**Mestrado em Engenharia Eletrónica e de Computadores**

**Laboratórios Integrados II**

**Relatório**

**Identificação de pessoas e**

**Keywords em imagem e som**

**2019/2020**

***Grupo 1:***

*Cláudia Pereira, nº 13289*

*Luís Martins, nº 17985*

# Objetivos

No âmbito da disciplina Laboratórios Integrados II, implementamos um sistema de autenticação com base em dois fatores: imagem e som. Além de identificação do utilizador o sistema permite a deteção de comandos de voz.

* Identificação de utilizador por mapeamento facial
* Identificação de utilizador por reconhecimento de voz
* Identificação de comandos de voz

# Tarefas executadas

1. Módulo de interface gráfico com o utilizador, em tempo real:
   1. Imagem da webcam e notações.
   2. Representação gráfica do som.
   3. Apresentação dos resultados de identificação.
2. Módulo de Imagem, obtida por uma câmara:
   1. Treino do modelo de identificação de utilizador por mapeamento facial.

Para o treino dos modelos de deteção facial, foi decidido que cada grupo, dos oito envolvidos neste trabalho, recolhia vinte fotos de rosto de um dos elementos de grupo. Estas imagens foram guardadas numa pasta identificando cada ficheiro com o grupo a que pertence a imagem.

Programou-se um iterador para obter um endereço de cada ficheiro que após leitura foi fornecido ao modelo de treino. Como modelo optamos pelo *Local Binary Pattern Histogram(LBPH)*. O modelo usa uma conjugação de binarização, mapeamento e analise de histograma da imagem, o qual será adicionado ao modelo para treino. Efetivamente o modelo faz uma comparação de histogramas das imagens, para gerar uma previsão da identidade do rosto detetado.

As imagens disponíveis para o trabalho foram divididas em dois grupos aleatórios, um para o treino com 80% das imagens sendo as restantes 20% reservadas para teste. Para o teste as imagens, estas tiveram de ser testadas uma de cada vez, guardando os resultados num array que posteriormente foi comparado com as etiquetas de identidade conhecida (resultados apresentados no capítulo seguinte).

* 1. Identificação de faces, na imagem em tempo real.

1. Módulo de som, obtido por um microfone:
   1. Organização dos clipes para treino.

Foi decido como formato comum para recolha dos clipes sonoros, para todos os grupos de trabalho, apenas um dos elementos do grupo gravou sons, têm um único comando em cada ficheiro o qual é identificado no seu nome pelo grupo e comando. Estes clipes de áudio, foram gravados em formato WAV a 16bits, com uma amostragem de 44100Hz. São oito grupos de trabalho, com oito comandos de voz (parar, recuar, direita, esquerda, baixo, centro, cima, avançar), para cada comando cada utilizador recolheu vinte repetições.

Como treinamos dois modelos, um para identificação de utilizador e outro para deteção de comando, optamos por dividir os clipes em duas pastas. A primeira pasta destinada ao treino de identificação do utilizador, os clipes estão divididos em subpasta cada uma com todos os comandos de cada grupo. Para o treino de comandos repetiu-se o formato de subpastas, mas neste caso cada pasta contém os clipes de cada comando de todos os grupos.

* 1. Leitura e processamento dos clipes para treino

A leitura de cada ficheiro de áudio é feita com a função “read” da biblioteca de SoundFile. O programa itera por todas subpastas de forma automática recolhendo a informação de endereçamento, que é fornecida à função mencionada atrás, e guarda num array as correspondentes tags de identificação de grupo ou de comando, dependendo do modelo treinado.

Sendo que a quantidade de clipes para treino é reduzida, optou-se por fazer Data Augmentation destes dados. Como primeiro passo detetou-se possíveis silêncios tanto no início do clipe como no final, os quais foram removidos. Como os clipes não têm uma intensidade uniforme, e o método utlizado na deteção de silêncios foi por limiar de potencia mínima, encontramos um compromisso de forma a retirar a maioria dos silêncios nos clipes com maior intensidade, sem remover zonas de sinal de comando nos clipes com menor intensidade. É de mencionar que o tempo a que corresponde à maior duração de clipe de treino após retirada de silêncios é de aproximadamente 2.2 segundo, bastante superior ao espectável dado a duração prevista dos comandos. Após a retirada dos silêncios normalizou-se a duração de todos os clipes para 2.3 segundo, o preenchimento dos dados em falta foram feitos com zeros. A data augmentation foi efetuada por deslocamento do ficheiro em saltos de 4410 posições, recorrendo à função “roll” da biblioteca Numpy. Para evitar que valores de sinal fossem deslocados pela função para o início dos clipes, restringimos este movimento a 11 saltos, correspondentes ao menor preenchimento com zeros feitos na etapa de normalização da duração dos clipes.

Na extração de caraterísticas de treino dos modelos optou-se por usar os coeficientes de Mel, para tal usamos a função “feature.mfcc” da biblioteca Bibrosa. Usamos como parâmetros da função num\_mfcc=13, n\_fft=2048, hop\_length=512. A informação recolhida de cada clipe de áudio foi adicionada a um array que foi passado para a função de treino dos respetivos modelos

* 1. Treino dos modelos

Para treino dos modelos ao contrário do que inicialmente estava previsto, optamos por uma abordagem de teste com todos os grupos/comandos preterindo o um-vs-todos. Após todas as caraterísticas serem extraídas no passo anterior, fizemos a divisão em 75% dados de treino e 25% dados de teste. Com os dados de treino executamos a função StandardScaler da biblioteca Scikit-learn, para normalizar os dados. Esta normalização foi aplicada a todos os dados, e guardamos o modelo para aplicação aquando da previsão em tempo real.

Para o treino optamos por utilizar o MLPClassifier da biblioteca Scikit-learn, utilizando 3 camadas com 52 neurónios na primeira camada, 26 neurónios na segunda e 13 na última. Como parâmetros do modelo usamos como função de ativação o “relu”, “batch\_size” de 30, taxa de aprendizagem de 0.001, máximo de iterações de 200 e solver do tipo “adam”.

Terminado o treino, guardamos o modelo e testamos a sua preformasse com os dados de teste reservados. Finalmente apresentamos a respetiva matriz de confusão e o relatório do modelo (resultados apresentados no capítulo seguinte).

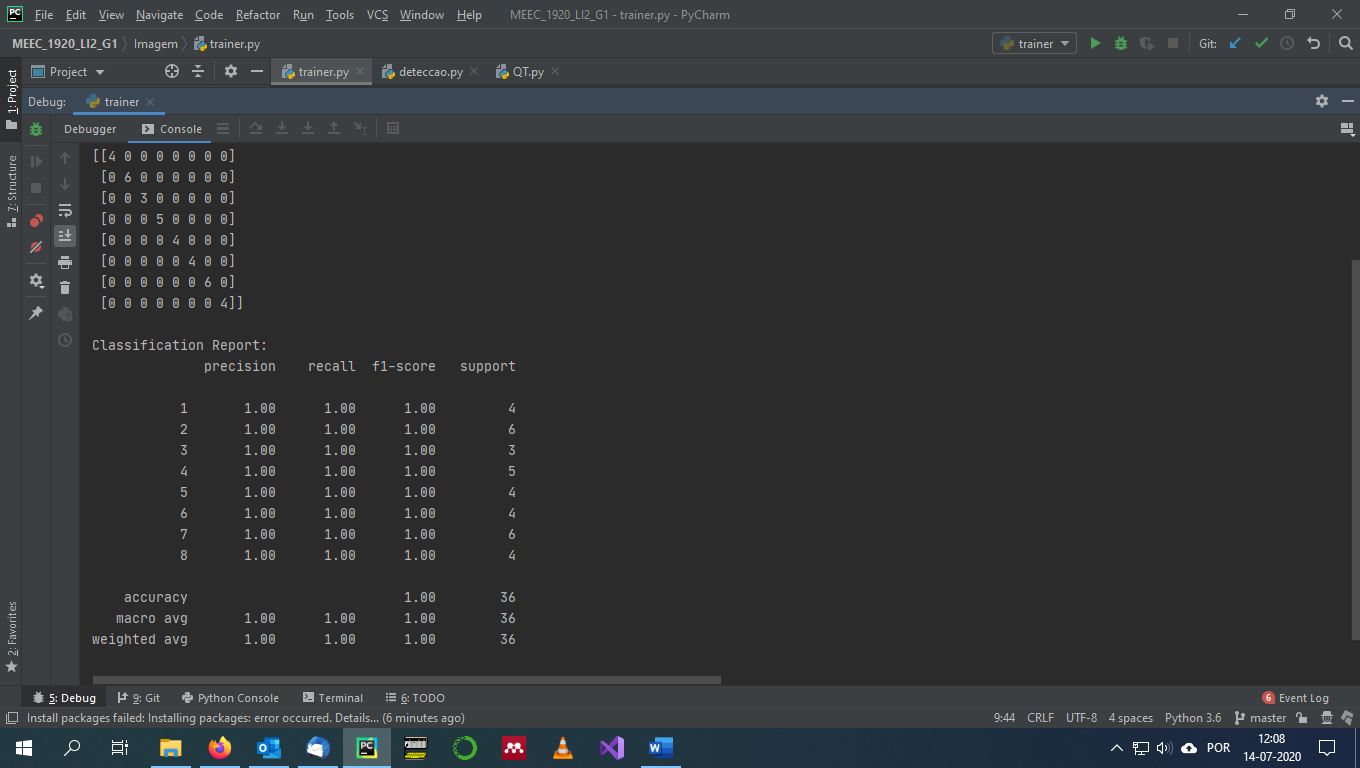
* 1. Aquisição do som a partir do microfone do computador.

A aquisição de som pelo microfone é feita em tempo real sendo processados os últimos 2.3 segundos finais com deslocação de janela de leitura de um quinto de segundo, o que proporciona uma sobreposição da amostra de som atual com a anterior de 2.1 segundos. Esta metodologia apesar de computacionalmente mais exigente, é necessária para garantir que o comando de voz será detetado integralmente pelo sistema. Após a recolha do clipe sonoro a testar, os dados passam pelo mesmo processo explicado no treino. Primeiro extraímos os valores de MFCC, aos quais é aplicado o modelo de normalização guardado no processo de treino, após a qual é enviado para a função “predict” do modelo MLP treinado, o qual devolve a previsão de identidade de utilizador assim como do comando detetado.

* 1. Apresentação da onda e dos resultados da previsão.

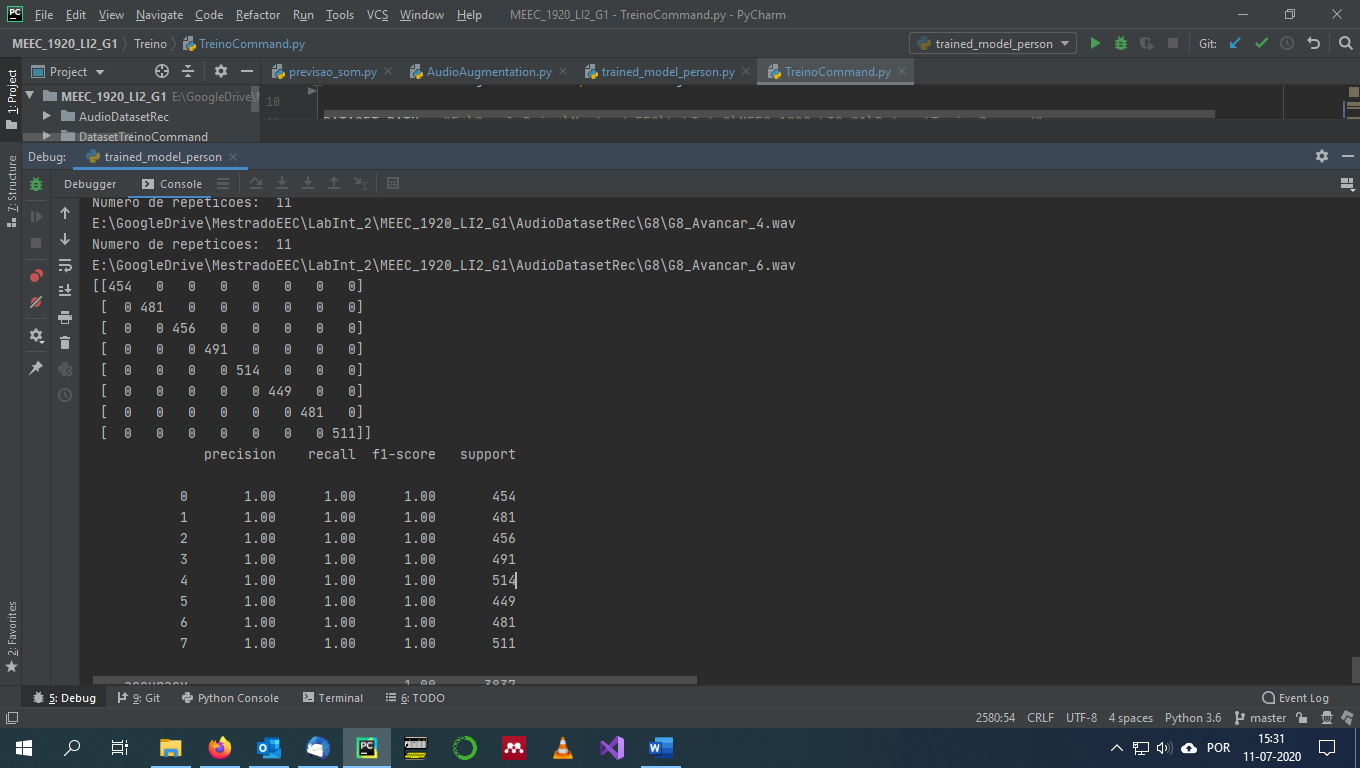
# Resultados

O teste do modelo treinado com *Local Binary Pattern Histogram*(LBPH), não apresenta qualquer falha.

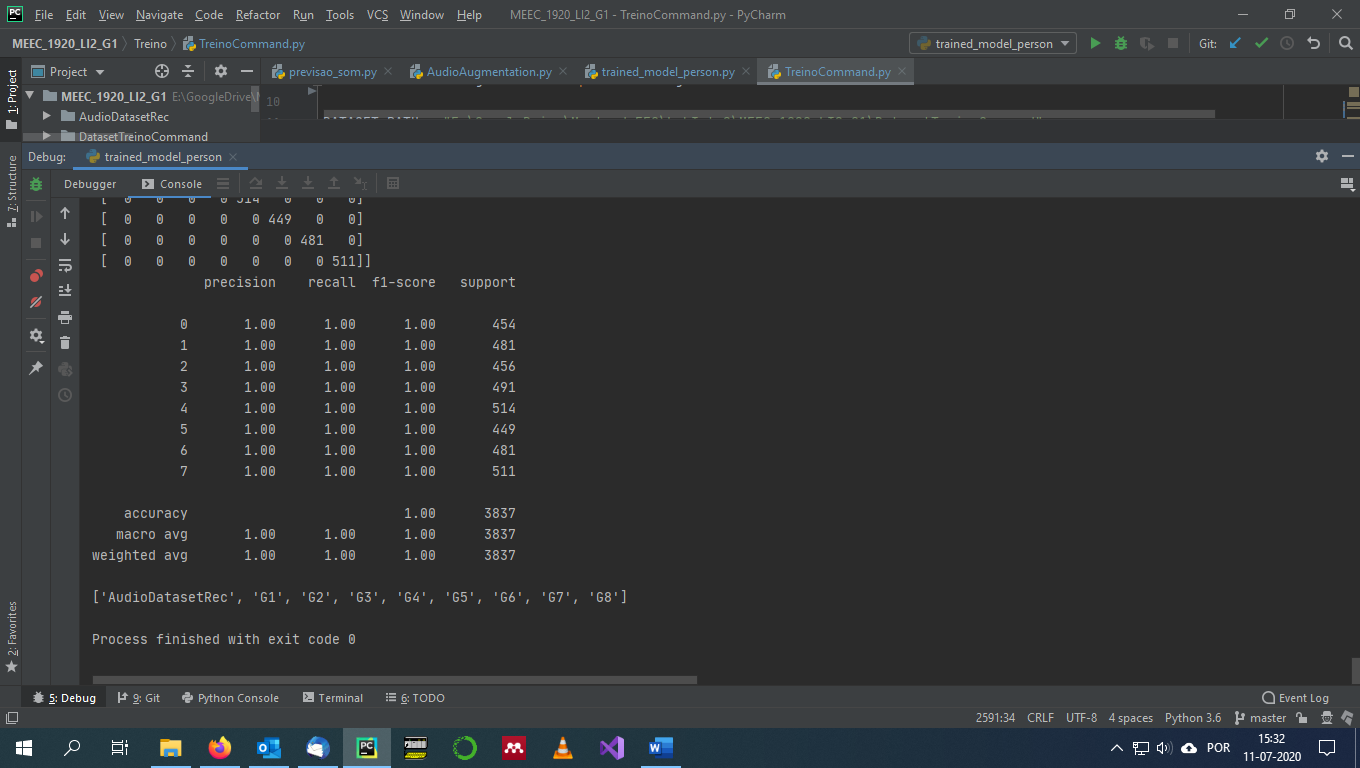


Matriz confusão e relatório – modelo identificação do utilizador imagem

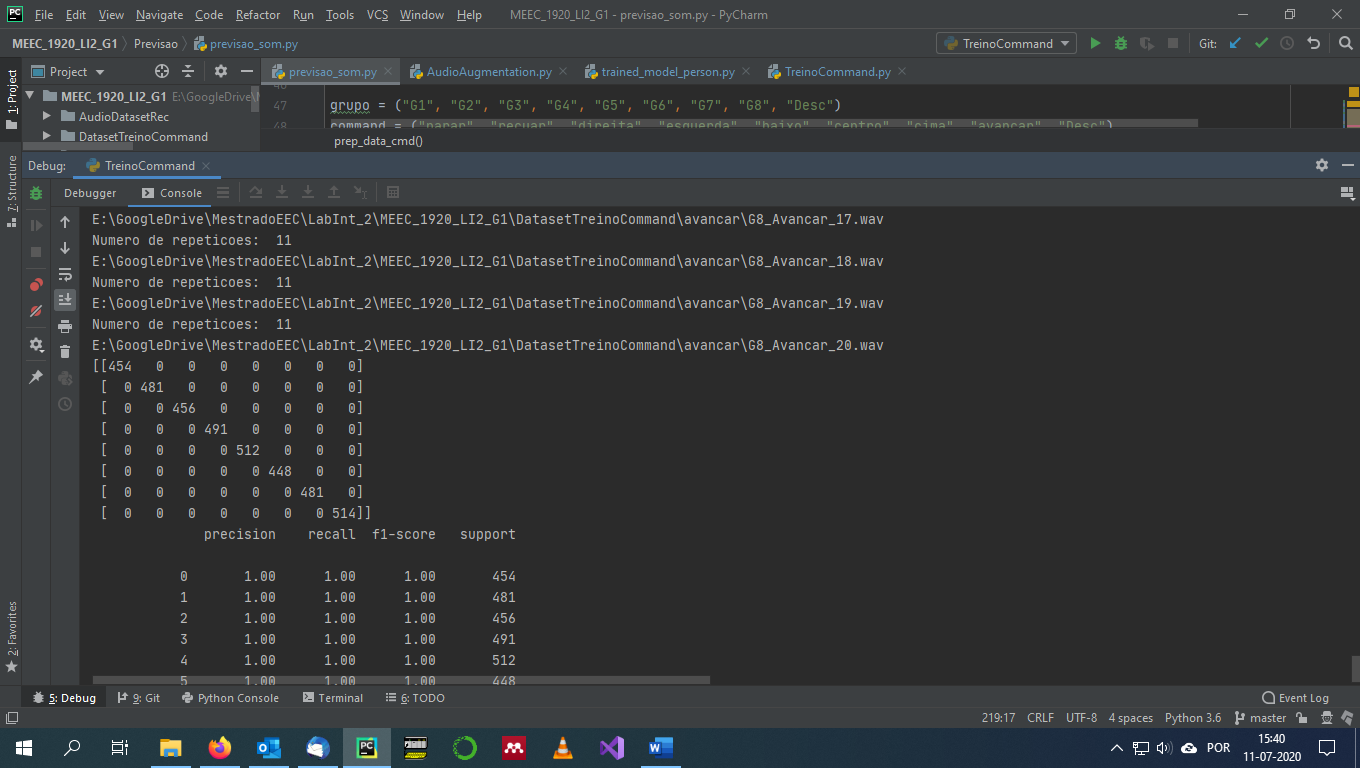
Abaixo são apresentados os resultados obtidos durante o treino dos modelos para a componente audio do sistema. Verifica-se para ambos os modelos ausência de falhas durante os testes, com os 25% de dados reservados.



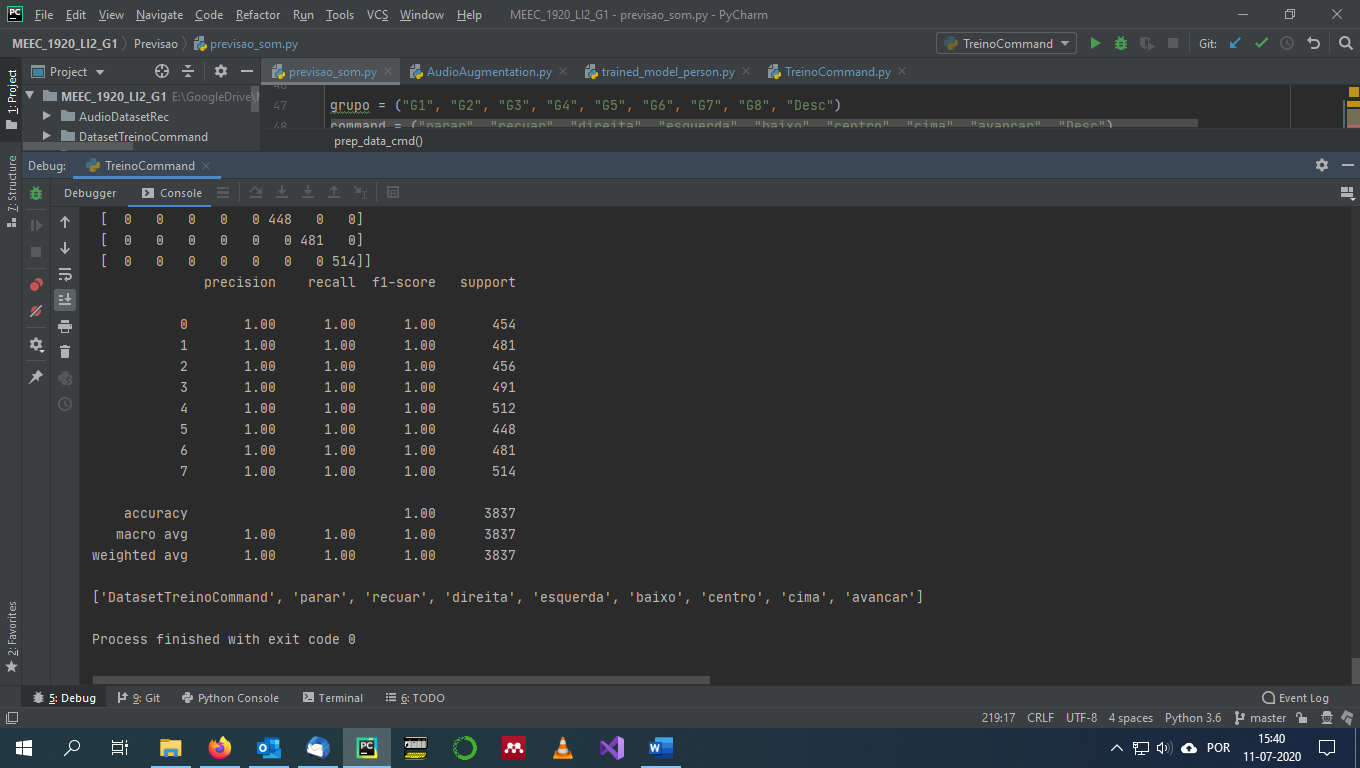
Matriz confusão – modelo identificação do utilizador



Relatório – modelo identificação do utilizador



Matriz confusão – modelo deteção de comandos



Relatório – modelo deteção de comandos

# Conclusão e trabalhos futuros

Os resultados de identificação, principalmente dos comandos de voz, menos favoráveis impele uma nova abordagem para futuro. Olhar para a deteção dos comandos não como uma palavra, mas como um conjunto de fonemas. Posteriormente procurar identificar a palavra correspondente aquele conjunto de fonemas. Como esta é uma abordagem consideravelmente diferente da adotada inicialmente neste trabalho, e requer uma elevada carga de nova programação e teste, não foi possível implementar no espaço temporal disponível para este trabalho.

Igualmente para continuação do trabalho no futuro, pensamos na implementação do sistema em C++. Isto advém de ao longo das pesquisas para este trabalho, vários técnicos na área de software/aplicativos áudio, mencionar o considerável aumento de preformasse dos sistemas áudio quando têm por base esta linguagem de programação, comparativamente a outras de mais auto nível como o Python.