Relatório do Projeto — Mineração de Dados

Nathalia Tescarollo Gonçalves e Renan Baisso

Novembro de 2019

[1]

1. Introdução

Este projeto tem como objetivo desenvolver um classificador capaz de fazer predições sobre gravações de diferentes caracteres.

A base de dados, sobre a qual as análises foram feitas, é formada por arquivos de áudio (extensão .wav), sendo que cada arquivo contem uma sequência de quatro caracteres. Cada caractere foi gravado separadamente e em ordem aleatória, utilizando uma ferramenta de gravação. A duração de cada gravação individual é de dois segundos, e os caracteres possíveis são: a,b,c,d,h,m,n,x,6,7.

Os arquivos de treinamento foram extraídos do diretório "TREINAMENTO", e os arquivos de teste foram extraídos do diretório "VALIDAÇÃO", ambos disponibilizados no Tidia da disciplina Mineração de Dados. A partir dessa base, então, foram realizados a segmentação e identificação das sequências de caracteres e o tratamento dos sinais. Quanto às bibliotecas utilizadas nessa etapa, temos as seguintes:

- **librosa**, tanto para o carregamento como para a extração de *features* dos áudios (*MFCC*, *Mel Frequency Cepstral Coefficients* e *Chroma Energy Normalized Statistics*);
- **scipy**, para a transformação dos sinais de áudio (*DCT*, *FFT* e Filtro de passa baixas de Butterworth);
- **random**, para embaralhar os arquivos da base de forma aleatória;
- **pandas** e **numpy**, para cálculos e manipulação dos dados.

Após realizarmos as devidas transformações na base de dados, aplicamos algoritmos de classificação e avaliamos suas predições (mais detalhes nas seções "Metodologia"e "Resultados"). Nesta etapa, utilizamos as seguintes bibliotecas:

- **sklearn**, tanto para treinar os modelos e realizar as predições como para extrair métricas;
- **matplotlib**, para gerar gráficos das matrizes de confusão dos modelos.

2. Análise Exploratória

Para realização das análises dos áudios, primeiramente, foi feita a segmentação dos áudios seguindo pelo fato de que todos os arquivos teriam aproximadamente oito segundos e que cada letra falada estaria em uma janela de dois segundos (exemplo de áudio capturado na Figura 1). Por isso, o áudio foi segmentado em quatro partes para extração de dados que deveriam ser lidos e classificados.

Com os áudios isolados, foi possível começar com as análises necessárias. À princípio, foi pensado em apenas utilizar os áudios em sua forma original, porém, com

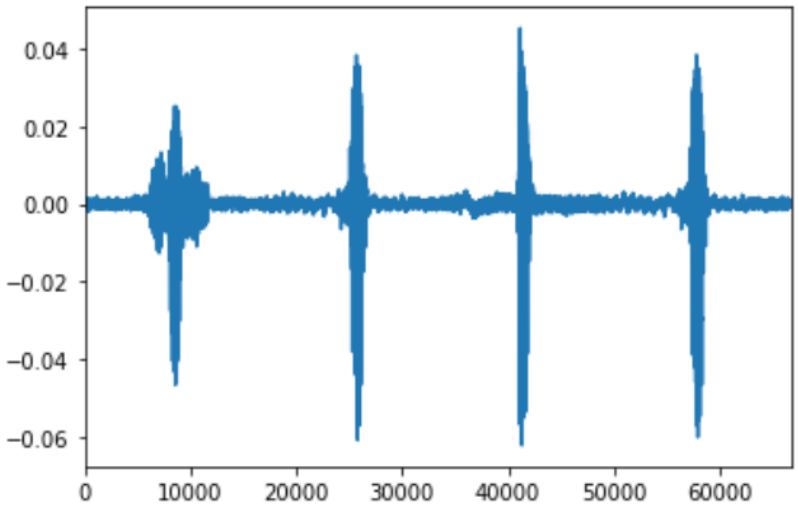

Figura 1. Exemplo de áudio capturado.

um pouco de observação sobre os dados, notou-se que os áudios nem sempre estavam centralizados nas janelas de dois segundos. Por isso, outro tipo de característica devia ser extraída, além apenas das amplitudes no tempo.

Com isso, foi necessário realizar algumas transformações para que os dados no tempo fossem analisados no domínio da frequência. Por isso, primeiramente foram realizadas as Transformadas Rápidas de Fourier (FFT) (Figura 2) e a Transformada Discreta de Cossenos (DCT) (Figura 3).

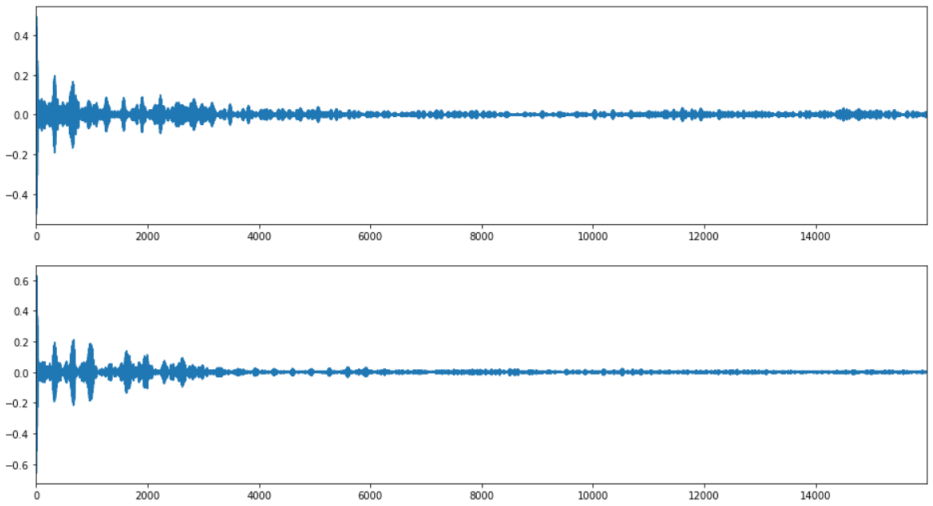

Figura 2. Espectro DCT para letras B.

À partir dos espectros gerados, foi possível verificar que os áudios de mesma letra tinham espectros semelhantes, tanto para FFT quanto para DCT, o que poderiam servir de auxílio para extração de características dos dados. Apesar disso, essas transformações são sensíveis aos ruídos captados na geração dos dados. Sendo assim, foi considerado utilizar um filtro passa baixas de Butterworth para diminuição de ruídos de alta frequência, pois

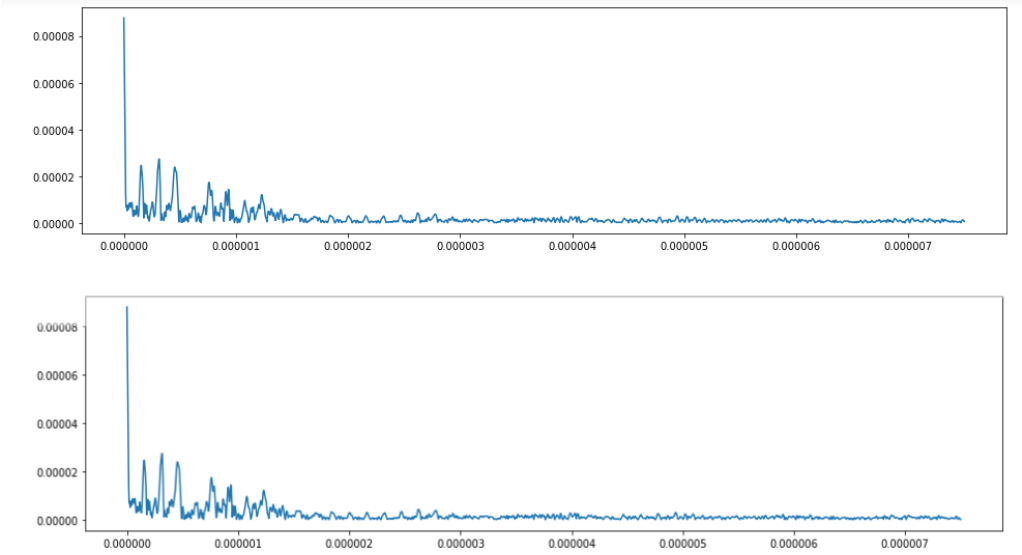

Figura 3. Espectro FFT para letras B.

os sons que os humanos naturalmente escutam estão entre 20Hz e 20kHz.

Ainda assim, a FFT e a DCT por si só não levam em conta a característica humana para resolução de sons. Desta maneira, foram escolhidas outras duas transformações de áudio para estas análises. A *Mel Frequency Cepstral Coefficients*, que demonstra em uma escala o relacionamento da frequência percebida de um tom com a frequência real medida, é calculada através da combinação da FFT, DCT e diferentes filtros.

No Python, a implementação do MFCC na biblioteca librosa resulta em um array de coeficientes das frequências para cada janela gerada pelo algoritmo. Por isso, normalmente a média entre as janelas é calculada para cada coeficiente. Na Figura 4 podemos observar um mesmo coeficiente para diferentes letras. A linha azul representa a curva para letra 'x', e as demais para diferentes áudios de letras 'b'.

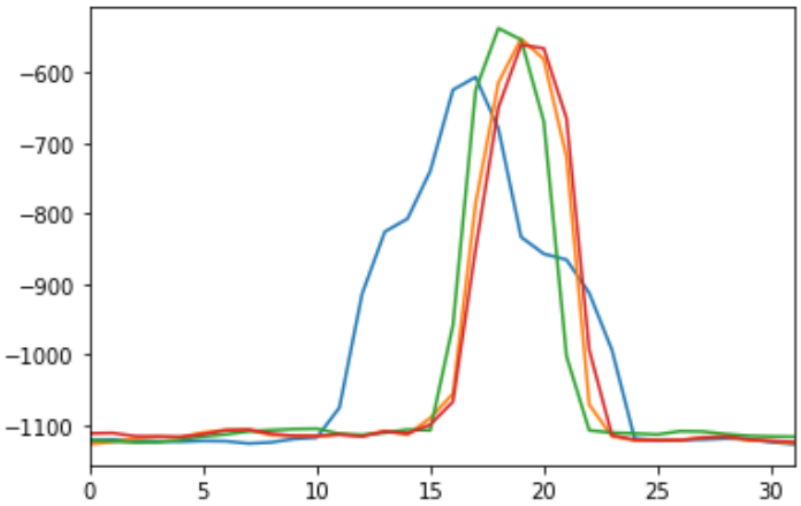

Figura 4. Coeficiente MFCC para as letras X e B.

Outra característica utilizada foi a Chroma Energy Normalized, a qual se baseia nos tons que os humanos percebem e na energia do espectro da frequência, o que facilita a diferenciação de timbres e articulações de fala.

3. Metodologia

Foram considerados principalmente três algoritmos de classificação, a fim de obtermos dados para comparação e definição do melhor algoritmo para o problema proposto. Abaixo encontram-se breve explicações sobre cada um deles, além de detalhes sobre as escolhas de implementação feitas pelo grupo:

- ***k-Nearest-Neighbours* (kNN) –** Algoritmo que utiliza a estratégia de modelar os dados de treinamento com base nos dados de teste. Esta abordagem considera os k elementos de treinamento (vizinhos) mais próximos de um elemento de teste para classificá-lo.
  Um ponto importante a ser definido para um modelo kNN é o valor de k. Se escolhido um k muito pequeno, o kNN estará sujeito a *overfitting*, já um k muito grande pode levar a muitas classificações incorretas. Obtivemos empiricamente os melhores resultados para o modelo com $k = 3$ vizinhos.
- ***Support Vector Machine* (SVM), ou *Support Vector Classification* (SVC) -** Algoritmo originário da teoria do aprendizado estatístico, muito utilizado para problemas de classificação. Realiza a separação dos dados com base em hiperplanos de margem máxima, i.e., hiperplanos com a maior distância entre elementos de diferentes classes. Essa abordagem busca manter a qualidade da predição para exemplos não vistos anteriormente. Para essa análise, instanciamos um modelo de SVC com *kernel* linear.
- ***Random Forest* –** Algoritmo pertencente a uma técnica de classificação conhecida como *ensemble* (em português, "comitê"), a qual agrega as predições de múltiplos classificadores para chegar em um "consenso" e melhorar a acurácia da classificação final. Métodos do tipo *ensemble* tendem a ter um desempenho melhor do que qualquer classificador singular.
  Duas importantes decisões a serem tomadas ao trabalhar com *Random Forest* estão relacionadas ao número de classificadores/árvores e à profundidade máxima dos mesmos. Conforme recomendado em aula – e após confirmarmos que este valor gerou melhores resultados que outros -, utilizamos 500 classificadores (parâmetro *n_estimators*) como padrão de instanciação. Já a profundidade máxima (parâmetro *max_depth*) foi definida como 50.

Quanto à metodologia de avaliação, treinamos os modelos e consideramos a acurácia (*score*) das predições feitas para a base de validação como métrica para avaliar seus respectivos desempenhos. A acurácia, também conhecida como taxa de acerto, calcula a porcentagem de amostras que foram classificadas corretamente.

Além disso, geramos as matrizes de confusão dos modelos de duas formas. Ao executar o programa ("*project.py*"), as matrizes de confusão (em representação numérica) de cada modelo são impressas na tela e, logo em seguida, são abertas, em janelas separadas, imagens de seus respectivos gráficos de cores.

Por último, é importante ressaltar que os testes foram feitos utilizando o Sistema Operacional Ubuntu (Linux) e a versão 3.6 da linguagem de programação Python.

4. Resultados

Dentre os três algoritmos de classificação mencionados anteriormente, e considerando a metodologia de avaliação adotada, o modelo que obteve melhores resultados foi o *Random Forest*.

Apesar de os outros dois apresentarem acurácias não tão distintas um do outro, em todos os testes que realizamos o modelo SVC foi melhor do que o modelo kNN. Portanto, descartaremos este último.

Em relação aos valores de acurácia obtidos, nos testes realizados a acurácia do modelo SVC variou de aproximadamente 58% a 61%. Já o modelo *Random Forest* apresentou acurácia de aproximadamente 72% a 74%. A Figura 5 apresenta a saída do programa para o modelo *Random Forest* com a máxima acurácia obtida nos testes realizados pelo grupo, de 74,16%.

```
nath-tescarollo@nathtescarollo-LenovoZ400:~/Documentos/Mineração de Dados/audio_captcha_recog$ python project.py

--- Bem vindo ao Projeto de Reconhecimento de Audio ---

Realizando Imports de Libs necessarias...

Buscando dados nas Pastas TREINAMENTO/ e VALIDACAO/ internas do projeto...

Preparando DataSet para Treino
310
Preparando DataSet para Teste/Validacao
8


Instanciando Modelo Random Forest

Treinando Modelo...
Realizando Classificacao...

Acuracia do modelo Random Forest = 0.7416

Matriz de Confusao Geral do Modelo Random Forest

['c', 'b', 'h', 'x', '6', '7', 'd', 'm', 'a', 'n']

[[ 74   3   0   2  26   2   0   0   0   0]
 [  1  64   0   0   1   0  34   5   0   1]
 [  0   0 116   0   0   2   1   0   3   0]
 [  1   0   0  89   4   0   0   1   0   0]
 [ 10   1   0   6  83   0   0   0   0   0]
 [  1   0   1   3   0 109   0   0   0   0]
 [  2  44   0   0   0   1  55   3   0   5]
 [  0   9   0   1   1   0   7  53   0  33]
 [  0   0   4   0   0   1   0   0  92   0]
 [  0   4   0   0   1   1   4  46   0  57]]
```

Figura 5. Exemplo de saída do programa *python.py* para o modelo *Random Forest*.

Comparando o gráfico das matrizes de confusão dos modelos SVC e *Random Forest* (Figura 6 e Figura 7, respectivamente) conseguimos ver facilmente que o *Random Forest* foi o que teve melhor desempenho, visto que a concentração de predições na diagonal principal da matriz é maior para ele do que para o SVC. Nesta representação, a diagonal principal da matriz diz respeito aos elementos que foram classificados corretamente.

Além disso, conseguimos perceber uma grande dificuldade dos algoritmos em diferenciar os seguintes pares de caracteres: m e n; b e d; c e 6.

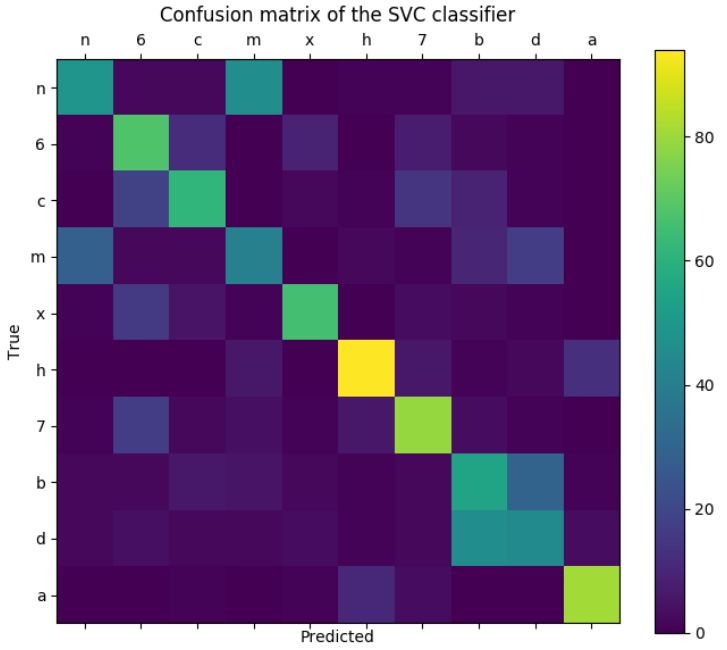

Figura 6. Exemplo de Matriz de Confusão do modelo SVC.

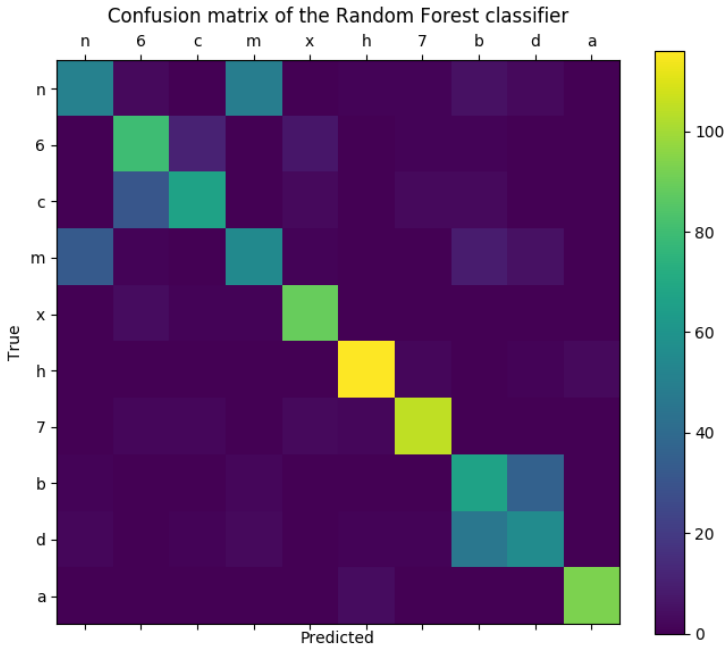

Figura 7. Exemplo de Matriz de Confusão do modelo *Random Forest*.

5. Comentários Finais

Resumidamente, obtivemos bons resultados iniciais utilizando o classificador *Random Forest*, o qual sobressaiu-se em relação aos demais algoritmos considerados nesta fase do

projeto.

Além disso, as *features* implementadas durante o pré-processamento da base de dados, para transformação dos sinais de áudio, teve um papel importante no aumento da acurácia dos classificadores.

Dentre as maiores dificuldades encontradas, podemos citar a definição do processo ideal de tratamento/transformação de arquivos de áudio e a determinação das métricas a serem utilizadas para fins de avaliação dos modelos. Estes dois tópicos são, portanto, potenciais campos a serem aprimorados durante a segunda fase do projeto.