IA048 - Exercícios de Fixação de Conceitos 2

Kaleb Roncatti de Souza **RA: 171360**
Gabriel Teixeira Callado **RA: 168172**

Parte 1 – Classificação Binária

O primeiro exercício trata-se de classificação binária na qual é preciso analisar um conjunto de dados baseados em propriedades acústicas da voz e determinar se esses dados provém de uma fala masculina ou feminina. O dataset possui 19 *features* e um *label* indicando a resposta esperada da classificação (homem ou mulher). As *features* são grandezas estatísticas da frequência da voz da pessoa (média, desvio-padrão, etc) e, deste modo, grande maioria das *features* é representada em frequências entre 0 e 280Hz. Além do mais, todos os dados já tinham sido pré-tratados em R.

Para a realizar a classificação, primeiro faremos uma análise das *features* apresentadas e a seguir aplicaremos um modelo de regressão logística, cuja escolha de parâmetros será detalhada e todos os resultados serão comentados.

Antes de começarmos a visualizar os dados, a primeira verificação que fizemos foi a procura por elementos de tipo *Null* no conjunto de dados. Verificamos então que todas as 3168 amostras possuíam valores *Non-Null's* para todos os dados.

Ademais, quando aplica-se qualquer método de *Machine Learning* que seja supervisionado, é muito importante determinarmos se o conjunto de dados está balanceado com respeito aos *label's* - ou seja, ter um número equivalente de amostras para cada *label*. O balanço na quantidade de amostras para cada classe é determinante para o modelo ter um bom funcionamento e também para a escolha de uma métrica de avaliação que seja coerente com este balanço. A distribuição dos dados do nosso problema é mostrada na Figura 1 na forma de um histograma. Neste caso, observamos há o mesmo número de homens e mulheres sendo classificados - o que será considerado no futuro para a escolha de nossa métrica de avaliação do modelo.

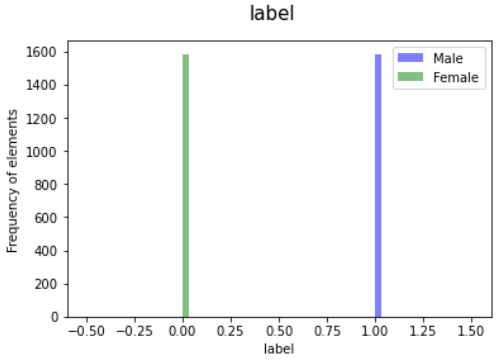

Figura 1: Frequência de cada *label* no dataset.

Antes de aplicarmos o modelo propriamente dito, observamos os histogramas para todas as 19 *features* presentes em função das *label's* como mostrado nas Figuras 2, 3 e 4.

A ideia de classificarmos em uma determinada categoria (homem ou mulher) com relação à um determinado atributo (*feature*) está diretamente ligada à separação dos dados dentre estas categorias com respeito à este atributo. O histograma é uma maneira simples de realizarmos esta visualização quando o número de atributos e de categorias não é tão alto.

Na Figura 2, conseguimos ver que o desvio padrão das frequências de voz para homens é maior do que para mulheres em geral - deste modo, a *feature* desvio padrão será um bom indicativo para dizer se aquela amostra pertence à classe homem ou mulher. Para a mediana das frequências vocais, vemos que os valores aparentemente concentram-se em torno do mesmo valor de frequência (200Hz) - tal mistura sugere que esse indicador não consegue diferenciar bem homens e mulheres. Para o primeiro quantil (Q25 em kHz),

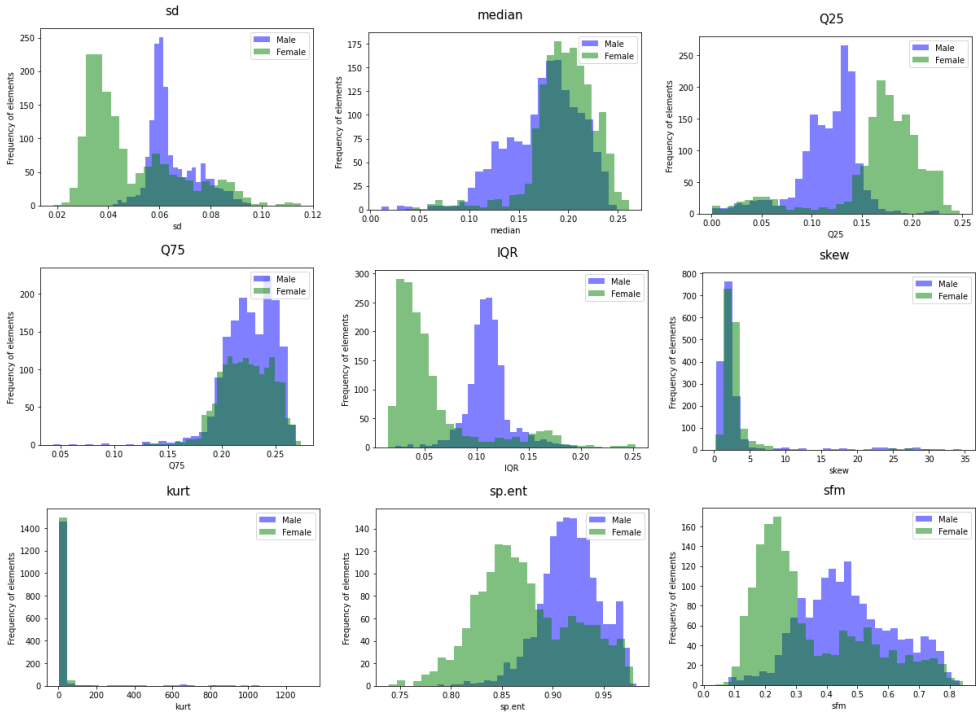

Figura 2: Histogramas em função do número de homens e mulheres.

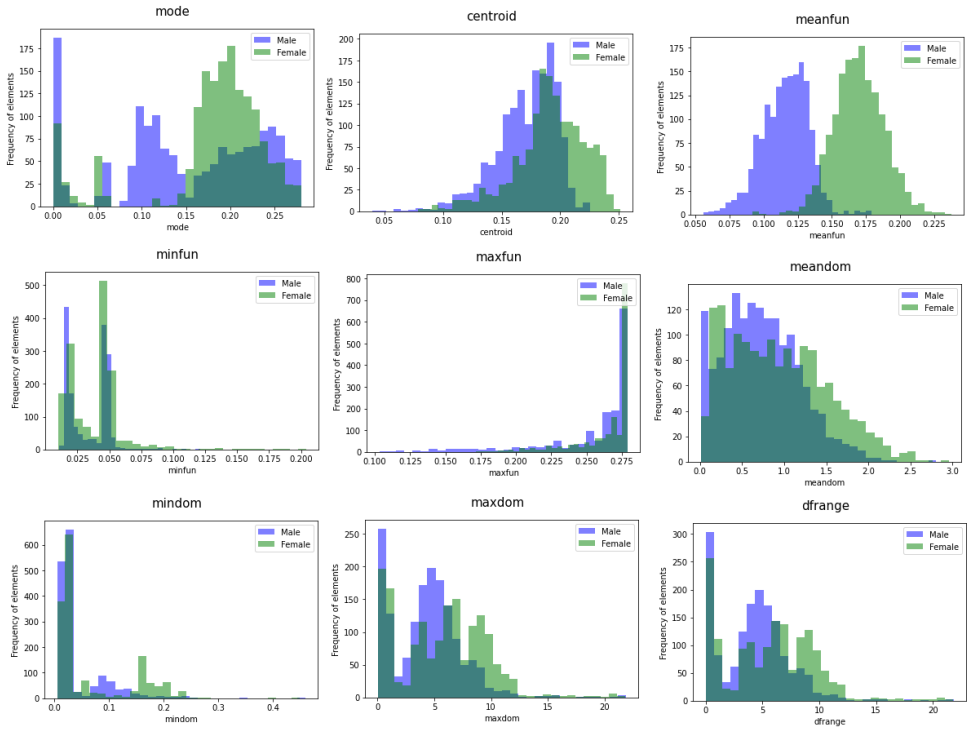

Figura 3: Histogramas em função do número de homens e mulheres.

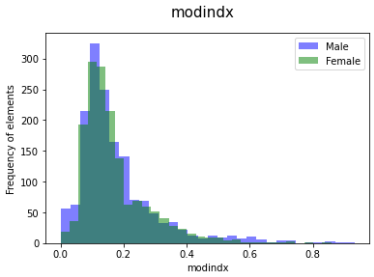

Figura 4: Histogramas em função do número de homens e mulheres.

estamos na verdade observando os 25% dos dados de frequência de voz mais baixa de cada conjunto de dados (masculino / feminino). Desta forma, podemos inferir que os homens com voz grave apresentam uma frequência de voz mais baixa que as mulheres com voz grave. Novamente, trata-se de uma boa *feature* para diferenciar homens e mulheres.

Para o terceiro quantil (Q75 em kHz), estamos de fato observando 25% dos dados de frequência de voz mais alta de cada conjunto de dados (masculino / feminino). É mais difícil distinguir as duas classes e, desta forma, se pudéssemos escolher as *features* (atributos) para alimentar nosso modelo, poderíamos deixar este atributo de fora. Porém, também podemos perceber que a quantidade de frequências mais altas para os homens neste quantil é maior do que para as mulheres. Por motivos de simplificação, não continuaremos a analisar histograma por histograma dado que a ideia geral da análise já foi esclarecida.

Assim sendo, conseguimos chegar à conclusão de que, apenas observando os histogramas apresentados anteriormente, conseguiríamos selecionar as *features* mais relevantes para a separação das classes e ignorar algumas menos relevantes, o que resultaria numa maior simplificação do modelo e em alguns casos numa maior eficácia.

Uma maneira mais simples de escolhermos as melhores *features* para alimentarmos o modelo levando em conta o mesmo aspecto de "separação"entre os dados aos quais discorremos anteriormente é através de uma análise de correlação entre as *features* e as *label's*. Uma correlação é considerada como a relação entre duas variáveis aleatórias. Mais especificamente, através da correlação somos capazes de determinar quão próximas

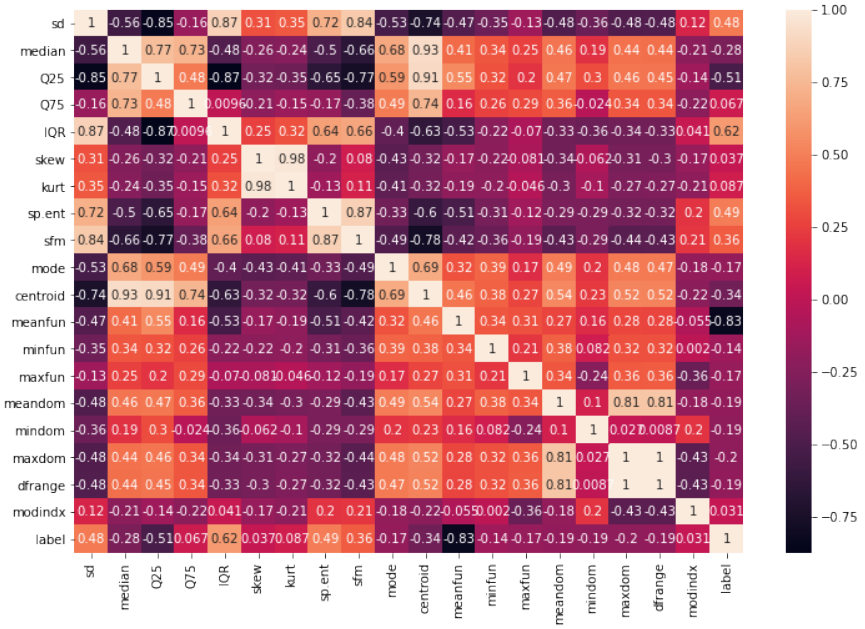

Figura 5: Correlação entre atributos e rótulos.

duas variáveis estão de ter um relacionamento linear entre si. Isto significa que, se duas características têm alta correlação, em tese, poderíamos abandonar uma delas e trabalhar apenas com a outra, já que a adição da segunda não inseriria mais informações do que já tínhamos apenas com a primeira. Da mesma forma, se uma *feature* tem uma grande correlação com as *labels*, esta *feature* é de grande importância para o modelo de classificação pois indicaria com certa precisão à qual categoria a amostra pertence. Um *heatmap* para a correlação é apresentado na Figura 5.

De acordo com a correlação de Pearson usada anteriormente, um valor mais próximo de 0 implica numa baixa correlação, um valor mais próximo de 1 implica correlação positiva mais forte e um valor mais próximo de -1 implica correlação negativa mais forte. Assim sendo, podemos analisar os valores de correlação em módulo, pois o fato de haver uma alta correlação positiva ou negativa são recíprocos para fins de classificação.

Para a escolha das melhores *features* podemos observar a última linha ou a última coluna da Figura 5 que representam a correlação entre cada atributo e os rótulos. Note que, o atributo *meanfun* apresenta a maior correlação em módulo dentre todos os atributos. Se retornarmos na Figura 3 e observarmos o histograma do atributo *meanfun* chegaremos à conclusão de que, dentre TODOS os histogramas, este é o que apresenta maior "separação"entre homens e mulheres com relação aos dados de frequência, o que representaria uma ótima *feature* para alimentar o modelo.

Uma maneira de usar a correlação seria definir um valor de *threshold* para o módulo da correlação entre as *features* e as *labels*, acima do qual escolheríamos tais *features* para utilizar no modelo. Um exemplo usando-se 0.3 como *threshold* é apresentado na Figura 6.

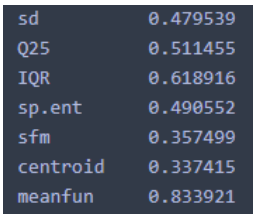

Figura 6: Módulo da Correlação de Pearson maiores do que threshold = 0.3.

Novamente, se observarmos os histogramas das Figuras 2, 3 e 4 para os atributos da Figura 6, veremos que são os atributos que melhor separam os dados dentre homens e mulheres, resultando em melhores características para classificação. Então, caso fossemos escolher alguns atributos para simplificar o modelo no pré-tratamento, estes seriam os atributos mais adequados para tal finalidade.

Para prosseguirmos com a aplicação do modelo de regressão logística, utilizamos **TODAS** as *features* e separamos 20% dos dados num conjunto de validação aplicando a técnica de ***hold-out*** como sugerido pelos docentes.

Para a separação dos dados em conjunto de treinamento e teste, percebeu-se que todos os dados não estavam randomizados. Desta forma, aplicamos uma randomização para que os dados com *label* '1' (homem) e '0' (mulher) se misturassem e não houvesse nenhuma tendência específica durante a aprendizagem do modelo.

Analisando-se os dados de entrada nos histogramas, percebemos que alguns atributos não são dados de frequência (kHz) propriamente ditos (como *kurtosis* e *skewness*) e portanto temos dados que não estão entre 0 e 0.28. Por consequência, decidimos realizar uma normalização do tipo MinMaxScaler() para que todos os atributos variem no mesmo *range* e não haja distorções em decorrência disso.

Para a aplicação do modelo de regressão logística escolhemos penalizar o modelo através de uma regularização l_2 para uma melhora na estabilidade numérica e para evitar *overfitting* nos dados de treinamento.

Com relação à função de perda que pode ser obtida através do princípio de máxima verossimilhança, utilizou-se a entropia cruzada binária, a qual representa a melhor maneira de penalizar um modelo de classificação binário por ter acertado ou errado o rótulo.

Com relação ao treinamento a ser utilizado para chegarmos no mínimo da função de perda, escolhemos o modelo conhecido como Stochastic Average Gradient (SAG), o qual representa uma variação do modelo de gradiente descente (SGD) que usamos em classe. A diferença entre os modelos é que, o modelo SAG, ao invés de calcular o gradiente para TODOS os dados do conjunto a cada iteração, ele escolhe aleatoriamente um dado e calcula o seu gradiente a cada iteração, e desta foram todos os outros gradientes são mantidos em

seu valor anterior. Este passo que damos em direção ao gradiente randomicamente escolhido garante uma otimização rápida pois somos obrigados a calcular apenas um gradiente por iteração. De qualquer forma, a escolha da forma de treinamento (SGD, Método de Newton, SAG, etc.) neste caso específico pode apenas melhorar ou piorar a velocidade com a qual iremos convergir ao mínimo de perda, pois a função de perda na classificação binária é convexa e possui um mínimo global.

Mudando o *threshold* na regressão logística para o qual atribuímos uma determinada predição da probabilidade à classe masculina ou feminina, fomos capazes de obter a curva ROC como mostrado na Figura 7.

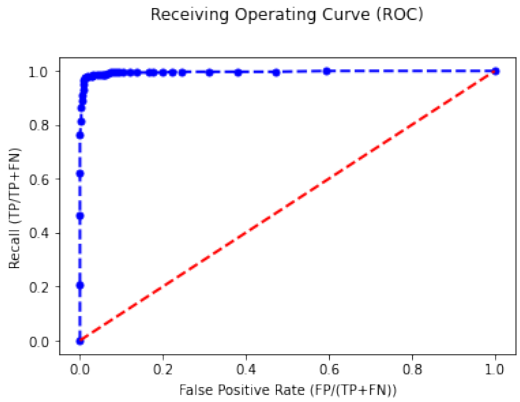

Figura 7: Gráfico ROC para diferentes valores de *threshold*.

Uma das maneiras de se avaliar o modelo é verificando o quão acima da curva y = x (classificador aleatório) está nossa curva ROC. Um modelo perfeito passaria pelo ponto (false positive rate = 0,recall = 1). Então, o fato de conseguirmos chegar muito perto deste ponto mostra que nosso modelo possui um bom desempenho. Os melhores *thresholds* para classificação são os pontos mais próximos de um *recall* máximo (1) e de um *false positive rate* mínimo (0).

A verdadeira utilidade da curva ROC é evidenciada quando tem-se múltiplos modelos. Plotando-se as curvas ROC para múltiplos modelos no mesmo gráfico e calculando-se a maior área abaixo da curva é possível determinar qual modelo desempenha melhor.

Assim sendo, calculamos o valor da F_1score em função de cada uma dos *thresholds* como mostrado na Figura 8. Observe que, existe uma gama de valores de *threshold* para os quais a métrica de avaliação apresenta alto *score*.

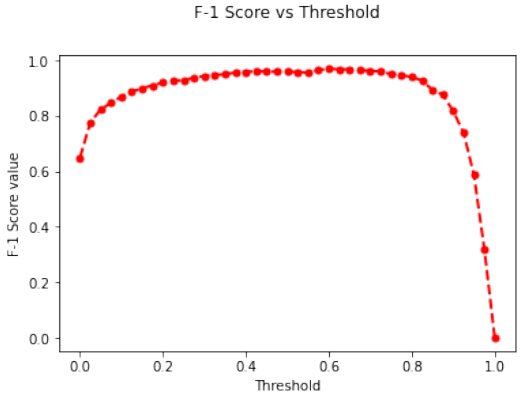

Figura 8: F_1score em função dos valores de *threshold*.

Para a escolha do melhor modelo, decidimos usar o *threshold* com maior F_1score da Figura 8. Para explicarmos o porquê a escolha de tal métrica ser coerente com o problema precisamos entrar em mais detalhes na definição da F_1score. Observe que:

$$F_1 = 2\frac{recall(\hat{y}(x)) \times precision(\hat{y}(x))}{recall(\hat{y}(x)) + precision(\hat{y}(x))}$$

onde

$$recall(\hat{y}(x)) = \frac{TP}{TP + FN}$$

$$precision(\hat{y}(x)) = \frac{TP}{TP + FP}$$

Note que o ***recall*** quantifica a taxa de verdadeiros positivos com relação à todos os elementos que são de fato positivos. De forma parecida, a **precisão** quantifica a taxa de quantos elementos são verdadeiramente positivos com relação à todos os elementos classificados como positivos. Desta forma, como o F_1score pondera *recall* e *precisão*, a métrica dá ênfase aos rótulos positivos. Se o *dataset* é não-balanceado, devemos tomar cuidado com qual classe é considerada como positiva e qual classe é considerada como negativa, pois se nos equivocarmos com qual classe tem maior "importância", o F_1score pode ser alto com um desempenho inadequado do modelo. No nosso caso, as classes estão completamente balanceadas como mostrado na Figura 1, e assim sendo, a métrica se mostra adequada.

Observe também que, a métrica dá o mesmo peso para o recall e para a precisão, o que é coerente com nosso problema visto que não existe nenhuma razão particular pela qual deveríamos priorizar falsos positivos em detrimento aos falsos negativos. Por exemplo, para o caso de um classificador que determina se um filme é adequado para uma criança assistir ou não, a precisão e o recall não deveriam possuir o mesmo peso pois seria muito mais adequado treinar um modelo que rejeita filmes que são adequados (baixo recall) mas mantendo os que são seguramente adequados (alta precisão). O que é completamente diferente do exemplo em que desejamos determinar se um vídeo de uma câmera de segurança representa um roubo ou não, pois neste caso desejamos treinar um modelo que, mesmo com alarmes falsos (baixa precisão), selecione todos os vídeos em que realmente um roubo acontece (alto recall).

Também poderíamos ter utilizado a acurácia visto que as classes são completamente balanceadas e falsos negativos/falsos positivos são igualmente importantes. A acurácia em função do *threshold* é mostrada na Figura 9.

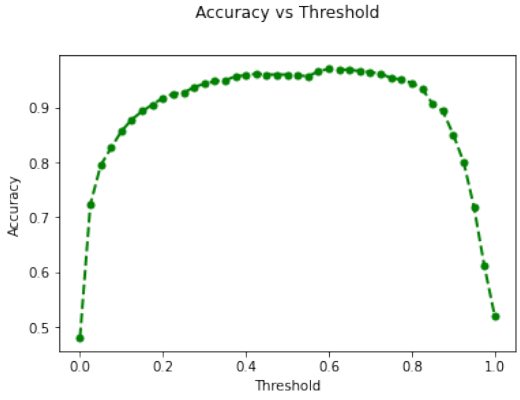

Figura 9: Acurácia em função dos valores de *threshold*.

Como conclusão da nossa análise, o esperado é que ambas as métricas F_1score e acurácia apresentem um valor próximo de *threshold* ótimo visto que as duas métricas são adequadas. Chegamos ao valor ótimo de ***threshold* de 0.6** em ambas métricas, nas quais observamos F_1score = 0.969 e Acurácia = 0.971.

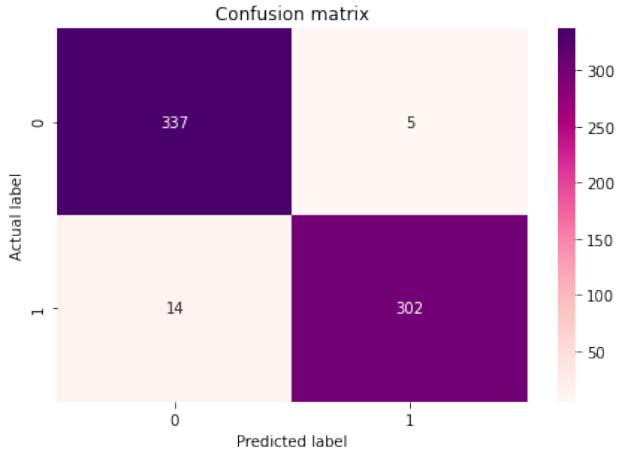

Figura 10: Matriz de confusão para o *threshold* de melhor desempenho $threshold = 0.6$.

A matriz de confusão para o melhor modelo é apresentado na Figura 10. Observe que, o modelo tem mais dificuldade em classificar um dado quando o rótulo real é '1' (mulher) pois o número de falsos positivos é três vezes maior do que o número de falsos negativos. Com relação à acurácia, o modelo tem um bom desempenho pois 97% das amostras de validação foram classificadas corretamente. Como dito anteriormente, esta métrica é extremamente pertinente para classificar o modelo visto que o conjunto de dados é balanceado e o peso entre falsos positivos e negativos deve ser o mesmo.

Parte 2 – Classificação Multi-classe

A segunda parte do EFC2 trata-se de um exercício de classificação multi-classe. O conjunto de dados disponibilizado foi extraído de sinais de acelerômetro e giroscópio de um smartphone durante diversas atividades de um voluntário humano. Esses dados, apresentados no domínio do tempo e da frequência, vão nos ajudar à classificar as amostras nos seguintes rótulos: caminhada (0), subindo escadas (1), descendo escadas (2), sentado (3), em pé (4) e deitado (5).

Há disponível 7352 amostras de treinamento e 2947 amostras de teste, cada uma com 561 *features* temporais ou espectrais - note que a separação dos dados em teste e treino já foi definida.

Como desafio, foi sugerido explorar dois método diferentes de classificação multi-classe: regressão logística e *k-nearest neighbors*.

A métrica global de avaliação do desempenho será a acurácia. Vamos perceber, pela matriz de confusão dos modelos, que os dados estão separados de maneira equilibrada nas diferentes classes e não temos um grande interesse em enfatizar a precisão ou o recall.

Regressão Logística

A Regressão Logística foi realizada com o auxílio da biblioteca *sklearn*. Como toda a função já está implementada e os dados já estão separados para o uso, nossa única dificuldade foi escolher os parâmetros para a regressão.

Para a regressão logística, testamos duas abordagens disponíveis pelo *sklearn*: *softmax* e um-contra-todos. Mantendo todos os parâmetros fixos, notamos que o desempenho das abordagens é praticamente o mesmo - método de um-contra-todos apresentou uma acurácia um pouco melhor (96% contra 95,9%). Afim de ilustrar as previsões, exibimos a matriz de confusão em ambas as abordagens (Figuras 11 e 12).

Através das matrizes de confusão, notamos que os resultados dos modelos abordados são extremamente similares. Podemos ainda visualizar a disposição dos dados entre as classes e notar a equivalência no número de amostras entre elas (apesar de não conter exatamente a mesma quantidade de dados em cada classe). Vale mencionar que a maior dificuldade dos modelos, segundo a matriz de confusão, é prever quando a pessoa está sentada - o nosso maior erro ocorre quando dizemos que ela está em pé quando na verdade está sentada (56 erros para a *softmax* e 53 erros para o método um-contra-todos).

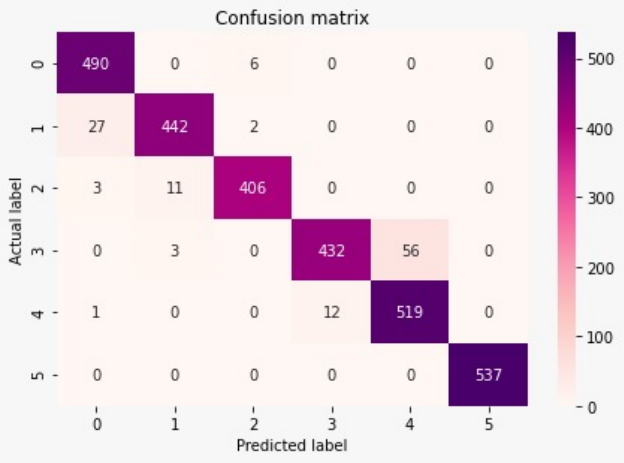

Figura 11: Matriz de confusão para a abordagem da softmax

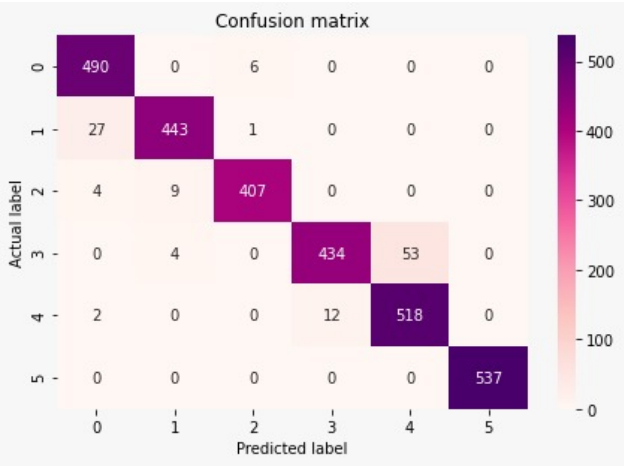

Figura 12: Matriz de confusão para a abordagem de um-contra-todos

Uma grande desvantagem do método um-contra-todos, entretanto, é o seu tempo de execução - é necessário um tempo muito maior que para o método da *softmax*. Deste modo, considerando o baixo tempo de execução e a ótima acurácia, optamos por usar o método da *softmax*.

Vale ressaltar também que tivemos que escolher um *solver* para o treinamento do modelo. O escolhido foi o *SAG* (explicado na parte 1). Sabemos que a *softmax* é uma função convexa e, portanto, apresenta apenas um mínimo global - logo, a escolha do *solver* mudará apenas o tempo de convergência do modelo. Com relação à regularização, testamos *l1*, *l2* e *none*. Para a regularização *l2*, obtivemos uma acurácia de 95,9%; para a *l1*, 95,6%; para o *none*, 95,9%. Deste modo, o melhor modelo apresentado é o modelo da figura 11 com a regularização *l2*.

K-Nearest Neighbors

Para a abordagem do *K-Nearest Neighbors*, utilizamos novamente a biblioteca *sklearn* para a geração do modelo. A parte determinante da implementação foi escolher os hiperparâmetros e analisar os resultados dos diferentes modelos.

Além do evidente hiperparâmetro k número de vizinhos, podemos escolher o tipo de ponderação (uniforme ou ponderado pela distância até o vizinho) e o tipo de métrica usada para calcular distâncias (p=1 ou p=2 para a distância de Minskowski). O gráfico abaixo mostra o acurácia do modelo para diferentes ponderações e tipos de distância:

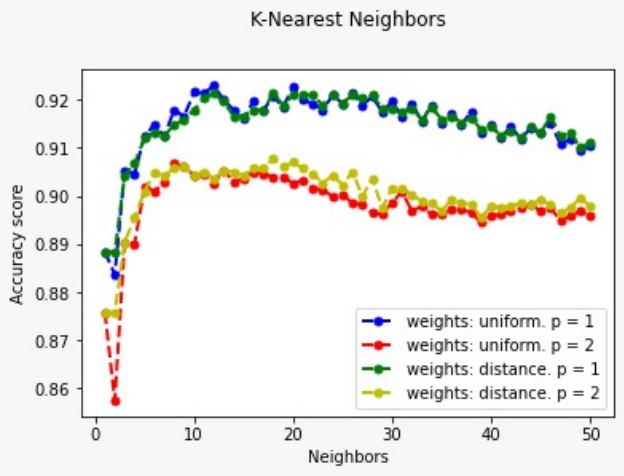

Figura 13: Acurácia do KNN para diferentes modelos

Pela figura 14, vemos que os modelos com o tipo de distância p=1 tiveram um desempenho melhor que os modelos do tipo p=2. Com relação ao tipo de ponderação, vemos que no caso p=2 os pesos ponderados pela distância tiveram um desempenho melhor que os pesos uniformes ao aumentarmos o número de vizinhos. No caso p=1, eles apresentaram quase a mesma acurácia.

Por fim, vemos que o melhor modelo de *k-nearest neighbors* teve, com um número de vizinhos k = 12, p=1 e pesos uniformes, uma acurácia de 92,3% - um pouco abaixo do desempenho apresentado pela *softmax* no modelo de regressão logística. Podemos visualizar os erros apresentados pelo modelo de kNN com a matriz de confusão abaixo:

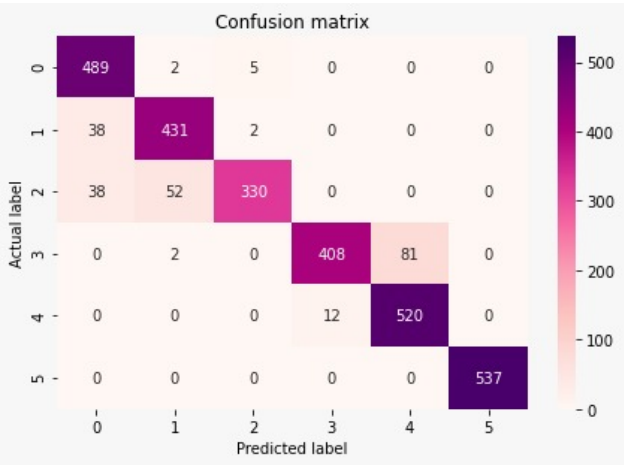

Figura 14: Matriz de confusão do modelo kNN

Pela matriz de confusão apresentada, vemos que o modelo kNN teve ainda mais dificuldade que o modelo de regressão logística para diferenciar a pessoa sentada e de pé. Ainda, no caso da pessoa descendo a escada (*actual label* = 2), vemos que o modelo kNN aumentou bastante o erro e previu diversas que a pessoa estava caminhando ou subindo as escadas (*predicted label* = 0 para caminhada e 1 para subindo as escadas).

A abordagem de *K-Nearest Neighbors* é bastante interessante pela sua simplicidade e a não necessidade de treino do modelo. Entretanto, para uma quantidade de dados relativamente alta (mais de 10000 amostras), o modelo se mostrou bastante lento. Além disso, teve um desempenho inferior ao modelo de regressão logística com a *softmax*, que conseguiu diferenciar classes similares com mais exatidão.