)',
406                 'num_cesarean_labor':'Número de
partos cesáreos (partos)'
407                 ,'gestaional_week':'Semana de
gestação (por intervalos)',
408                 'tp_newborn_presentation':'Tipo
de apresentação de recém-nascido'
409                 }, inplace=True)

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py

```
410
411 plt.subplots(figsize=(15, 10))
412 plt.tick_params(labelsize=12)
413 sns.heatmap(X.corr(), annot=True, cmap="RdYlGn")
414 plt.show()
415
416 """# Distribuição por raça"""
417
418 ax = sns.boxplot(y="maternal_age", x=
    "tp_maternal_race", data=df)
419 ax.set(xlabel="Raça da mãe")
420 ax.set(ylabel="Idade da mãe")
421
422 ax = sns.boxplot(y="newborn_weight", x=
    "tp_maternal_race", data=df)
423 ax.set(xlabel="Raça da mãe")
424 ax.set(ylabel="Peso ao nascer")
425
426 ax=sns.boxplot(y="gestaional_week", x=
    "tp_maternal_race", data=df)
427 ax.set(xlabel="Raça da mãe")
428 ax.set(ylabel="Semanas de gestação")
429
430 """# Distribuição por estado civil"""
431
432 ax=sns.boxplot(y="newborn_weight", x=
    "tp_marital_status", data=df)
433 ax.set(xlabel="Estado civil da mãe")
434 ax.set(ylabel="Peso ao nascer")
435
436 ax=sns.boxplot(y="gestaional_week", x=
    "tp_marital_status", data=df)
437 ax.set(xlabel="Estado civil da mãe")
438 ax.set(ylabel="Semanas de gestação")
439
440 ax=sns.boxplot(y="maternal_age", x=
    "tp_marital_status", data=df)
441 ax.set(xlabel="Estado civil da mãe")
442 ax.set(ylabel="Idade da mãe")
443
444 """# Distribuição por tp_maternal_schooling"""
```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_09_06_2021_analiseexploratória_basebrasil_2016ate2018.py

```
445  
446 ax=sns.boxplot(y="maternal_age", x="  
    tp_maternal_schooling", data=df)  
447 ax.set(xlabel="Escolaridade da mãe")  
448 ax.set(ylabel="Idade da mãe")  
449  
450 ax=sns.boxplot(y="newborn_weight", x="  
    tp_maternal_schooling", data=df)  
451 ax.set(xlabel="Escolaridade da mãe")  
452 ax.set(ylabel="Peso ao nascer")  
453  
454 ax=sns.boxplot(y="gestaional_week", x="  
    tp_maternal_schooling", data=df)  
455 ax.set(xlabel="Escolaridade da mãe")  
456 ax.set(ylabel="Semanas de gestação")
```

APÊNDICE D

	maternal_age	tp_maternal_schooling	tp_marital_status	tp_maternal_race
count	6760222.0	6760222.0	6760222.0	6760222.0
mean	26.401345991300285	3.998663949201668	2.353586908832284	2.891211856652045
std	6.6611898645498	0.876181220667121	1.674026551700602	1.790012710421487
min	8.0	0.0	1.0	1.0
25%	21.0	4.0	1.0	1.0
50%	26.0	4.0	2.0	4.0
75%	31.0	4.0	2.0	4.0
max	55.0	9.0	9.0	9.0

APÊNDICE E

	num_live_births	num_fetal_losses	num_previous_gestations	num_normal_labors	num Cesarean_labor	tp_pregnancy
count	6760222.0	6760222.0	6760222.0	6760222.0	6760222.0	6760222.0
mean	0.9434919444953139	0.21755853579956397	1.142718981713914	0.6418286263380107	0.32911951708094794	1.027113458700025
std	1.2190498593787504	0.5296863372364051	1.3921779943827164	1.182946998566041	0.6264943476705581	0.2557842198613431
min	0.0	0.0	0.0	0.0	0.0	1.0
25%	0.0	0.0	0.0	0.0	0.0	1.0
50%	1.0	0.0	1.0	0.0	0.0	1.0
75%	1.0	0.0	2.0	1.0	1.0	1.0
max	10.0	5.0	10.0	10.0	6.0	9.0

APÊNDICE F

	num_prenatal_appointments	tp_labor	tp_robson_group
count	6760222.0	6760222.0	6760222.0
mean	7.806817438835589	1.5749163267123476	4.009188751493664
std	2.8997939770324033	0.5282566476986162	2.676933729338736
min	0.0	1.0	1.0
25%	6.0	1.0	2.0
50%	8.0	2.0	3.0
75%	10.0	2.0	5.0
max	40.0	9.0	11.0

APÊNDICE G

	newborn_weight	gestaional_week	cd_apgar1	cd_apgar5	has_congenital_malformation	tp_newborn_presentation	tp_labor	was Cesarean before labor	was_labor_induced	tp_childbirth_care	is_neonatal_death
count	6760222.0	6760222.0	6760222.0	6760222.0	6760222.0	6760222.0	6760222.0	6760222.0	6760222.0	6760222.0	6760222.0
mean	3187.3027962691167	38.4816275264333	8.348666360365089	9.355560364733584	2.114988679365855	1.153266712247024	1.5749163267123476	4.1499194848926555	2.0903014723480973	3.716102222678486	0.006069475233209797
std	550.6572641717921	2.195726353135084	1.2145860172741434	0.8703954805230871	0.9272452881421853	0.9509163220331359	0.5282566476986162	3.442838027682753	1.4422798471046538	1.904822287294678	0.07767005598204807
min	0.0	15.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0
25%	2905.0	38.0	8.0	9.0	2.0	1.0	1.0	1.0	2.0	2.0	0.0
50%	3215.0	39.0	9.0	9.0	2.0	1.0	2.0	2.0	2.0	5.0	0.0
75%	3522.0	40.0	9.0	10.0	2.0	1.0	2.0	9.0	2.0	5.0	0.0
max	6000.0	45.0	10.0	10.0	9.0	9.0	9.0	9.0	9.0	9.0	1.0

APÊNDICE H

File - /Users/carlosbeluzo/Downloads/tcc_caiomota_arvoredecisao_brneodeath_2016_2018_final.py

```

1 # -*- coding: utf-8 -*-
2 """TCC_CaioMota_ArvoreDecisao_BRNeodeath_2016_2018_FINAL.ipynb
3
4 Automatically generated by Colaboratory.
5
6 Original file is located at
7     https://colab.research.google.com/drive/1lWavsUi1NOM-
8     HTOFKWiYbJUU92NadjRa
9
10 **Implementação de modelo utilizando algoritmo de Árvore de Decisão**
11 <br>
12 **Autor**: Caio Augusto de Souza Mota (*caiomota802@gmail.com*)
13
14 Data: 09/06/2021
15
16 **Revisor**: Carlos Eduardo Beluzo (*cbeluzo@gmail.com*)
17 <br>
18 <br>
19 *Código adaptado de Diogo Cortiz disponível em: https://github.com/
diogocortiz/Curso-IA-para-todos/tree/master/ArvoreDecis%C3%A3o*
20
21 ---
22 Este código é parte do Trabalho de Conclusão de Curso apresentado como
exigência parcial para obtenção do diploma do Curso de Tecnologia em
Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação
, Ciência e Tecnologia de São Paulo Câmpus Campinas.
23
24 # 1. Importação de Bibliotecas, carga de dados e funções
25 """
26
27 # Instalando o Synapse Client
28 ! pip install synapsecient
29
30 # Util
31 import itertools
32 import synapsecient as syna
33 from getpass import getpass
34
35 # Data
36 import numpy as np
37 import pandas as pd
38
39 from sklearn.tree import DecisionTreeClassifier, export_graphviz
40 from sklearn import preprocessing
41 from sklearn.model_selection import train_test_split, cross_val_score
, cross_validate, cross_val_predict
42 from sklearn.externals.six import StringIO
43 from sklearn.metrics import accuracy_score, classification_report,
precision_score, recall_score, confusion_matrix,
precision_recall_curve, roc_curve, auc, log_loss
44
45 # Images
46 import matplotlib.pyplot as plt
47 import pydotplus
48 import matplotlib.image as mpimg
49 from IPython.display import Image
50
51 """## 1.1 Carregando base de dados disponível no Synapse"""
52
53 # BRNeodeath
54 # Recuperando a base de dados do repositório de dados Synapse
55 syn = syna.Synapse()
56 syn.login(input('Sybapse User: '), getpass('Passwd: '))
57
58 # Obtendo um ponteiro e baixando os dados
59 dataset = syn.get(entity='syn25575811') # ID do dataset BRNeodeath
60
61 df_ori = pd.read_csv(dataset.path)
62 df_ori.shape
63

```

```

File - /Users/carlosbeluzo/Downloads/tcc_caiomota_arvoredecisao_brneodeath_2016_2018_final.py
64 # Amostra para testes rápidos
65 df = df_ori.sample(frac=1)
66
67 # Exibindo as 5 primeiras linhas
68 print(df.head(5))
69
70 count_row = df.shape[0] # Número de linhas
71 count_col = df.shape[1] # Número de colunas
72
73 print(count_row)
74 print(count_col)
75
76 # Distribuição entre registros de vivos (negativo = 0) e mortos (positivo = 1)
77 print ('Total de registros negativos: ', df[df['is_neonatal_death'] == 0 ].shape[0])
78 print ('Total de registros positivos: ', df[df['is_neonatal_death'] == 1 ].shape[0])
79
80 """# 2. Aplicação de modelos de Machine Learning: Decision Tree
81
82 Precisamos converter o Dataframe para um Array Numpy, que é o tipo de
dados que iremos usar no treinamento.
83
84 Um com as features de entrada, e outro com os labels (etiquetas,
rótulos do registro).
85 """
86
87 # "Features" do modelo
88 nome_features = ['maternal_age', 'tp_maternal_schooling', 'tp_marital_status',
89                   'tp_maternal_race', 'num_live_births',
90                   'num_fetal_losses',
91                   'num_previous_gestations', 'num_normal_labors',
92                   'num Cesarean_labor',
93                   'tp_pregnancy', 'newborn_weight', 'gestational_week',
94                   'cd_apgar1', 'cd_apgar5', 'has_congenital_malformation',
95                   'tp_newborn_presentation', 'num_prenatal_appointments',
96                   'tp_labor',
97                   'was_cesarean_before_labor', 'was_labor_induced',
98                   'tp_childbirth_care',
99                   'tp_robson_group']
100
101 target = ['is_neonatal_death']
102
103 # Array de features
104 X = df[nome_features].values
105 Y = df[target].values
106
107 """## Criando modelo usando divisão da base em 90%/10% para treino
e teste."""
108
109 # "Split" do dataset para Treino e Teste do modelo
110 # Usando treino com 90% dos dados e 10% dos dados para teste
111 xTreino, xTeste, yTreino, yTeste = train_test_split(X, Y,
112                                                       test_size=0.1,
113                                                       shuffle=False,
114                                                       random_state=7)
115
116 # Instanciando algoritmo
117 algoritmo_arvore = DecisionTreeClassifier(criterion='entropy',
118                                             max_depth=5)
119
120 # "Treinamento"/Construção do modelo
121 modelo_90_10 = algoritmo_arvore.fit(xTreino, yTreino)
122
123 """## Exibindo a árvore que o modelo montou
124
125 A árvore de decisão pode ser considerada um modelo White Box, ou seja

```

```

File - /Users/carlosbeluzo/Downloads/tcc_caiomota_arvoredecisao_brneodeath_2016_2018_final.py
121 , um modelo que podemos entender melhor o que ele aprendeu e como ele
122 decide. Podemos mostrar a árvore para isso.
123 """
124 importances = modelo_90_10.feature_importances_
125 indices = np.argsort(importances)[::-1]
126 print("Importância de cada variável para o modelo:")
127
128 for f in range(X.shape[1]):
129     print("%d. feature %d (%f)" % (f + 1, indices[f], importances[
130     indices[f]]))
130 f, ax = plt.subplots(figsize=(10, 5))
131 plt.title("Importância de cada variável para o modelo", fontsize = 14)
132 plt.bar(range(X.shape[1]), importances[indices],
133         color="b",
134         align="center")
135 plt.xticks(range(X.shape[1]), indices)
136 plt.xlim([-1, X.shape[1]])
137 plt.ylabel("Importância", fontsize = 12)
138 plt.xlabel("Features (índice)", fontsize = 12)
139
140 plt.savefig("importance_features_tree.png", dpi=300)
141
142 plt.show()
143
144 #Indice das features
145 # 1 - 'tp_birth_place,
146 # 2 - 'maternal_age'
147 # 3 - 'tp_marital_status'
148 # 4 - 'tp_maternal_education_years'
149 # 5 - 'num_live_births'
150 # 6 - 'num_fetal_losses'
151 # 7 - 'tp_pregnancy_duration'
152 # 8 - 'tp_pregnancy'
153 # 9 - 'tp_labor'
154 # 10 - 'tp_prenatal_appointments'
155 # 11 - 'cd_apgar1'
156 # 12 - 'cd_apgar5'
157 # 13 - 'newborn_weight'
158 # 14 - 'has_congenital_malformation'
159 # 15 - 'tp_maternal_skin_color'
160 # 16 - 'num_gestations'
161 # 17 - 'num_normal_labors'
162 # 18 - 'num Cesarean_labors'
163 # 19 - 'num_gestational_weeks'
164 # 20 - 'tp_presentation_newborn'
165 # 21 - 'tp_childbirth_assistance'
166 # 22 - 'tp_fill_form_responsible'
167 # 23 - 'cd_robson_group'
168
169 # Montando imagem da árvore de decisão
170 dot_data = StringIO()
171 export_graphviz(modelo_90_10, out_file=dot_data, filled=True,
172                  feature_names=nome_features,
173                  class_names=['Vivo', 'Morto'], rounded=True,
174                  special_characters=True,
175                  node_ids=True, proportion=True, max_depth=5, rotate=True)
176
177 graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
178 Image(graph.create_png())
179 graph.write_png("arvore.png")
180 Image('arvore.png')
181
182 # Montando imagem da árvore de decisão
183 dot_data = StringIO()
184 export_graphviz(modelo_90_10, out_file=dot_data, filled=True,
185                  feature_names=nome_features,
186                  class_names=['Vivo', 'Morto'], rounded=True,
187                  special_characters=True),

```

```

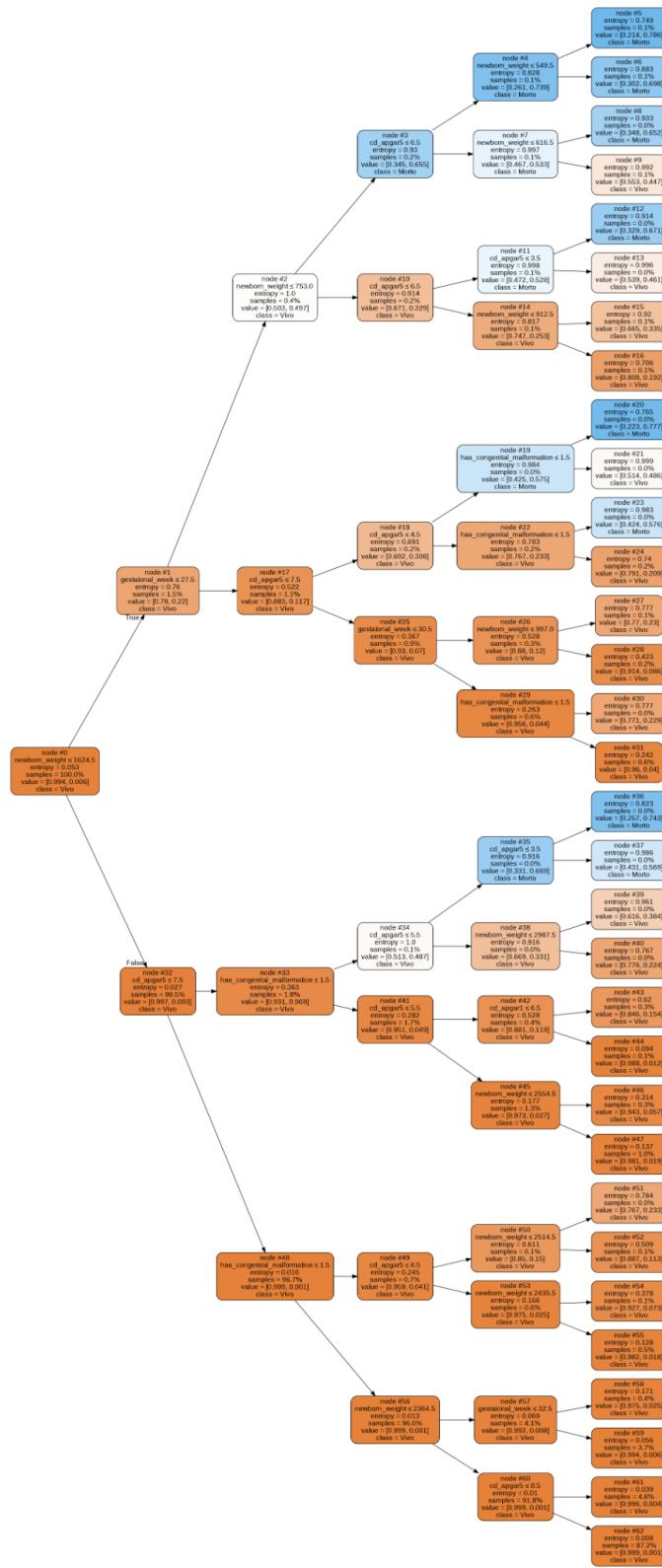
File - /Users/carlosbelozo/Downloads/tcc_caiomota_arvoredecisao_brneodeath_2016_2018_final.py
184             node_ids=True, proportion=True, max_depth=2)
185
186 graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
187 Image(graph.create_png())
188 graph.write_png("arvore_short.png")
189 Image('arvore_short.png')
190
191 """### Curva ROC
192 """
193 """
194
195 y_pred = modelo_90_10.predict(xTeste)
196 y_pred_proba = modelo_90_10.predict_proba(xTeste)[:, 1]
197 [fpr, tpr, thr] = roc_curve(yTeste, y_pred_proba)
198
199 print('Treino/Teste resultados para %s:' % modelo_90_10.__class__.
200       __name__)
200 print("Accuracy = %.3f" % accuracy_score(yTeste, y_pred))
201 print("Log_loss = %.3f" % log_loss(yTeste, y_pred_proba))
202 print("AUC = %.3f" % auc(fpr, tpr))
203
204 idx = np.min(np.where(tpr > 0.95))
205
206 #plt.figure(figsize=(13, 13))
207 plt.plot(fpr, tpr, color='coral', label='Curva ROC (area = %.3f)' %
208           auc(fpr, tpr))
208 plt.plot([0, 1], [0, 1], 'k--')
209 plt.plot([0, fpr[idx]], [tpr[idx], tpr[idx]], 'k--', color='blue')
210 plt.plot([fpr[idx], fpr[idx]], [0, tpr[idx]], 'k--', color='blue')
211 plt.xlim([0.0, 1.0])
212 plt.ylim([0.0, 1.05])
213 plt.xlabel('Taxa de falso positivo (1 - especificidade)', fontsize=14)
214 plt.ylabel('Taxa de verdadeiro positivo (recall)', fontsize=14)
215 plt.title('Curva ROC')
216 plt.legend(loc="lower right")
217
218 plt.savefig("ROC.png", dpi=300)
219 print("\n*Nota: Usando um limite de %.3f " % thr[idx] +
220       "garante uma sensibilidade de %.3f " % tpr[idx] +
221       "\ne uma especificidade de %.3f" % (1-fpr[idx]) +
222       ", ou seja, uma taxa de falsos positivos de %.2f%%.\n" % (np.
223         array(fpr[idx])*100))
223
224 plt.show()
225
226 """Métricas do modelo usando divisão 90/10
227
228 - PRECISÃO: DAS CLASSIFICAÇÕES QUE O MODELO FEZ PARA UMA DETERMINADA
229   CLASSE
230 - RECALL: DOS POSSÍVEIS DATAPOINTS PERTECENTES A UMA DETERMINADA
231   CLASSE
232 """
233
233 Y_predicoes = modelo_90_10.predict(xTeste)
234
235 print("ACURÁCIA DA ÁRVORE: ", accuracy_score(yTeste, Y_predicoes))
236 print(classification_report(yTeste, Y_predicoes))
237
238 """ # 3. Criando modelo usando validação cruzada"""
239
240 # k = 10 na divisão da base para treino e teste
241 algortimo_arvore = DecisionTreeClassifier(criterion='entropy',
242                                           max_depth=3)
242 scoring = {'accuracy': 'accuracy', 'log_loss': 'neg_log_loss', 'auc':
243             'roc_auc'}
243
244 modelo2 = cross_validate(algortimo_arvore, X, Y, cv=10,
245                           scoring=list(scoring.values()),
246                           return_train_score=False)

```

```
File - /Users/carlosbeluzo/Downloads/tcc_caiomota_arvoredecisao_brneodeath_2016_2018_final.py
247
248 print('Resultado usando K-fold = 10 no cross-validation:\n')
249 for sc in range(len(scoring)):
250     print("%s: %.3f (+/-%.3f)" % (list(scoring.keys())[sc], -modelo2[
251         'test_%s' % list(scoring.values())[sc]].mean()
251         if list(scoring.values())[sc]=='neg_log_loss'
252         else modelo2['test_%s' % list(scoring.
253             values())[sc]].mean(),
253             values())[sc]).std()))
254 """
255 """Matriz de confusão
256 """
257 """
258
259 algortimo_arvore = DecisionTreeClassifier(criterion='entropy',
260 max_depth=3)
260 modelo2_predicts = cross_val_predict(algortimo_arvore, X, Y, cv=10)
261 cm_modelo2 = confusion_matrix(Y, modelo2_predicts)
262 print(cm_modelo2)
263
264 # generate report
265 print(classification_report(Y, modelo2_predicts))
```

APÊNDICE I

Para uma melhor visualização a imagem esta disponível em
https://github.com/CaioSMota/TCC_CaioMota_09-06-2021_TADS_ModalidadeNeonatal/blob/main/ArvoreCompleta.png



APÊNDICE J

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py

```

1 # -*- coding: utf-8 -*-
2 """TCC_CaioMota_RegLogic_BRNeoDeath_2016_2018_FINAL.
3 ipynb
4
5 Automatically generated by Colaboratory.
6
7 Original file is located at
8     https://colab.research.google.com/drive/
9         1J4dedmeN3DC5sL0vvJ0hBmaJu0xch0c3
10
11
12 **Implementação de modelo utilizando algoritmo de
13 Regressão Logística**
14 <br>
15 <br>
16 **Autor**: Caio Augusto de Souza Mota (*caiomota802@
17 gmail.com*)
18 <br>
19 <br>
20 *Código adaptado de Baligh Mnassri disponível em:
21     https://www.kaggle.com/mnassrib/titanic-logistic-
22 regression-with-python/notebook*
23
24 ---
```

22 Este código é parte do Trabalho de Conclusão de Curso apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo Câmpus Campinas.

23

24 # 1. Importação de Bibliotecas, carga de dados e funções

25 """

26

27 # instalando o Synapse Client

28 ! pip install synapseclient

```
File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py
29
30 import os
31 import synapseclient as syna
32 from getpass import getpass
33
34 import numpy as np
35 from math import sqrt
36 import pandas as pd
37
38 from sklearn.linear_model import LogisticRegression
39 from sklearn.model_selection import train_test_split
    , cross_val_score, cross_validate, cross_val_predict
    , GridSearchCV, StratifiedKFold
40 from sklearn.feature_selection import RFE, RFECV
41 from sklearn.metrics import roc_curve, plot_roc_curve
    , accuracy_score, auc, log_loss, confusion_matrix,
    classification_report
42
43 import matplotlib.pyplot as plt
44 plt.rc("font", size=14)
45
46 import seaborn as sns
47 sns.set(style="white") #white background style for
    seaborn plots
48 sns.set(style="whitegrid", color_codes=True)
49
50 import warnings
51 warnings.simplefilter(action='ignore')
52
53 """#### Funções auxiliares"""
54
55 # Matrix de Confusão
56 def my_cm(p_cm, p_acc):
57     plt.figure(figsize=(3,3))
58     sns.heatmap(p_cm, annot=True, fmt=".0f", linewidths=.9, square=True, cmap='Blues_r')
59     plt.ylabel('Valores reais')
60     plt.xlabel('Valores preditos')
61     plt.title('Acurácia: %2.3f' % p_acc, size=12)
62
63 """## 1.1 Carregando base de dados disponível no
```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py

```

63 Synapse"""
64
65 # BRNeodeath
66 # Recuperando a base de dados do repositório de
67 # dados Synapse
67 syn = syna.Synapse()
68 syn.login(input('Sybapse User: '), getpass('Passwd:'))
69
70 # Obtendo um ponteiro e baixando os dados
71 dataset = syn.get(entity='syn25575811') # ID do
72 # dataset BRNeodeath
73 df_ori = pd.read_csv(dataset.path)
74
75 """## 1.2 Divisão da base em conjunto para treino e
76 # teste do modelo de ML"""
76
77 df = df_ori.sample(frac=1)
78 print("Shape:", df.shape, "\n")
79 df.head()
80
81 # Todas as "Features" da base de dados
82 features = ['maternal_age', 'tp_maternal_schooling',
83               , 'tp_marital_status',
84               'tp_maternal_race', 'num_live_births',
85               , 'num_fetal_losses',
86               'num_previous_gestations',
87               'num_normal_labors', 'num Cesarean_labor',
88               'tp_pregnancy', 'newborn_weight',
89               'gestational_week',
90               'cd_apgar1', 'cd_apgar5',
91               'has_congenital_malformation',
92               'tp_newborn_presentation',
93               'num_prenatal_appointments', 'tp_labor',
94               'was_Cesarean_before_labor',
95               'was_labor_induced', 'tp_childbirth_care',
96               'tp_robson_group']

91 target = ['is_neonatal_death']
92

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py

```

93 # Separação das features e do "target" para serem
   usados no modelo posteriormente
94 xArray = df[features]
95 yArray = df[target]
96
97 # "Split" do dataset para Treino e Teste do modelo
   usando 90% para o treino e 10% para teste
98 xTreino, xTeste, yTreino, yTeste = train_test_split(
   xArray, yArray, test_size=0.1 ,random_state=42)
99
100 print(xTreino.shape)
101 print(xTeste.shape)
102 print(yTreino.shape)
103 print(yTeste.shape)
104
105 """# 2. Aplicação de modelos de Machine Learning:
   Logistic Regression
106
107 ---
108
109 ## Usando particionamento 90/10
110 """
111
112 logreg = LogisticRegression()
113
114 logreg.fit(xTreino, yTreino)
115 y_pred = logreg.predict(xTeste)
116 y_pred_proba = logreg.predict_proba(xTeste)[:, 1]
117
118 [fpr, tpr, thr] = roc_curve(yTeste, y_pred_proba)
119
120 print('Treino/Teste resultados divididos %s:' %
   logreg.__class__.__name__)
121 print("Accuracy is %2.3f" % accuracy_score(yTeste,
   y_pred))
122 print("Log_loss is %2.3f" % log_loss(yTeste,
   y_pred_proba))
123 print("AUC is %2.3f" % auc(fpr, tpr))
124
125 """#### Curva ROC"""
126

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py

```

127 idx = np.min(np.where(tpr > 0.95))
128
129 plt.figure()
130 plt.plot(fpr, tpr, color='coral', label='Curva ROC ( area = %0.3f)' % auc(fpr, tpr))
131 plt.plot([0, 1], [0, 1], 'k--')
132 plt.plot([0,fpr[idx]], [tpr[idx],tpr[idx]], 'k--',
           color='blue')
133 plt.plot([fpr[idx],fpr[idx]], [0,tpr[idx]], 'k--',
           color='blue')
134 plt.xlim([0.0, 1.0])
135 plt.ylim([0.0, 1.05])
136 plt.xlabel('Taxa de falso positivo (1 - especificidade)', fontsize=14)
137 plt.ylabel('Taxa de verdadeiro positivo (recall)', fontsize=14)
138 plt.title('Curva de característica de operação do receptor (ROC)')
139 plt.legend(loc="lower right")
140 plt.show()
141
142 plt.savefig('ROC_90_10.png', dpi=300)
143
144 print("Usando um limite de %.3f " % thr[idx] + " garante uma sensibilidade de %.3f " % tpr[idx] +
145       "e uma especificidade de %.3f" % (1-fpr[idx]
146       ]) +
147       ", ou seja, uma taxa de falsos positivos de %.2f%%." % (np.array(fpr[idx])*100))
148 """#### Matrix de Confusão"""
149
150 acc = accuracy_score(yTeste, y_pred)
151 my_cm(confusion_matrix(yTeste, y_pred), acc)
152 plt.savefig('CM_90_10.png', dpi=300)
153
154 """#### Relatório de Classificação"""
155
156 print(classification_report(yTeste, y_pred))
157 report = classification_report(yTeste, y_pred,
           output_dict=True)

```

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py
158 round(pd.DataFrame(report).transpose(),2).to_csv('
    ClassRep_90_10.csv')
159
160 """## Cross validation com k = 10
161
162 **No Trabalho incluir apenas o relatório de
    classificação para comparar com o resultado sem
    cross-validation, não precisar incluir matrix de
    conf.**
163 """
164
165 # Regressão logística de validação cruzada de 10
    vezes
166 logreg = LogisticRegression()
167 scoring = {'accuracy': 'accuracy', 'log_loss': '
    neg_log_loss', 'auc': 'roc_auc'}
168
169 modelo_KFold = cross_validate(logreg, xArray, yArray
    , cv=10,
170                                     scoring=list(scoring.values
    ()),
171                                     return_train_score=False)
172
173 print('K-fold cross-validation results:')
174 for sc in range(len(scoring)):
175     print("%s: %.3f (+/-.3f)" % (list(scoring.keys
    ())[sc], -modelo_KFold['test_%s' % list(scoring.
        values())[sc]].mean()
176                                     if list(scoring.
        values())[sc]=='neg_log_loss'
177                                     else modelo_KFold['
        test_%s' % list(scoring.values())[sc]].mean(),
178                                     modelo_KFold['test_%s
        ' % list(scoring.values())[sc]].std()))
179
180 """#### CURVA ROC para CROSS VALIDATION"""
181
182 #https://scikit-learn.org/stable/auto_examples/
    model_selection/plot_roc_crossval.html
183 X = xArray.copy()
184 y = yArray.copy()

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py

```

185
186 X.reset_index(inplace=True)
187 X.drop(columns={'index'}, inplace=True)
188
189 y.reset_index(inplace=True)
190 y.drop(columns={'index'}, inplace=True)
191
192 n_samples, n_features = X.shape
193
194 # ##########
195 # Classification and ROC analysis
196
197 # Run classifier with cross-validation and plot ROC
198 # curves
199 cv = StratifiedKFold(n_splits=10)
200 classifier = LogisticRegression()
201
202 tprs = []
203 aucs = []
204 mean_fpr = np.linspace(0, 1, 100)
205
206 fig, ax = plt.subplots()
207 fig.set_figwidth(10)
208 fig.set_figheight(10)
209
210 for i, (train, test) in enumerate(cv.split(X, y)):
211     classifier.fit(X[X.index.isin(train)], y[y.index
212 .isin(train)])
213     viz = plot_roc_curve(classifier, X[X.index.isin(
214 test)], y[y.index.isin(test)],
215                         name='ROC fold {}'.format(i
216 ),
217                         alpha=0.3, lw=1, ax=ax)
218     interp_tpr = np.interp(mean_fpr, viz.fpr, viz.
219 tpr)
220     interp_tpr[0] = 0.0
221     tprs.append(interp_tpr)
222     aucs.append(viz.roc_auc)
223
224 ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color=

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py

```

219 'r',
220         label='Chance', alpha=.8)
221
222 mean_tpr = np.mean(tprs, axis=0)
223 mean_tpr[-1] = 1.0
224 mean_auc = auc(mean_fpr, mean_tpr)
225 std_auc = np.std(aucs)
226 ax.plot(mean_fpr, mean_tpr, color='b',
227         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)'
228             % (mean_auc, std_auc),
229             lw=2, alpha=.8)
230
231 std_tpr = np.std(tprs, axis=0)
232 tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
233 tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
234 ax.fill_between(mean_fpr, tprs_lower, tprs_upper,
235                 color='grey', alpha=.2,
236                 label=r'$\pm$ 1 std. dev.')
237
238 ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],
239         title="Receiver operating characteristic
240             example")
241 ax.legend(loc="lower right")
242 plt.show()
243
244 """
245 y_pred = cross_val_predict(logreg, xArray, yArray,
246     cv=10)
247 acc = modelo_KFold['test_accuracy'].mean()
248 my_cm(confusion_matrix(yArray, y_pred), acc)
249 plt.savefig('CM_KFold.png', dpi=300)
250
251 """
252 print(classification_report(yArray, y_pred))
253 report = classification_report(yArray, y_pred,
254     output_dict=True)
255 round(pd.DataFrame(report).transpose(),2).to_csv('

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py

```
254 ClassRep_KFold.csv')
255
256 """## RFE (Eliminação recursiva de features)
257
258 A ideia da eliminação recursiva de feature (RFE) é
    selecionar features considerando recursivamente
    conjuntos cada vez menores de features, isso ocorre
    da seguinte forma. Primeiro, o estimador é treinado
    no conjunto inicial de features e a importância de
    cada feature é obtida por meio de um atributo "coef_"
    ou por meio de um atributo "feature_importances_".
    Em seguida, os features menos importantes são
    removidos do conjunto atual de features. Esse
    procedimento é repetido recursivamente no conjunto
    removido até que o número desejado de features a
    serem selecionados seja finalmente alcançado.
259
260 ### Criando um modelo com número de features
    otimizado usando RFECV
261
262 RFECV executa RFE em um loop de validação cruzada
    para encontrar o número ideal ou o melhor número de
    features. No código abaixo temos uma eliminação de
    feature recursiva aplicada na regressão logística
    com ajuste automático do número de features
    selecionados com validação cruzada.
263
264 Como resultado o código trouxe 6 features que é o
    numero ideia de features para o modelo e trouxe
    também as melhores features sendo elas:
265
266 - 'tp_maternal_schooling'
267 - 'gestaional_week'
268 - 'cd_apgar1'
269 - 'cd_apgar5'
270 - 'has_congenital_malformation'
271 - 'tp_labor'
272 """
273
274 # Crie o objeto RFE e calcule uma pontuação com
    validação cruzada.
```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py

```

275 # A pontuação de "precisão" é proporcional ao número
    # de classificações corretas
276 rfecv = RFECV(estimator=LogisticRegression(), step=1
    , cv=10, scoring='accuracy')
277 rfecv.fit(xArray, yArray)
278
279 print("Número ideal de Features: %d" % rfecv.
    n_features_)
280 print('Features Selecionadas: %s' % list(xArray.
    columns[rfecv.support_]))
281
282 # Número do lote das Features VS. pontuações de
    # validação cruzada
283 plt.figure(figsize=(7,5))
284 plt.xlabel("Número de Features selecionadas")
285 plt.ylabel("Pontuação de validação cruzada (nb de
    # classificações corretas)")
286 plt.plot(range(1, len(rfecv.grid_scores_) + 1),
    rfecv.grid_scores_)
287 plt.show()
288
289 """## Modelo Final: Usar apenas variáveis
    # selecionadas e aplicar o GridSearchCV
290
291 No RFECV vc esta esclhendo mehiores features de sua
    # base, no gridSearch vc esta escolhendo parametros do
    # algorítmo.
292 """
293
294 xArray_RFE = xArray[list(xArray.columns[rfecv.
    support_])]
295 xArray_RFE.head()
296
297 param_grid = {'C': np.arange(1e-05, 3, 0.1)}
298 scoring = {'Accuracy': 'accuracy', 'AUC': 'roc_auc'
    , 'Log_loss': 'neg_log_loss'}
299
300 gs = GridSearchCV(LogisticRegression(),
    return_train_score=True,
301                 param_grid=param_grid, scoring=
    scoring, cv=10, refit='Accuracy')
```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py

```

302
303 gs.fit(xArray_RFE, yArray)
304 results = gs.cv_results_
305
306 print('='*20)
307 print("Best params: " + str(gs.best_estimator_))
308 print("Best params: " + str(gs.best_params_))
309 print('Best score:', gs.best_score_)
310 print('='*20)
311
312 plt.figure(figsize=(10, 5))
313 plt.title("GridSearchCV evaluating using multiple
scorers simultaneously", fontsize=16)
314
315 plt.xlabel("Inverse of regularization strength: C")
316 plt.ylabel("Score")
317 plt.grid()
318
319 ax = plt.axes()
320 ax.set_xlim(0, param_grid['C'].max())
321 ax.set_ylim(0.0, 1.2)
322
323 X_axis = np.array(results['param_C'].data, dtype=
float)
324
325 for scorer, color in zip(list(scoring.keys()), ['g'
, 'k', 'b']):
326     for sample, style in (('train', '--'), ('test',
'-')):
327         sample_score_mean = -results['mean_%s_%s' %
(sample, scorer)] if scoring[scorer]=='_
neg_log_loss' else results['mean_%s_%s' % (sample,
scorer)]
328         sample_score_std = results['std_%s_%s' % (
sample, scorer)]
329         ax.fill_between(X_axis, sample_score_mean -
sample_score_std,
330                         sample_score_mean +
sample_score_std,
331                         alpha=0.1 if sample == 'test
' else 0, color=color)

```



```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py
359 modelo_Final = cross_validate(logReg_GS, xArray_RFE
, yArray, cv=10,
360                                     scoring=list(scoring.values
()),,
361                                     return_train_score=False)
362
363 print('K-fold cross-validation results:')
364 for sc in range(len(scoring)):
365     print("%s: %.3f (+/-.3f)" % (list(scoring.keys
())[sc], -modelo_Final['test_%s' % list(scoring.
values())[sc]].mean()
366                                     if list(scoring.
values())[sc]=='neg_log_loss'
367                                     else modelo_Final['
test_%s' % list(scoring.values())[sc]].mean(),
368                                     modelo_Final['test_%s
' % list(scoring.values())[sc]].std()))
369
370 """#### Curva ROC"""
371
372 #https://scikit-learn.org/stable/auto_examples/
model_selection/plot_roc_crossval.html
373 X = xArray_RFE.copy()
374 y = yArray.copy()
375
376 X.reset_index(inplace=True)
377 X.drop(columns={'index'}, inplace=True)
378
379 y.reset_index(inplace=True)
380 y.drop(columns={'index'}, inplace=True)
381
382 n_samples, n_features = X.shape
383
384 # #####
#####
385 # Classification and ROC analysis
386
387 # Run classifier with cross-validation and plot ROC
curves
388 cv = StratifiedKFold(n_splits=10)
389 classifier = logReg_GS

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py

```

390
391 tprs = []
392 aucs = []
393 mean_fpr = np.linspace(0, 1, 100)
394
395 fig, ax = plt.subplots()
396 fig.set_figwidth(10)
397 fig.set_figheight(10)
398
399 for i, (train, test) in enumerate(cv.split(X, y)):
400     classifier.fit(X[X.index.isin(train)], y[y.index
        .isin(train)])
401     viz = plot_roc_curve(classifier, X[X.index.isin(
            test)], y[y.index.isin(test)],
402                           name='ROC fold {}'.format(i
        )),
403                           alpha=0.3, lw=1, ax=ax)
404     interp_tpr = np.interp(mean_fpr, viz.fpr, viz.
        tpr)
405     interp_tpr[0] = 0.0
406     tprs.append(interp_tpr)
407     aucs.append(viz.roc_auc)
408
409 ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color=
        'r',
410          label='Chance', alpha=.8)
411
412 mean_tpr = np.mean(tprs, axis=0)
413 mean_tpr[-1] = 1.0
414 mean_auc = auc(mean_fpr, mean_tpr)
415 std_auc = np.std(aucs)
416 ax.plot(mean_fpr, mean_tpr, color='b',
417          label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' %
        (mean_auc, std_auc),
418          lw=2, alpha=.8)
419
420 std_tpr = np.std(tprs, axis=0)
421 tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
422 tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
423 ax.fill_between(mean_fpr, tprs_lower, tprs_upper,
        color='grey', alpha=.2,

```

File - C:\Users\Caio Mota\Desktop\tcc_caiomota_reglogic_brneodeath_2016_2018_final.py

```
424             label=r'$\pm$ 1 std. dev.')
425
426 ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],
427         title="Receiver operating characteristic
428 example")
429 plt.legend(loc="lower right")
430 plt.show()
431 plt.savefig('ROC_KFold_Final.png', dpi=300)
432
433 """#### Matriz de confusão"""
434
435 y_pred = cross_val_predict(logReg_GS, xArray_RFE,
436                             yArray, cv=10)
437 acc = modelo_Final['test_accuracy'].mean()
438 my_cm(confusion_matrix(yArray, y_pred), acc)
439 plt.savefig('CM_Final.png', dpi=300)
440
441 """#### Relatório de Classificação"""
442 print(classification_report(yArray, y_pred))
443 report = classification_report(yArray, y_pred,
444                                 output_dict=True)
445 round(pd.DataFrame(report).transpose(),2).to_csv('
446 ClassRep_Final.csv')
```