escola superior de tecnologia e gestão
instituto politécnico de leiria

Comparação do desempenho de metodologias de classificação sexual baseada em critérios métricos craniomandibulares em ortopantomografias para Identificação Forense

Mestrado em Ciência de Dados

João Ricardo Gregório Alves

Leiria, setembro de 2023

escola superior de tecnologia e gestão
instituto politécnico de leiria

Comparação do desempenho de metodologias de classificação sexual baseada em critérios métricos craniomandibulares em ortopantomografias para Identificação Forense

Mestrado em Ciência de Dados

João Ricardo Gregório Alves

Trabalho de Projeto realizado sob a orientação do Professor Doutor Rui Filipe Vargas de Sousa Santos e da Professora Doutora Cristiana Palmela Pereira

Leiria, setembro de 2023

Originalidade e Direitos de Autor

O presente relatório de projeto é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Agradecimentos

Pretendo com este capítulo agradecer às pessoas que sem as quais e que direta ou indiretamente contribuíram para o desenvolvimento deste projeto.

Primeiramente tenho de agradecer ao meu orientador, o Professor Doutor Rui Filipe Vargas de Sousa Santos não só pela orientação, o incentivo constante, e a valiosa partilha de conhecimentos, mas também pela disponibilidade e o empenho ao longo de todo o projeto. E que sem esta contribuição não teria sido possível desenvolver este projeto.

A minha coorientadora a Professora Doutora Cristiana Palmela Pereira pela partilha de todos os dados utlizados neste projeto, e pelos conhecimentos transmitidos.

Aos colegas de turma e amigos que foram feitos no decorrer do mestrado em ciência de dados, que de certa forma encorajaram e moldaram o percurso do mestrado, em especial ao Zé que foi o meu companheiro dos trabalhos.

Por último, mas não menos importante, aos meus familiares e amigos, em especial ao Pedro pelo apoio e encorajamento necessários para que fosse possível concluir esta jornada acadêmica.

A todos vocês, o meu sincero Obrigado.

Resumo

As estruturas ósseas craniomandibulares, por serem mais resistentes aos processos de tafonomia, são relevantes na diagnose sexual de esqueletos adultos. Este passo é primordial na vertente reconstrutiva de um cadáver não identificado. Assim, com base numa amostra obtida por estudantes da Faculdade de Medicina Dentária da Universidade de Lisboa e através de um conjunto de medições efetuadas em ortopantomografias (radiografias panorâmicas), neste trabalho é avaliado o desempenho de diferentes metodologias de classificação do sexo. Algumas das metodologias avaliadas são baseadas nas medições realizadas, como a regressão logística, a analise discriminante, os k-vizinhos mais próximos, entre outras. É igualmente avaliada a aplicação de redes neuronais pré-treinadas, como a VGG16, a RESNET-50 e a INCEPTION V-3, que concretizam a classificação diretamente das ortopantomografias. A amostra utilizada foi aleatoriamente dividida em 80 por cento para a estimação dos parâmetros de cada metodologia (treino) e as restantes 20 por cento para avaliação do desempenho (teste). A comparação do desempenho foi baseada na matriz de confusão e medidas associadas (acurácia, sensibilidade, especificidade, valores preditivos e F-score) e na área sob a curva ROC.

Abstract

Craniomandibular bone structures, as they are more resistant to taphonomy processes, are relevant in the sexual diagnosis of adult skeletons. This step is essential in the reconstruction of an unidentified corpse. Thus, based on a sample obtained by students of the Faculty of Dental Medicine of the University of Lisbon through a set of measurements carried out in orthopantomography (panoramic radiographs), this work assesses the performance of different gender classification methodologies. Some of the evaluated methodologies are based on measurements taken, such as logistic regression, discriminant analysis, k-nearest neighbors, among others. The application of pre-trained neural networks, such as the VGG16, the RESNET-50 and the INCEPTION V-3, which perform the classification directly from the orthopantomography's, is also evaluated. The sample used was randomly divided into 80% for estimating the parameters of each methodology (training) and the remaining 20% for performance evaluation (test). Performance comparison was based on the confusion matrix and associated measures (accuracy, sensitivity, specificity, predictive values and F-score) and on the area under the ROC curve.

Índice

Lista de Figuras

Lista de Tabelas

Lista de siglas e acrónimos

AUC	Área abaixo da curva
CNN	Rede Neuronal Convolucional
ESP	Especificidade
FMDUL	Faculdade de Medicina Dentária da Universidade de Lisboa
FN	Falso Negativo
FP	Falso Positivo
KNN	K-Vizinhos mais próximos
LDA	Análise Discriminante Linear
ML	Machine Learning
OPG	Ortopantomografias
QDA	Análise Discriminante Quadrática
ROC	Característica de Operação do Recetor
SENS	Sensibilidade
SVM	Máquinas de Vetores de Suporte
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo
VPN	Valor Preditivo Negativo
VPP	Valor Preditivo Positivo

1. Introdução

Em acidentes e catástrofes com um elevado número de vítimas mortais, a identificação das vítimas é primordial. Quando os cadáveres estão esqueletizados a diagnose sexual é o primeiro passo na vertente reconstrutiva do cadáver, seguido da estimativa da idade e da altura, a estimação da idade e da altura em ambos os casos são afetados pelo sexo. [1]

Vários estudos foram feitos sobre a diagnose sexual em esqueletos utilizando diversos ossos, tais como o osso coxal, crânio e mandíbula [2]. Se o crânio não estiver disponível ou estiver gravemente danificado, a mandíbula pode ser utilizada para a diagnose sexual uma vez que apresenta diferenças entre homens e mulheres [3].

A radiografia panorâmica foi introduzida em 1950 e é, desde então, um exame complementar imagiológico usado como rotina clínica. A radiografia panorâmica é uma radiografia extra-oral que permite a visualização da mandíbula, maxila, articulação temporomandíbular e estruturas relacionadas apenas numa radiografia. Foram propostos vários índices morfométricos mandibulares para a diagnose sexual, tais como a largura bicondilar, a distância intercoronóide e algumas outras distâncias verticais e horizontais medidas através das radiografias panorâmicas [4–6]. Atualmente a inteligência artificial é aplicada nas mais diversas áreas, especialmente na área da saúde [7], segurança informática [8], reconhecimento de linguagem [9], condução autónoma [10], bem como em muitos outros contextos. No entanto, o aumento na quantidade de dados que são registados sobre os pacientes, acompanhado pela evolução das técnicas de diagnostico, como é o exemplo das radiografias panorâmicas, fez emergir novas possibilidades na utilização de técnicas de aprendizagem automática (*machine learning*) com os dados obtidos. As áreas onde a aplicação dessas técnicas tem maior impacto é no diagnostico, na deteção e na previsão de patologias.[11–13]

O objetivo deste projeto consiste na comparação do desempenho de metodologias de classificação sexual baseada em critérios métricos craniomandibulares em ortopantomografias utilizados atualmente na identificação forense, com o desempenho de

modelos de regressão logística, análise discriminante, algoritmos de classificação e da aplicação de redes neuronais convolucionais (CNN) às radiografias panorâmicas.

Este projeto está dividido em duas partes. A primeira parte tem como objetivo a criação de modelos de regressão logística, análise discriminante e algoritmos de classificação com recurso a uma base de dados com as medidas lineares obtidas através da medição de 14 pontos. Na segunda parte do trabalho foi utilizado um conjunto de radiografias panorâmicas (OPG) na classificação de imagens, i.e., para a criação, treino e validação de algumas redes neuronais convolucionais profundas e, desta forma, tentar validar uma metodologia para a classificação do sexo através da aplicação de redes CNN.

As medidas lineares e as imagens foram recolhidas no âmbito da dissertação de mestrado [14] realizado na Faculdade de Medicina Dentária da Universidade de Lisboa (FMDUL). Os dados são o resultado da medição de 14 pontos na mandíbula de 206 indivíduos portugueses de ambos os sexos. Na referida dissertação, e através da análise descritiva e na aplicação de regressão logística, foi concluído que algumas medidas lineares podem ser utilizadas para a obtenção de uma classificação do sexo com uma acurácia de aproximadamente 83%. Recorrendo a estes dados na Secção 3.3, aplicamos modelos de regressão logística, análise discriminante e outros algoritmos de classificação da aprendizagem automática supervisionada com recurso ao software R em ambiente RStudio. Na Secção 3.2 foram utilizadas 195 imagens de radiografias panorâmicas (OPG) para treinar, validar e testar três redes neuronais convolucionais profundas para a classificação das imagens com recurso à linguagem de programação *Python* em ambiente *notebook* na plataforma *Kaggle*, e com recurso a algumas bibliotecas tais como o *Tensorflow* e *Keras*.

Nos Capítulos 4 e 5 apresentamos, discutimos e comparamos os valores obtidos nos modelos criados nas Secções 3.2 e 3.3 com as metodologias utilizadas atualmente. Deste modo, pretendemos demonstrar que a aplicação de técnicas de aprendizagem automática e inteligência artificial na classificação sexual para a identificação forense pode ser utilizada com fiabilidade.

2. Revisão Bibliográfica

Após análise de diversas publicações relacionadas com o dimorfismo sexual em indivíduos, é possível constatar que as primeiras referências de que a mandíbula possui características diferenciadoras do sexo são de 1946 e foram feitas por Earnest Albert Hooton [15]. Nas décadas seguintes, vários trabalhos surgem utilizando esse princípio, como por exemplo Krogman [16] e Giles [17]. Sendo as redes neuronais um tema mais recente, é possível encontrar trabalhos referenciando a utilização de redes neuronais para análise e diagnostico a partir da década de 80 [18,19]. Vários artigos propõem a classificação utilizando radiografias panorâmicas, no entanto existe especial foco na estimação da idade do individuo deixando a classificação do sexo para segundo plano. De seguida, são apresentados por ordem cronológica (inversa) trabalhos publicados recentemente sobre ambos os temas abordados neste projeto.

2.1. Estudos relacionados com a aplicação de modelos estatísticos para a classificação do sexo (classificação via medições)

Em 2021, Victoria Ionel [14] na dissertação de mestrado sobre a diagnose sexual baseada em parâmetros radiológicos craniomandibulares, utilizou a regressão logística para identificar as variáveis com maior significância estatística na diagnose sexual. Foram utilizadas 206 ortopantomografias de indivíduos com idade superior a 25 anos, de onde foram realizadas 14 medições (14 variáveis). Com recurso às variáveis com maior significância estatística foi obtida uma acurácia de 83% na classificação por sexo.

No mesmo ano, Coelho *et al*. [20] propõem a caracterização de variações biológicas como a idade e o sexo de indivíduos. A análise foi efetuada em imagens 3D de 215 mandíbulas, nas quais foram utilizadas um total de 13 pontos cefalométricos. O objetivo deste estudo foi verificar a influência do sexo e da idade sobre as variáveis, bem como determinar o seu valor preditivo. Para tal, foram aplicados o teste ANOVA (análise da variância) e a regressão logística. Neste estudo, na análise de todas as variáveis, foi obtida uma sensibilidade de 67.8% no modelo para a classificação do sexo.

Em 2016, Vandakara *et al*. [21] propõem a determinação do dimorfismo sexual em humanos através das medidas lineares obtidas nas radiografias panorâmicas. No estudo foi utlizada análise discriminante linear com o objetivo de determinar quais das variáveis tem maior capacidade discriminante do sexo. Tendo sido consideradas todas as variáveis, foi obtida uma acurácia de 79.5% para o lado direito e 77% para o lado esquerdo. Por conseguinte, concluíram que as medidas mandibulares podem ser utilizadas para a diagnose do sexo.

Ainda no ano de 2016, Damera *et. al*. [22] consideram que a mandíbula pode ser uma ferramenta para a classificação do sexo. Tendo em conta a metodologia utilizada e os resultados obtidos, conclui que as medições do ramo mandibular em ortopantomografias podem ser utilizadas como parâmetro fiável para a classificação do sexo. A taxa de previsão obtida (acurácia) para as variáveis estudadas foi de 83.8%.

Por fim, em 2010 Pereira *et al.* [23] demonstram, através da análise discriminante, a contribuição dos dentes para a identificação do sexo. O estudo foi efetuado em 80 moldes, 55 do sexo feminino e 25 do sexo masculino e, através de vários critérios métricos, foram obtidas 12 variáveis. Os autores concluíram que existem alguns dentes com maior relevância estatística para a classificação do sexo.

2.2. Estudos relacionados com a classificação do sexo em ortopantomografias (classificação via imagens médicas)

A classificação com recurso a redes neuronais realizada diretamente nas imagens médicas é uma área recente, na qual o número de publicações tem aumentado significativamente nos últimos anos.

Neste contexto, em 2022, Nithya e Sornam [24] propõem um novo método para a classificação humana de homens e mulheres utilizando imagens panorâmicas de raios X. O modelo proposto é construído usando redes neurais convolucionais profundas sequenciais, para executar a classificação binária. O treino do modelo foi efetuado utilizando imagens panorâmicas de raios X e o modelo conseguiu atingir uma acurácia de 95%.

Igualmente em 2022, Isa Atas [25] compara a utilização de diversas redes convolucionais profundas com aprendizagem por transferência (*deep transfer learning*), que incorporam modelos pré-treinados, para a classificação do sexo utilizando radiografias panorâmicas. A base de dados utilizada consistia em 24.000 imagens preparadas para a classificação binaria. Neste estudo foi obtida uma performance de 97.25% de acurácia utilizando a rede *DenseNet121*.

Em 2021, Vila-Blan *et. al.* [26] propõem a utilização de CNN utilizando radiográficas panorâmicas para a localização de pontos no contorno da mandíbula. O objetivo do estudo é comparar as medidas obtidas através da obtenção dos pontos com a metodologia atual, obtida através da distâncias lineares e ângulos da mandíbula. Desta forma, pretende-se criar um processo completamente automático para a classificação do sexo.

Illic *et al.* [27] em 2018 propõem a utilização e uma rede CNN para a classificação do sexo através de radiografias panorâmicas. Para o treino da rede foi utilizado uma base de dados de 4155 imagens, sendo composta por 60% de imagens do sexo feminino e 40% do sexo masculino. A rede utilizada foi uma rede convolucional profunda VGG16, obtendo, na melhor performance, 94.3% de acurácia utilizando aprendizagem por transferência com a rede pré-treinada.

No mesmo ano, Milošević *et al.* [28] propõem a utilização de redes CNN e radiografias panorâmicas para classificar o sexo. Para tal, utilizaram uma base de dados de 4000 radiografias panorâmicas de pacientes de origem europeia para treinar uma rede convolucional, obtendo uma precisão de 96.87%.

3. Metodologia

Neste capítulo e respetivas secções apresentamos, de forma resumida, as metodologias aplicadas para a realização da classificação sexual baseada em ortopantomografias. Deste modo, inicialmente descrevemos a base de dados utilizada, de seguida os modelos de classificação aplicados, quer os baseados diretamente nas imagens radiográficas quer os baseado nos dados numéricos, e, por fim, são apresentados os critérios de avaliação de desempenho aplicados na comparação dos modelos de classificação.

3.1. Descrição das bases de dados utilizadas

As radiografias panorâmicas utilizadas neste projeto foram recolhidas no Centro Universitário de Radiologia Oro-Maxilo-Facial da FMDUL e encontram-se anonimizadas, para que não haja nenhuma violação de direitos dos indivíduos envolvidos neste projeto. A recolha foi realizada de acordo com as normas éticas definidas pela Comissão de Ética para a Saúde da FMDUL, sob o número 202121. As medidas lineares registadas no ficheiro Excel foram obtidas através das medições dos exames radiográficos dos indivíduos selecionados utilizando critérios de inclusão e exclusão.[14]

A fonte de dados Excel contém 14 variáveis obtidas através da medição da mandíbula nas ortopantomografias [14, 21] e a variável sexo e idade dos indivíduos. Na Tabela 1 estão apresentadas estas variáveis e a respetiva descrição.

Variável	Descrição
Sexo	Sexo do Individuo
Idade	Idade do Individuo
ACD	Altura da coronóidea direita
ACE	Altura da coronóidea esquerda
AMD	Altura da mandíbula direita
AME	Altura da mandíbula esquerda
AQ	Altura do queixo

ARMD	Altura do ramo da mandíbula direita
ARME	Altura do ramo da mandíbula esquerda
LMRMD	Largura mínima do ramo da mandíbula direita
LMRME	Largura mínima do ramo da mandíbula esquerda
DG	Distância entre gónion de ambos os lados
DI1	Distância intercondilar
DI2	Distância intercoronoideia
AGD	Ângulo goníaco direito
AGE	Ângulo goníaco esquerdo

Tabela 1 – Lista das variáveis

Podemos ver na Figura 1 um exemplo das medidas obtidas, através da radiografia panorâmica, e que se encontram registadas na base de dados em Excel.

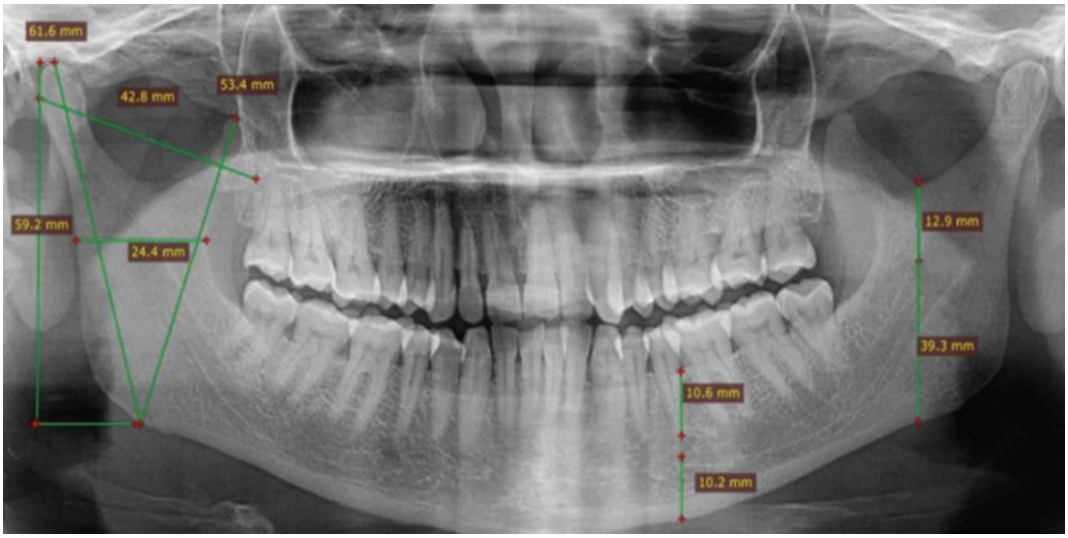

Figura 1 - Descrição das medidas obtidas

Os dados registados são referentes a 206 indivíduos com as idades compreendidas entre os 25 e os 79 anos dos quais 116 são do sexo feminino e 90 do sexo masculino. A distribuição dos dados pode ser verificada na Figura 2.

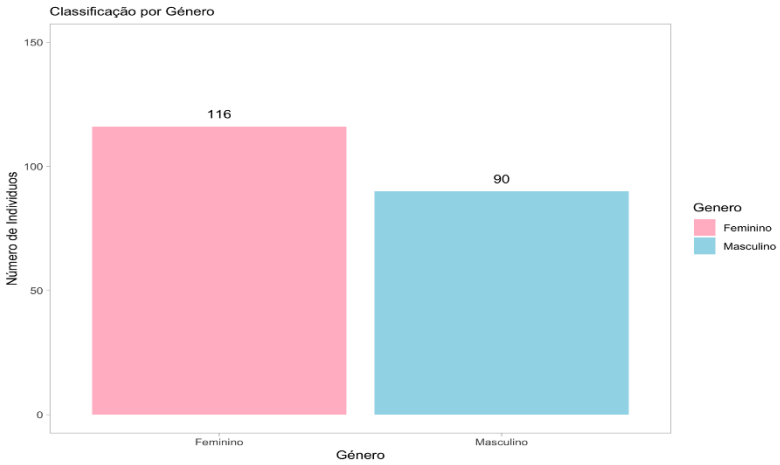

Figura 2 - Distribuição da base de dados por sexo

A base de dados de imagens para o treino das redes neuronais contém 195 radiografias panorâmicas divididas em 108 imagens do sexo feminino e 87 do sexo masculino. As imagens têm uma dimensão de 1201 x 849 pixéis em RGB, por requisito das redes neuronais e, para que seja possível a otimização dos resultados, foi necessário efetuar transformações nas imagens. Na Secção 3.2.2 é descrito o pré-processamento utilizado nas imagens. Exemplos de radiografias panorâmicas são apresentados na Figura 3.

Figura 3 - Radiografias panorâmicas de indivíduos do sexo masculino (esquerda) e do sexo feminino (direita)

Nas imagens podemos detetar a presença de diversas condições, como por exemplo próteses, implantes e aparelhos ortodônticos. No entanto, devido a dimensão da base de dados, essas imagens não foram excluídas. Ambas as fontes de dados não estão balanceadas (igual número de indivíduos de ambos os sexos), porém foram utilizadas técnicas de balanceamento na regressão logística, e *data augmentation* no treino das redes convolucionais, numa tentativa de obtermos os melhores resultados possíveis.

3.2. Modelos de classificação baseados nas imagens radiográficas

3.2.1. Contextualização

Os modelos de classificação baseados em imagens através da aplicação de redes neuronais são um campo da inteligência artificial que têm ganho bastante destaque nos últimos anos, devido aos excelentes resultados e à sua aplicação em diversas áreas, com especial foco na área médica. O desenvolvimento de uma rede convolucional para classificação de imagens envolve um processo de diversas etapas, a escolha de arquitetura, a definição dos parâmetros, o treino da rede e por fim a avaliação. Foi essencial compreender que o tamanho da base de dados disponível para este projeto iria constituir um problema, porque a quantidade de elementos de treino tem uma relação direta com os resultados obtidos pela rede [29]. Depois de uma análise a alguns dos artigos mais recentes e haver uma crescente aplicação de redes CNN profundas para a análise de imagens, e a sua utilização em bases de dados de pequenas dimensões, decidimos utilizar três arquiteturas de redes convolucionais profundas.

As três redes escolhidas para este projeto foram a VGG16, a RESNET-50 e a INCEPTION V-3, não só por serem consideradas das redes mais poderosas para classificação de imagens disponíveis atualmente, mas também por serem das mais utilizadas em alguns dos projetos similares. A popularidade deste género de rede neuronal prende-se em parte pelo facto destas redes poderem ser utilizadas pré-treinadas. Neste sentido, as redes utilizadas foram treinadas utilizado a base de dados da *imageNet* com 1.2 milhões de imagens distribuídas por 10.000 classes. Através de uma técnica, denominada *transfer learning*, é possível aproveitar o conhecimento, os parâmetros e os pesos utilizados anteriormente e que se encontram otimizados para o reconhecimento de características complexas e padrões nas imagens, com o objetivo de obter um melhor desempenho na performance das redes.

Na tentativa de encontrar os parâmetros com melhores resultados foram utilizados os seguintes optimizadores para treinar a rede VGG16:

- *RMSProp* (lr=1e-4) - O algoritmo *RMSProp* utiliza a média móvel dos gradientes quadráticos para adaptar a taxa de aprendizagem de cada parâmetro do modelo. Em vez de usar uma taxa de aprendizagem global fixa, o *RMSProp* ajusta a taxa de

aprendizagem de cada parâmetro com base em estimativas recentes dos gradientes quadráticos acumulados.

- *Adam* (lr=1e-4) - O algoritmo *Adam* utiliza a média móvel dos gradientes e os momentos de ordem superior para atualizar os valores dos parâmetros, recorrendo a duas médias móveis exponenciais: a primeira média móvel (média dos gradientes) e a segunda média móvel (média dos gradientes ao quadrado). Deste modo, estas médias são usadas para calcular as atualizações dos parâmetros.
- *SGD* (lr=1e-4) – O algoritmo *SGD* realiza atualizações dos parâmetros utilizando uma abordagem de descida do gradiente estocástico. Nesse algoritmo, os parâmetros são atualizados em pequenos lotes de amostras de treinamento, em vez de usar o conjunto completo de dados.

E foram testados os optimizadores com as seguintes funções de perda:

- *Binary_crossentropy* – É uma função de perda que mede a diferença entre as probabilidades previstas pelo modelo e os rótulos reais dos exemplos de treino. Quanto menor o valor da perda, melhor o modelo está ajustado aos dados.
- *Focal_loss*() - É uma função de perda desenvolvida para lidar com o problema do desequilíbrio de classes durante a classificação. O objetivo associado à função focal_loss é atribuir uma maior importância à correção das amostras mal classificadas e reduzir a importância das amostras corretamente classificadas. O que ajuda a rede a focar-se nas amostras mais difíceis, que frequentemente são as classes minoritárias. [30]

Para desenvolver este projeto foi utilizada a linguagem de programação *Python*, na plataforma *Kaggle NoteBoo*k. A escolha do *Kaggle* prendeu-se pelo fato desta plataforma permitir a utilização de aceleradores, como o GPU P100, para otimizar o treino das redes neuronais. Nos próximos capítulos será descrito o processo utilizado neste projeto, bem como as etapas utilizadas no pré-processamento, na implementação e no treino das redes convolucionais.

3.2.2. Pré-Processamento das Imagens

O pré-processamento das imagens é uma etapa fundamental no treino de redes convolucionais. As decisões tomadas nesta etapa variam de acordo com o tipo de rede escolhido e com o tipo de imagens a serem utilizadas para o treino da rede convolucional. Esta etapa é considerada fundamental porque diferentes configurações poderão afetar significativamente a performance da rede. Algumas das técnicas mais comuns que podem ser utilizadas são o redimensionamento, normalização, *data augmentation* e filtros. Como foi previamente referido, as configurações utilizadas neste processo são as mesmas para toda a implementação e, após alguns testes, concluímos que o rebordo que as radiografias panorâmicas tinham estavam a prejudicar o treino da rede. Por conseguinte, foi efetuado um *crop* em todas as imagens. Durante o processo de investigação inicial, foram testadas algumas funções na tentativa de obter o *crop* automático das imagens. No entanto, nenhuma das funções produziram o resultado pretendido e, como o tamanho da base de dados não é muito extenso, decidimos aplicar o *crop* manualmente em todas as imagens. Este processo, apesar de moroso, permitiu obter uma normalização da área da imagem.

Por razões técnicas e de performance as imagens foram redimensionadas para 224 x 224, que é a resolução *standard* utilizada pelas redes VGG16 e RESNET-50, não sendo, no entanto, necessário para a INCEPTION-V3. Este processo é utilizado por diversas razões, tais como:

- Eficiência - desta forma as imagens são convertidas para matrizes multidimensionais com dimensões fixas, permitindo assim um processamento mais rápido e a utilização de menos memória.
- Uniformização - ao redimensionar as imagens para 224 x 224 ajuda a garantir que a rede identifica as características da imagem com mais precisão.
- Consistência - como as redes utilizadas são redes pré-treinadas com a base de dados da imageNet que utiliza imagens 224 x 224, isto permite às camadas pré-treinadas obter resultados mais precisos.

Outro método aplicado na implementação foi o *data augmentation*, que foi utilizado quer no processo de treino quer no de validação. A utilização desta técnica permite aumentar a quantidade de imagens através da aplicação de métodos de transformação às imagens

existentes. Assim, tem como objetivo compensar o facto do tamanho e o balanceamento da base de dados não serem o ideal para a implementação de uma rede convolucional, ajudando, desta forma, a aumentar a precisão e a robustez do modelo. Um exemplo ilustrativo pode ser consultado na Figura 4.

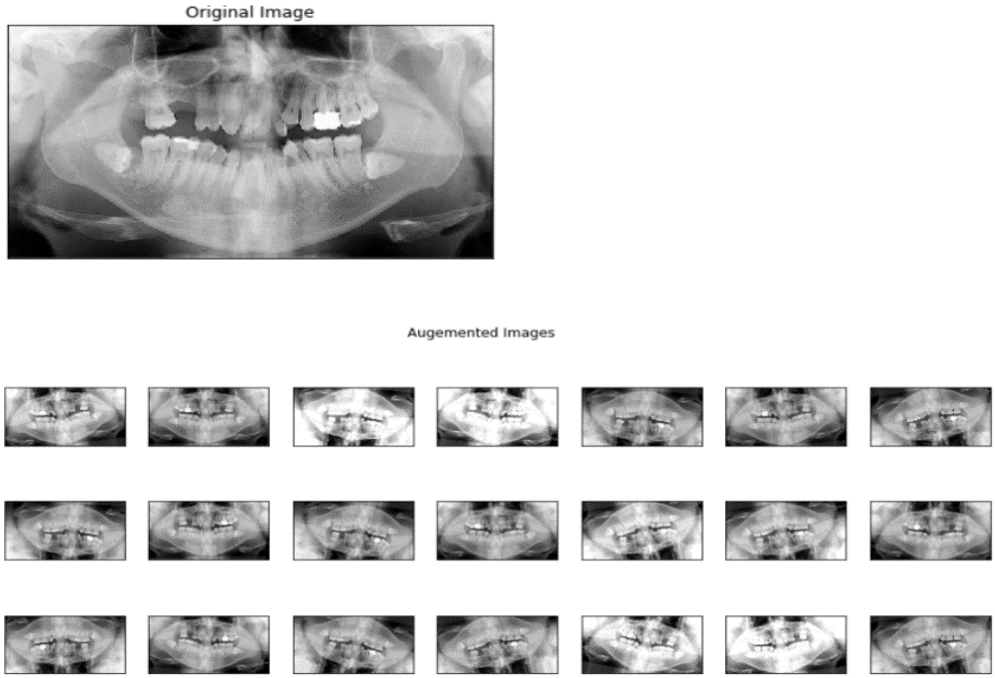

Figura 4 - Imagem com *data augmentation*

Os parâmetros utilizados foram a rotação, mudança de brilho e de escala. Como iremos verificar no Capítulo 4, a utilização de *data augmentation* foi essencial para atingir os resultados obtidos.

Para treinar, validar e testar a rede é necessário ter as imagens divididas, tendo o rácio escolhido para a divisão das 195 imagens sido 80% para o treino e 20 % para a validação. Como a base de dados é pequena, para ter algumas imagens que permitisse testar a classificação da rede, retiramos 9 imagens ao treino e 1 da validação para utilizar no procedimento de teste. Na Figura 5 podemos ver a distribuição da base de dados.

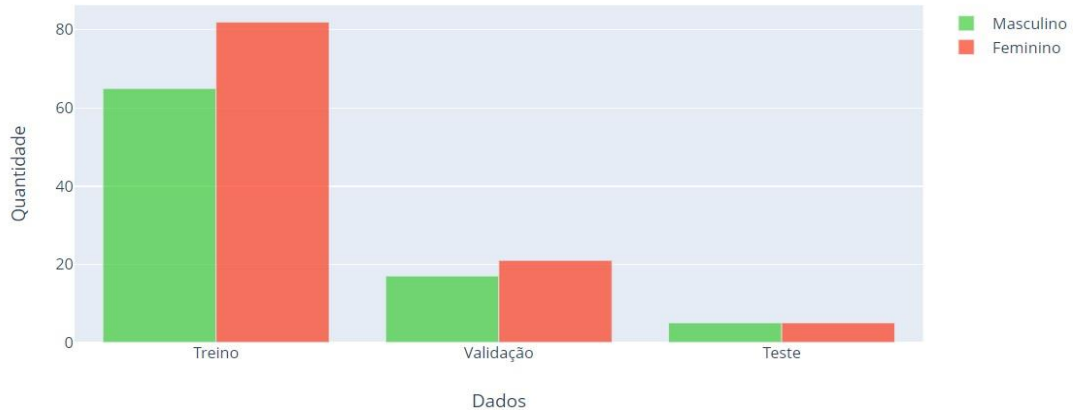

Figura 5 - Distribuição da base de dados

O resultado da distribuição é o seguinte, 82 imagens do sexo feminino e 65 do sexo masculino para o treino, para a validação temos 21 imagens para o sexo feminino e 17 para o sexo masculino e, por fim, para o teste sobram 10 imagens, 5 para cada sexo. Esta distribuição da base de dados foi utilizada nas três arquiteturas.

3.2.3. Implementação e treino das redes neuronais

3.2.3.1. VGG16

A rede VGG16 é um tipo de rede CNN que é considerada, por muitos, o melhor modelo de visão computacional lançado até a data. Esta arquitetura foi apresentada em 2014 pelo Karen Simonyan & Andrew Zisserman [31]. A VGG16 é uma versão melhorada (profunda) da rede VGGNet, e foi uma das primeiras arquiteturas a demonstrar eficácia na utilização de redes profundas no reconhecimento e classificação de imagens.

O 16 em VGG16 refere-se ao número de camadas que contêm parâmetros possíveis de ser afinados (13 camadas convolucionais e 3 camadas totalmente conectadas) no entanto a rede VGG16 é composta por, 13 camadas convolucionais (aplicação de filtros ou *kernels*), 5 camadas *max pooling* (diminuição do tamanho da matriz da imagem utilizando o maior valor de determinada região da matriz para representar toda a região na formação da matriz de saída) e 3 camadas densas ou totalmente conectadas. As camadas convolucionais são formadas por filtros que ao serem aplicados extraem da imagem de entrada algumas características, tais como texturas, rebordos, formas, padrões, etc. Em seguida, as camadas de *max pooling* reduzem o tamanho da imagem para metade, contudo mantendo as características importantes da imagem. Por fim, as camadas densas, que são igualmente denominadas por camadas totalmente conectadas, atuam como um classificador utilizando as características extraídas nas camadas anteriores e produzindo uma saída que é a probabilidade para cada uma das classes (cf. Figura 6. [1]).

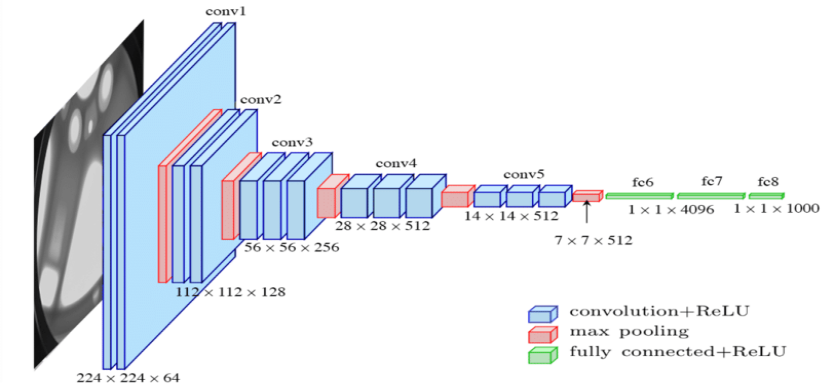

Figura 6 - Arquitetura VGG16

[1] https://www.researchgate.net/figure/VGG16-architecture-16_fig2_321829624

Na implementação da VGG16 utilizamos o conceito de *transfer learning* e, para isso, é necessário "congelar" as primeiras camadas para que a rede utilize o modelo pré-treinado (vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5) como ponto de partida. Desta forma, usamos a última camada convolucional como entrada para a camada totalmente conectada. Por conseguinte, evita que os pesos da rede sejam integralmente ajustados durante o treino.

A rede VGG16 que contém 14739777 parâmetros, porém apenas utiliza 25088 parâmetros para treino. Como é possível ver na Figura 7, o número de camadas utilizadas na rede está reduzido a 4.

- A primeira camada é a camada *base_model* que é a rede neuronal VGG16 pré-treinada, produzindo uma saída de *(None, 7, 7, 512)*, ou seja, foram extraídos 512 parâmetros da imagem de entrada.
- A segunda camada é a camada *flatten* que serve para converter a saída da camada anterior num vetor unidimensional de 25088 parâmetros, necessário para a próxima camada densa.
- A terceira é a camada d*ropout*, com 50% de probabilidade de desativação utilizada como técnica para reduzir *overfitting*.
- A quarta é a camada densa, que é a camada de saída reduzida a uma unidade e com a função de ativação s*igmoid, usual n*a classificação binária.

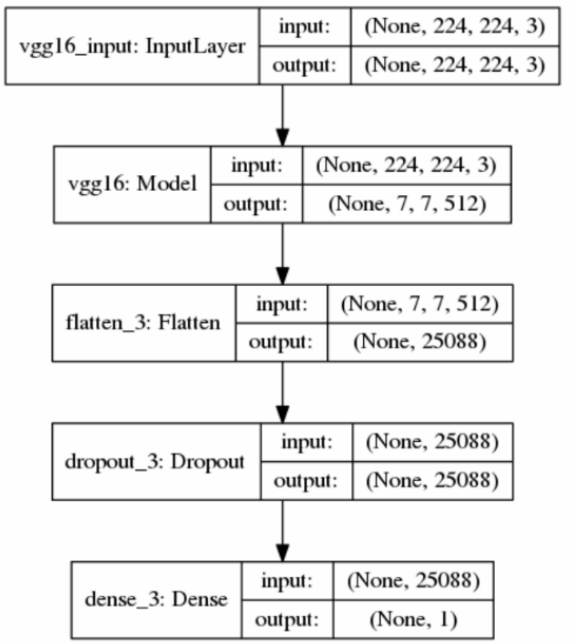

Figura 7 - Arquitetura usada VGG16

3.2.3.2. RESNET-50

À semelhança da rede VGG16, também a RESNET-50 é uma rede convolucional profunda e foi introduzida em 2015 por He *et al.* [32] tornando-se uma das arquiteturas mais utilizadas no reconhecimento de objetos e classificação. É composta por 50 camadas, distribuídas por 1 camada de entrada, 1 camada de saída e 48 blocos residuais. A camada de entrada é uma camada convolucional que recebe uma imagem e extrai as suas características. A RESNET-50 destaca-se pela utilização de blocos residuais para tentar colmatar o problema de degradação de desempenho que acontece nas redes profundas. Estes blocos são compostos por ligações *skip* que permitem a rede comparar a diferença entre a entrada e a saída. Permite, desta forma, que as camadas adicionais aprendam a identificar características na imagem que não foram identificadas nas camadas anteriores (cf. Figura 8). [2]

À semelhança da VGG também a RESNET-50 é uma rede pré-treinada com recurso à base de dados da imageNet. No entanto a Resent-50 tem uma desvantagem em relação aos requisitos computacionais, a rede necessita de elevado poder computacional o que, dependendo do tamanho da base de dados e da resolução das imagens, pode significar um baixo desempenho.

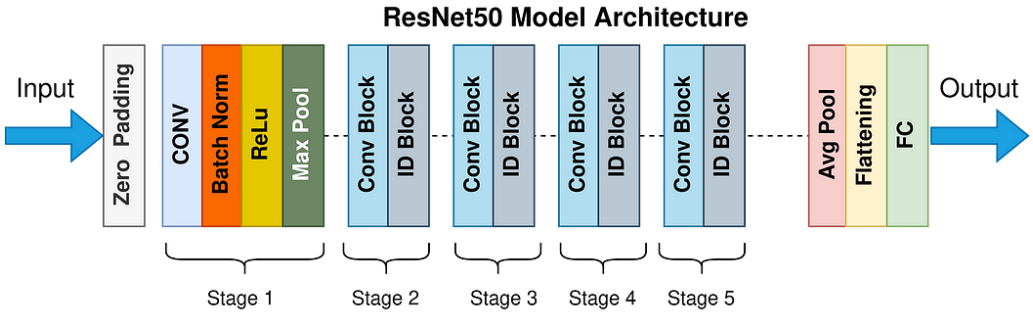

Figura 8 - Arquitetura RESNET-50

À semelhança da configuração para a rede VGG16, foram configurados os pesos (*keras-pretrained-models/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5)* e, desta forma, usar a RESET-50 no modo de *transfer learning*. A RESNET-50 tem 23688065 parâmetros, que ao congelar as primeiras camadas ficamos com "unicamente" 100353 parâmetros disponíveis para treino. A arquitetura foi configurada da forma ilustrada na Figura 9.

[2] https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758

A primeira camada da RESNET-50 contém a rede pré-treinada com 23587712 parâmetros não treináveis, e produz uma saída *(None, 1, 1, 2048)* que indica que foram extraídas 2048 características a partir da imagem de entrada.

- A segunda camada é o *dropout* configurado com 30% de probabilidade de desativação.
- A terceira camada é o *flatten*, utilizada para transformar a saída da camada anterior num vetor unidimensional de tamanho 100352 (parâmetros).
- A quarta camada é um *dropout* mas, desta vez, configurado com 50% de probabilidade de desativação.
- A quinta camada é a camada densa com um neurónio e ativação *sigmoid*, responsável pela classificação binária, e com o tamanho de 100352 parâmetros treináveis.

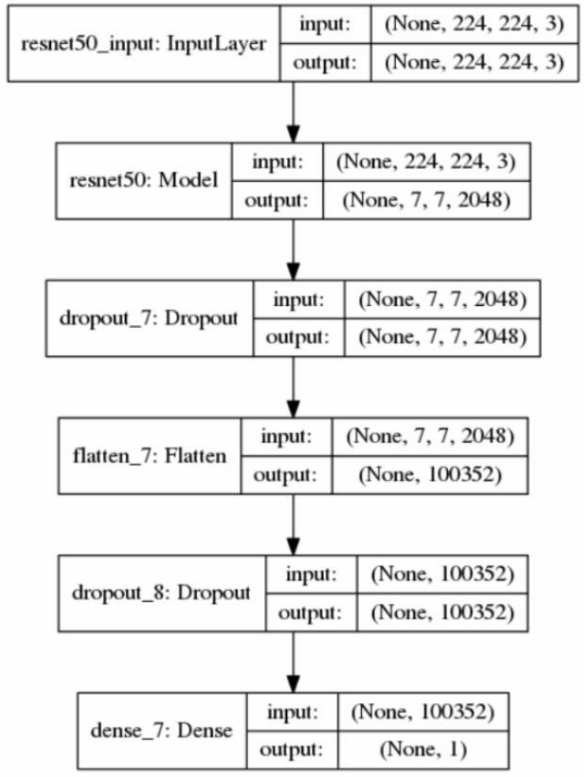

Figura 9 - Arquitetura utilizada RESNET-50

3.2.3.3. INCEPTION V3

A rede INCEPTION-v3, tal como as anteriores, é uma rede convolucional profunda que foi desenvolvida pela equipa da Google Brain [33], tendo sido essencialmente projetada para classificar imagens de alta resolução. A rede é composta por 42 camadas, divididas em 39 camadas convolucionais com diferentes tamanhos de filtros e de *kernel*, o que permite utilizar imagens de diferentes tamanhos e resoluções. Ainda conta com camadas de p*ooling* para a redução do tamanho da imagem, camadas de normalização (*batch normalization*) e camadas totalmente conectadas (densas) com 1000 neurónios (cf. Figura 10). [3]

Esta rede também foi pré treinada com recurso à base de dados da ImageNet. E, como é uma rede profunda muito complexa, apresenta também as desvantagens deste tipo de rede, como referido anteriormente para a VGG-16 e a Resnet-50.

Figura 10 - Arquitetura INCEPTION V3

Na configuração para o *transfer learning* na rede INCEPTION V-3 utilizamos os pesos (*keras-pretrained-models/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5*) para congelar as primeiras camadas da rede. A INCEPTION V-3 tem 21853985 parâmetros, no entanto, através desta técnica, apenas 51201 parâmetros são treinados. A arquitetura foi utilizada com as seguintes camadas, conforme ilustra a Figura 11.

- A primeira camada utilizada é o modelo pré-treinado produzindo uma saída *(None, 5,5, 2048)* com 2048 parâmetros treináveis.
- A segunda camada é um *dropout* com uma probabilidade de desativação de 30%.

[3] https://www.researchgate.net/figure/Google-Inception-v3-architecture-A-schematic-view-of-the-Inception-v3-model-Each-layer_fig1_329229460

- A terceira camada é uma camada *flatten* utilizada para transformar a entrada num vetor unidimensional com 51200 parâmetros.
- A quarta camada é uma camada *dropout* com uma probabilidade de desativação de 50%.
- A quinta camada é uma camada *densa* com saída *(None, 1)*, para a classificação binária com 51200 parâmetros.

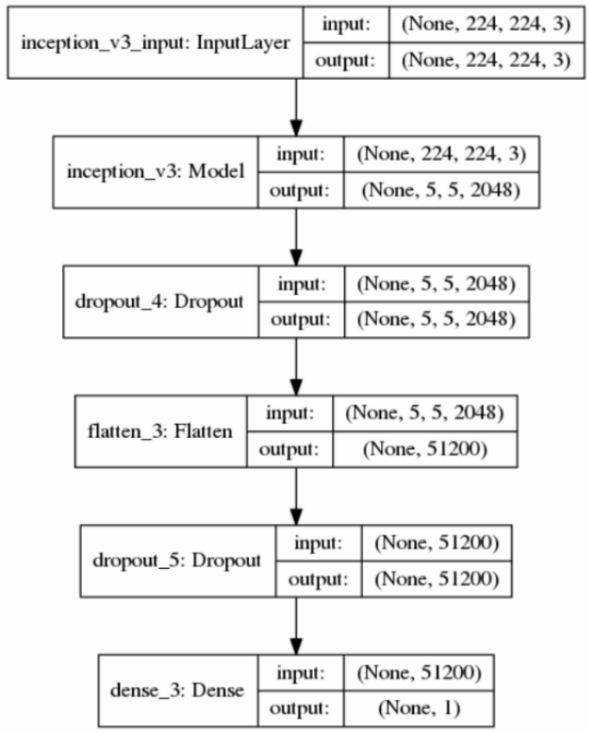

Figura 11 - Arquitetura utilizada INCEPTION V3

3.3. Modelos de classificação baseado nos dados numéricos

O objetivo desta Secção é a de apresentar os métodos aplicados na classificação do sexo de cada paciente em função dos resultados das medições realizadas nas ortopantomografias. Deste modo, pretendemos aplicar métodos de classificação, que podem ser métodos estatísticos ou de inteligência artificial, e em particular métodos de aprendizagem supervisionada. Todavia, o paradigma subjacente à aplicação dos métodos estatísticos, baseados na análise das distribuições que caracterizam os erros de estimação e/ou de classificação, é distinto do associado aos métodos da inteligência artificial, baseados na divisão da amostra em amostra de treino, utilizada na estimação dos parâmetros do modelo, e amostra de teste, para validação da generalização do modelo, isto é, avaliação da capacidade do modelo ser aplicado a outro conjunto de dados com características semelhantes. Por conseguinte, de forma a ser possível comparar o desempenho obtido por cada modelo, e uma vez que não é possível aplicar o paradigma dos modelos estatísticos a todos os métodos aplicados no presente trabalho, optamos por utilizar o referido paradigma da inteligência artificial.

Nesta análise, para criar e aplicar os modelos de classificação, os dados foram tratados estatisticamente com recurso ao software R em ambiente RStudio. Foi convertido o ficheiro que se encontrava em SPSS para RData, e foi feita a divisão dos dados de acordo com a divisão feita para a base de dados de imagens, i.e. 80% para treino e 20% para teste. Aplicou-se também um *oversampling* com o objetivo de balancear os dados e perceber que efeito o balanceamento tem no desempenho dos modelos finais. Na criação dos modelos tentamos aplicar as mesmas amostras, sendo aplicados os modelos com as seguintes combinações:

- Amostra completa
- Amostra de treino e teste
- Amostra de treino e teste aplicando o *oversampling*
- Amostra de treino e teste aplicando validação cruzada (*crossvalidation*)
- Amostra de treino e teste com *oversampling* e validação cruzada

Esta foi a metodologia aplicada na criação da maioria dos modelos na tentativa de encontrar os melhores resultados, e trabalhar com o facto da amostra de dados ser limitada em termos de tamanho.

3.3.1. Regressão logística

Os modelos de regressão são uma das ferramentas estatísticas mais importantes na análise estatística de dados, quando se pretende modelar relações entre variáveis. A regressão logística é uma técnica estatística que tem como objetivo modelar, a partir de um conjunto de observações, a relação "logística" entre uma variável resposta dicotómica e uma serie de variáveis explicativas numéricas (continuas, discretas) e/ou categóricas. Estes modelos são utilizados na análise de risco na atribuição de crédito, previsão de comportamentos, deteção de spam, entre outras aplicações. [34]

Para implementar o modelo de regressão logística, foi utilizada a função glm() no RStudio. O GLM (Modelos Lineares Generalizados) é uma classe de modelos de regressão que suporta dados que não apresentam necessariamente uma distribuição normal. Deste modo, esta função permite ao utilizador aplicar vários modelos de regressão, como a regressão linear, a regressão logística e a regressão de Poisson. Como o objetivo é encontrar o melhor modelo de classificação foram aplicados vários modelos utilizando, como variável resposta, a variável Sexo.

No primeiro modelo Rlog1 a variável de resposta foi estimada com base em todas as outras variáveis da base de dados. No parâmetro *family*, foi utilizada a distribuição binomial, uma vez que a variável resposta é dicotómica, correspondendo à aplicação da regressão logística.

```
Rlog1 <- glm(Genero ~., family = binomial, data = df)
```

Modelo 1 = Genero ~ Idade + ACD + ACE + AMD + AME + AQ + ARMD + ARME + LMRMD + LMRME + DG + DI1 + DI2 + AGD + AGE

De seguida, foi criada uma variável que contém as probabilidades estimadas (valores ajustados), ou seja, a probabilidade prevista para cada paciente no conjunto de dados ser do sexo masculino com base nos parâmetros estimados do modelo. Através desta variável, é possível calcular o ponto de corte para a distribuição, a matriz de confusão e a curva ROC do modelo.

No segundo modelo Rlog2 o objetivo é encontrar o modelo mais parcimonioso e, para tal, foram removidas, sequencialmente, as variáveis com menos significância (sem poder explicativo relevante). O modelo final obtido foi o seguinte:

Modelo2 = Genero ~ AMD + AME + ARMD

Para testar a significância global do modelo e a contribuição de cada variável para o poder explicativo do modelo, foi efetuado o teste *anova()* entre os dois modelos. E foi também verificado o teste de hipóteses para os valores de AUC, entre a curva ROC do primeiro modelo e do modelo parcimonioso.

Para o modelo Rlog3 foi utilizada a amostra de treino e teste, tendo também aqui o objetivo de encontrar o modelo parcimonioso. À semelhança do modelo 2, foram removidas as variáveis com menor significância, e no final o modelo obtido foi o seguinte:

Modelo3 = Genero ~ AME + ARMD

Com o modelo criado, foi utilizada a função *predict()* para a amostra de treino e para a amostra de teste, e, desta forma, obtidas previsões da probabilidade de ser do sexo masculino para cada paciente, quer nos dados de treino quer nos de teste. Desta forma, é possível obter as matrizes de confusão e as curvas ROC e aplicar o *roc.test()* de comparação nas duas curvas.

O quarto modelo Rlog4 foi criado recorrendo ao conjunto de dados com *oversampling*, tendo também neste modelo sido utilizado o modelo parcimonioso.

Modelo 4 = Genero ~ Idade + AQ + ARMD

Neste modelo foram aplicadas as mesmas funções que no modelo 3. No Capítulo 4 são apresentados os valores obtidos para o desempenho dos quatro modelos criados.

3.3.2. Análise discriminante linear

A análise discriminante linear (LDA, *linear discriminant analysis*) é uma ferramenta usada para classificação, redução de dimensão e visualização de dados. O objetivo de se aplicar técnicas de redução de dimensões é remover as características redundantes e dependentes (no sentido de fortemente correlacionadas) ao transformar características de um espaço dimensional maior para um espaço com dimensão inferior.

O método foi introduzido, na sua forma inicial, para problemas com duas classes por Fisher em 1936, mas é particularmente útil com amostras de dados multinomiais, assumindo que as variáveis de entrada seguem uma distribuição gaussiana multivariada e que as matrizes de covariância (ou de correlação) de cada grupo são iguais (homocedasticidade). A LDA tem aplicação em diversos campos do conhecimento, como na biologia (para classificações taxonómicas), nas finanças (o Z-score de Altman para previsão de falências) e na tecnologia (para reconhecimento de dígitos), entre outros. [35]

Antes de se iniciar a análise discriminante linear efetuamos o teste, utilizando a função HDoutliers(), para verificar a existência de outliers multivariados. Como a LDA assume que os dados seguem uma distribuição normal multivariada e a existência de homocedasticidade, foi aplicado o teste de hipóteses mshapiro.test() e desta forma verificar se as variáveis quantitativas seguem uma distribuição normal multivariada. Foi ainda aplicado o teste *boxM()* para verificar se a matriz de variância-covariância é igual nas duas categorias da variável Sexo.

À semelhança da metodologia utilizada para a regressão logística, também na análise discriminante linear foram criados vários modelos com o intuído de encontrar o modelo com os melhores resultados. Para a criação dos modelos LDA utilizamos a função lda() da biblioteca MASS que tem como objetivo encontrar um discriminante linear que maximize a separação entre classes. O primeiro modelo criado utiliza a variável de resposta Sexo em relação a todas as outras variáveis do conjunto de dados.

```
lda_0 = lda (Genero~., data = df)
```

Com o modelo criado, foi utilizada a função *predict()* para criar uma variável que contém a previsão das probabilidades, e desta forma obter o gráfico com as projeções dos dados no espaço discriminante, a matriz de confusão e a curva ROC para o modelo.

O segundo modelo foi criado à semelhança do primeiro, no entanto com a utilização da amostra de treino e de teste. Foi utilizada a função *predict()* com o modelo, em ambas as amostras, com o intuído de comparar as probabilidades de ser do sexo feminino ou masculino e verificar a distribuição dos valores da função discriminante por sexo.

O terceiro modelo foi criado com recurso à amostra de dados de treino com *oversampling*, que tem como objetivo aumentar a representatividade da classe minoritária, visto que o conjunto de dados não se encontra balanceado.

Na criação dos modelos seguintes foi utilizada a técnica de validação cruzada (*cross validation).* Esta técnica é utilizada para avaliar a capacidade de generalização do modelo através da divisão da amostra de dados em *folds*. Recorrendo a função trainControl() definimos que o nosso conjunto de dados será divido em 10 *folds,* ou seja, o conjunto de dados será dividido em 10 partes de tamanho igual. Para treinar o modelo foi utilizada a função *train()* da biblioteca *caret*, e foi treinada utilizando o algoritmo LDA, a métrica acurácia, e com o argumento do controlo criado anteriormente.
Nestes modelos a função *predict()* foi utlizada com a opção type= "prob" porque pretendemos a probabilidade das classes para cada observação no conjunto de dados de treino. Iniciamos com a criação de um modelo com a amostra total, aplicando a técnica de validação cruzada:

```
lda_0CV <- lda(Genero ~., CV = TRUE, data = df)
```

Foi possível obter a precisão do teste para este modelo, como também obtivemos as respetivas curvas ROC. Nos modelos seguintes utilizamos os mesmos parâmetros, com a amostra de treino e a amostra de teste, e com a amostra com *oversampling*.

3.3.3. Análise discriminante quadrática

A análise discriminante quadrática (QDA, *quadratic discriminant analysis*) é um método estatístico utilizado para classificar objetos em diferentes grupos ou categorias com base em variáveis de entrada. É uma extensão do LDA e assume que as variáveis de entrada seguem uma distribuição gaussiana, porém as matrizes de covariância não têm de ser iguais (pode haver heterocedasticidade). A QDA estima a função de densidade de probabilidade para cada grupo usando uma distribuição gaussiana multivariada. A partir dessas estimativas, é possível calcular a probabilidade de um objeto pertencer a cada grupo. O indivíduo é então classificado no grupo com a maior probabilidade. [36]

O primeiro modelo criado com a análise discriminante quadrática foi com a amostra total dos dados.

```
qda_0 = qda(Genero~., data = df)
```

Seguindo a mesma metodologia aplicada à análise discriminante linear, foram criados modelos com a amostra de treino e teste, e também com a amostra com *oversampling*, tendo sido também utilizada a técnica de validação cruzada à amostra. Todos os modelos foram criados com a amostra de treino, e foi aplicada a função *predict*() à amostra de teste.

3.3.4. Árvores de decisão

As árvores de decisão são uma técnica de aprendizagem automática supervisionada utilizada em problemas de classificação e de regressão. Elas funcionam criando uma estrutura de árvore que é usada para modelar a relação entre os recursos e o resultado desejado. Assim, é construída através da divisão recursiva do conjunto de dados em subconjuntos menores de acordo com determinados critérios e de forma que os conjuntos finais tenham uma elevada homogeneidade. Uma das vantagens das árvores de decisão é a interpretabilidade. Como a estrutura é visual, é possível compreender facilmente como o algoritmo tirou as conclusões [37, 38]. Um exemplo de árvore de decisão pode ser consultado na Figura 12.

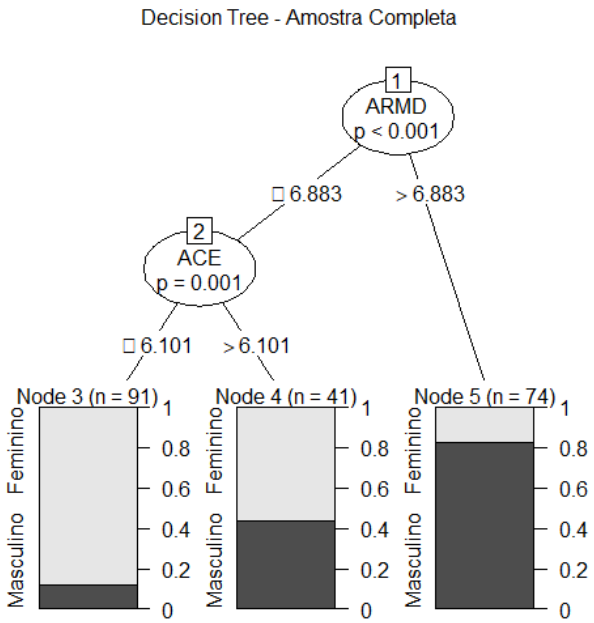

Figura 12 - Árvore de decisão

Para a criação dos modelos com recurso às arvores de decisão foi utilizada a função *ctree()* da biblioteca *party*. Assim, para o primeiro modelo recorremos à amostra completa.

```
model_tree0 <- ctree(Genero ~ ., df)
```

Na Figura 12 podemos ver o resultado do *plot()* obtido do primeiro modelo. Nos modelos seguintes foi utilizada a amostra de treino para o treino do modelo e a amostra de teste para a função *predict()*. Também foi criado um modelo com a amostra de dados em *oversampling*. Obtivemos as curvas ROC, as matrizes de confusão e a representação de todos os modelos estimados.

3.3.5. Naive Bayes

Naive Bayes é um algoritmo de classificação baseado no teorema de Bayes, sendo utilizado sobretudo para tarefas de classificação de texto. O algoritmo assume, para simplificar, que as características ou atributos de cada classe são independentes entre si, ou seja, a presença ou ausência de características não influencia a presença ou ausência de outras características.

Deste modo, uma das vantagens do algoritmo Naive Bayes é ser um algoritmo pouco complexo, escalável, que permite lidar com dados de elevada dimensão. Atualmente este algoritmo é utilizado em filtros de *spam*, classificação de documentos, entre outros. [39]

Seguindo a metodologia, o primeiro modelo foi criado utilizando a amostra completa e utilizando a função naiveBayes disponível no RStudio.

```
model_naive0 <- naiveBayes(Genero ~ ., data = df)
```

À semelhança dos anteriores modelos, depois de criado e treinado o modelo naiveBayes, foi utilizada a função *predict()* para obter as previsões do modelo. Desta forma, podemos obter a matriz de confusão e a curva ROC.

Os modelos seguintes foram criados recorrendo à divisão do conjunto de dados, em treino e teste, e também à aplicação do *oversampling* nos dados de treino.

3.3.6. *k*-vizinhos mais próximos

O modelo *k*-vizinhos mais próximos (KNN, *k-nearest neighbours*) é um algoritmo de aprendizagem automática supervisionada utilizado para classificação e para regressão. O algoritmo classifica uma amostra nova com base nas classes das amostras vizinhas (mais próximas). A distância entre as observações é geralmente determinada usando a distância euclidiana, mas existem diversas distâncias que podem ser aplicadas. É um algoritmo simples de implementar e robusto. A principal desvantagem deste método prende-se com a complexidade computacional para classificar novos indivíduos crescer linearmente com o aumento da dimensão da amostra de treino.[39, 40]

Antes de criar os modelos foi necessário criar um conjunto de dados estandardizados e, para tal, foi utilizada a função *scale()* nas variáveis quantitativas. Esta estandardização consiste em subtrair a média de cada variável e dividir pelo desvio padrão, pois desta forma todas as variáveis têm uma média nula e um desvio padrão de um. Deste modo, todas as variáveis terão um peso semelhante no agrupamento, pois as variáveis com maior variabilidade

(dispersão) tendem a ser mais relevantes na determinação das distâncias entre indivíduos. O primeiro modelo criado foi utilizado o *k* (número de vizinhos) igual a dois (por se pretender classificar em duas categorias).

```
knn0 = knn(train=df_scale, test=df_scale, cl=df$Genero, k=2)
```

Depois de analisar as métricas, verificou-se que tanto *k*=2 como *k*=3 são os valores onde obtemos melhores resultados com o modelo. Para o segundo modelo, utilizamos o método de validação cruzada e, para tal, foi criada uma variável para controlar o treino dos dados com a função *traincontrol(),* com o número de *folds* igual a 10 e o número de repetições igual a 5. Ou seja, os dados serão divididos em 10 subconjuntos para a validação cruzada, e 5 é o número de repetições que o processo de validação é feito.

Depois de criada a variável de controlo, o treino do modelo é feito com recurso à função *train(),* utilizando a amostra completa dos dados e com o método *knn,* foi também utilizado os parâmetros *preProcess*, que permite aplicar um pré-processamento aos dados e o *tuneLength* igual a 10 que especifica o número de valores *k* a serem testados durante a etapa de ajuste dos hiperparâmetros do modelo.

Para o terceiro modelo, recorremos à amostra dividida em treino e teste, e aplicamos a metodologia usada no primeiro modelo. O modelo foi criado inicialmente com *k* igual a 2, e, depois de analisar as métricas, observou-se que a melhor performance era obtida para *k* igual a 9.

O quarto modelo foi criado com a amostra de dados de treino e com a aplicação do método de validação cruzada. A função de treino é semelhante à utilizada anteriormente. Para encontrar o número *K-folds* necessários para criar o modelo, foi desenvolvida uma função que testa o valor de *K* no modelo e devolve o valor da acurácia para o modelo. Desta forma, é possível observar que existem várias opções para o qual *K gera* melhor acurácia.

Para o quinto modelo foi utilizado o *oversampling* à amostra de dados de treino. Porém, antes de criar o modelo, foi necessário estandardizar as variáveis quantitativas da amostra de dados de treino recorrendo a função *scale()*. Aplicando a função para percorrer um

determinado número de valores de *k*, observamos que a melhor acurácia obtida foi para *k* igual a 15. Valor esse que foi utilizado na criação do modelo do final.

O sexto modelo foi criado recorrendo à validação cruzada, e foi aplicado à amostra de dados com *oversampling*.

3.3.7. Máquinas de vetores de suporte

Máquinas de vetores de suporte (SVM – *support vector machine*) é um algoritmo de aprendizagem automática supervisionada utilizado para classificação ou regressão. O SVM funciona dividindo as classes em dois espaços separados por um hiperplano de forma que a margem entre duas classes seja maximizada. Desta forma, tentando encontrar a melhor fronteira de separação entre classes para um dado conjunto de dados que seja linearmente separável (hiperplano). Quando os dados não são linearmente separáveis, o SVM utiliza um núcleo (*kernel*) para transformar o espaço de características num espaço de dimensão superior, onde as classes possam ser separadas por um hiperplano. A vantagem deste algoritmo é ser robusto e eficaz, mesmo quando o conjunto de dados apresenta alta dimensionalidade. Como desvantagem, é um algoritmo lento comparado com outros, e é de difícil interpretação e visualização. [40, 41]

Para criar um modelo do tipo máquinas de vetores de suporte, no RStudio utilizamos a função *train()* incluindo no parâmetro *method* o svmLinear. A função *train()* utiliza a mesma variável de controlo dos modelos anteriores, com os mesmos parâmetros. De seguida, para obter o vetor de previsões utilizamos a função *predict()* com a amostra de teste. No segundo modelo criado utilizamos a amostra de dados com *oversampling*, à semelhança do modelo anterior, o *train()* com a amostra de treino, e o *predict()* com a amostra de teste.

Utilizando esta metodologia, é possível obter todas as métricas necessárias para comprar o modelo na amostra de treino e na de teste, bem como obter as matrizes de confusão associadas.

3.3.8. Florestas aleatórias

À semelhança dos algoritmos anteriores, também as florestas aleatórias são um algoritmo de aprendizagem automática supervisionada, que combina várias árvores de decisão para classificação ou regressão. Foi introduzido em 2011 por Breiman [42] e é um método que evoluiu das árvores de decisão. O algoritmo cria várias árvores de decisão e cada uma é treinada com uma subamostra aleatória do conjunto de dados de treino. O resultado é um conjunto de árvores não correlacionadas. Assim, durante o processo cada arvore é usada para prever a classe. Por conseguinte, são obtidas diversas classificações independentes, provavelmente cada uma sem grande fiabilidade, mas que em conjunto permitem obter uma classificação mais fiável. As vantagens deste algoritmo é a precisão, simplicidade e flexibilidade.[43, 44]

À semelhança das máquinas de vetores, para criar um modelo do tipo florestas aleatórias utilizamos a função *train(),* no entanto o parâmetro *method* é igual a "rf". Também a variável de controlo é a mesma utilizada para o treino de todos os outros modelos. Depois de criado o modelo, foi utilizado para obter as previsões na amostra de teste. No segundo modelo recorremos à amostra de treino com *oversampling*.

3.4. Critérios de avaliação de desempenho dos modelos

A avaliação do desempenho é uma etapa crucial no processo de construção de modelos de classificação. Para determinar a performance desse modelo é necessário definir os critérios de avaliação apropriados. Como o objetivo deste projeto é comparar a performance dos modelos de classificação e das redes neuronais, foram escolhidos os seguintes critérios.

A matriz de confusão é umas das ferramentas mais comuns, usada para avaliar o desempenho dos modelos de classificação, e é utilizada para identificar o número de indivíduos classificados correta ou incorretamente para cada classe, (cf. exemplificado na Tabela 2). Todavia, na classificação é relevante determinar qual é a categoria positiva e a negativa, que na área dos testes clínicos é evidente (por exemplo, num teste ao Covid-19), mas na classificação sexual é indiferente. Deste modo, de forma a simplificar a comparação entre os diferentes métodos aplicados, foi definido que a categoria positiva seria o sexo masculino.

		Classes Previstas	
		Masculino	Feminino
Classes Reais	Masculino	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Feminino	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Tabela 2 - Matriz de confusão

A matriz de confusão é composta por quatro valores[4]:

- VP (verdadeiro positivo) – o indivíduo foi classificado como sendo do sexo masculino, e é realmente do sexo masculino.

- FP (falso positivo) – o indivíduo foi classificado como sendo do sexo masculino, mas efetivamente é do sexo feminino, logo foi incorretamente classificado.

[4] No exemplo retratado na Tabela 2, considerou-se como positivo ser do sexo masculino, mas o mesmo procedimento poderia ser aplicado considerando o oposto.

- VN (verdadeiro negativo) – o indivíduo foi classificado como sendo do sexo feminino, e é efetivamente do sexo feminino.

- FN (falso negativo) – o indivíduo foi classificado como sendo do sexo feminino, e é efetivamente do sexo masculino, portanto foi incorretamente classificado.

Através da matriz de confusão é possível calcular diversas métricas de desempenho, tais como:

- Acurácia (A): representa a proporção de indivíduos classificados corretamente em relação ao total de indivíduos.

$$\mathrm{A} = \frac{\mathrm{VP} + \mathrm{VN}}{\mathrm{VP} + \mathrm{VN} + \mathrm{FP} + \mathrm{FN}}$$

- Precisão ou valor preditivo positivo (VPP): representa a proporção de indivíduos classificados como positivos em relação ao total de positivos.

$$\mathrm{VPP} = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FP}}$$

- Especificidade (ESP): representa a proporção de indivíduos negativos que foram corretamente classificados como negativos pelo modelo. É uma métrica especialmente relevante em aplicações médicas, em que falsos positivos podem levar a diagnósticos errados cujos tratamentos acarretem riscos para o paciente.

$$\mathrm{ESP} = \frac{\mathrm{VN}}{\mathrm{VN} + \mathrm{FP}}$$

- Sensibilidade (SENS) ou *recall*: representa a proporção de indivíduos positivos classificados corretamente em relação ao número total de indivíduos positivos.

$$\mathrm{SENS} = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FN}}$$

- F1 score: esta métrica toma em consideração a precisão e o *recall* do modelo. O F1 score varia entre 0 e 1, sendo que valores próximos de 1 indicam um bom desempenho do modelo em ambas as métricas, correspondendo à média harmónica da precisão e *recall*.

$$\text{F1 Score} = 2 \text{ x } \frac{\text{Precisão x Recall}}{\text{Precisão} + \text{Recall}}$$

Outra das métricas utilizadas neste projeto, como critério de avaliação, é a curva ROC (*receiver operating characteristic*) e a área sob a curva ROC (AUC – *area under the curve*). A curva ROC é um gráfico utilizado para avaliar o desempenho de um modelo de classificação, e representa graficamente a relação entre a taxa de verdadeiros positivos (sensibilidade) e a taxa de falsos positivos (1- especificidade), permitindo assim calcular a sensibilidade e a especificidade para diferentes pontos de corte [45]. Um exemplo de uma curva ROC é apresentado na Figura 13.[5]

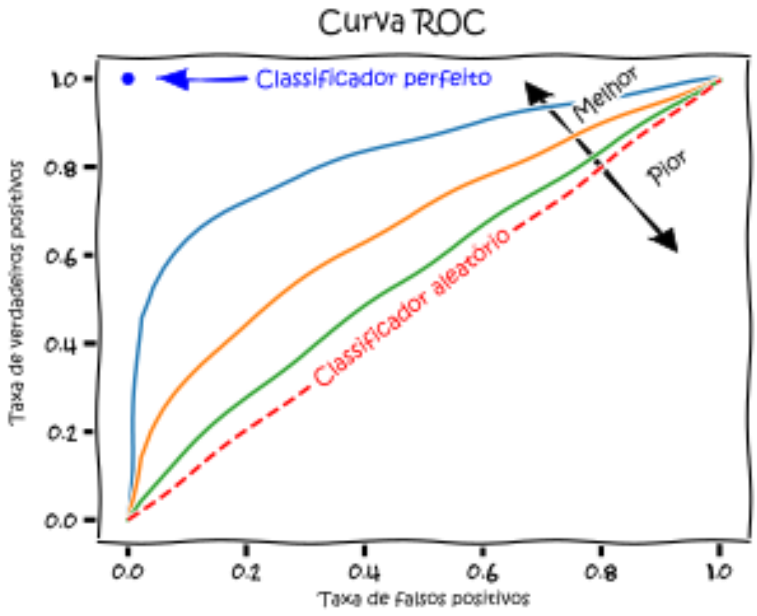

Figura 13 - Exemplo de curva ROC

[5] https://eme.cochrane.org/razao-de-verossimilhanca-e-curva-roc/

Quanto mais a curva ROC se aproximar do canto superior esquerdo do gráfico, melhor é a performance do modelo de classificação. Num modelo perfeito a curva passaria pelo ponto (0,1), o que significaria 100% de sensibilidade e 100% de especificidade.

Além da curva ROC, também a AUC (a área sob a curva) foi utilizada para avaliar a performance do modelo de classificação. A área sob a curva ROC é uma métrica de desempenho que mede a capacidade de um modelo de classificação binária distinguir entre as categorias positiva e negativa. Quanto maior a área sob a curva ROC, melhor o desempenho do modelo em termos de sensibilidade e especificidade.

O valor AUC varia de 0 a 1, onde 0 indica um modelo que classifica todos os indivíduos incorretamente; 0.5 corresponde a classificação aleatória e 1 indica um modelo perfeito, que classifica todos os indivíduos corretamente[6], cf. Figura 14.[7]

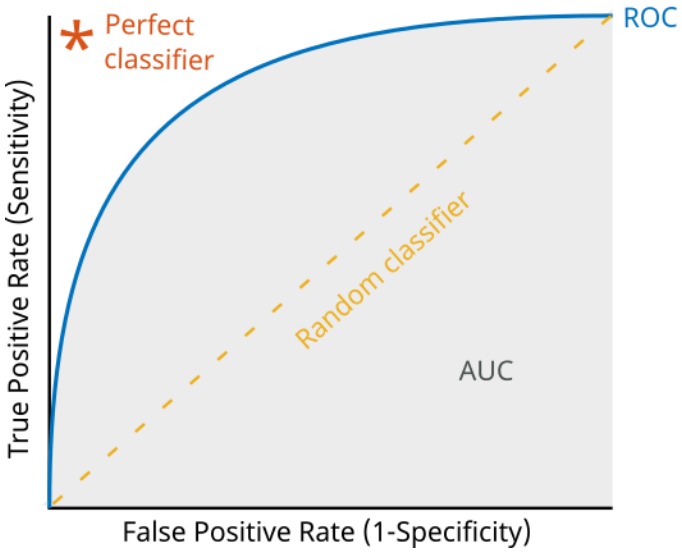

Figura 14 - Exemplo da métrica AUC

[6] Usualmente é referido que a AUC varia entre 0.5 e 1, pois valores perto de zero indicam que o modelo consegue distinguir os dois grupos, mas a classificação está trocada (se AUC for zero, então é trocar todas as classificações e passamos a ter AUC igual a 1). Deste modo, AUC abaixo de 0.5 indicia que terá existido algum problema no processo de classificação. O valor 0.5 é que indica incapacidade de discriminação do modelo (próximo da classificação aleatória).

[7] https://www.mathworks.com/help/deeplearning/ug/compare-deep-learning-models-using-ROC-curves.html

4. Resultados

A análise e discussão dos resultados da classificação sexual é uma etapa fundamental na avaliação do desempenho do modelo e na compressão da capacidade de previsão dos métodos estatísticos e dos modelos de inteligência artificial desenvolvidos. Nesta análise, iremos discutir os principais resultados obtidos, começando pela avaliação geral do desempenho dos modelos, seguido pelos critérios definidos na Secção 3.4 e pela análise dos gráficos mais relevantes. Um resumo dos resultados pode ser consultado na Secção 7.1.

4.1. Análise e discussão dos resultados da classificação sexual com recurso a redes neuronais

Todos os resultados apresentados nesta Secção foram obtidos utilizando a linguagem Python, estando o respetivo script utilizado presente no anexo II.

Como descrito no Capítulo 3, foi utilizada a mesma metodologia para a obtenção dos resultados. Depois de vários testes aos parâmetros (optimizadores, funções de perda (*loss function*), pesos, entre outros) foi encontrada a melhor combinação, combinação essa que foi replicada nas três redes neuronais. Ver os resultados obtidos na Tabela 3.

Rede	Data Augmentation	Pesos	Optimizador	Função Loss		Acurácia	Sexo	VPP	Recall	F1-Score
VGG16	SIM	SIM	RMSprop(lr=1e-4)	binary_crossentropy	Validação	0,95	homem	1	0,88	0,94
							mulher	0,91	1	0,95
					teste	0,7	homem	0,75	0,94	0,95
							mulher	0,67	0,95	0,95
RESNET50	SIM	SIM	RMSprop(lr=1e-4)	binary_crossentropy	Validação	0,58	homem	0,52	0,94	0,67
							mulher	0,86	1,29	0,43
					teste	0,6	homem	0,74	0,55	0,41
							mulher	0,76	0,5	0,38
INCEPTION V3	SIM	SIM	RMSprop(lr=1e-4)	binary_crossentropy	Validação	0,53	homem	0,48	0,82	0,61
							mulher	0,67	0,29	0,4
					teste	0,4	homem	0,57	0,55	0,5
							mulher	0,58	0,53	0,49

Tabela 3 - Resultados obtidos com recursos a Redes Neuronais

Como é possível ver na Tabela 3, a rede onde obtivemos melhores resultados foi a VGG16 com uma acurácia de 95% para a amostra de validação e 90% de acurácia para amostra de teste. Para as mesmas condições na rede RESNET 50 apenas foi possível obter 58% de acurácia na validação e 60% no teste, tendo a rede INCEPTION V3 devolvido os piores resultados, 53% de acurácia na validação e 40% no teste.

Através da análise aos gráficos de acurácia é possível ver que o modelo VGG16 apresenta uma curva crescente, o que nos indica que o modelo está a evoluir durante o processo de treino. Podemos constatar que a curva de acurácia para o conjunto de treino demonstra que o desempenho do modelo está a melhorar à medida que ele é exposto a mais exemplos. A estagnação da acurácia para o conjunto de validação pode indicar um sinal de *overfitting*, cf. Figura 15.

Na análise do gráfico de perda para ambos os conjuntos de dados, podemos verificar que o erro está a diminuir à medida que o treino avança, no entanto para o conjunto de dados de validação (dados nunca vistos pelo modelo) é possível verificar uma estagnação ou até uma ligeira subida, o que poderá indicar que o modelo poderá sofrer de *overfitting.*

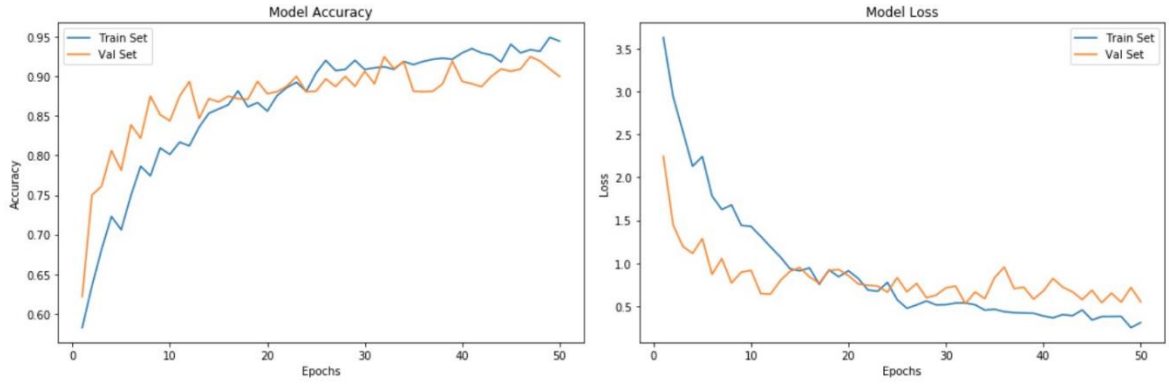

Figura 15 - Gráfico da acurácia (esquerda) e gráfico de perda (direita) da VGG16

Outro dos gráficos obtidos das redes neuronais foi a matriz de confusão, e na Figura 16 temos a matriz de confusão para o conjunto de dados de validação e a matriz de confusão para o conjunto de dados de teste. É possível verificar que a rede está a classificar praticamente a totalidade dos dados como verdadeiros positivos e verdadeiros negativos, classificando apenas dois elementos como falsos positivos, ou seja, dois elementos foram classificados como sendo do sexo feminino e eram, na realidade, do sexo masculino.

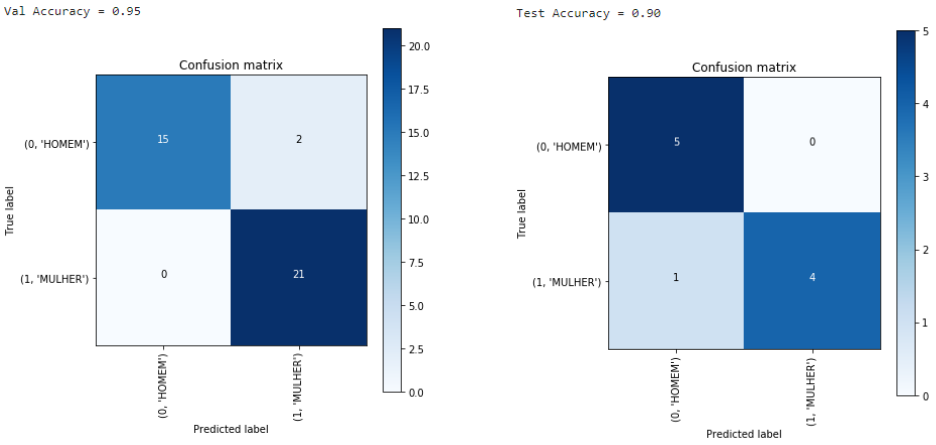

Figura 16 – Matriz de confusão na validação (esquerda) e no teste (direita)

Na matriz de confusão para o conjunto de dados de teste, houve um elemento classificado como sendo do sexo masculino, quando na realidade é do sexo feminino.

Na Figura 17 é possível ver o gráfico para a área abaixo da curva ROC em relação ao número de iterações utilizadas para treinar a rede. O valor obtido para a AUC nesta rede foi de 0,99, para valores compreendidos entre 0,9 e 1 a qualidade do teste é excelente.

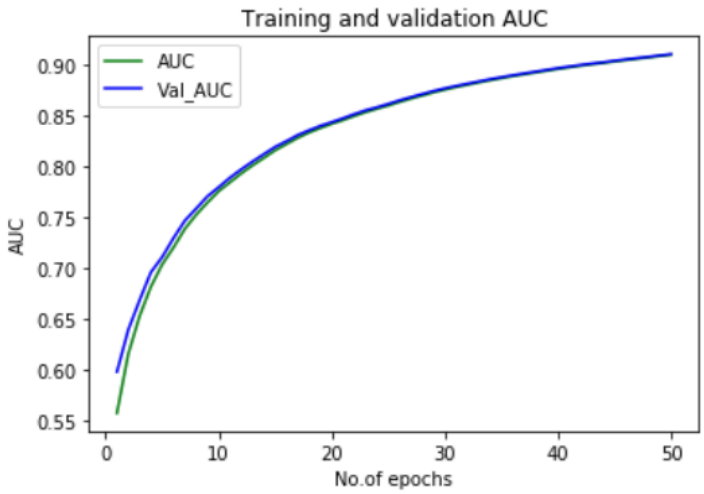

Figura 17 – Gráfico de progressão VGG16

Na Figura 18 podemos ver os gráficos de acurácia e perda para a rede RENET-50 e é notório no comportamento do gráfico que a rede melhora à medida que é treinada (conjunto de treino). No entanto, quando é apresentado o conjunto de dados de validação, a rede apresenta

um fraco desempenho na classificação. Deste modo, possivelmente o modelo está a ajustar-se aos dados de treino e não tem capacidade de generalizar para outros dados. O mesmo acontece com o gráfico de perda pois, quando são utilizados dados que a rede desconhece, a função de perda aumenta, evidenciando um claro sinal de *overfitting*.

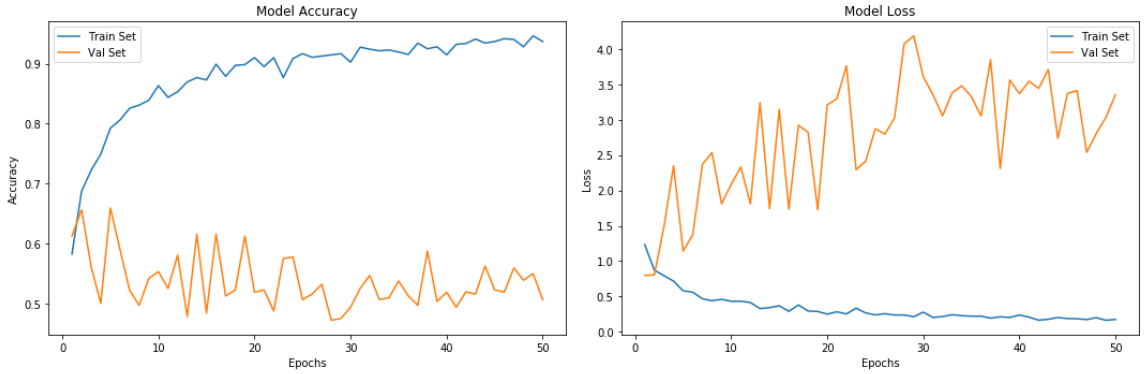

Figura 18 - Gráfico da acurácia (esquerda) e gráfico de perda (direita) da RESNET 50

Também através da análise das matrizes de confusão, conseguimos verificar a péssima capacidade de previsão da rede RESNET 50. Na Figura 19, é possível ver que a rede classificou quase a totalidade da amostra como verdadeiro positivo e falso positivo, ou seja, como sendo tudo do sexo masculino.

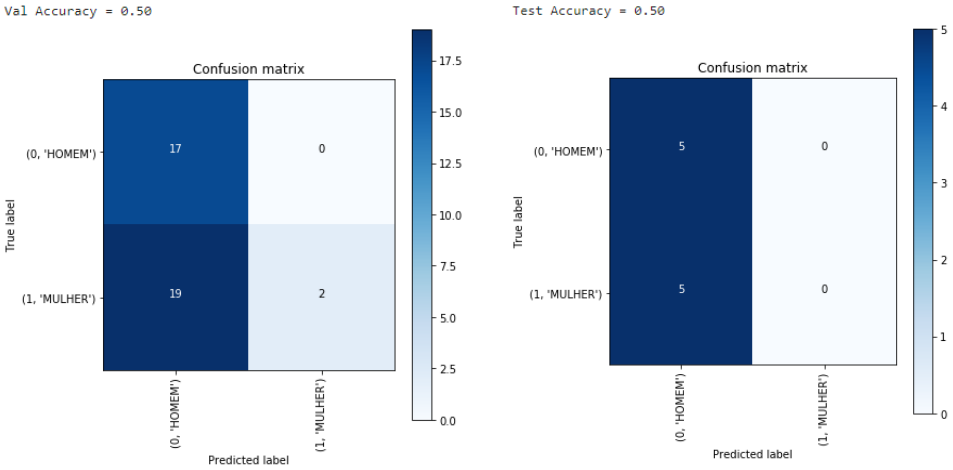

Figura 19 - Matriz de Confusão na Validação (esquerda) e no Teste (direita)

Também na matriz de confusão para a amostra de teste podemos ver o mesmo comportamento, onde a totalidade da amostra foi classificada como sendo do sexo masculino, cf. Figura 19.

Na rede INCEPTION V3 o comportamento da rede é idêntico ao da rede RESNET 50, sendo possível verificar que a rede está em *overfitting* pela diferenciação de comportamento dos gráficos nos dois conjuntos de dados, cf. Figura 20.

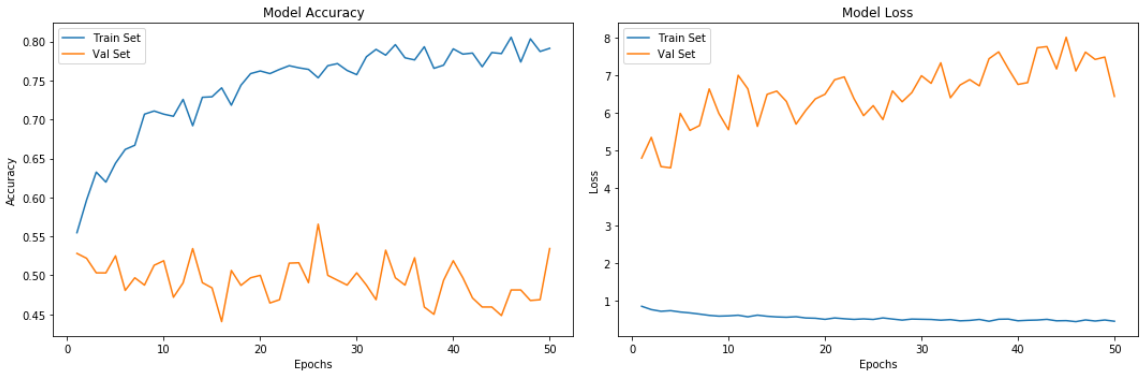

Figura 20 - Gráfico da acurácia (esquerda) e gráfico de perda (direita) da INCEPTION V3

Também as matrizes de confusão para o conjunto de dados de validação e para o conjunto de dados de teste são apresentadas na Figura 21. Podemos verificar que praticamente a totalidade dos elementos foram classificados como verdadeiros positivos.

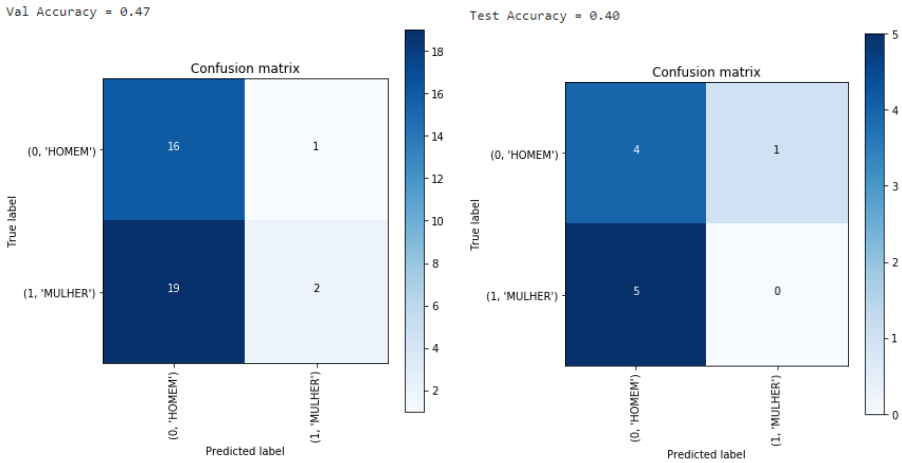

Figura 21 - Matriz de Confusão Validação

4.2. Análise e discussão dos resultados da análise exploratória dos dados e criação de modelos lineares

De forma a simplificar a apresentação dos resultados, nomeadamente por serem aplicadas diversas metodologias de classificação com diferentes divisões da amostra (completa, treino *versus* teste, com e sem *oversampling*, com e sem validação cruzada) os principais resultados são apresentados de forma resumida em anexo (Anexo I). De igual forma, a programação realizada em linguagem R é apresentada no anexo III.

Assim, iniciamos a análise estatística do conjunto de dados com a análise à capacidade discriminante das variáveis quantitativas. Para tal, representamos o diagrama de extremos e quartis (*boxplot*) na Figura 22 para cada variável em função do sexo. Através da análise visual, é possível verificar a existência de alguns *outliers*, e a dispersão dos dados interquartis e as medianas de cada variável em relação ao sexo.

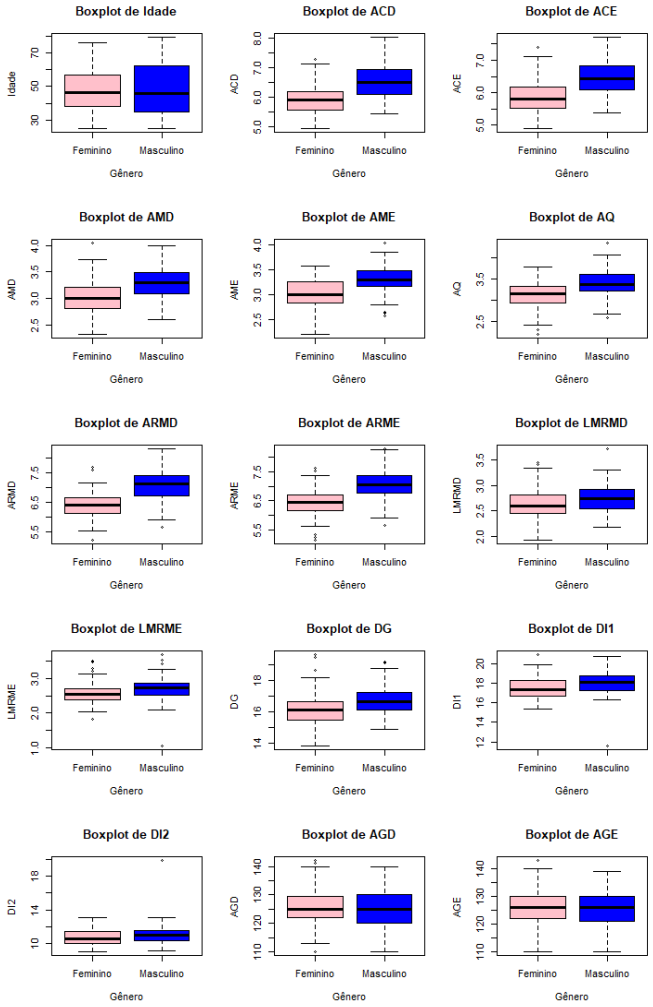

Figura 22 - Capacidade discriminante de cada variável (*boxplot* por sexo)

É possível constatar através de análise visual que existem variáveis que possuem dispersões, medianas, e quartis diferentes nos dois sexos, como é o caso da variável ARME, ARMD. Por outro lado, em variáveis como AGE e AGD os gráficos são similares para ambos os sexos, o que nos leva a considerar que as variáveis em questão possuem pouca capacidade discriminante.

Como foi descrito na Secção 3.4, uma das métricas mais utilizadas neste projeto é a AUC e, como tal, nesta análise inicial apresentamos também as curvas ROC para cada variável. Com é possível verificar na Figura 23, a curva com o melhor valor para o AUC é de 0.861 na variável ARMD (Altura do ramo da mandíbula direita). No entanto, a variável ARME possui um valor muito similar. Assim, para ambos os casos podemos considerar que as variáveis têm um desempenho discriminante muito bom.

É possível também verificar que as variáveis idade, AGD e AGE são as variáveis com o valor mais baixo de AUC, um valor próximo de 0.5. Estes resultados revelam que estas variáveis não tem capacidade discriminatória. Deste modo, as AUC confirmam a análise prévia, dos diagramas de extremos e quartis, na identificação das variáveis mais discriminantes.

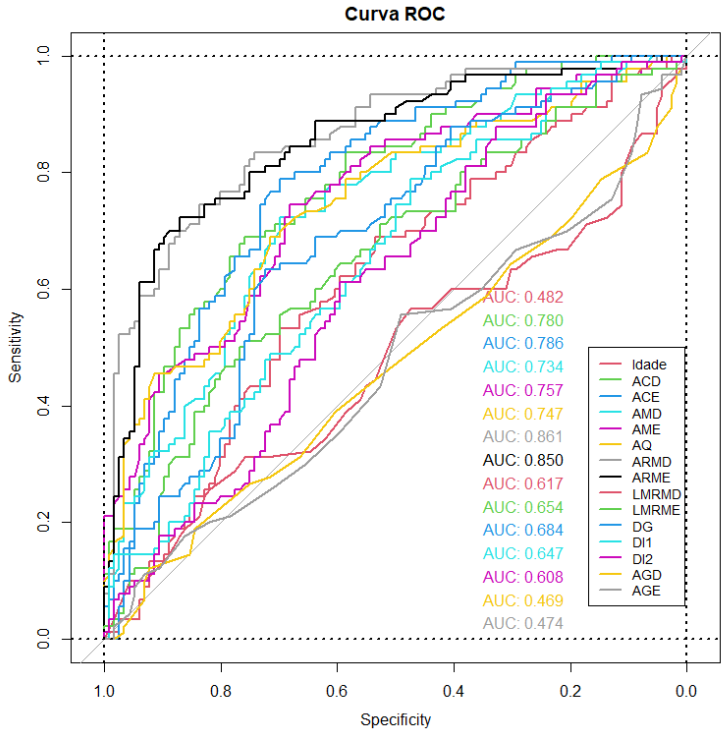

Figura 23 - Curva ROC toda as variáveis

Na Figura 24 temos o resultado obtido com *QQ plot* para todas as variáveis quantitativas. Com este gráfico é possível verificar se as variáveis seguem (aproximadamente) uma distribuição normal e quanto se afastam desta distribuição. No entanto, de seguida iremos utilizar o teste de Shapiro e avaliar os resultados.

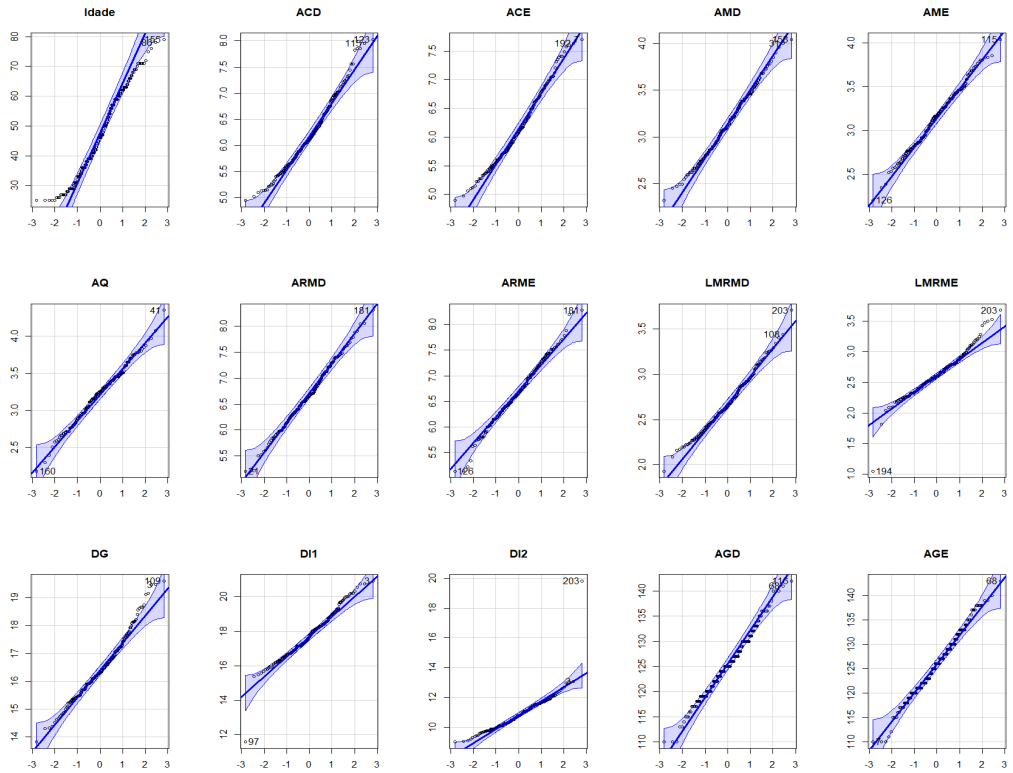

Figura 24 - QQPlot de todas as variáveis

Efetuamos o teste de hipóteses para a normalidade nas variáveis quantitativas do conjunto de dados para, desta forma, perceber se as variáveis seguem uma distribuição normal. O teste de normalidade efetuado foi o teste de Shapiro (função *shapiro.test)*, onde a hipótese nula (H_0) indica que a variável segue uma distribuição normal enquanto a hipótese alternativa (H_1) indica que a variável não segue uma distribuição normal. Na aplicação de testes de hipóteses foi, ao longo de todo o trabalho, considerado um nível de significância (α) de 0,05 (5%). Os resultados obtidos podem ser consultados na Tabela 4.

Variável	Valor-p	Variável	Valor-p	Variável	Valor-p
Idade	2e-04	**AQ**	0.4754	**DG**	0.0029
ACD	0.0019	**ARMD**	0.8967	**DI1**	1e-04
ACE	0.0082	**ARME**	0.5371	**DI2**	0
AMD	0.5259	**LMRMD**	0.0992	**AGD**	0.1504
AME	0.7489	**LMRME**	0	**AGE**	0.3102

Tabela 4 – Resultados do teste de normalidade

Como é possível verificar, há diversas variáveis com valores-p superiores a α (0.05), não existindo evidência estatística que essas variável não sigam uma distribuição normal e, portanto, deverão ter uma distribuição próxima da distribuição normal. É o caso das variáveis AMD, AME, AQ, ARMD, ARME, LMRMD, AGD e AGE. Para as restantes variáveis (idade, ACD, ACE, LMRME, DG, DI1 e DI2) foi rejeitada a hipótese nula e, deste modo, podemos concluir que existe evidência estatística que essas variáveis não seguem uma distribuição normal.

Foi igualmente aplicado o mesmo teste de normalidade às mesmas variáveis em cada sexo. Os resultados indicam que a normalidade é rejeitada em ambos os sexos nas variáveis idade, LMRME e DG; é rejeitada unicamente no sexo feminino nas variáveis ACD, ACE e LMRMD; é rejeitada unicamente no sexo masculino nas variáveis DI1 e DI2; não sendo rejeitada em ambos os sexos nas variáveis AMD, AME, AQ, ARMD, ARME, AGD e AGE.

De seguida foi feita a comparação das variâncias nas variáveis quantitativas entre as duas categorias da variável sexo, recorrendo ao desvio padrão e ao teste de homogeneidade das variâncias, onde a hipótese nula (H_0) corresponde à igualdade das variâncias nos dois sexos, enquanto a hipótese alternativa (H_1) a variância da variável em análise é diferente nos dois sexos. Nas variáveis em que não foi rejeitada a normalidade em nenhum dos sexos foi aplicado o teste F (função *var.test*), caso contrário foi aplicado o teste de Levene com base na mediana (função *leveneTest*). Os resultados obtidos estão resumidos na Tabela 5.

	Desvio Padrão		**Var.test**	**leveneTest**
Variável	Sexo Feminino	Sexo Masculino	Variável ~ Sexo	Variável ~Sexo
Idade	12.4874	14.7711	0.0901	0.0487
ACD	0.5025	0.5984	0.078	0.0734
ACE	0.5141	0.5254	0.8207	0.5116
AMD	0.3066	0.314	0.8046	
AME	0.2772	0.3038	0.3542	
AQ	0.2992	0.3277	0.358	
ARMD	0.4323	0.472	0.374	
ARME	0.4426	0.4685	0.5626	
LMRMD	0.2918	0.2781	0.6361	0.9455
LMRME	0.2778	0.3368	0.0519	0.2979
DG	1.0313	0.9483	0.0519	0.7068
DI1	1.1398	1.3167	0.1453	0.8288
DI2	0.8344	1.2063	2e-04	0.8755
AGD	5.7847	6.4808	0.2509	
AGE	5.9126	6.5331	0.313	

Tabela 5 – Resultados do teste de igualdade de variâncias

Apenas na variável DI2 rejeitamos a hipótese H_0, para um valor-*p* de $2\times10^{-04} < \alpha$ no teste F. Assim, segundo este teste, existe evidência estatística que a variável DI2 tem uma variância diferente nos dois sexos. Contudo, o teste de Levene não revela esta diferença.

Comparamos as médias das variáveis entre os dois sexos, e efetuamos o teste t (função *t.test*) para a comparação de médias de amostras independentes, considerando a normalidade e a igualdade de variância de acordo com o teste prévio (o argumento *var.equal* será verdadeiro ou falso consoante o resultado de homogeneidade das variáveis previamente realizado). Assim, neste teste onde a hipótese nula (H_0) corresponde à igualdade das médias nos dois sexos, enquanto a hipótese alternativa (H_1) indica que as médias da variável em análise são diferentes nos dois sexos.

Os resultados obtidos são apresentados na Tabela 6 onde, apesar de algumas varáveis não serem caracterizadas pela distribuição normal, optou-se por indicar o resultado para todas as variáveis.

	Médias		**t.test**
Variável	Sexo Feminino	Sexo Masculino	Variável ~ Sexo
Idade	46.8276	47.8333	0.5973
ACD	5.9306	6.5365	0
ACE	5.9004	6.4654	0
AMD	3.0244	3.2924	0
AME	3.0252	3.3183	0
AQ	3.1047	3.3909	0
ARMD	6.3976	7.0782	0
ARME	6.4144	7.0555	0
LMRMD	2.6377	2.745	0.0081
LMRME	2.5678	2.7018	0.002
DG	16.1739	16.7831	0
DI1	17.5152	18.0742	0.0013
DI2	10.6873	11.0751	0.007
AGD	125.6552	124.9	0.379
AGE	126.1207	125.6444	0.5845

Tabela 6 – Resultados para as Médias

Considerando os valores-*p* obtidos para o teste *t*, apenas nas variáveis idade, AGD e AGE são superiores ao nível de significância α, não sendo rejeitada a hipótese nula. No entanto, as restantes variáveis apresentam diferenças significativas entre as médias dos dois sexos.

De seguida determinamos a mediana das variáveis em cada sexo e aplicamos o teste de Wilcoxon (função *wilcox.test*) para comparar a distribuição da amostra (cf.Tabela 7) .

As hipóteses testadas são:

H_0: A variável tem a mesma distribuição em ambos os sexos

H_1: A variável tem distribuições distintas nos dois sexos.

	Mediana		**Wilcox.test**
Variável	Sexo Feminino	Sexo Masculino	Variável ~ Sexo
Idade	46.5	46	0.6559
ACD	5.909	6.4895	0
ACE	5.816	6.4335	0
AMD	3.0085	3.2885	0
AME	3.008	3.299	0
AQ	3.1425	3.3725	0
ARMD	6.403	7.121	0
ARME	6.434	7.0445	0
LMRMD	2.6	2.743	0.0041
LMRME	2.5445	2.7215	2e-04
DG	16.1105	16.667	0
DI1	17.329	18.088	3e-04
DI2	10.607	11.006	0.0081
AGD	125	125	0.4521
AGE	126	126	0.5172

Tabela 7 – Resultados para as Medianas

Uma vez mais, unicamente para as variáveis idade, AGD e AGE obtivemos valores-p superiores a α. No entanto, para as restantes variáveis rejeitamos H_0 e podemos concluir que as variáveis têm distribuições diferentes nos dois sexos. Notemos que os resultados do teste de Wilcoxon são convergentes com os obtidos no teste t, uma vez que evidenciam a existência de discriminação em todas as variáveis avaliadas com exceção da idade, AGD e AGE.

Efetuamos a análise da correlação entre as variáveis do conjunto de dados e obtivemos a matriz de correlação apresentada na Figura 25. Na Figura 26 é possível verificar os índices de correlação e a distribuição das variáveis por sexo.

Através da análise visual é possível verificar que existem variáveis como AMD, AME, ARMD, ARME, ACD e ACE que se encontram positiva e fortemente correlacionadas entre si, ou seja, têm valores perto de 1. E podemos também concluir que variáveis como a idade, AGD e AGE que apresentam valores maioritariamente perto de 0, ou seja, não possuem nenhum tipo de correlação entre elas (exceto entre AGD e AGE, estando estas variáveis negativamente correlacionadas com LMRMD, LMRM e DG).

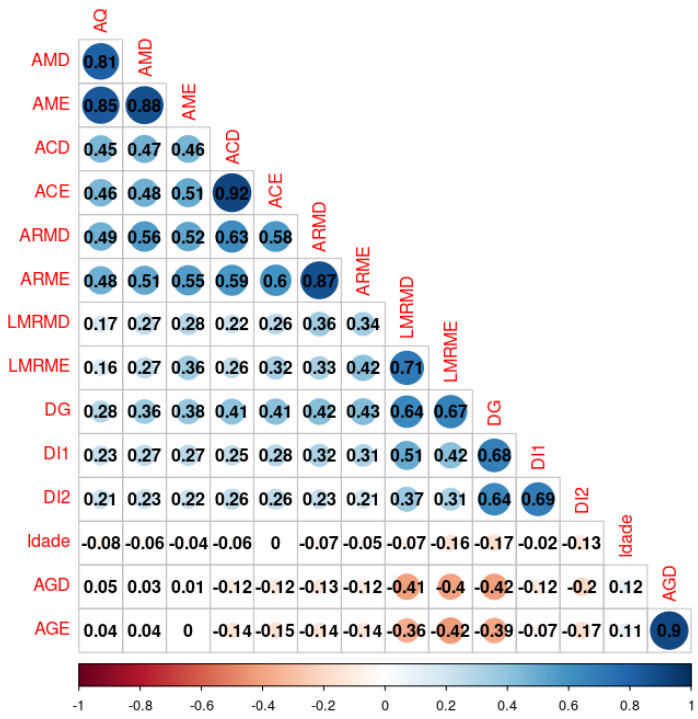

Figura 25 – Matriz de correlação

Na Figura 26 é possível verificar os valores para a correlação entre variáveis, porém conseguimos avaliar a distribuição para cada um dos sexos, assim como a capacidade discriminante das variáveis estudadas.

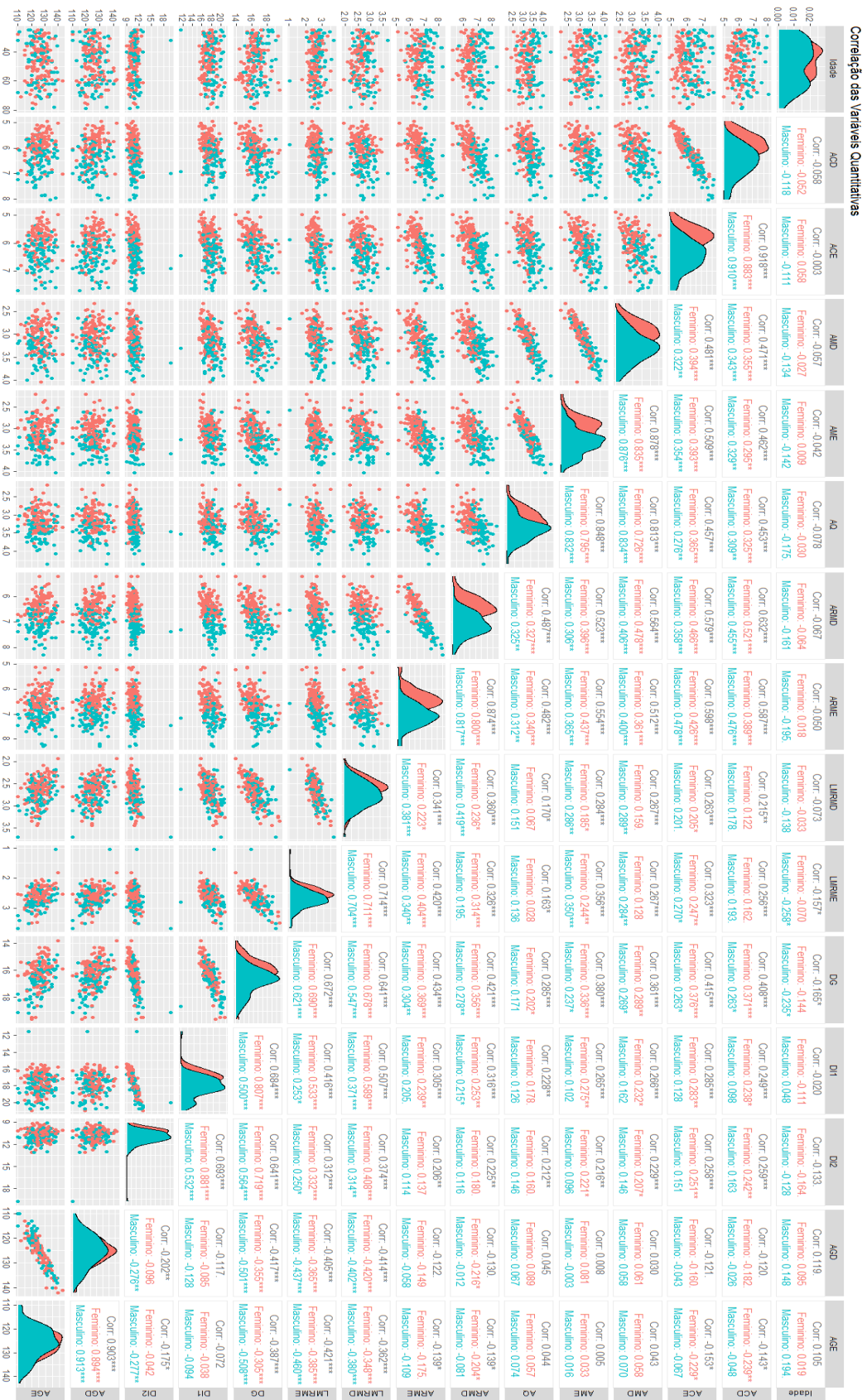

Figura 26 – Gráfico das variáveis quantitativas

Para o modelo de classificação que criamos utilizando a regressão logística, obtivemos os resultados AUC na curva ROC da Figura 27. O modelo com os melhores resultado foi criado com a amostra completa, obtendo um valor de AUC de 0.904. É possível constatar que para as outras combinação do conjuntos de dados os valores obtidos são muito similares, apesar das diferenças nos pontos de corte. Por outro lado, destaca-se os resultados obtidos nas amostras de teste que não são muito diferentes dos obtidos na amostra de treino, revelando assim capacidade de generalização, isto é, do modelo ser aplicado a outras amostras não utilizadas no processo de aprendizagem.

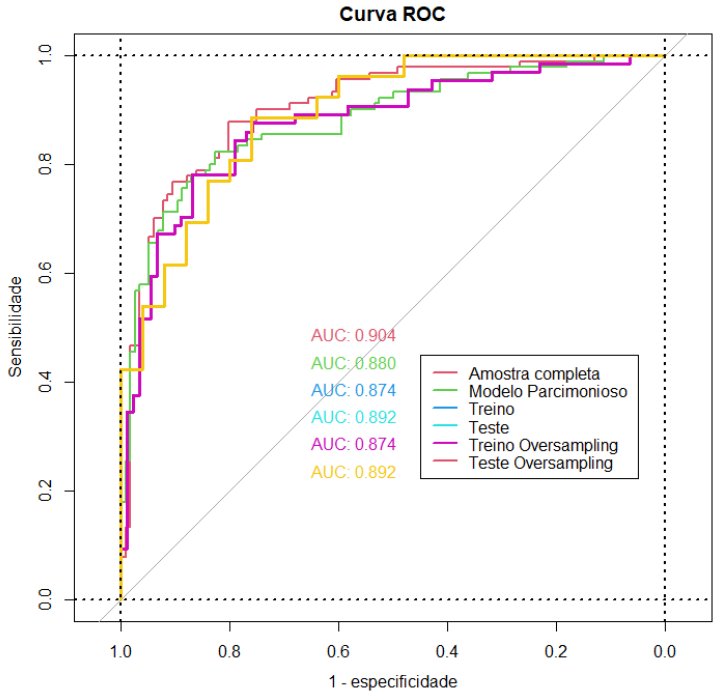

Figura 27 - Resultados regressão logística

Para valores de AUC acima de 0.9, o AUC indica-nos que o modelo tem um excelente desempenho e que possui uma elevada capacidade de classificação. Em anexo (Anexo I), na Tabela 10 e na Tabela 11, podemos consultar os resultados obtidos para os diversos modelos, com a acurácia, especificidade, entre outras medidas.

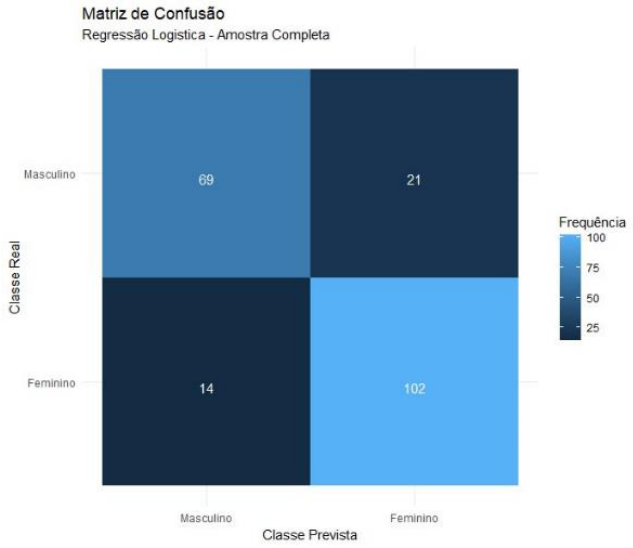

Figura 28 - Matriz de confusão da regressão logística

Na Figura 28 temos a matriz de confusão para o modelo de regressão logística com melhor resultado, o modelo criado com amostra de testes completa.

Após a análise aos dados, verificamos que o conjunto de dados não apresenta *outliers* multivariados. No entanto, através da análise visual é possível verificar um distanciamento das observações (cf. Figura 29).

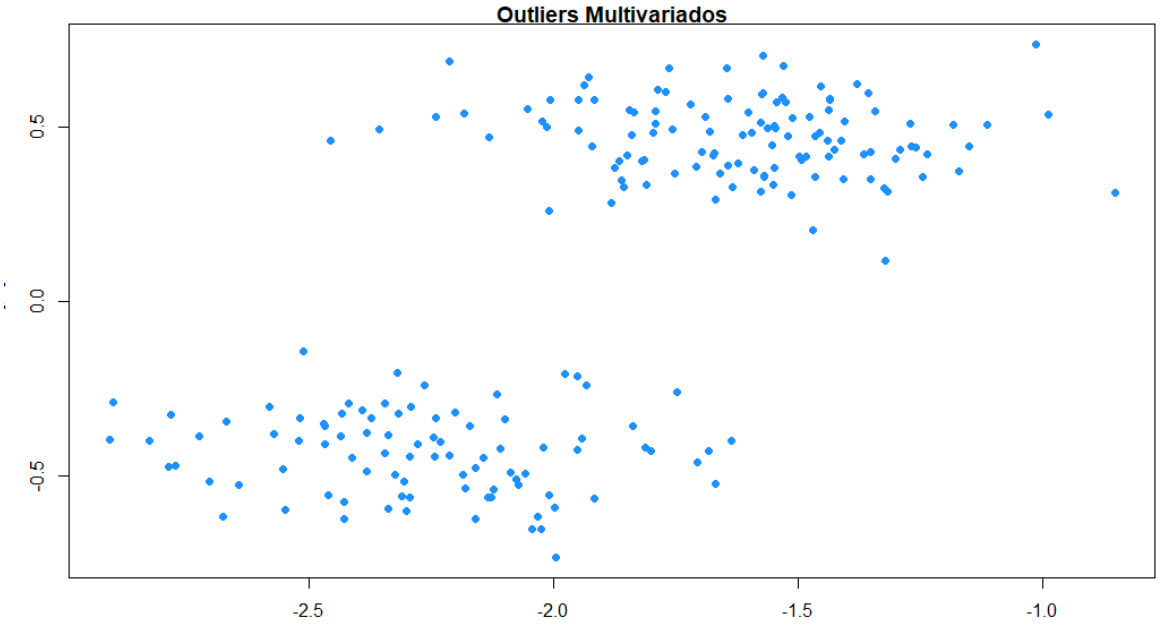

Figura 29 - Gráfico *outliers* multivariados

Foi feito o teste de normalidade multivariado de Shapiro-Wilk (função *mshapiro.test*) para verificar se as variáveis quantitativas seguem uma distribuição normal multivariada, onde a hipótese nula (H_0) indica que as variáveis quantitativas seguem uma distribuição normal multivariada, enquanto a hipótese alternativa (H_1) indica que as variáveis quantitativas não seguem uma distribuição normal multivariada.

O valor obtido para o valor-*p* foi inferior a 2.2×10^{-16}, logo menor que α. Desta forma, rejeitamos a hipótese nula e, portanto, existe evidência estatística que as variáveis quantitativas não seguem uma distribuição normal multivariada.

Com o objetivo de reforçar os resultados obtidos anteriormente, foi efetuado o teste de normalidade multivariada para cada sexo (variável qualitativa), obtendo um valor-*p* para o sexo masculino de 2.2×10^{-16} e para o sexo feminino de 7.145×10^{-14}. Assim, há evidência que as variáveis não seguem uma distribuição normal multivariadas em cada sexo.

Para verificar a igualdade da matriz de variância-covariância em ambas as categorias do variável sexo, foi efetuado o teste de homogeneidade *Box's M-test*. Neste teste temos as seguintes hipóteses:

H_0: Matriz de variância-covariância das variáveis quantitativas é igual nas duas categorias da variável sexo.
H_1: Matriz de variância-covariância das variáveis quantitativas não é igual nas duas categorias da variável sexo.

Obtendo assim um valor-*p* de 9.489×10^{-10}, menor que o nível de significância, logo rejeitando a hipóteses nula. Desta forma, existe evidencia estatística que a matriz de variância-covariância das variáveis quantitativas não é igual nas duas categorias da variável sexo.

Também para os modelos de análise discriminante linear (LDA), criamos uma serie de modelos com as diferentes combinações descritas na metodologia. Obtivemos modelos com igual desempenho, sendo que o melhor resultado obtido foi de AUC 0.925 para a amostra de teste, treino com *oversampling* e treino com *oversampling* e validação cruzada (Figura 33).

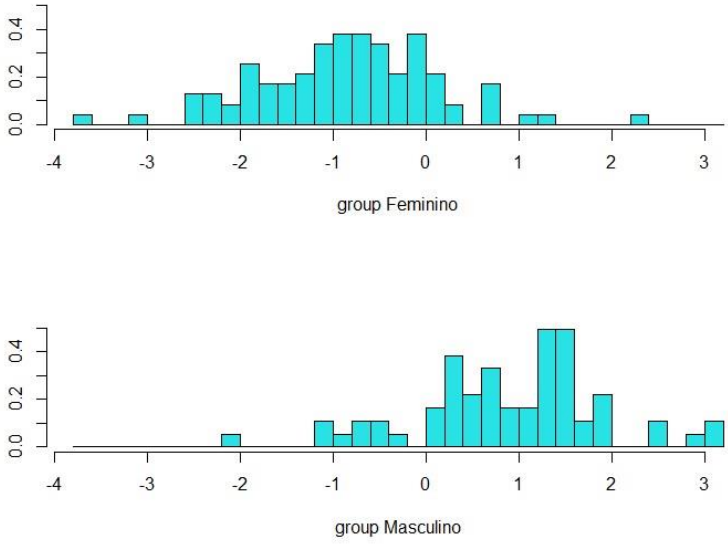

Figura 30 – Histograma do modelo LDA por sexo

Os resultados obtidos para todos os modelos LDA podem ser consultados em anexo (Anexo I) na Tabela 12. Na Figura 30 podemos ver o histograma do modelo LDA com a amostra de treino com *oversampling*, e é possível verificar que existe uma sobreposição das classes, o que nos leva a concluir que o modelo apesar de um valor AUC elevado, tem alguma dificuldade na classificação do sexo.

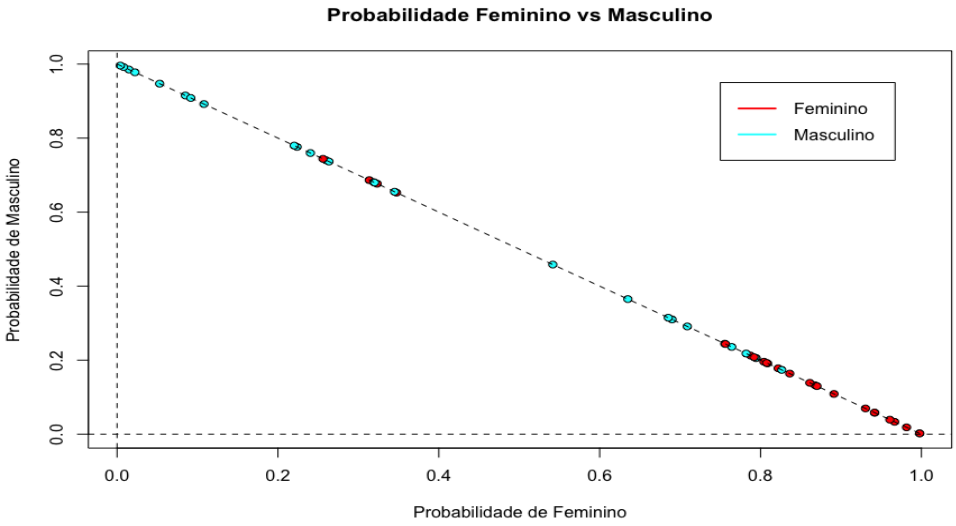

Figura 31 - Gráfico dispersão das probabilidades

O gráfico de dispersão das probabilidades na Figura 31 mostra-nos a dispersão das probabilidade de pertencer a classe “Feminino” ou “Masculino”. É possível constatar alguma sobreposição das classes através da análise visual do gráfico, no entanto a maioria das probabilidades está bem separada o que nos leva concluir que o modelo está a classificar

bem as classes. A Figura 32 mostra a matriz de confusão associada à análise discriminante linear.

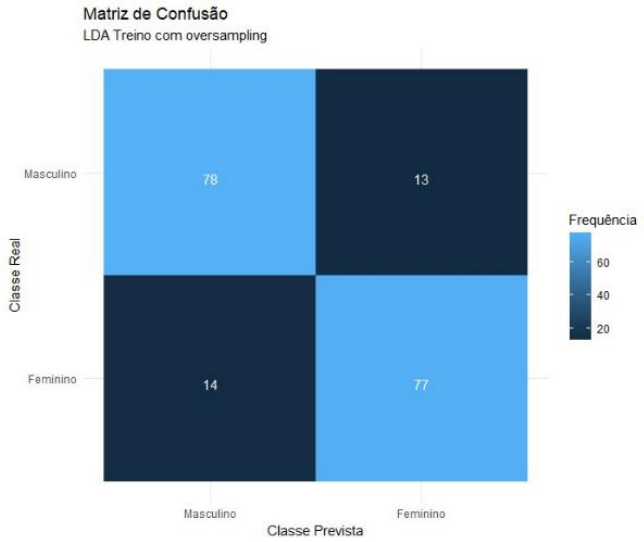

Figura 32 - Matriz de confusão LDA

Os resultados para todos os modelos criados podem ser observados na Figura 33, na qual temos 3 modelos com os mesmo valores de 0.925. Deste modo, têm um valor de AUC próximo de 1, o que significa que estes modelos têm um desempenho excelente na classificação. Mesmo o valor mais baixo que se encontra na amostra de treino é um valor acima de 0.8, o que significa um desempenho muito bom apesar de duas das condições de aplicabilidade do modelos não se verificarem, nomeadamente a normalidade e a homocedasticidade.

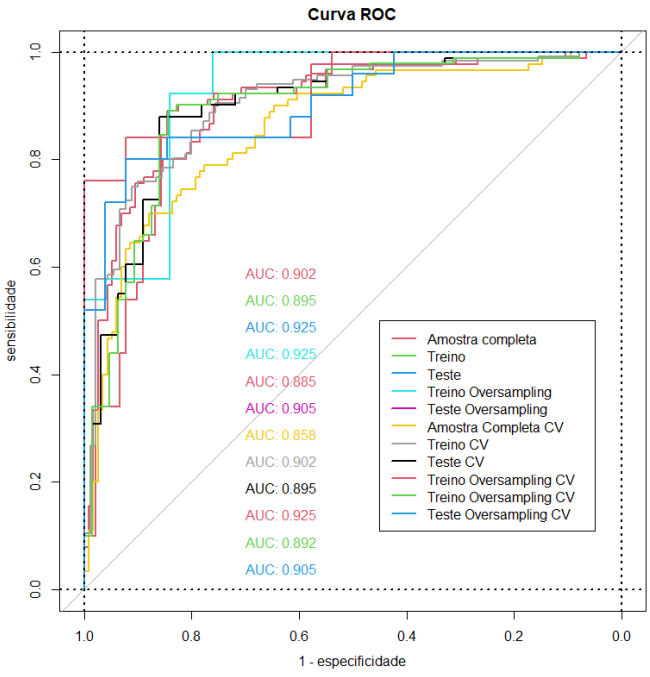

Figura 33 – Resultados dos modelos LDA

À semelhança do que foi criado com os modelos de regressão logística e análise discriminante linear, também para a análise discriminante quadrática (QDA) criámos uma serie de modelos para as diferentes amostras. E o modelo com o melhor valor de AUC foi o modelo criado com a amostra de treino com a aplicação de *oversampling*, com um valor de AUC de 0.937. Na Figura 34 podem ser comparados os diferentes modelos de aplicação da QDA.

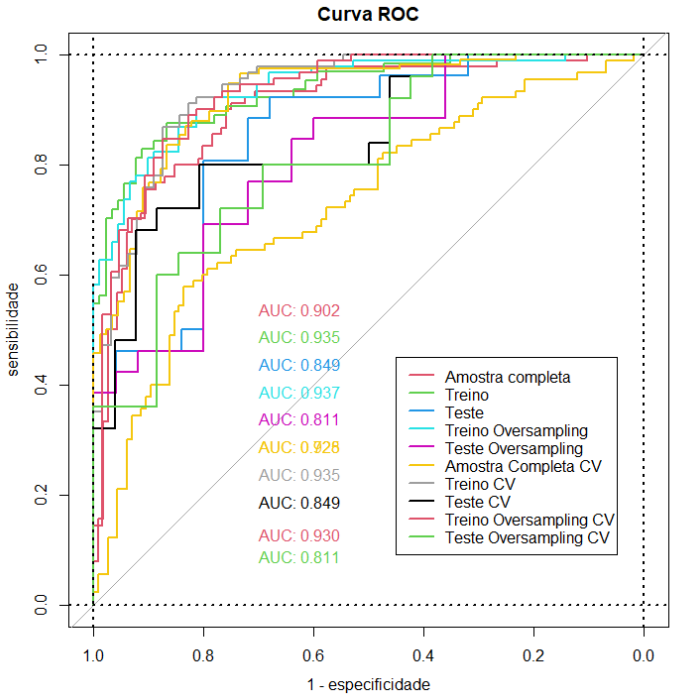

Figura 34 – Resultados QDA

Em anexo (Anexo I), na Tabela 13, estão os resultados obtidos para os modelos de análise discriminante quadrática (QDA) e, na Figura 35, podemos observar a matriz de confusão associada.

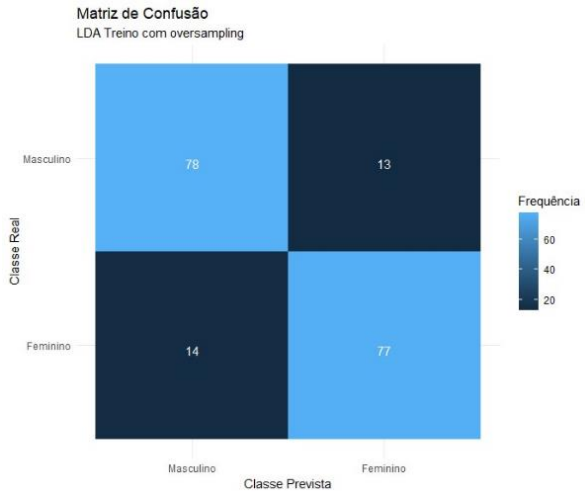

Figura 35 - Matriz de Confusão QDA

Nos modelos criados utilizando as árvores de decisão, a métrica utilizada foi a acurácia, por não termos a métrica AUC disponível (apesar de existirem funções que permitem representar a curva ROC, a sua interpretação não é imediata, nem comparável com os modelos previamente analisados). É possível verificar os valores obtidos para este modelos, recorrendo às diversas amostras, em anexo (Anexo I) na Tabela 14 e na Figura 36. Nesta figura são comparados os valores da acurácia, sensibilidade, especificidade, valor preditivo positivo e valor preditivo negativo. O modelo com melhor resultado utilizou a amostra de treino com *oversampling*, obtendo uma acurácia de 83%.

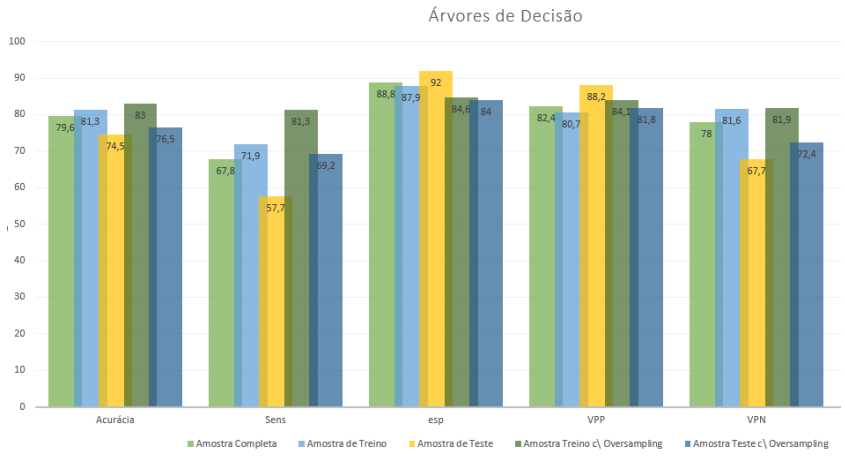

Figura 36 - Resultados árvores de decisão

Na Figura 37 é possível verificar as variáveis que o modelo está a utilizar para fazer a divisão, o valor de corte e os ramos que indicam as diferentes saídas possíveis. Assim, na classificação apresentada unicamente foram utilizadas três variáveis: ARMD, ARME e AME.

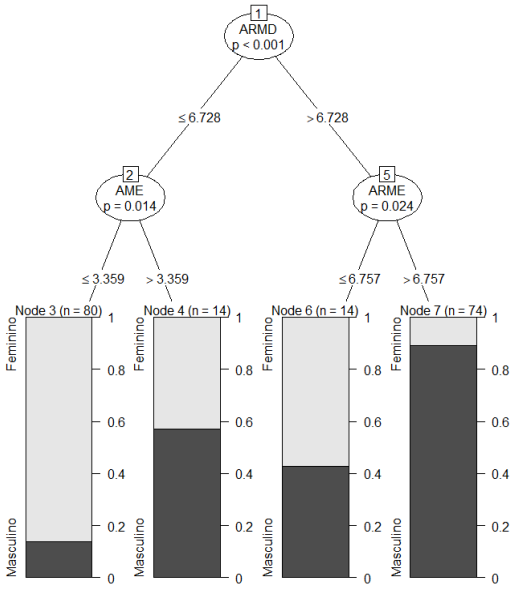

Figura 37 – Representação da árvore de decisão

Na Figura 38 podemos verificar que o modelo tem um bom desempenho na identificação dos verdadeiros positivos e verdadeiros negativos, significando que o modelo tem capacidade para classificar corretamente a maioria das amostras positivas e negativas.

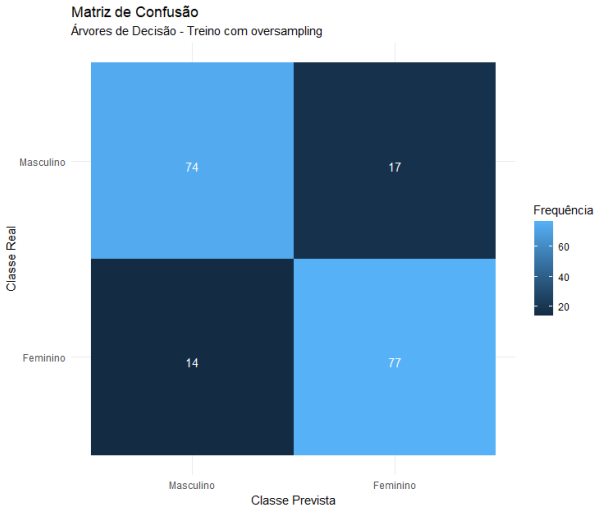

Figura 38 - Matriz de confusão árvore de decisão

No modelo seguinte foi aplicada a metodologia Naive Bayes para classificar o sexo. Os modelos obtiveram quase todos uma fiabilidade semelhante, com AUC próxima 0.87, quer nas amostras de treino quer nas amostras de teste (cf. Figura 39). Esta diferença diminuta entre os resultados da amostra de treino em relação à amostra de teste revela que o modelo parece ser generalizável, pois não perde fiabilidade quando aplicada a uma amostra que não foi utilizada na aprendizagem.

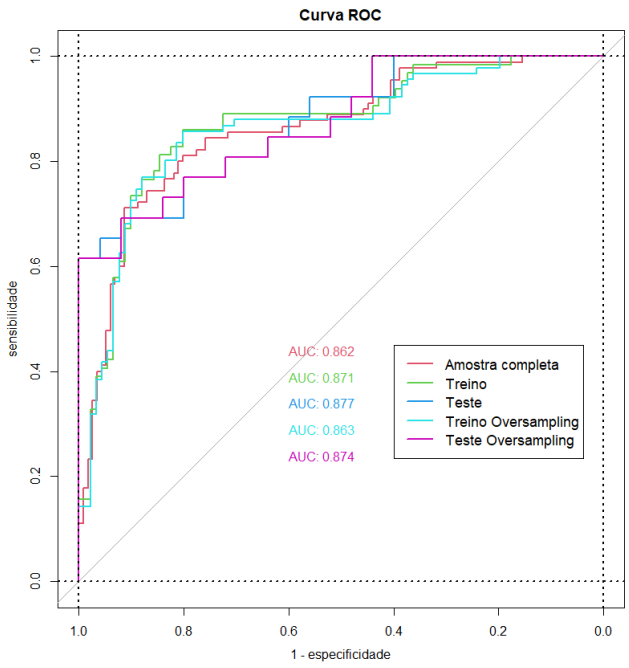

Figura 39- Curva ROC Naive Bayes

Com um valor de AUC próximo de 0.87 o modelo tem muito bom poder de classificação, conforme se constata igualmente na matriz de confusão apresentada na Figura 42. Na Tabela 15 do Anexo I estão presentes as outras métricas obtidas para os modelos criados.

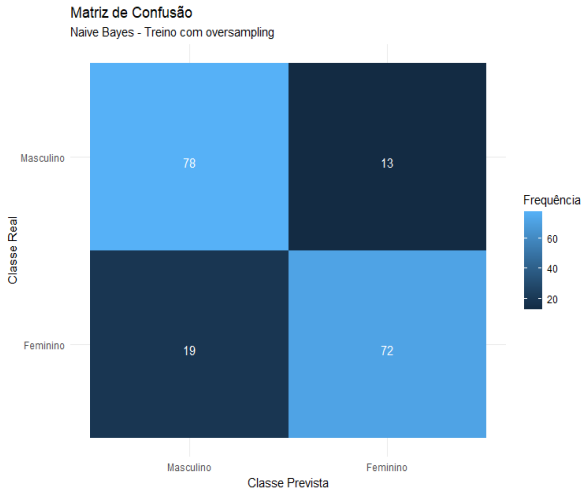

Figura 40 – Matriz de confusão Naive Bayes

Para os modelos criados com o método *k*-vizinhos mais próximos, foi necessário utilizar a amostra de dados com a aplicação do comando *scale*() que permite padronizar as variáveis quantitativas, o que significa subtrair a média e dividir pelo desvio padrão para que as variáveis tenham média zero e desvio padrão igual a um. Na Tabela 16 (Anexo I) estão os valores de acurácia, sensibilidade, especificidade, valor preditivo positivo e valor preditivo negativo obtidos pelos modelos criados. Na Figura 43 podemos verificar a matriz de confusão.

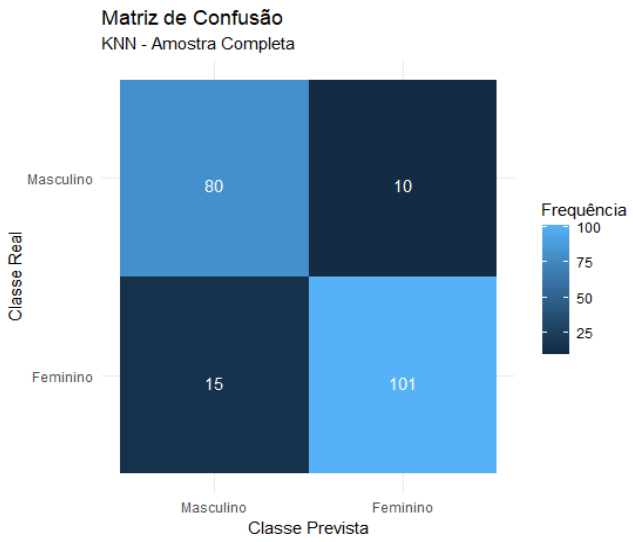

Figura 41 - Matriz de confusão KNN

Na Figura 42 é possível constatar que o melhor valor de acurácia obtido é igual nos dois modelos criados com a amostra completa, tanto para o número de vizinho 2 como para 3.

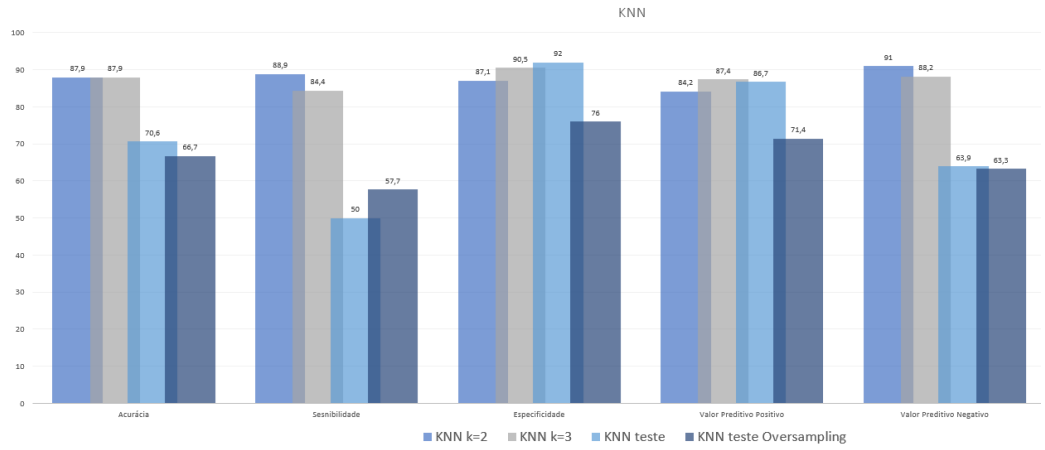

Figura 42 - Resultados KNN

À semelhança dos modelos criados com as árvores de decisão e os *k*-vizinhos mais próximos, também para os modelos de máquina de vetores de suporte foi utilizada a acurácia para determinar a qualidade do modelo. O modelo com melhor acurácia foi criado com o conjunto de dados de teste, obtendo 74,5% de acurácia (cf. Figura 43).

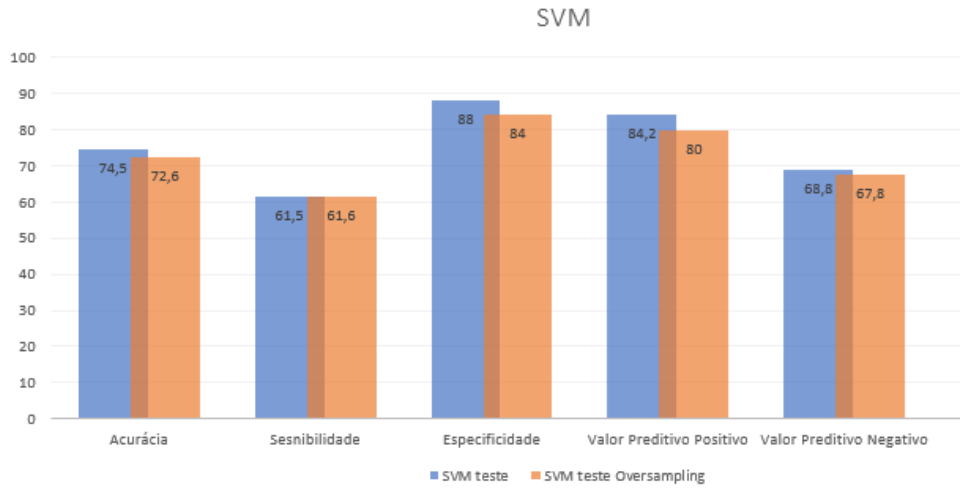

Figura 43 – Resultados Máquinas de Suporte de Vetores

Na Figura 44 podemos ver a matriz de confusão para o modelo de máquinas de suporte de vetores.

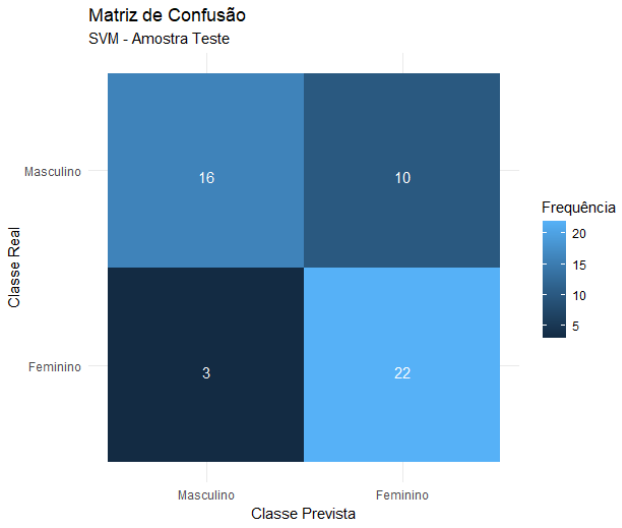

Figura 44 – Matriz de confusão SVM

Dos modelos criados com o método das florestas aleatórias, o melhor resultado obtido foi para o modelo criado com a amostra de teste, com um valor de AUC de 0.866 (Figura 45).

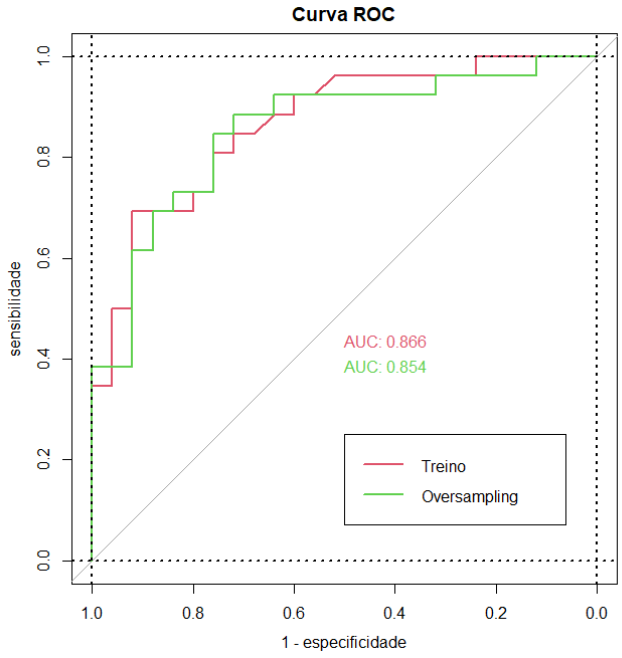

Figura 45 – Resultados das florestas aleatórias

O resultado obtido de 0.866 indica-nos que o modelo tem um ótimo desempenho e capacidade de discriminação. Na Figura 46 podemos verificar a Matriz de confusão para o modelo criado com as florestas aleatórias.

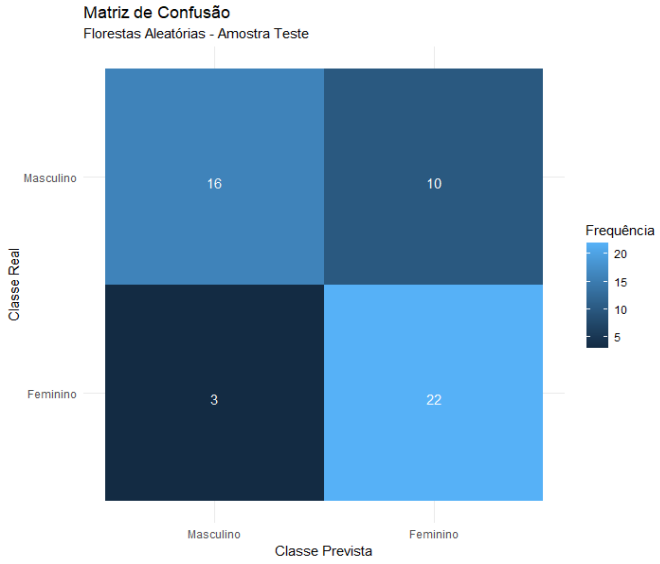

Figura 46 - Matriz de confusão florestas aleatórias

4.3. Comparação dos resultados com outros trabalhos relacionados

Ao analisar os resultados obtidos no nosso projeto em relação a trabalhos relacionados, procuramos identificar possíveis semelhanças, divergências e fornecer um contexto através de trabalhos prévios. Depois da pesquisa sobre trabalhos realizados dentro do mesmo tema deste projeto para a caraterização do estado de arte, os trabalhos apresentados na Tabela 8 foram os que encontramos com a maior semelhança e que, desta forma, nos permitiram estabelecer um ponto de comparação.

Alguns destes trabalhos apresentam o tamanho do conjunto de dados, o método de classificação utilizado e a métrica usada para avaliar a performance, a acurácia.

Autor	**Ano**	**Conjunto de dados**	**Método**	**Acurácia**
Este trabalho	2023	206	Vários Métodos	74,5 – 95%
Vijayakumari et *al.* [46]	2023	285	Redes Neuronais	94%
Atas [25]	2022	24.000	Redes Neuronais Profundas	97.25%
Balan *et al.* [47]	2022	-	Vários Métodos	77.5 – 93.8%
Victoria [14]	2021	206	Regressão Logística	83%
Nithya *et al.* [48]	2021	-	Redes Neuronais Profundas	95%
Rajee *et al.*[49]	2021	1000	Redes Neuronais	98.27%
Vila-Blanco *et al.* [50]	2020	3400	Redes Neuronais	75-96,24%
Milošević et *al.* [51]	2019	4000	Redes convolucionais	96.87%
Gamba *et al.* [52]	2016	160	Métodos Estatísticos	95.1%
Sairam *et al.* [21]	2016	200	Métodos Estatísticos	76-79.5%
Damera *et al.* [22]	2016	80	Métodos Estatísticos	83.3%

Kim *et al.* [53]	2013	104	Análise Discriminante	65.4 – 89.4%
Indira *et al.* [54]	2012	100	Métodos Estatísticos	75.8%
Saini *et al.* [55]	2011	116	Métodos Estatísticos	80.20%

Tabela 8 - Resultados de Trabalhos Relacionados

Os resultados obtidos neste projeto variam de acordo com os modelos criados, e as amostras de dados utilizadas. Na Figura 47 podemos verificar que o método com a acurácia mais baixa foi as máquinas de vetores de suporte com 74,5% de acurácia, e o maior foi 95% com a aplicação das redes neuronais profundas VGG16 ao conjunto de dados disponível.

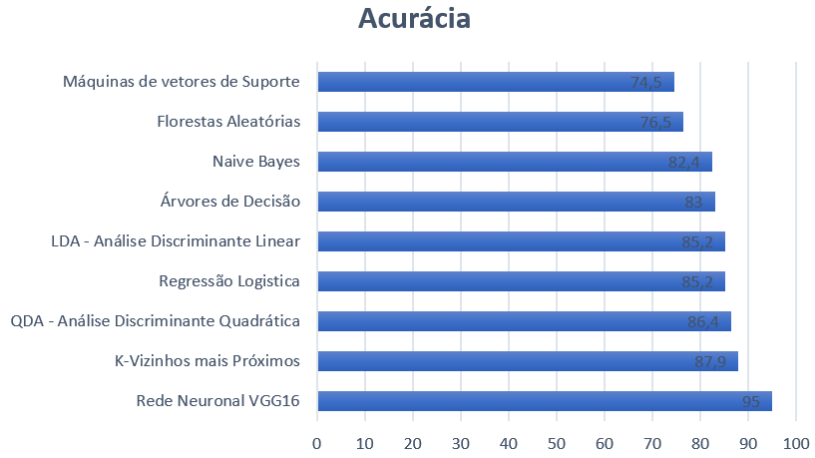

Figura 47 – Resultados da acurácia por método de classificação

Analisando os resultados dos vários trabalhos na área, os valores que obtivemos no decorrer do projeto são bastante similares havendo apenas algumas diferenças que acreditamos se deverem a dimensão da nossa amostra de dados. Por exemplo, comparando com os valores obtidos por Balan et al. [47], trabalho onde também foram aplicados vários métodos, nós obtemos melhor acurácia para as redes neuronais e para o naive Bayes, no entanto eles obtiveram uma acurácia de 90% nas florestas aleatórias e 82.5% nas máquinas de vetores de suporte. Para as redes neuronais, o trabalho com o valor mais alto da acurácia foi obtido por Rajee et al. [49] com um valor de 98.27% e, nesse trabalho, foi utilizada a rede neuronal Resnet50, no entanto neste projeto teve um desempenho claramente inferior.

5. Conclusão

Neste capítulo iremos resumir algumas das principais conclusões do trabalho, como também abordar eventuais desafios e limitações encontradas durante o processo, tais como eventuais melhorias e direções futuras.

5.1. Discussão dos resultados obtidos em relação aos objetivos e hipóteses

Podemos concluir que os resultados obtidos nesse projeto excederam as nossas expectativas iniciais. Devido às limitações identificadas no início do projeto, e que se encontram descritas abaixo, julgámos que os resultados iriam ficar aquém dos valores que outros investigadores apresentam nos diversos trabalhos publicados. Os resultados obtidos que se encontram no Capítulo 4, e em anexo (Anexo I) de forma mais resumida, permitem-nos verificar que apenas os modelos criados com recurso às máquinas de vetores, florestas aleatórias e naïves Bayes, estão abaixo do valor de referência. Valor esse obtido pela Ionel [14] utilizando o mesmo conjunto de dados e recorrendo a métodos de regressão logística na tese de mestrado "Diagnose sexual baseada em parâmetros radiológicos craniomandibulares".

Consultando a bibliografia deste tema e tendo em consideração o método atual utilizado pelos profissionais para classificação sexual, que se baseia nos critérios métricos craniomandibulares, podemos concluir que existem modelos de classificação e redes neuronais que permitem obter melhores resultados do que o método utilizado atualmente.

5.2. Contribuições da pesquisa para a área de análise de imagem médica com recurso a redes neurais

Como foi identificado e mencionado acima existem diversos trabalhos de classificação utilizando radiografias panorâmicas, no entanto a grande maioria dos trabalhos abordam a classificação e/ou estimação da idade. A classificação sexual, quando mencionada, na maioria dos casos é deixada para segundo plano. No entanto, nos últimos anos têm sido apresentados alguns trabalhos nesta área com foco na classificação sexual, em parte pelo

crescente interesse nas redes neuronais, porém ainda num número muito inferior ao que podemos encontrar para o estudo da idade.

De acordo com os valores obtidos e que podem ser consultados no Capítulo 4, a rede VGG16 com uma acurácia de 95% pode ser utilizada para classificar as radiografias panorâmicas sem que seja necessária grande preparação. O valor obtido de 95% de acurácia é bastante superior ao obtido atualmente para classificação através dos métodos de medição 83% [14] (resultados na Tabela 9 do anexo I) o que nos leva a concluir que a utilização de redes neuronais é um processo válido, e que com o avanço das redes neuronais e com trabalhos, como o aqui desenvolvido, fica demonstrada a capacidade das redes neuronais em classificar. Acreditamos, deste modo, que a aplicação de *Machine Learning* nesta área será, num futuro próximo, uma clara mais-valia.

5.3. Limitações da pesquisa e sugestões para trabalhos futuros

No decorrer do projeto a maior limitação encontrada, e aquela que pode ter a maior implicação nos resultados, é a dimensão do conjunto de dados. Apesar de terem sido aplicadas técnicas para tentar minimizar os efeitos, como por exemplo a aplicação de *data augmentation* nas redes neuronais de forma a aumentar o número de imagens, e a aplicação de *oversampling* no conjunto de dados numa tentativa de equilibrar as classes, acreditamos que com um conjunto de dados maior iríamos obter melhores resultados. Outra das questões que nos deparamos foi o número limitado de trabalhos nesta área, e os trabalhos existentes focarem-se na componente técnica da classificação de imagem, e na apresentação de resultados, e não tanto na componente prática das redes neuronais escolhidas, e dos parâmetros utilizados. Assim, o nosso projeto foca-se mais na aplicação, e não no aspeto teórico das redes neuronais, e do métodos de classificação utilizados.

Como sugestão para trabalhos futuros, seria a aplicação dos modelos de classificação e as redes neuronais criadas no projeto para o desenvolvimento de uma aplicação web ou móvel, que pudesse ser utilizada pelos profissionais da área para classificar as radiografias panorâmicas.

Outra sugestão seria usar um conjunto de dados significativamente maior e voltar e aplicar os modelos de classificação com o intuito de perceber o eventual impacto nos resultados, e observar se as variáveis, que identificámos como sendo as que têm maior capacidade discriminante, se manteriam.

Por fim, no caso das redes neuronais, voltar a treinar as redes com o conjunto de dados maior, testar outros valores para os hiperparâmetros, e perceber se a VGG16 continuaria a ser a rede com melhor performance das redes testadas, visto que a maioria dos trabalhos na área identificaram outras redes como tendo melhores resultados.

6. Bibliografia

[1] A. Schmitt, E. Cunha, and J. E. S. Pinheiro, “Forensic anthropology and medicine: complementary sciences from recovery to cause of death,” 2006.

[2] K.-S. Hu, K.-S. Koh, S.-H. Han, K.-J. Shin, and H.-J. Kim, “Sex determination using nonmetric characteristics of the mandible in Koreans,” *J Forensic Sci*, vol. 51, no. 6, pp. 1376–1382, 2006.

[3] M. Durić, Z. Rakočević, and D. Donić, “The reliability of sex determination of skeletons from forensic context in the Balkans,” *Forensic Sci Int*, vol. 147, no. 2–3, pp. 159–164, 2005.

[4] M. R. Vaishali, K. S. Ganapathy, and K. Srinivas, “Evaluation of the precision of dimensional measurements of the mandible on panoramic radiographs,” *Journal of Indian Academy of Oral Medicine and Radiology*, vol. 23, no. Suppl 1, pp. S323–S327, 2011.

[5] K. Samatha, S. M. Byahatti, R. A. Ammanagi, P. Tantradi, C. K. Sarang, and P. Shivpuje, “Sex determination by mandíbular ramus: A digital orthopantomographic study,” *J Forensic Dent Sci*, vol. 8, no. 2, p. 95, 2016.

[6] R. Schulze, F. Krummenauer, F. Schalldach, and B. d’Hoedt, “Precision and accuracy of measurements in digital panoramic radiography.,” *Dentomaxillofacial Radiology*, vol. 29, no. 1, pp. 52–56, 2000.

[7] C. Ozdemir, M. A. Gedik, and Y. Kaya, “Age Estimation from Left-Hand Radiographs with Deep Learning Methods.,” *Traitement du Signal*, vol. 38, no. 6, 2021.

[8] Ö. B. Dinler and C. B. Şahin, “Prediction of phishing web sites with deep learning using WEKA environment,” *Avrupa Bilim ve Teknoloji Dergisi*, no. 24, pp. 35–41, 2021.

[9] Ö. BATUR and N. Aydin, “An optimal feature parameter set based on gated recurrent unit recurrent neural networks for speech segment detection,” *Applied Sciences*, vol. 10, no. 4, p. 1273, 2020.

[10] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, “A survey of deep learning techniques for autonomous driving,” *J Field Robot*, vol. 37, no. 3, pp. 362–386, 2020.

[11] Y. Zhao, E. P. Wood, N. Mirin, S. H. Cook, and R. Chunara, "Social determinants in machine learning cardiovascular disease prediction models: a systematic review," *Am J Prev Med*, vol. 61, no. 4, pp. 596–605, 2021.

[12] F. Jiang *et al.*, "Artificial intelligence in healthcare: past, present and future," *Stroke Vasc Neurol*, vol. 2, no. 4, 2017.

[13] A. Esteva *et al.*, "A guide to deep learning in healthcare," *Nat Med*, vol. 25, no. 1, pp. 24–29, 2019.

[14] V. Ionel, "Diagnose sexual baseada em parâmetros radiológicos craniomandibulares," 2021. Dissertação de Mestrado Integrado em Medicina Dentária da Faculdade de Medicina Dentária da Universidade de Lisboa, 2021

[15] E. A. Hooton, *Up from the Ape*, vol. 10. Macmillan, 1946.

[16] W. M. Krogman, "Sexing skeletal remains," *The human skeleton in forensic medicine*, 1962.

[17] E. Giles, "Sex determination by discriminant function analysis of the mandible," *Am J Phys Anthropol*, vol. 22, no. 2, pp. 129–135, 1964, doi: https://doi.org/10.1002/ajpa.1330220212.

[18] B. C. Conrath, C. M. W. Daft, and W. D. O'Brien, "Applications of neural networks to ultrasound tomography," in *Proceedings., IEEE Ultrasonics Symposium,* 1989, pp. 1007–1010.

[19] D. L. Hudson, M. E. Cohen, and M. F. Anderson, "Determination of testing efficacy in carcinoma of the lung using a neural network model," in *Proceedings of the Annual Symposium on Computer Application in Medical Care*, 1988, p. 251.

[20] J. Coelho *et al.*, "Sex and age biological variation of the mandible in a Portuguese population-a forensic and medico-legal approaches with three-dimensional analysis," *Science & Justice*, vol. 61, no. 6, pp. 704–713, 2021.

[21] S. Vankadara, M. Geethamalika, B. Kumar, G. Naresh, and G. Raju Potturi, "Determination of sexual dimorphism in humans by measurements of mandible on digital panoramic radiograph," *Contemp Clin Dent*, vol. 7, pp. 434–439, Mar. 2016, doi: 10.4103/0976-237X.194110.

[22] A. Damera, J. Mohanalakhsmi, P. Yellarthi, and B. Rezwana, "Radiographic evaluation of mandíbular ramus for gender estimation: Retrospective study," *J Forensic Dent Sci*, vol. 8, p. 74, Mar. 2016, doi: 10.4103/0975-1475.186369.

[23] C. Pereira, M. Bernardo, D. Pestana, J. C. Santos, and M. C. de Mendonça, "Contribution of teeth in human forensic identification–discriminant function sexing odontometrical techniques in Portuguese population," *J Forensic Leg Med*, vol. 17, no. 2, pp. 105–110, 2010.

[24] L. Nithya and M. Sornam, "Deep Convolutional Networks in Gender Classification Using Dental X-Ray Images," in *Artificial Intelligence and Evolutionary Computations in Engineering Systems: Computational Algorithm for AI Technology, Proceedings of ICAIECES 2020*, 2022, pp. 375–380.

[25] I. Atas, "Human Gender Prediction Based on Deep Transfer Learning from Panoramic Radiograph Images," *arXiv preprint arXiv:2205.09850*, 2022.

[26] N. Vila-Blanco, P. Varas-Quintana, Á. Aneiros-Ardao, I. Tomás, and M. J. Carreira, "Automated description of the mandible shape by deep learning," *Int J Comput Assist Radiol Surg*, vol. 16, no. 12, pp. 2215–2224, 2021.

[27] I. Ilić, M. Vodanović, and M. Subašić, "Gender Estimation from Panoramic Dental X-ray Images using Deep Convolutional Networks," in *IEEE EUROCON 2019 -18th International Conference on Smart Technologies*, 2019, pp. 1–5. doi: 10.1109/EUROCON.2019.8861726.

[28] D. Milošević, M. Vodanović, I. Galić, and M. Subašić, "Estimating Biological Gender from Panoramic Dental X-Ray Images," in *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*, 2019, pp. 105–110. doi: 10.1109/ISPA.2019.8868804.

[29] A. Neelakantan *et al.*, "Adding Gradient Noise Improves Learning for Very Deep Networks." 2015.

[30] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[33] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[34] C. I. S. Cabral, "Aplicação do modelo de regressão logística num estudo de mercado," Universidade de Lisboa Faculdade de Ciências, 2013. Accessed: Sep. 18, 2023. [Online]. Available: https://repositorio.ul.pt/bitstream/10451/10671/1/ulfc106455_tm_Cleidy_Cabral.pdf

[35] S. Sawla, "Linear discriminant analysis," *Medium: Data Science*, vol. 5, 2018.

[36] G. J. McLachlan, *Discriminant analysis and statistical pattern recognition*. John Wiley & Sons, 2005.

[37] J. R. Quinlan, "Induction of decision trees," *Mach Learn*, vol. 1, pp. 81–106, 1986.

[38] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[39] S. Parsons, "Introduction to Machine Learning, Second Editon by Ethem Alpaydin, MIT Press, 584 pp., 55.00. ISBN 978-0-262-01243-0," *Knowl Eng Rev*, vol. 25, no. 3, p. 353, 2010.

[40] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer, 2009.

[41] B. Schölkopf, A. J. Smola, F. Bach, and others, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[42] L. Breiman, "Random Forests," *Mach Learn*, vol. 45, no. 1, pp. 5–32, 2001.

[43] A. Liaw, M. Wiener, and others, "Classification and regression by randomForest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[44] L. Breiman, "Random forests," *Mach Learn*, vol. 45, pp. 5–32, 2001.

[45] R. Santos, M. Felgueiras, J. P. Martins, and L. F. L. Ferreira, "Accuracy Measures for Binary Classification Based on a Quantitative Variable," *REVSTAT-Statistical Journal*, vol. 17, no. 2, pp. 223–244, 2019.

[46] B. Vijayakumari, S. Vidhya, and J. Saranya, "Deep learning-based gender classification with dental X-ray images," *Int J Biomed Eng Technol*, vol. 42, no. 1, pp. 109–121, 2023.

[47] H. Balan, A. F. Alrasheedi, S. S. Askar, and M. Abouhawwash, "An Intelligent Human Age and Gender Forecasting Framework Using Deep Learning Algorithms," *Applied Artificial Intelligence*, vol. 36, no. 1, p. 2073724, 2022.

[48] L. Nithya and M. Sornam, "Deep Convolutional Networks in Gender Classification Using Dental X-Ray Images," in *Artificial Intelligence and Evolutionary Computations in Engineering Systems: Computational Algorithm for AI Technology, Proceedings of ICAIECES 2020*, 2022, pp. 375–380.

[49] M. V Rajee and C. Mythili, “Gender classification on digital dental x-ray images using deep convolutional neural network,” *Biomed Signal Process Control*, vol. 69, p. 102939, 2021, doi: https://doi.org/10.1016/j.bspc.2021.102939.

[50] N. Vila-Blanco, R. R. Vilas, M. J. Carreira, and I. Tomás, “Towards deep learning reliable gender estimation from dental panoramic radiographs,” in *Proceedings 9th European Starting AI Researchers’ Symposium co-located with 24th European Conference on Artificial Intelligence (ECAI 2020)*, 2020.

[51] D. Milošević, M. Vodanović, I. Galić, and M. Subašić, “Estimating Biological Gender from Panoramic Dental X-Ray Images,” in *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*, 2019, pp. 105–110. doi: 10.1109/ISPA.2019.8868804.

[52] T. de Oliveira Gamba, M. C. Alves, and F. Haiter-Neto, “Mandíbular sexual dimorphism analysis in CBCT scans,” *J Forensic Leg Med*, vol. 38, pp. 106–110, 2016, doi: https://doi.org/10.1016/j.jflm.2015.11.024.

[53] D.-I. Kim, Y.-S. Kim, U.-Y. Lee, and S.-H. Han, “Sex determination from calcaneus in Korean using discriminant analysis,” *Forensic Sci Int*, vol. 228, no. 1, pp. 177.e1-177.e7, 2013, doi: https://doi.org/10.1016/j.forsciint.2013.03.012.

[54] A. P. Indira, A. Markande, and M. P. David, “Mandíbular ramus: An indicator for sex determination-A digital radiographic study,” *J Forensic Dent Sci*, vol. 4, no. 2, p. 58, 2012.

[55] V. Saini, R. Srivastava, R. K. Rai, S. N. Shamal, T. B. Singh, and S. K. Tripathi, “Mandíbular ramus: An indicator for sex in fragmentary mandible,” *J Forensic Sci*, vol. 56, pp. S13–S16, 2011.

7. Anexos

7.1. Anexo I – Resumo dos resultados obtidos

Ionel [14] **Regressão Logística**	**Acurácia**	**SENS**	**ESP**	**AUC**
Variável ARMD	79,1	85,0	71,0	86,1
Variável ARME	79,6	85,0	72,0	85,0
Variáveis Altura Corónidea Esq + ARMD + Altura corpo mandíbula D + ACME	83,0	86,0	79,0	88,7

Tabela 9 - Resultados da dissertação de mestrado [14]

Regressão Logística	**Acurácia**	**SENS**	**ESP**	**VPP**	**VPN**	**AUC**
Variável ARMD	79,1	71,1	85,3	79,0	79,2	86,1
Variável ARME	79,6	72,2	85,3	79,3	79,8	85,0
Variável ACMD	68	55,6	77,6	65,8	69,2	73,4
Variável ACME	67,5	58,9	74,1	63,9	69,9	75,7
Todas as Variáveis	83,0	76,7	87,9	83,1	82,9	90,0
Varieties ARMD + ACMD + ACME	83,0	75,6	88,8	84,0	82,4	88,0

Tabela 10 - Resultados obtidos regressão logística

Regressão Logística	**Acurácia**	**SENS**	**ESP**	**VPP**	**VPN**	**AUC**
Amostra de Treino (AME + ARMD)	84,5	76,6	90,1	84,5	84,5	89,7
Amostra de Teste (AME + ARMD)	74,5	61,5	88,0	84,2	68,8	90,9
Amostra de Treino (Idade + AQ + ARMD) c\ Oversampling	85,2	84,4	85,7	80,6	88,6	88,2
Amostra de Teste (Idade + AQ + ARMD) c\ Oversampling	70,6	57,7	84,0	79,0	65,6	86,2

Tabela 11 - Resultados obtidos regressão logística

LDA	Acurácia	SENS	ESP	VPP	VPN	AUC
Amostra completa	83,0	75,6	88,8	84,0	82,4	90,2
Amostra de Treino	85,2	78,1	90,1	84,8	85,4	89,5
Amostra de Teste	72,6	61,5	84,0	80,0	67,7	92,5
Amostra de Treino c\ Oversampling	85,2	85,7	84,6	84,8	85,6	88,5
Amostra de Teste c\ Oversampling	72,6	61,5	84,0	80,0	67,8	90,5
LDA CV	78,2	70,0	84,5	77,8	78,4	85,8
LDA CV 2	83,1	75,6	88,8	84,0	82,4	90,2
Amostra de Treino c\ Validação Cruzada	85,2	78,1	90,1	84,8	85,4	89,5
Amostra de Teste c\ Validação Cruzada	72,6	61,5	84,0	80,0	67,7	92,5
Amostra de Treino c\ Validação Cruzada e Oversampling	85,2	85,7	84,6	84,8	85,6	89,2
Amostra de Teste c\ Validação Cruzada e Oversampling	72,5	61,5	84,0	80,0	67,8	90,5

Tabela 12 - Resultados obtidos análise discriminante linear

QDA	Acurácia	SENS	ESP	VPP	VPN	AUC
Amostra Completa	83,0	75,6	88,8	84,0	82,4	90,2
Amostra de Treino	87,1	76,6	94,5	90,7	85,2	93,5
Amostra de Teste	64,7	50,0	80,0	72,2	60,6	84,9
Amostra de Treino c\ Oversampling	85,2	76,9	93,4	92,1	80,2	93,7
Amostra de Teste c\ Oversampling	64,7	50,0	80,0	72,2	60,6	81,1
QDA Amostra Completa c\ Validação Cruzada	70,8	53,3	84,5	72,7	70,0	72,8
QDA Amostra Completa c\ Validação Cruzada 2	86,4	73,3	96,6	94,3	82,4	92,5

Amostra de treino c\ Validação Cruzada	87,1	76,6	94,5	90,7	85,2	93,5
Amostra de Teste c\ Validação Cruzada	64,7	50,0	80,0	72,2	60,6	84,9
Amostra de Treino c\ Validação Cruzada e Oversampling	85,2	76,9	93,4	92,1	80,2	93
Amostra de Teste c\ Validação Cruzada e Oversampling	64,7	50	80	60,6	72,2	81,1

Tabela 13 - Resultados obtidos análise discriminante quadrática

Arvores de Decisão	**Acurácia**	**SENS**	**ESP**	**VPP**	**VPN**	**AUC**
Amostra Total	79,6	67,8	88,8	82,4	78	
Amostra de Treino	81,3	71,9	87,9	80,7	81,6	
Amostra de Teste	74,5	57,7	92	88,2	67,7	
Amostra de Treino c\ Oversampling	83	81,3	84,6	84,1	81,9	
Amostra de Teste c\ Oversampling	76,5	69,2	84	81,8	72,4	

Tabela 14 - resultados obtidos árvores de decisão

Naive Bayes	**Acurácia**	**SENS**	**ESP**	**VPP**	**VPN**	**AUC**
Amostra Total	80,1	80	80,1	75,8	83,8	86,2
Amostra de Treino	81,9	84,3	80,2	75	88	87,1
Amostra de Teste	78,4	65,4	92	84,5	71,9	87,7
Amostra de Treino c\ Oversampling	82,4	85,7	79,1	80,4	84,7	86,3
Amostra de Teste c\ Oversampling	78,4	69,2	88	85,7	73,3	87,4

Tabela 15 - Resultados obtidos Naives Bayes

K-Vizinhos mais próximos	Acurácia	SENS	ESP	VPP	VPN	AUC
k=2	87,9	88,9	87,1	84,2	91	
k=3	87,9	84,4	90,5	87,4	88,2	
Amostra de teste	70,6	50	92	86,7	63,9	
Amostra de teste c\ Oversampling	66,7	57,7	76	71,4	63,3	

Tabela 16 - Resultados obtidos K-vizinhos mais próximos

Máquinas de Vetores de Suporte	Acurácia	SENS	ESP	VPP	VPN	AUC
Amostra de Teste	74,5	61,5	88	84,2	68,8	
Amostra de Teste c\ Oversampling	72,6	61,6	84	80	67,8	

Tabela 17 - Resultados obtidos máquinas de vetores de suporte

Florestas Aleatórias	Acurácia	SENS	ESP	VPP	VPN	AUC
Amostra de Teste	76,5	61,5	92	88,9	69,7	
Amostra de Teste c\ Oversampling	74,5	61,5	88	84,2	68,8	

Tabela 18 - Resultados obtidos florestas aleatórias

7.2.Anexo II – Programa em linguagem Python elaborado para a classificação baseada em imagens radiográficas

```
#BIBLIOTECAS

from IPython.display import clear_output
!pip install imutils
!apt-get install tree
clear_output()

import numpy as np
from tqdm import tqdm
import cv2
import os
import shutil
import itertools
import imutils
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_curve
from keras import backend as K
from sklearn import metrics
import tensorflow as tf
from sklearn.metrics import roc_auc_score
from tensorflow.keras.utils import plot_model
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt



import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot
from plotly import tools

from keras.preprocessing.image import ImageDataGenerator
from keras.applications.vgg16 import VGG16, preprocess_input
from keras import layers
from keras.models import Model, Sequential
from keras.optimizers import Adam, RMSprop
from keras.callbacks import EarlyStopping

import keras as keras
from keras.applications.resnet50 import ResNet50,resnet50
from keras.applications.inception_v3 import InceptionV3,inception_v3

init_notebook_mode(connected=True)
RANDOM_SEED = 123

#CRIAÇÃO DA ESTRUTURA DE PASTAS
!mkdir TRAIN TEST VAL TRAIN/HOMEM TRAIN/MULHER TEST/HOMEM TEST/MULHER VAL/HOMEM VAL/MULH
ER
!tree -d

#ORIGEM DAS IMAGENS
IMG_PATH = '/kaggle/input/opg-crop-homem-mulher/OPG_CROP_M_W/'



#DIVISÃO DO DATASET EM TREINO|VALIDAÇÃO|TESTE
```

```
for CLASS in os.listdir(IMG_PATH):
    if not CLASS.startswith('.'):
        IMG_NUM = len(os.listdir(IMG_PATH + CLASS))
        for (n, FILE_NAME) in enumerate(os.listdir(IMG_PATH + CLASS)):
            img = IMG_PATH + CLASS + '/' + FILE_NAME
            if n < 5:
                shutil.copy(img, 'TEST/' + CLASS.upper() + '/' + FILE_NAME)
            elif n < 0.8*IMG_NUM:
                shutil.copy(img, 'TRAIN/'+ CLASS.upper() + '/' + FILE_NAME)
            else:
                shutil.copy(img, 'VAL/'+ CLASS.upper() + '/' + FILE_NAME)


#FUNÇÃO PARA TRANSFORMAR AS IMAGENS EM ARRAYS
def load_data(dir_path, img_size=(100,100)):
    X = []
    y = []
    i = 0
    labels = dict()
    for path in tqdm(sorted(os.listdir(dir_path))):
        if not path.startswith('.'):
            labels[i] = path
            for file in os.listdir(dir_path + path):
                if not file.startswith('.'):
                    img = cv2.imread(dir_path + path + '/' + file)
                    X.append(img)
                    y.append(i)
            i += 1
    X = np.array(X)
    y = np.array(y)
    print(f'{len(X)} images loaded from {dir_path} directory.')
    return X, y, labels


#FUNÇÃO PARA O PLOT DA MATRIZ DE CONFUSÃO
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    plt.figure(figsize = (6,6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    cm = np.round(cm,2)
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()

TRAIN_DIR = 'TRAIN/'
TEST_DIR = 'TEST/'
VAL_DIR = 'VAL/'
```

```
IMG_SIZE = (224,224)

#DIVISÃO DO DATASET
X_train, y_train, labels = load_data(TRAIN_DIR, IMG_SIZE)
X_test, y_test, _ = load_data(TEST_DIR, IMG_SIZE)
X_val, y_val, _ = load_data(VAL_DIR, IMG_SIZE)

#GRAFICO DE BARRAS DA DISTRIBUIÇÃO
y = dict()
y[0] = []
y[1] = []
for set_name in (y_train, y_val, y_test):
    y[0].append(np.sum(set_name == 0))
    y[1].append(np.sum(set_name == 1))

trace0 = go.Bar(
    x=['Treino', 'Validação', 'Teste'],
    y=y[0],
    name='No',
    marker=dict(color='#33cc33'),
    opacity=0.7
)
trace1 = go.Bar(
    x=['Treino', 'Validação', 'Teste'],
    y=y[1],
    name='Yes',
    marker=dict(color='#ff3300'),
    opacity=0.7
)
data = [trace0, trace1]
layout = go.Layout(
    title ='',
    xaxis={'title': 'Dados'},
    yaxis={'title': 'Quantidade'}
)
fig = go.Figure(data, layout)
iplot(fig)



#FUNÇÃO PARA A PRÉ-VISUALIZAÇÃO DO DATASET(PLOT)
def plot_samples(X, y, labels_dict, n=50):

    for index in range(len(labels_dict)):
        imgs = X[np.argwhere(y == index)][:n]
        j = 10
        i = int(n/j)

        plt.figure(figsize=(15,6))
        c = 1
        for img in imgs:
            plt.subplot(i,j,c)
            plt.imshow(img[0])

            plt.xticks([])
            plt.yticks([])
            c += 1
        plt.suptitle('{}'.format(labels_dict[index]))
        plt.show()

plot_samples(X_train, y_train, labels, 30)

#DATA AUGMENTATION
```

```
TRAIN_DIR = 'TRAIN/'
VAL_DIR = 'VAL/'

train_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    brightness_range=[0.5, 1.5],
    horizontal_flip=True,
    vertical_flip=True,
    preprocessing_function=preprocess_input
)

test_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    brightness_range=[0.5, 1.5],
    horizontal_flip=True,
    vertical_flip=True,
    preprocessing_function=preprocess_input
)


train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    color_mode='rgb',
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary',
    seed=RANDOM_SEED
)


validation_generator = test_datagen.flow_from_directory(
    VAL_DIR,
    color_mode='rgb',
    target_size=IMG_SIZE,
    batch_size=16,
    class_mode='binary',
    seed=RANDOM_SEED
)

#SEM DATA AUGMENTATION

TRAIN_DIR = 'TRAIN/'
VAL_DIR = 'VAL/'


image_gen = ImageDataGenerator(rescale = 1./255)
test_data_gen = ImageDataGenerator(rescale = 1./255)

train_generator = image_gen.flow_from_directory(
      TRAIN_DIR,
      target_size=IMG_SIZE,
      color_mode='rgb',
      class_mode='binary',
      batch_size=32,
     seed=RANDOM_SEED
      )
```

```
validation_generator = test_data_gen.flow_from_directory(
      VAL_DIR,
      target_size=IMG_SIZE,
      color_mode='rgb',
      class_mode='binary',
      batch_size=16,
      seed=RANDOM_SEED
      )
```

Modelo VGG 16

```
#MODELO VGG16 COM PESOS

vgg16_weight_path = '../input/keras-pretrained-models/vgg16_weights_tf_dim_ordering_tf_k
ernels_notop.h5'
base_model = VGG16(
    weights=vgg16_weight_path,
    include_top=False,
    input_shape=IMG_SIZE + (3,)
)


#MODELO VGG16 SEM PESOS

base_model = VGG16(
    weights=None,
    include_top=False,
    input_shape=IMG_SIZE + (3,)
)



#FUNÇÕES DE PERDA

def weighted_binary_crossentropy(zero_weight, one_weight):
    def weighted_binary_crossentropy(y_true, y_pred):
        b_ce = keras.backend.binary_crossentropy(y_true, y_pred)
        # weighted calc
        weight_vector = y_true * one_weight + (1 - y_true) * zero_weight
        weighted_b_ce = weight_vector * b_ce
        return keras.backend.mean(weighted_b_ce)

    return weighted_binary_crossentropy



def focal_loss(alpha=0.25,gamma=2.0):
    def focal_crossentropy(y_true, y_pred):
        bce = K.binary_crossentropy(y_true, y_pred)

        y_pred = K.clip(y_pred, K.epsilon(), 1.- K.epsilon())
        p_t = (y_true*y_pred) + ((1-y_true)*(1-y_pred))

        alpha_factor = 1
        modulating_factor = 1

        alpha_factor = y_true*alpha + ((1-alpha)*(1-y_true))
        modulating_factor = K.pow((1-p_t), gamma)

        # compute the final loss and return
        return K.mean(alpha_factor*modulating_factor*bce, axis=-1)
    return focal_crossentropy
```

```
#FUNÇÃO PARA A MÉTRICA AUC

def auc(y_true, y_pred):
    auc = tf.metrics.auc(y_true, y_pred)[1]
    K.get_session().run(tf.local_variables_initializer())
    return auc


#CRIAÇÃO DA REDE COM OS PARAMETROS

NUM_CLASSES = 1

model_VGG16 = Sequential()
model_VGG16.add(base_model)
model_VGG16.add(layers.Flatten())
model_VGG16.add(layers.Dropout(0.5))
model_VGG16.add(layers.Dense(NUM_CLASSES, activation='sigmoid'))

model_VGG16.layers[0].trainable = False

model_VGG16.compile(
    loss='binary_crossentropy',
    optimizer=Adam(lr=1e-4),
    metrics=['accuracy',auc]
)

model_VGG16.summary()

#TREINO DA REDE

EPOCHS = 50

history_vgg16 = model_VGG16.fit_generator(
    train_generator,
    steps_per_epoch=50,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=25,
)


#MÉTRICAS

acc = history_vgg16.history['acc']
val_acc = history_vgg16.history['val_acc']
loss = history_vgg16.history['loss']
val_loss = history_vgg16.history['val_loss']
epochs_range = range(1, len(history_vgg16.epoch) + 1)

plt.figure(figsize=(15,5))

plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Train Set')
plt.plot(epochs_range, val_acc, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Train Set')
```

```
plt.plot(epochs_range, val_loss, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')

plt.tight_layout()

plt.savefig('vgg16_data_acc_loss.png')

#MÉTRICAS

AUC = history_vgg16.history['auc']
val_AUC = history_vgg16.history['val_auc']
epochs_range = range(1, len(history_vgg16.epoch) + 1)

plt.subplot(1, 1, 1)
plt.plot(epochs_range, AUC, 'g', label='AUC')
plt.plot(epochs_range, val_AUC, 'b', label='Val_AUC')
plt.xlabel('No.of epochs')
plt.ylabel('AUC')
plt.title('Training and validation AUC')
plt.legend()
plt.figure()

plt.savefig('roc_auc_vgg16.png')

#PLOT DA ARQUITETURA DO MODELO

plot_model(model_VGG16, show_shapes = True)

#MATRIZ DE CONFUSÃO ( VALIDAÇÃO E TESTE)

pred_Y = model_VGG16.predict(X_val_prep,
                             batch_size = 32,
                             verbose = True)


print("VALIDATION")
print(classification_report(y_val, pred_Y>0.5, target_names = ['Homem', 'Mulher']))

pred_Y = model_VGG16.predict(X_test_prep,
                             batch_size = 32,
                             verbose = True)


print(classification_report(y_test, pred_Y>0.5, target_names = ['Homem', 'Mulher']))

#VALOR DE AUC

auc = roc_auc_score(y_test, pred_Y)
print('ROC AUC: %f' % auc)

#CURVA ROC

def plot_roc_curve(y_test, pred_Y):
    """
    plots the roc curve based of the probabilities
    """

    fpr, tpr, thresholds = roc_curve(y_test, pred_Y)
    plt.plot(fpr, tpr)
```

```
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')

plot_roc_curve(y_test, pred_Y)
print(f'model 1 AUC score: {roc_auc_score(y_test, pred_Y)}')
plt.savefig('roc_curve_vgg16.png')

#SALVAR O MODELO
model_VGG16.save('VGG_model_SGD_Data_final.h5')
```

MODELO RESNET 50

```
#MODELO RESNET50 COM PESOS

ResNet50_weight_path = '../input/keras-pretrained-models/resnet50_weights_tf_dim_orderin
g_tf_kernels_notop.h5'
resnet50_x = ResNet50(
    weights=ResNet50_weight_path,
    include_top=False,
    input_shape=IMG_SIZE + (3,)
)

#CRIAÇÃO DO MODELO RESNET 50 COM OS PARAMETROS

def auc(y_true, y_pred):
    auc = tf.metrics.auc(y_true, y_pred)[1]
    K.get_session().run(tf.local_variables_initializer())
    return auc


NUM_CLASSES = 1

resnet50 = Sequential()
resnet50.add(resnet50_x)
resnet50.add(layers.Dropout(0.3))
resnet50.add(layers.Flatten())
resnet50.add(layers.Dropout(0.5))
resnet50.add(layers.Dense(NUM_CLASSES, activation='sigmoid'))

resnet50.layers[0].trainable = False

resnet50.compile(loss='binary_crossentropy', optimizer=keras.optimizers.Adam(lr=0.0003,
beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False), metrics=["accuracy"])

resnet50.summary()

#TREINO DA REDE

EPOCHS = 50
resnet50_history = resnet50.fit_generator(
    train_generator,
    steps_per_epoch=50,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=25,
)
```

```
#MÉTRICAS

acc = resnet50_history.history['acc']
val_acc = resnet50_history.history['val_acc']
loss = resnet50_history.history['loss']
val_loss = resnet50_history.history['val_loss']
epochs_range = range(1, len(resnet50_history.epoch) + 1)

plt.figure(figsize=(15,5))

plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Train Set')
plt.plot(epochs_range, val_acc, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Train Set')
plt.plot(epochs_range, val_loss, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')

plt.tight_layout()
plt.show()

#MATRIZ DE CONFUSÃO
pred_Y = resnet50.predict(X_val_prep,
                          batch_size = 32,
                          verbose = True)


print("VALIDATION")
print(classification_report(y_val, pred_Y>0.5, target_names = ['Homem', 'Mulher']))

pred_Y = resnet50.predict(X_test_prep,
                          batch_size = 32,
                          verbose = True)
print("TESTE")

print(classification_report(y_test, pred_Y>0.5, target_names = ['Homem', 'Mulher']))

#MÉTRICAS

AUC = resnet50_history.history['auc']
val_AUC = resnet50_history.history['val_auc']
epochs_range = range(1, len(resnet50_history.epoch) + 1)

plt.subplot(1, 1, 1)
plt.plot(epochs_range, AUC, 'g', label='AUC')
plt.plot(epochs_range, val_AUC, 'b', label='Val_AUC')
plt.xlabel('No.of epochs')
plt.ylabel('AUC')
plt.title('Training and validation AUC')
plt.legend()
plt.figure()


plt.savefig('ROC_RESNET50.jpg')
plt.show()
```

```
auc = roc_auc_score(y_test, pred_Y)
print('ROC AUC: %f' % auc)

#PLOT DA ARQUITETURA DA REDE

plot_model(resnet50, show_shapes = True)

#CURVA ROC

def plot_roc_curve(y_test, pred_Y):

    fpr, tpr, thresholds = roc_curve(y_test, pred_Y)
    plt.plot(fpr, tpr)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')

plot_roc_curve(y_test, pred_Y)
print(f'model 1 AUC score: {roc_auc_score(y_test, pred_Y)}')

#SALVAR O MODELO
resnet50.save('Resnet50_SEM_DATA_final.h5')
```

INCEPTION V3

```
#CARREGAR O MODELO INCEPTION V3 COM PESOS

InceptionV3_weight_path = '../input/keras-pretrained-models/inception_v3_weights_tf_dim_
ordering_tf_kernels_notop.h5'
inceptionV3 = InceptionV3(
     weights=InceptionV3_weight_path,
    include_top=False,
    input_shape=IMG_SIZE + (3,)
)

#CRIAÇÃO DA REDE COM OS PARAMETROS

def auc(y_true, y_pred):
    auc = tf.metrics.auc(y_true, y_pred)[1]
    K.get_session().run(tf.local_variables_initializer())
    return auc


NUM_CLASSES = 1

inception_v3 = Sequential()
inception_v3.add(inceptionV3)
inception_v3.add(layers.Dropout(0.3))
inception_v3.add(layers.Flatten())
inception_v3.add(layers.Dropout(0.5))
inception_v3.add(layers.Dense(NUM_CLASSES, activation='sigmoid'))

inception_v3.layers[0].trainable = False

inception_v3.compile(
    loss='binary_crossentropy',
    optimizer=RMSprop(lr=1e-4),
    metrics=['accuracy',auc]
)

inception_v3.summary()
```

```
#TREINO DA REDE
EPOCHS = 50

inception_v3_history = inception_v3.fit_generator(
    train_generator,
    steps_per_epoch=50,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=25,
)

#MÉTRICAS
acc = inception_v3_history.history['acc']
val_acc = inception_v3_history.history['val_acc']
loss = inception_v3_history.history['loss']
val_loss = inception_v3_history.history['val_loss']
epochs_range = range(1, len(inception_v3_history.epoch) + 1)

plt.figure(figsize=(15,5))

plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Train Set')
plt.plot(epochs_range, val_acc, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Train Set')
plt.plot(epochs_range, val_loss, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')

plt.tight_layout()
plt.show()

#MÉTRICAS

AUC = inception_v3_history.history['auc']
val_AUC = inception_v3_history.history['val_auc']
epochs_range = range(1, len(inception_v3_history.epoch) + 1)

plt.subplot(1, 1, 1)
plt.plot(epochs_range, AUC, 'g', label='AUC')
plt.plot(epochs_range, val_AUC, 'b', label='Val_AUC')
plt.xlabel('No.of epochs')
plt.ylabel('AUC')
plt.title('Training and validation AUC')
plt.legend()
plt.figure()


plt.savefig('ROC_RESNET50.jpg')
plt.show()


#PLOT DA ARQUITETURA DA REDE
plot_model(inception_v3, show_shapes = True)
```

```
#MATRIZ DE CONFUSÃO
pred_Y = inception_v3.predict(X_val_prep,
                              batch_size = 32,
                              verbose = True)


print("VALIDATION")
print(classification_report(y_val, pred_Y>0.5, target_names = ['Homem', 'Mulher']))

pred_Y = inception_v3.predict(X_test_prep,
                              batch_size = 32,
                              verbose = True)


print("TESTE")
print(classification_report(y_test, pred_Y>0.5, target_names = ['Homem', 'Mulher']))

#AUC
auc = roc_auc_score(y_test, pred_Y)
print('ROC AUC: %f' % auc)

#CURVA ROC

def plot_roc_curve(y_test, pred_Y):

    fpr, tpr, thresholds = roc_curve(y_test, pred_Y)
    plt.plot(fpr, tpr)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')

plot_roc_curve(y_test, pred_Y)
print(f'model 1 AUC score: {roc_auc_score(y_test, pred_Y)}')

#SALVAR O MODELO
inception_v3.save('InceptionV3_Data.h5')
```

7.3. Anexo III – Programa em linguagem R (via RStudio) elaborado para a classificação baseada nos dados numéricos

```
library(foreign)
library(caret)
library(psych)
library(carData)
library(car)
library(mvnormtest)
library(HDoutliers)
library(corrplot)
library(pROC)
library(heplots)
library(mfx)
library(MASS)
library(e1071)
library(caTools)
library(caret)
library(partykit)
library(class)
library(ggplot2)
library(car)
library(ggpairs)
library(GGally)


#DATASET
read.spss('Diagnose Sexual.sav', reencode='utf-8')

df <- read.spss('Diagnose Sexual.sav', to.data.frame = TRUE, reencode='utf-8')

View(df)
head(df)
attach(df)

## Renomear variáveis de forma a ser mais fácil perceber gráficos...

names(df)
library(dplyr)
df <- rename(df, ACD=Altura_coronoideia_Dta)
df <- rename(df, ACE=Altura_coronoideia_Esq)
df <- rename(df, AMD=Altura_mandíbula_Dta)
df <- rename(df, AME=Altura_mandíbula_Esq)
df <- rename(df, AQ=Altura_queixo)
df <- rename(df, ARMD=Altura_ramo_mandíbula_Dta)
df <- rename(df, ARME=Altura_ramo_mandíbula_Esq)
df <- rename(df, LMRMD=Largura_minima_ramo_mandíbula_Dta)
df <- rename(df, LMRME=Largura_minima_ramo_mandíbula_Esq)
df <- rename(df, DG=Distancia_gonion)
df <- rename(df, DI1=Distancia_intercondilar)
df <- rename(df, DI2=Distancia_Intercoronoideia)
df <- rename(df, AGD=Angulo_goniaco_Dto)
df <- rename(df, AGE=Angulo_goniaco_Esq)
names(df)

##########
sum(is.na(df))
```

```
#########
prop.table(table(Genero))*100
round(prop.table(table(Genero))*100,2)
barplot(table(Genero), main="Genéro",col= c('red', 'green'))

# DIVISÃO DO DATASET TESTE-TREINO (80-20)
set.seed(123)
split <- sample.split(df, SplitRatio = 0.8)
df_treino <- subset(df, split == "TRUE")
df_teste <- subset(df, split == "FALSE")

### Oversampling
library(UBL);
set.seed(123)
df_treino_O <- RandOverClassif(Genero~., df_treino, "balance")

#CRIAÇÃO DATASET VARIAVEIS QUANTITATIVAS
df_vq <- df[,c(2:16)]

## Analisar a capacidade discriminante em cada variável


par(mfrow = c(5, 3))

# Loop pelas variáveis do conjunto de dados
for (variable in names(df_vq)) {
  # Criar o boxplot
  boxplot(df[[variable]] ~ df$Genero, col = c('pink', 'blue'),
          main = paste("Boxplot de", variable),
          xlab = "", ylab = variable)
}

# Restaurar o layout padrão
par(mfrow = c(1, 1))


#Normalidade
for(i in names(df_vq)){print(c(i,round(shapiro.test(df[,i])$p.value,4)))}
library(car)
par(mfrow=c(3,5))
for(i in names(df_vq)){qqPlot(df[,i], main=paste(i), xlab=" ", ylab=" ")}
par(mfrow=c(1,1))


## Comparando variâncias (considerando normalidade)
for(i in names(df_vq)){print(c(i,
                               round(tapply(df[,i], df$Genero, sd),4),
                               round(var.test(df[,i] ~ df$Genero)$p.value,4)))}

install.packages("car")


for(i in names(df_vq)){print(c(i,
                               round(tapply(df[,i], df$Genero, sd),4),
                               round(leveneTest(df[,i] ~ df$Genero)$p.value,4)))}



## LeveneTest
for (column in names(df_vq)){print(c(column,
                                     round(leveneTest(df[,    column]    ~    df$Genero,
center=median)$`Pr(>F)`[1],4)))}
```

```
## comparando médias (considerando normalidade e igualdade de variâncias)
for(i in names(df_vq)){print(c(i,
                               round(tapply(df[,i], df$Genero, mean),4),
                               round(t.test(df[,i]    ~    df$Genero,    var.equal    =
TRUE)$p.value,4)))}



## Comparando medianas
for(i in names(df_vq)){print(c(i,
                               round(tapply(df[,i], df$Genero, median),4),
                               round(wilcox.test(df[,i] ~ df$Genero)$p.value,4)))}



##### Curva ROC para todas as variáveis

variaveis <- names(df_vq)
valor <- 0.60
valor1 <- FALSE

for (i in seq_along(variaveis)) {
  variavel <- variaveis[i]

  formula <- paste("df$Genero ~ df$", variavel, sep = "")
  plot(roc(as.formula(formula)),col = i+1 ,
       main = "Curva ROC", print.auc.y=valor , print.auc.x = 0.30,
       add=valor1, print.auc=TRUE)
  valor1 <- TRUE
  valor <- valor - 0.04
}
abline(h=c(0,1), v=c(0,1), lty=3, lwd=2);
legend(0.17, 0.60, variaveis,
       col = c(2:16), lwd =2, ,cex = 0.65, text.width = 0.05)


########  MATRIZ DE CONFUSÃO ######

grafico_matriz_confusao2 <- function(conf_matrix, texto){
  conf_matrix_df <- as.data.frame(conf_matrix$table)
  ggplot(conf_matrix_df, aes(x = Reference, y = Prediction, fill = Freq)) +
    geom_tile() +
    geom_text(aes(label = Freq), vjust = 1, color="white") +  # Adicione os valores nas
células do heatmap
    labs( x = "Classe Real", y = "Classe Prevista", fill = "Frequência") +
    theme_minimal() +
    ggtitle("Matriz de Confusão ", paste(texto))
}

grafico_matriz_confusao2 <- function(conf_matrix, texto) {
  conf_matrix_df <- as.data.frame(conf_matrix$table)
  ggplot(conf_matrix_df, aes(x = Prediction, y = Reference, fill = Freq)) +
    geom_tile() +
    geom_text(aes(label = Freq), vjust = 1, color = "white") +  # Adicione os valores nas
células do heatmap
    labs(x = "Classe Prevista", y = "Classe Real", fill = "Frequência") +
    theme_minimal() +
    ggtitle("Matriz de Confusão ", paste(texto))
}

grafico_matriz_confusao <- function(conf_matrix, texto) {
  conf_matrix_df <- as.data.frame(conf_matrix$table)

  # Reordena as classes previstas para a ordem desejada
```

```
  conf_matrix_df$Prediction <- factor(conf_matrix_df$Prediction, levels =
rev(levels(conf_matrix_df$Prediction)))

  ggplot(conf_matrix_df, aes(x = Prediction, y = Reference, fill = Freq)) +
    geom_tile() +
    geom_text(aes(label = Freq), vjust = 1, color = "white") +  # Adicione os valores nas
células do heatmap
    labs(x = "Classe Prevista", y = "Classe Real", fill = "Frequência") +
    theme_minimal() +
    ggtitle("Matriz de Confusão ", paste(texto))
}


################################################
## Análise da correlação

summary(df_vq)


round(cor(df_vq),2)  #matrix de coorelação

corrplot(cor(df_vq), addCoef.col = TRUE,
         type = "lower",
         order="hclust",
         diag = FALSE)


ggpairs(df_vq,title="Correlação das Variáveis Quantitativas",
ggplot2::aes(colour=df$Genero))


#REGRESSÃO LOGISTICA

Rlog1 <- glm(Genero ~., family = binomial, data = df)

summary(Rlog1)
coefficients(Rlog1)
exp(Rlog1$coefficients)

#AIC : 193.25

Prob_Rlog1 <- Rlog1$fitted.values
Prob_Rlog1

Classe_Rlog1 <- ifelse(Prob_Rlog1 >=0.5, "Masculino","Feminino")
table(Classe_Rlog1,df$Genero)


Rlog1_matriz <- confusionMatrix(as.factor(Classe_Rlog1),df$Genero, positive = "Feminino")
Rlog1_matriz2 <- confusionMatrix(as.factor(Classe_Rlog1),df$Genero, positive =
"Masculino", mode = "prec_recall")
print(Rlog1_matriz)
print(Rlog1_matriz2)


grafico_matriz_confusao(Rlog1_matriz2, "Regressão Logistica - Amostra Completa")


#ROC
roc1 <- roc(df$Genero ~ Prob_Rlog1)
plot(roc1, print.auc=TRUE, col=2,  main = "Curva ROC", xlab = "1 - especificidade",
     ylab = "Sensibilidade", print.auc.x=0.65)

#AUC: 0.904
```

```
#outro exemplo (PARCIMONIOSO)

Rlog2=glm(Genero~., family = binomial(link="logit"), data = df)
summary(Rlog2)

logitor(Genero~.,data=df)
exp(coef(Rlog2))
anova(Rlog2, test = "Chisq")

## alterei esta parte
Rlog2 = update(Rlog2,~. - DI1)
summary(Rlog2)
Rlog2 = update(Rlog2, ~. -DG)
summary(Rlog2)
Rlog2 = update(Rlog2, ~. -ACD)
summary(Rlog2)
Rlog2 = update(Rlog2, ~. -LMRME)
summary(Rlog2)
Rlog2 = update(Rlog2, ~. -DI2)
summary(Rlog2)
Rlog2 = update(Rlog2, ~. -ARME)
summary(Rlog2)
Rlog2 = update(Rlog2, ~. -AQ)
summary(Rlog2)
Rlog2 = update(Rlog2, ~. -Idade)
summary(Rlog2)
Rlog2 = update(Rlog2, ~. -LMRMD)
summary(Rlog2)
Rlog2 = update(Rlog2, ~. -AGD)
summary(Rlog2)
Rlog2 = update(Rlog2, ~. -AGE)
summary(Rlog2)
Rlog2 = update(Rlog2, ~. -ACE)
summary(Rlog2)
anova(Rlog1,Rlog2, test = "Chisq")

Prob_Rlog2 <- Rlog2$fitted.values
Classe_Rlog2 <- ifelse(Prob_Rlog2 >=0.5, "Masculino","Feminino")
confusionMatrix(as.factor(Classe_Rlog2),df$Genero, positive = "Masculino")
confusionMatrix(as.factor(Classe_Rlog2),df$Genero,  positive  =  "Masculino",  mode  =
"prec_recall")

#ROC 2
roc2 <- roc(df$Genero ~ Prob_Rlog2)
plot(roc2, print.auc=TRUE, col=3, add=TRUE, print.auc.y=0.45,  print.auc.x=0.65)

#AUC: 0.880

roc.test(roc1, roc2)

# H0: AUC semelhantes
# H1: AUC distintas
# p-value = 0.05319 > 0.05 -> Não há evidência que as AUC são distintas.


#outro exemplo com amostra treino e teste
Rlog3 <- glm(Genero ~., family=binomial, data=df_treino)
summary(Rlog3)
Rlog3 = update(Rlog3,~. - ACE)
summary(Rlog3)
Rlog3 = update(Rlog3, ~. -DG)
summary(Rlog3)
Rlog3 = update(Rlog3, ~. -DI1)
summary(Rlog3)
Rlog3 = update(Rlog3, ~. -ACD)
```

```
summary(Rlog3)
Rlog3 = update(Rlog3, ~. -AGE)
summary(Rlog3)
Rlog3 = update(Rlog3, ~. -AGD)
summary(Rlog3)
Rlog3 = update(Rlog3, ~. -LMRMD)
summary(Rlog3)
Rlog3 = update(Rlog3, ~. -AQ)
summary(Rlog3)
Rlog3 = update(Rlog3, ~. -Idade)
summary(Rlog3)
Rlog3 = update(Rlog3, ~. -AMD)
summary(Rlog3)
Rlog3 = update(Rlog3, ~. -DI2)
summary(Rlog3)
Rlog3 = update(Rlog3, ~. -LMRME)
summary(Rlog3)
Rlog3 = update(Rlog3, ~. -ARME)
summary(Rlog3)


Prob_treino <- predict(Rlog3,df_treino, type="response")
Classes_treino <- ifelse(Prob_treino >= 0.5, "Masculino", "Feminino")

Prob_teste <- predict(Rlog3, df_teste, type="response")
Classes_teste <- ifelse(Prob_teste >= 0.5, "Masculino", "Feminino")

confusionMatrix(as.factor(Classes_treino),df_treino$Genero, positive="Masculino")
confusionMatrix(as.factor(Classes_teste),df_teste$Genero, positive="Masculino")

roc_treino = roc(df_treino$Genero ~ Prob_treino)
roc_teste = roc(df_teste$Genero ~ Prob_teste)

plot(roc_treino,  print.auc  =  TRUE,  col  =  4,  lwd  =3,  add=TRUE,  print.auc.y=0.40,
print.auc.x=0.65);
plot(roc_teste, add=TRUE , print.auc = TRUE, col = 5, lwd =3,
     print.auc.y = 0.35,  print.auc.x=0.65);

roc.test(roc_treino, roc_teste)
#ROC1 AUC : 0.9051724
#ROC2 AUC: 0.8824451

### Oversampling
Rlog4 <- glm(Genero ~., family=binomial, data=df_treino_O)
summary(Rlog4)
Rlog4 = update(Rlog4,~. - ACE)
summary(Rlog4)
Rlog4 = update(Rlog4, ~. -LMRME)
summary(Rlog4)
Rlog4 = update(Rlog4, ~. -LMRMD)
summary(Rlog4)
Rlog4 = update(Rlog4, ~. -AGD)
summary(Rlog4)
Rlog4 = update(Rlog4, ~. -AGE)
summary(Rlog4)
Rlog4 = update(Rlog4, ~. -ACD)
summary(Rlog4)
Rlog4 = update(Rlog4, ~. -DI1)
summary(Rlog4)
Rlog4 = update(Rlog4, ~. -AMD)
summary(Rlog4)
Rlog4 = update(Rlog4, ~. -AME)
summary(Rlog4)
Rlog4 = update(Rlog4, ~. -DG)
summary(Rlog4)
```

```
Rlog4 = update(Rlog4, ~. -DI2)
summary(Rlog4)
Rlog4 = update(Rlog4, ~. -ARME)
summary(Rlog4)

Prob_treino <- predict(Rlog4,df_treino, type="response")
Classes_treino <- ifelse(Prob_treino >= 0.5, "Masculino", "Feminino")
Prob_teste <- predict(Rlog4, df_teste, type="response")
Classes_teste <- ifelse(Prob_teste >= 0.5, "Masculino", "Feminino")

confusionMatrix(as.factor(Classes_treino), df_treino$Genero, positive="Masculino")
confusionMatrix(as.factor(Classes_teste), df_teste$Genero, positive="Masculino")

roc_treino = roc(df_treino$Genero ~ Prob_treino)
roc_teste = roc(df_teste$Genero ~ Prob_teste)

plot(roc_treino,print.auc  =  TRUE,  col  =  6,  lwd  =3,  add=TRUE,  print.auc.y=0.30,
print.auc.x=0.65);
plot(roc_teste, add=TRUE , print.auc = TRUE, col = 7, lwd =3, print.auc.x=0.65,
     print.auc.y = 0.25);
abline(h=c(0,1), v=c(0,1), lty=3, lwd=2)

legend(0.35, 0.45, c("Amostra completa", "Modelo Parcimonioso",
                     "Treino","Teste",
                     "Treino Oversampling","Teste Oversampling"),
       col = c(2,3,4,5,6), lwd =2, ,cex = 0.65, text.width = 0.2)


roc.test(roc_treino, roc_teste)
#AUC TREINO: 0.8818681
#AUC TESTE: 0.8615385


### ANALISE DISCRIMINANTE ###

pairs(df_vq, col=df$Genero, pch=22)

outHD <-HDoutliers(df);
outHD;
plotHDoutliers(df, outHD)
#NÃO TEM OUTLIERS MULTIVARIADOS

mshapiro.test(t(df_vq))

# H0: as variaveis quantitativas seguem uma distribuição
# normal multivariada
# H1: as variaveis quantitativas não seguem uma distribuição
# normal multivariada

# p-value = 2.2e-16 < alfa = 0.05 -> rejeitar H0, ...

mshapiro.test(t(df[df$Genero == "Masculino",c(2:16)]))
# Masculino: p-value = 2.2e-16  < alfa = 0.05 -> rejeitar H0
mshapiro.test(t(df[df$Genero == "Feminino",c(2:16)]))
# Feminino: p-value = 7.145e-14  < alfa = 0.05 -> rejeitar H0


boxM(df[,c(2:16)], df$Genero)
# H0: Matriz de variância-covariância das variáveis quantitativas
#é igual nas duas categorias da variável class
# H1: Matriz de variância-covariância das variáveis quantitativas
#não é igual nas duas categorias da variável class

# p-value < 9.489e-10 < alfa = 0.05 -> rejeitar H0
```

```
#########################
####análise discriminante linear (LDA)

lda_0 = lda(Genero~., data = df)
lda_0
lda_0_pred <- predict(lda_0, df);

ldahist(lda_0_pred$x[,1], g=df$Genero)

plot(lda_0_pred$x,
     col=c("red","blue"), pch=22, main="", xlab="", ylab="")
legend("bottomright", legend = levels(df_treino$Genero), col = c("red", "blue"), pch = 22)

confusionMatrix(as.factor(lda_0_pred$class), df$Genero, positive="Masculino")

roc_lda_0 <- roc(df$Genero ~ as.numeric(lda_0_pred$x))
plot(roc_lda_0, print.auc=TRUE, col=2,
     main="Curva ROC", xlab="1 - especificidade", ylab="sensibilidade",
     print.auc.x=0.7, print.auc.y=0.60)


# LDA Treino vs. teste

lda_1 = lda(Genero~., data = df_treino)
lda_1

lda_1_treino <- predict(lda_1, df_treino);

head(lda_1_treino$class,5)
head(lda_1_treino$posterior,5)
head(lda_1_treino$x,5)

ldahist(lda_1_treino$x[,1], g=df_treino$Genero)

plot(lda_1_treino$x[,1],
     col=c("red","blue"), pch=22, main="", xlab="", ylab="")
legend("bottomright", legend = levels(df_treino$Genero), col = c("red","blue"), pch = 22)

confusionMatrix(as.factor(lda_1_treino$class), df_treino$Genero,
                mode = "prec_recall")
confusionMatrix(as.factor(lda_1_treino$class), df_treino$Genero, positive="Masculino")

plot(roc(df_treino$Genero == "Feminino" ~ lda_1_treino$posterior[,1]),
     print.auc=TRUE, col=3,add=TRUE, print.auc.x=0.7 , print.auc.y=0.55)


#ACURACIA TREINO 85,81%

lda_1_teste <- predict(lda_1, df_teste)
lda_1_teste$class

confusionMatrix(as.factor(lda_1_teste$class),
                df_teste$Genero)
confusionMatrix(as.factor(lda_1_teste$class),
                df_teste$Genero,
                mode="prec_recall")
confusionMatrix(as.factor(lda_1_teste$class), df_teste$Genero, positive="Masculino")

# ACURACIA TESTE LDA : 78,43%

head(lda_1_teste$posterior)
plot(lda_1_teste$posterior[,1],
     lda_1_teste$posterior[,2],
     pch=21, bg=rainbow(2)[df_teste$Genero],
```

```
     xlab="Probabilidade de Feminino",
     ylab= "Probabilidade de Masculino", main="Probabilidade Feminino vs Masculino");
abline(h=c(0), v=c(0), lty=2); segments(0,1,1,0, lty=2);
legend(0.75, 0.95, c("Feminino", "Masculino"), col = rainbow(2), lwd =2)

r1 = roc(df_teste$Genero == "Feminino" ~ lda_1_teste$posterior[,1]);
r2 = roc(df_teste$Genero == "Masculino" ~ lda_1_teste$posterior[,2]);

plot(r1, print.auc=TRUE, col=4, print.auc.y=0.50, add= TRUE, print.auc.x=0.7)

plot(r2, print.auc=TRUE, col=5, print.auc.y=0.45, add=TRUE, print.auc.x=0.7)



### LDA Oversampling
lda_2 = lda(Genero~., data = df_treino_O)
lda_2_treino <- predict(lda_2, df_treino_O);
confusionMatrix(as.factor(lda_2_treino$class), df_treino_O$Genero,
                mode = "prec_recall")
lda_2_confMatrix  <-  confusionMatrix(as.factor(lda_2_treino$class),  df_treino_O$Genero,
positive="Masculino")

grafico_matriz_confusao(lda_2_confMatrix,"LDA Treino com oversampling")

head(lda_2_treino$class,5)
head(lda_2_treino$posterior,5)
head(lda_2_treino$x,5)
ldahist(lda_2_treino$x[,1], g=df_treino$Genero)
plot(lda_2_treino$x[,1],
     col=c("red","blue"), pch=22, main="", xlab="", ylab="")
legend("bottomright", legend = levels(df_treino$Genero), col = c("red","blue"), pch = 22)

head(lda_2_teste$posterior)
plot(lda_2_teste$posterior[,1],
     lda_2_teste$posterior[,2],
     pch=21, bg=rainbow(2)[df_teste$Genero],
     xlab="Probabilidade de Feminino",
     ylab= "Probabilidade de Masculino", main="Probabilidade Feminino vs Masculino");
abline(h=c(0), v=c(0), lty=2); segments(0,1,1,0, lty=2);
legend(0.75, 0.95, c("Feminino", "Masculino"), col = rainbow(2), lwd =2)


plot(roc(df_treino_O$Genero == "Feminino" ~ lda_2_treino$posterior[,1]),
     print.auc=TRUE, col=2, print.auc.y=0.40, print.auc.x=0.7, add=TRUE)


lda_2_teste <- predict(lda_2, df_teste)
confusionMatrix(as.factor(lda_2_teste$class),
                df_teste$Genero)
confusionMatrix(as.factor(lda_2_teste$class),
                df_teste$Genero,
                mode="prec_recall")
confusionMatrix(as.factor(lda_2_teste$class), df_teste$Genero, positive="Masculino")

plot(roc(df_teste$Genero == "Feminino" ~ lda_2_teste$posterior[,1]),
     print.auc=TRUE, col=6, print.auc.y=0.35, add=TRUE, print.auc.x=0.7)



#com cross-validation - amostra total
lda_0CV <- lda(Genero ~., CV = TRUE, data = df)
cat("Precisão Treino: ", mean(lda_0CV$class == df$Genero))
confusionMatrix(as.factor(lda_0CV$class), df$Genero, positive="Masculino")
roc_lda_0CV <- roc(df$Genero ~ as.numeric(lda_0CV$posterior[,1]))
```

```
plot(roc_lda_0CV, print.auc=TRUE, col=7, add=TRUE, print.auc.x=0.7, print.auc.y=0.30)


control <- trainControl(method = "cv", number = 10)
lda_0CV <- train(Genero ~.,  data = df,
                 method = "lda",  metric = "Accuracy", trControl = control)
lda_0CV_pred <- predict(lda_0CV, df)
confusionMatrix(lda_0CV_pred, df$Genero, positive="Masculino")
lda_0CV_p <- predict(lda_0CV, df, type="prob")
r = roc(df$Genero == "Feminino" ~ lda_0CV_p[,1]);
plot(r, print.auc=TRUE, col=8, print.auc.y=0.25, add=TRUE, print.auc.x=0.70)


#com cross-validation - treino vs. teste
control <- trainControl(method = "cv", number = 10)
lda_1CV <- train(Genero ~.,  data = df_treino,
               method = "lda",  metric = "Accuracy", trControl = control,
               pred=TRUE)
lda_1CV_treino <- predict(lda_1CV, df_treino)
confusionMatrix(lda_1CV_treino, df_treino$Genero, positive="Masculino")
lda_1CV_teste <- predict(lda_1CV, df_teste)
confusionMatrix(lda_1CV_teste, df_teste$Genero, positive="Masculino")
lda_1CV_treinop <- predict(lda_1CV, df_treino, type="prob")
lda_1CV_testep <- predict(lda_1CV, df_teste, type="prob")

r1 = roc(df_treino$Genero == "Feminino" ~ lda_1CV_treinop[,1]);
r2 = roc(df_teste$Genero == "Feminino" ~ lda_1CV_testep[,1]);
plot(r1, print.auc=TRUE, col=9, print.auc.y=0.20, add=TRUE, print.auc.x=0.7)

plot(r2, print.auc=TRUE, col=10, print.auc.y=0.15, add=TRUE, print.auc.x=0.7);


#com cross-validation - treino vs. teste - Oversampling
control <- trainControl(method = "cv", number = 10)
lda_2CV <- train(Genero ~.,  data = df_treino_O,
                 method = "lda",  metric = "Accuracy", trControl = control,
                 pred=TRUE)
lda_2CV_treino <- predict(lda_2CV, df_treino_O)
confusionMatrix(lda_2CV_treino, df_treino_O$Genero, positive="Masculino")
lda_2CV_teste <- predict(lda_2CV, df_teste)
confusionMatrix(lda_2CV_teste, df_teste$Genero, positive="Masculino")
lda_2CV_treinop <- predict(lda_2CV, df_treino, type="prob")
lda_2CV_testep <- predict(lda_2CV, df_teste, type="prob")

r1 = roc(df_treino$Genero == "Feminino" ~ lda_2CV_treinop[,1]);
r2 = roc(df_teste$Genero == "Feminino" ~ lda_2CV_testep[,1]);
plot(r1, print.auc=TRUE, col=11, print.auc.y=0.10, add=TRUE, print.auc.x=0.7)
plot(r2, print.auc=TRUE, col=12, print.auc.y=0.05, add=TRUE, print.auc.x=0.7);
abline(h=c(0,1), v=c(0,1), lty=3, lwd=2);

legend(0.45, 0.50, c("Amostra completa", "Treino",
                     "Teste","Treino Oversampling",
                     "Teste Oversampling","Amostra Completa CV",
                     "Treino CV", "Teste CV",
                     "Treino   Oversampling   CV",   "Treino   Oversampling   CV","Teste
Oversampling CV"),
       col = c(2,3,4,5,6,7,8,9,10,11,12), lwd =2, ,cex = 0.65, text.width = 0.3)

###################################
######## QDA

qda_0 = lda(Genero~., data = df)
qda_0
qda_0_pred <- predict(qda_0, df);
```

```
confusionMatrix(as.factor(qda_0_pred$class), df$Genero, positive="Masculino")

oc_qda_0 <- roc(df$Genero ~ as.numeric(qda_0_pred$x))
plot(roc_qda_0, print.auc=TRUE, print.auc.y=0.55, print.auc.x=0.7,
     col=2, xlab="1 - especificidade", ylab="sensibilidade", main="Curva ROC")

# QDA treino vs. teste
qda_1 <- qda(Genero ~., data = df_treino)
qda_1_predict_treino <- data.frame(predict(qda_1, df_treino))
confusionMatrix(as.factor(qda_1_predict_treino$class),                  df_treino$Genero,
positive="Masculino")
roc_qda_1_treino                <-                roc(df_treino$Genero                ~
as.numeric(qda_1_predict_treino$posterior.Feminino))
plot(roc_qda_1_treino, print.auc=TRUE, col=3, add=TRUE, print.auc.y=0.5,print.auc.x=0.7)

qda_1_predict <- data.frame(predict(qda_1, df_teste))
qda_1_predict$class
confusionMatrix(as.factor(qda_1_predict$class),
                df_teste$Genero)
confusionMatrix(as.factor(qda_1_predict$class), df_teste$Genero, positive="Masculino")
roc_qda_1_teste <- roc(df_teste$Genero ~ as.numeric(qda_1_predict$posterior.Feminino))
plot(roc_qda_1_teste, print.auc=TRUE, col=4, add=TRUE, print.auc.y=0.45,print.auc.x=0.7)


### QDA Oversampling
qda_2 <- qda(Genero ~., data = df_treino_O)
qda_predict_treino <- data.frame(predict(qda_2, df_treino_O))
qda_conf_matrix          <-          confusionMatrix(as.factor(qda_predict_treino$class),
df_treino_O$Genero, positive="Masculino")
roc_qda_treino                <-                roc(df_treino_O$Genero                ~
as.numeric(qda_predict_treino$posterior.Feminino))
plot(roc_qda_treino, print.auc=TRUE, col=5, print.auc.y=0.40, add=TRUE, print.auc.x=0.7)

grafico_matriz_confusao(qda_conf_matrix,"QDA Treino com oversampling")

qda_predict <- data.frame(predict(qda_2, df_teste))
confusionMatrix(as.factor(qda_predict$class),
                df_teste$Genero)
confusionMatrix(as.factor(qda_predict$class), df_teste$Genero, positive="Masculino")
roc_qda_teste <- roc(df_teste$Genero ~ as.numeric(qda_predict$posterior.Feminino))
plot(roc_qda_teste, print.auc=TRUE, col=6, print.auc.y=0.35, add=TRUE, print.auc.x=0.7)



#com cross-validation - amostra total
qda_0CV <- qda(Genero ~., CV = TRUE, data = df)
cat("Precisão Treino: ", mean(qda_0CV$class == df$Genero))
confusionMatrix(as.factor(qda_0CV$class), df$Genero, positive="Masculino")
roc_qda_0CV <- roc(df$Genero ~ as.numeric(qda_0CV$posterior[,1]))
plot(roc_qda_0CV, print.auc=TRUE, col=7, print.auc.y=0.30, add=TRUE, print.auc.x=0.7)

control <- trainControl(method = "cv", number = 10)
qda_0CV <- train(Genero ~.,  data = df,
                 method = "qda",  metric = "Accuracy", trControl = control)
qda_0CV_pred <- predict(qda_0CV, df)
confusionMatrix(qda_0CV_pred, df$Genero, positive="Masculino")
qda_0CV_p <- predict(qda_0CV, df, type="prob")
r = roc(df$Genero == "Feminino" ~ qda_0CV_p[,1]);
plot(r, print.auc=TRUE, col=7, print.auc.y=0.30, add=TRUE, print.auc.x=0.7)


#com cross-validation - treino vs. teste
control <- trainControl(method = "cv", number = 10)
qda_1CV <- train(Genero ~.,  data = df_treino,
                 method = "qda",  metric = "Accuracy", trControl = control,
```

```
                pred=TRUE)
qda_1CV_treino <- predict(qda_1CV, df_treino)
confusionMatrix(qda_1CV_treino, df_treino$Genero, positive="Masculino")
qda_1CV_teste <- predict(qda_1CV, df_teste)
confusionMatrix(qda_1CV_teste, df_teste$Genero, positive="Masculino")
qda_1CV_treinop <- predict(qda_1CV, df_treino, type="prob")
qda_1CV_testep <- predict(qda_1CV, df_teste, type="prob")

r1 = roc(df_treino$Genero == "Feminino" ~ qda_1CV_treinop[,1]);
r2 = roc(df_teste$Genero == "Feminino" ~ qda_1CV_testep[,1]);
plot(r1, print.auc=TRUE, col=8, print.auc.y=0.25, add=TRUE, print.auc.x=0.7)
plot(r2, print.auc=TRUE, col=9, print.auc.y=0.20, add=TRUE, print.auc.x=0.7)



#com cross-validation - treino vs. teste - Oversampling
control <- trainControl(method = "cv", number = 10)
qda_2CV <- train(Genero ~.,  data = df_treino_O,
                 method = "qda",  metric = "Accuracy", trControl = control,
                 pred=TRUE)
qda_2CV_treino <- predict(qda_2CV, df_treino_O)
confusionMatrix(qda_2CV_treino, df_treino_O$Genero, positive="Masculino")
qda_2CV_teste <- predict(qda_2CV, df_teste)
confusionMatrix(qda_2CV_teste, df_teste$Genero, positive="Masculino")
qda_2CV_treinop <- predict(qda_2CV, df_treino, type="prob")
qda_2CV_testep <- predict(qda_2CV, df_teste, type="prob")

r1 = roc(df_treino$Genero == "Feminino" ~ qda_2CV_treinop[,1]);
r2 = roc(df_teste$Genero == "Feminino" ~ qda_2CV_testep[,1]);
plot(r1, print.auc=TRUE, col=10, print.auc.y=0.14, add=TRUE, print.auc.x=0.7)
plot(r2, print.auc=TRUE, col=11, print.auc.y=0.10, add=TRUE, print.auc.x=0.7);

abline(h=c(0,1), v=c(0,1), lty=3, lwd=2);
legend(0.45, 0.50, c("Amostra completa", "Treino",
                     "Teste","Treino Oversampling",
                     "Teste Oversampling","Amostra Completa CV",
                     "Treino CV", "Teste CV",
                     "Treino Oversampling CV", "Teste Oversampling CV"),
       col = c(2,3,4,5,6,7,8,9,10,11), lwd =2, ,cex = 0.65, text.width = 0.3)



######################
##### DECISION TREE - amostra completa

model_tree0 <- ctree(Genero ~ ., df)
plot(model_tree0)
predict_model_tree0 <- predict(model_tree0, df)
y <- confusionMatrix(as.factor(predict_model_tree0), df$Genero, positive = "Masculino")

y_pred_tree0 <- predict(model_tree0, newdata = df, type="prob")

roc_tree_completa = roc(df$Genero == "Masculino" ~ y_pred_tree0[,1]);


##### DECISION TREE - treino vs. teste

model_tree1 <- ctree(Genero ~ ., df_treino)
plot(model_tree1)
predict_model_tree1_treino <- predict(model_tree1, df_treino)
confusionMatrix(as.factor(predict_model_tree1_treino),   df_treino$Genero,   positive   =
"Masculino")
predict_model_tree1_teste <- predict(model_tree1, df_teste)
```

```
confusionMatrix(as.factor(predict_model_tree1_teste),    df_teste$Genero,    positive    =
"Masculino")

y_pred_tree0_treino <- predict(model_tree0, newdata = df_treino, type="prob")
roc_tree_treino = roc(df_treino$Genero == "Masculino" ~ y_pred_tree0_treino[,1]);

y_pred_tree0_teste<- predict(model_tree0, newdata = df_teste, type="prob")
roc_tree_teste = roc(df_teste$Genero == "Masculino" ~ y_pred_tree0_teste[,1]);


##### DECISION TREE - treino vs. teste - Oversampling

model_tree2 <- ctree(Genero ~ ., df_treino_O)
plot(model_tree2)
predict_model_tree2_treino_O <- predict(model_tree2, df_treino_O)
tree_conf_matriz              <-confusionMatrix(as.factor(predict_model_tree2_treino_O),
df_treino_O$Genero, positive = "Masculino")
predict_model_tree2_teste <- predict(model_tree2, df_teste)
confusionMatrix(as.factor(predict_model_tree2_teste),    df_teste$Genero,    positive    =
"Masculino")

grafico_matriz_confusao(tree_conf_matriz, "Árvores de Decisão - Treino com oversampling")

y_pred_tree0_treino_o <- predict(model_tree2, newdata = df_treino_O, type="prob")
roc_tree_treino_o = roc(df_treino_O$Genero == "Masculino" ~ y_pred_tree0_treino_o[,1]);

y_pred_tree0_teste<- predict(model_tree2, newdata = df_teste, type="prob")
roc_tree_teste_o = roc(df_teste$Genero == "Masculino" ~ y_pred_tree0_teste[,1]);

## CURVA ROC

plot(roc_tree_completa, print.auc=TRUE, col=2, print.auc.y=0.45,
     main="Curva ROC", xlab="1 - especificidade", ylab="sensibilidade");
plot(roc_tree_treino, print.auc=TRUE, col=3, print.auc.y=0.40, add=TRUE);
plot(roc_tree_teste, print.auc=TRUE, col=4, print.auc.y=0.35, add=TRUE);
plot(roc_tree_treino_o, print.auc=TRUE, col=5, print.auc.y=0.30, add=TRUE);
plot(roc_tree_teste_o, print.auc=TRUE, col=6, print.auc.y=0.25, add=TRUE);

abline(h=c(0,1), v=c(0,1), lty=3, lwd=2);
legend(0.5, 0.25,
       c("Amostra    Completa",    "Treino","Teste    ,"    Treino    Oversampling","Teste
Oversampling"),
       col = c(2,3,4),
       lwd =2,cex = 0.6, text.width = 0.3)


####################
## NAIVE BAYES - amostra completa

set.seed(123)
model_naive0 <- naiveBayes(Genero ~ ., data = df)
y_pred_naive <- predict(model_naive0, newdata = df)
confusionMatrix(as.factor(y_pred_naive), df$Genero, positive = "Masculino")
y_pred_naive_p <- predict(model_naive0, newdata = df, type="raw")
roc_nv_completa = roc(df$Genero == "Masculino" ~ y_pred_naive_p[,1]);


## NAIVE BAYES ## treino vs teste
set.seed(123)
model_naive1 <- naiveBayes(Genero ~ ., data = df_treino)
y_pred_naive1_treino <- predict(model_naive1, newdata = df_treino)
confusionMatrix(as.factor(y_pred_naive1_treino),     df_treino$Genero,     positive     =
"Masculino")
y_pred_naive1_teste <- predict(model_naive1, newdata = df_teste)
```

```
naive_conf_matrix   <-confusionMatrix(as.factor(y_pred_naive1_teste),   df_teste$Genero,
positive = "Masculino")
y_pred_naive1_treino_p <- predict(model_naive1, newdata = df_treino, type="raw")
y_pred_naive1_teste_p <- predict(model_naive1, newdata = df_teste, type="raw")
roc_nv_treino = roc(df_treino$Genero == "Masculino" ~ y_pred_naive1_treino_p[,1]);
roc_nv_teste = roc(df_teste$Genero == "Masculino" ~ y_pred_naive1_teste_p[,1]);

grafico_matriz_confusao(naive_conf_matrix, "Naive BAyes - Amostra teste")

## NAIVE BAYES ## treino vs teste - oversampling
set.seed(123)
model_naive2 <- naiveBayes(Genero ~ ., data = df_treino_O)
y_pred_naive2_treino <- predict(model_naive2, newdata = df_treino_O)
matrix_treino  <-  confusionMatrix(as.factor(y_pred_naive2_treino),  df_treino_O$Genero,
positive = "Masculino")
y_pred_naive2_teste <- predict(model_naive2, newdata = df_teste)
confusionMatrix(as.factor(y_pred_naive2_teste), df_teste$Genero, positive = "Masculino")
y_pred_naive2_treino_p <- predict(model_naive2, newdata = df_treino_O, type="raw")
y_pred_naive2_teste_p <- predict(model_naive2, newdata = df_teste, type="raw")
roc_nv_treino_o = roc(df_treino_O$Genero == "Masculino" ~ y_pred_naive2_treino_p[,1]);
roc_nv_teste_o = roc(df_teste$Genero == "Masculino" ~ y_pred_naive2_teste_p[,1]);

grafico_matriz_confusao(matrix_treino, "Naive Bayes - Treino com oversampling")

## CURVA ROC

plot(roc_nv_completa,          print.auc=TRUE,          col=2,          print.auc.y=0.45,
print.auc.x=0.6,main="Curva ROC", xlab="1 - especificidade", ylab="sensibilidade");
plot(roc_nv_treino, print.auc=TRUE, col=3, print.auc.y=0.40, add=TRUE, print.auc.x=0.6);
plot(roc_nv_teste, print.auc=TRUE, col=4, print.auc.y=0.35, add=TRUE, print.auc.x=0.6);
plot(roc_nv_treino_o, print.auc=TRUE, col=5, print.auc.y=0.30, add=TRUE, print.auc.x=0.6);
plot(roc_nv_teste_o, print.auc=TRUE, col=6, print.auc.y=0.25, add=TRUE, print.auc.x=0.6);

abline(h=c(0,1), v=c(0,1), lty=3, lwd=2);

legend(0.35, 0.40, c("Amostra completa", "Treino",
                     "Teste","Treino Oversampling",
                     "Teste Oversampling"),
       col = c(2,3,4,5,6,7,8,9,10,11,12), lwd =2, ,cex = 0.55, text.width = 0.2)

########################
## KNN - Amostra Completa

df_scale <- scale(df[,c(2:16)])

set.seed(123)
knn0 = knn(train=df_scale, test=df_scale,
           cl=df$Genero, k=2)
knn_conf_matrix <- confusionMatrix(knn0, df$Genero, positive = "Masculino")

grafico_matriz_confusao(knn_conf_matrix, "KNN - Amostra Completa")

set.seed(123)
acc <- vector()
for (i in 2:25){set.seed(123)
  knn0 = knn(train=df_scale, test=df_scale,
                cl=df$Genero, k=i)
  acc <- c(acc, as.numeric(confusionMatrix(knn0, df$Genero)$overall[[1]]))}
acc
# melhor k=2 e k=3

set.seed(123)
knn0 = knn(train=df_scale, test=df_scale,
           cl=df$Genero, k=3)
confusionMatrix(knn0, df$Genero, positive = "Masculino")
```

```
probs <- ifelse(knn0 == "Masculino", 1, 0)  # Converter para probabilidades (1 para
"Masculino" e 0 para outras classes)

# Calcular curva ROC
roc_knn <- roc(response = df$Genero, predictor = probs)


### Com CV

trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 5)
set.seed(123)
knn_fit <- train(Genero ~ ., data=df, method = "knn",
                 trControl=trctrl,
                 preProcess = c("center", "scale"),
                 tuneLength = 10)
knn_fit
test_pred <- predict(knn_fit, df)
confusionMatrix(test_pred, df$Genero, positive = "Masculino")

probs <- predict(knn_fit, newdata = df, type = "prob")
probs_pos <- probs[, "Masculino"]  # Probabilidades da classe positiva
roc_knn_cv <- roc(response = df$Genero, predictor = probs_pos)



## KNN ## treino vs. amostra

treino_scale <- scale(df_treino[,c(2:16)])
teste_scale <- scale(df_teste[,c(2:16)])

# k=2
set.seed(123)
knn1 = knn(train=treino_scale, test=teste_scale,
           cl=df_treino$Genero, k=2)
confusionMatrix(knn1, df_teste$Genero, positive = "Masculino")



acc <- vector()
for (i in 2:25){set.seed(123)
  knn1 = knn(train=treino_scale, test=teste_scale,
             cl=df_treino$Genero, k=i)
  acc <- c(acc, as.numeric(confusionMatrix(knn1, df_teste$Genero)$overall[[1]]))}
acc
# melhor é k=9
set.seed(123)
knn1 = knn(train=treino_scale, test=teste_scale,
           cl=df_treino$Genero, k=9)
confusionMatrix(knn1, df_teste$Genero, positive = "Masculino")




### Com CV
set.seed(123)
knn_fit1 <- train(Genero ~ ., data=df_treino, method = "knn",
                 trControl=trctrl,
                 preProcess = c("center", "scale"),
                 tuneLength = 10)
knn_fit1
test_pred1 <- predict(knn_fit1, df_teste)
confusionMatrix(test_pred1, df_teste$Genero, positive = "Masculino")
```

```
## k=14 e 15
sqrt(206)
knn.14 <- knn(train=treino_scale, test=teste_scale, cl=df_treino$Genero, k=14)
knn.15 <- knn(train=treino_scale, test=teste_scale, cl=df_treino$Genero, k=15)
ACC.14 <- 100 * sum(df_teste$Genero == knn.14)/NROW(df_teste$Genero)
ACC.15 <- 100 * sum(df_teste$Genero == knn.15)/NROW(df_teste$Genero)
ACC.14
ACC.15

cm_knn<-table(knn.14 ,df_teste$Genero)
cm_knn
confusionMatrix(cm_knn)

#função para encontrar o melhor número de "vizinhos"
i=1
k.optm=1
for (i in 1:28){
   knn.mod <- knn(train=treino_scale, test=teste_scale, cl=df_treino$Genero, k=i)
   k.optm[i] <- 100 * sum(df_teste$Genero == knn.mod)/NROW(df_teste$Genero)
   k=i
   cat(k,'=',k.optm[i],'
')}
#ACURACIA KNN : 80.39%


## KNN ## treino vs. amostra oversampling
treino_O_scale <- scale(df_treino_O[,c(2:16)])

# k=2
set.seed(123)
knn2 = knn(train=treino_O_scale, test=teste_scale,
           cl=df_treino_O$Genero, k=2)
confusionMatrix(knn2, df_teste$Genero, positive = "Masculino")

acc <- vector()
for (i in 2:25){set.seed(123)
  knn1 = knn(train=treino_O_scale, test=teste_scale,
             cl=df_treino_O$Genero, k=i)
  acc <- c(acc, as.numeric(confusionMatrix(knn1, df_teste$Genero)$overall[[1]]))}
which.max(acc)
acc
# melhor é k=16
set.seed(123)
knn1 = knn(train=treino_scale, test=teste_scale,
           cl=df_treino$Genero, k=16)
confusionMatrix(knn1, df_teste$Genero, positive = "Masculino")

### Com CV
set.seed(123)
knn_fit2 <- train(Genero ~ ., data=df_treino_O, method = "knn",
                  trControl=trctrl,
                  preProcess = c("center", "scale"),
                  tuneLength = 10)
knn_fit2
test_pred2 <- predict(knn_fit2, df_teste)
confusionMatrix(test_pred1, df_teste$Genero, positive = "Masculino")




###################################
## SVM - Support Vector Machine ##
```

```
model_svm <- train(Genero ~., data = df_treino, method = "svmLinear",
                   trControl=trctrl,
                   preProcess = c("center", "scale"),
                   tuneLength = 10)

model_svm
#ACURACIA TREINO SVM  79%

predict_treino_svm <- predict(model_svm, newdata = df_treino)
predict_teste_svm <- predict(model_svm, newdata = df_teste)
conf_svm_matrix  <-  confusionMatrix(predict_teste_svm,  df_teste$Genero,  positive  =
"Masculino")

grafico_matriz_confusao(conf_svm_matrix, "SVM - Amostra Teste")

roc_svm_treino      <-      roc(response      =      df_treino$Genero,      predictor
=as.numeric(predict_treino_svm))


#ACURACIA TESTE SVM  78,43%


roc_svm_test <- roc(response = df_teste$Genero, predictor =as.numeric(predict_teste_svm))



## SVM - Support Vector Machine com oversampling
set.seed(123)
model_svm2 <- train(Genero ~., data = df_treino_O, method = "svmLinear",
                    trControl=trctrl,
                    preProcess = c("center", "scale"),
                    tuneLength = 10)
model_svm2
predict_teste_svm2 <- predict(model_svm2, newdata = df_teste)
confusionMatrix(predict_teste_svm2, df_teste$Genero, positive = "Masculino")

roc_svm_over <- roc(response = df_teste$Genero, predictor =as.numeric(predict_teste_svm2))



####################
## RANDOM-FOREST ##

set.seed(123)
model_rf <- train(Genero ~.,
                  data = df_treino,
                  method = 'rf',
                  trControl = trctrl)
plot(model_rf)
predict_teste_rf <- predict(model_rf, newdata = df_teste)
confusionMatrix(predict_teste_rf, df_teste$Genero, positive = "Masculino")

probs <- predict(model_rf, newdata = df_teste, type = "prob")
probs_pos <- probs[, "Masculino"]  # Probabilidades da classe positiva
roc_RF<- roc(response = df_teste$Genero, predictor = probs_pos)

auc <- auc(roc_RF)  # Área sob a curva ROC


## RANDOM-FOREST com oversampling
set.seed(123)
model_rf2 <- train(Genero ~.,
                   data = df_treino_O,
                   method = 'rf',
                   trControl = trctrl)
```

```
plot(model_rf2)
predict_teste_rf2 <- predict(model_rf2, newdata = df_teste)
rf_conf_matrix<-    confusionMatrix(predict_teste_rf2,    df_teste$Genero,    positive    =
"Masculino")

grafico_matriz_confusao(rf_conf_matrix, "Florestas Aleatórias - Amostra Teste")

probs <- predict(model_rf2, newdata = df_teste, type = "prob")
probs_pos <- probs[, "Masculino"]  # Probabilidades da classe positiva
roc_FR_over <- roc(response = df_teste$Genero, predictor = probs_pos)

auc <- auc(roc_FR_over)  # Área sob a curva ROC


plot(roc_RF, print.auc=TRUE, col=2, print.auc.y=0.45,
     main="Curva ROC", xlab="1 - especificidade", ylab="sensibilidade");
plot(roc_FR_over, print.auc=TRUE, col=3, print.auc.y=0.40, add=TRUE);
abline(h=c(0,1), v=c(0,1), lty=3, lwd=2);
legend(0.5, 0.25, c("Treino", "Oversampling"), col = c(2,3), lwd =2)


grafico_matriz_confusao(conf_matrix, "Matriz de Confusão")
```