Universidade Federal do ABC
Programa de Pós-graduação em Ciência da Computação
Disciplina de Mineração de Dados

Relatório 2 - Mineração de Dados

João Carlos Pandolfi Santana, Patrícia Dias dos Santos

3 de setembro de 2018

1 Introdução

O objetivo do desafio proposto é construir um conjunto específico de classificadores para a detecção e classificação de áudios, capazes de distinguir gravações de diferentes caracteres. Primeiramente foi realizada a segmentação da sequência, separando o áudio referente a cada caractere. Após isso, cada classificador deveria identificar o caractere, sendo que a predição é a combinação das predições individuais para cada caractere.

1.1 Dataset

O conjunto de dados para o desafio de classificação de áudios consiste em gravações de 8 segundos de vários caracteres. Para simplificar o problema, foram considerados apenas os seguintes caracteres: a,b,c,d,h,m,n,x,6,7. O conjunto de dados de classificação de áudio consiste em 2 partes igualmente proporcionais, cada uma composta por aproximadamente 148 gravações compostas por um conjunto de 4 caracteres cada, sendo uma de treinamento e outra de validação. Neste dataset, os arquivos de som estão no formato ".wav".

2 Análise Exploratória

Um dos primeiros problemas encontrados foi em relação ao ruído de fundo dos áudios. Foi necessário normalizar o áudio e aplicar alguns filtros para melhorar a qualidade do áudio. Os ruídos também dificultaram na segmentação dos áudios.

O experimento consistiu de três etapas, as quais serão explicadas mais detalhadamente na metodologia:

1. Preparação dos dados
2. Treinamento dos modelos
3. Testes dos modelos

3 Metodologia

Nas subseções a seguir serão descritas as ferramentas utilizadas no nosso experimento.

3.1 Preparação dos Dados

O primeiro passo consistiu no tratamento dos dados. Isso precisou ser feito devido às características do áudio, que foi extraído de diferentes fontes, com vozes e entonações muito diferentes entre si.

O tratamento foi feito de acordo com as seguintes etapas:

- Primeira etapa: Utilização de um filtro por frequência (basicamente um passa-baixa da biblioteca Librosa, usamos 18db, conforme pode ser visto nas Figuras 1 e 2). Após isso, a onda foi normalizada (a onda original pode ser vista na Figura 3) e a onda obtida após a normalização pode ser vista na Figura 4. Como após todos estes processos, ainda havia ruído na amostra (Figura 5) foram adotados os seguintes procedimentos: filtragem do ruído remanescente por amostragem em que foi selecionado um pedaço da onda (200 primeiros valores); a onda foi espelhada no eixo X, levando todos os valores para o eixo positivo; calculou-se a média; foram selecionados os valores acima da média e mas uma porcentagem da média (média + média * 0,7); foi calculada uma nova média desses valores e se passou do threshold 0.15, estabilizou-se para 0.1. Por fim, subtraiu-se de toda a onda o valor obtido e conseguimos cancelar o ruído (Figura 6).

- Segunda etapa: Segmentação em 4 partes iguais (acreditando que cada letra estará dentro dos 2 segundos).

- Terceira etapa: Definição dos pedaços e separação dos que corresponderiam às mesmas letras.

Para retirada das *features* foi utilizada uma função de extração de MFCCs (Mel-frequency cepstral coefficients). Também foram utilizadas as seguintes features:

- stZCR: Quantidade de zeros por tempo
- stEnergy : Computa a energia do sinal
- stEnerfyentropy : Entropia da energia
- Foi executada uma transformada de Fourier na transposta da amostragem
- Foram utilizadas as seguintes features recomendadas na literatura de processamento de audio: Melspectogram[1], Chroma e Spectral Contrast [1], [2].

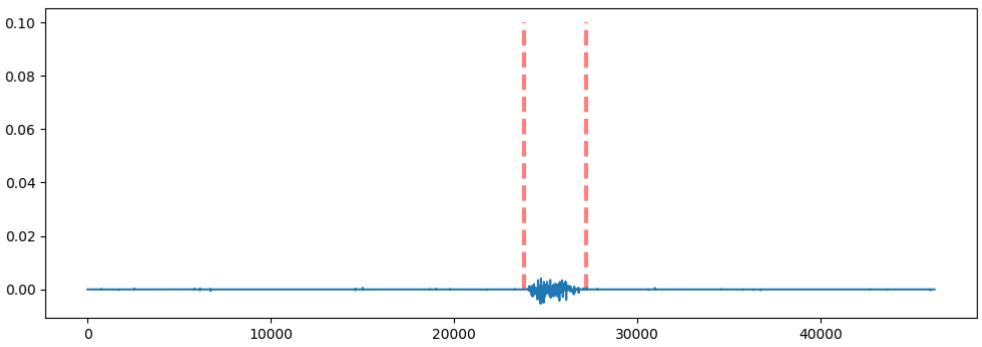

Figura 1: Exemplo de Aplicação do Filtro de Passa Baixo

Fonte: Produzido pelos autores.

3.2 Treinamento dos Dados

Para treinamento dos modelos, foi utilizada a biblioteca de aprendizado de máquina em Python *scikit-learn* [3], essa biblioteca foi escolhida por possuir uma interface consistente e orientada a tarefas, o que permitiu a comparação dos métodos de classificação do áudio. Os modelos escolhidos foram:

1. GridSearch com Random Forest
2. KNN - distance
3. KNN - uniform

[1]Ver: librosa.feature.melspectrogram

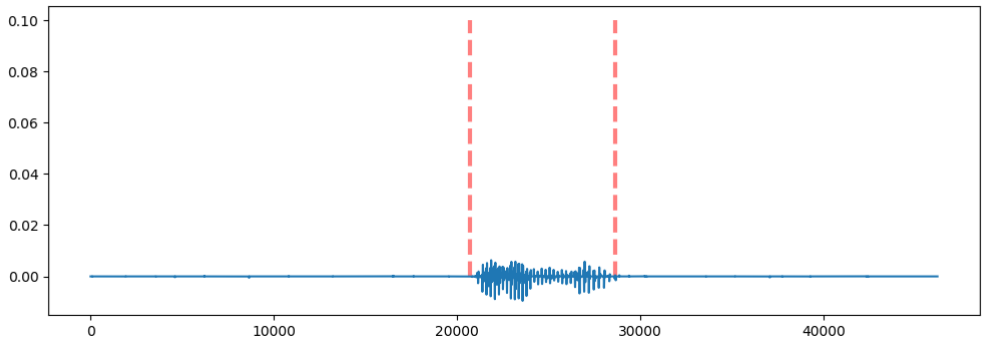

Figura 2: Exemplo de segmentação do áudio por caractere

Fonte: Produzido pelos autores.

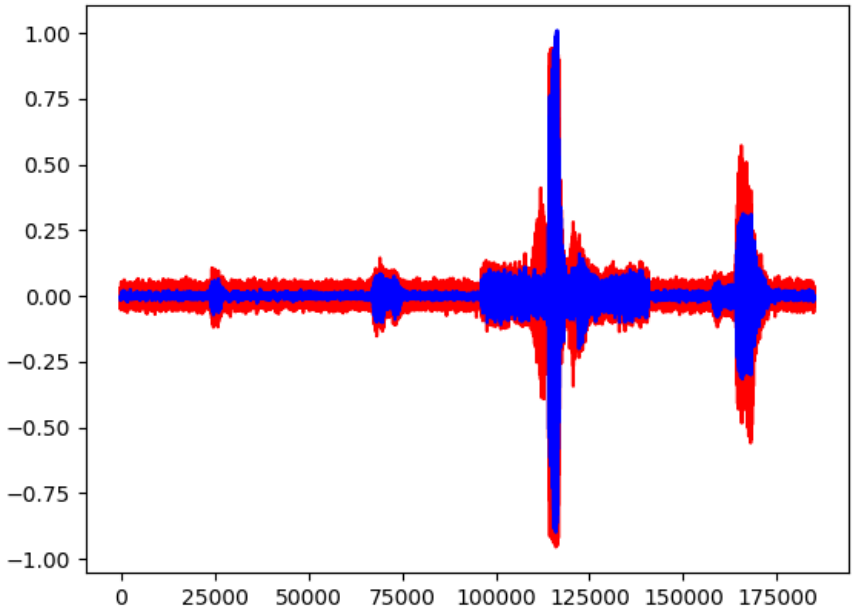

Figura 3: Onda Original em vermelho e onda filtrada em azul (Filtro passa baixa).

Fonte: Produzido pelos autores.

4. Naive Bayes
5. SVM
6. Gaussian Process
7. ExtraTreesClassifier

3.3 Teste dos Modelos

Na etapa de teste dos modelos foram executados os arquivos da pasta de validação dando um shuffle nos dados lidos. Assim obtivemos os valores da matriz de confusão e a estatística de acerto.

4 Resultados

Foram geradas as seguintes matrizes de confusão para cada modelo, conforme pode ser visto a seguir:

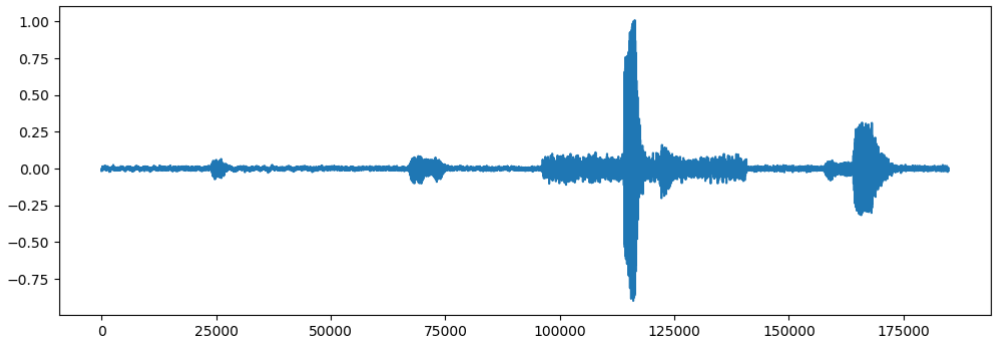

Figura 4: Onda antes da Normalização

Fonte: Produzido pelos autores.

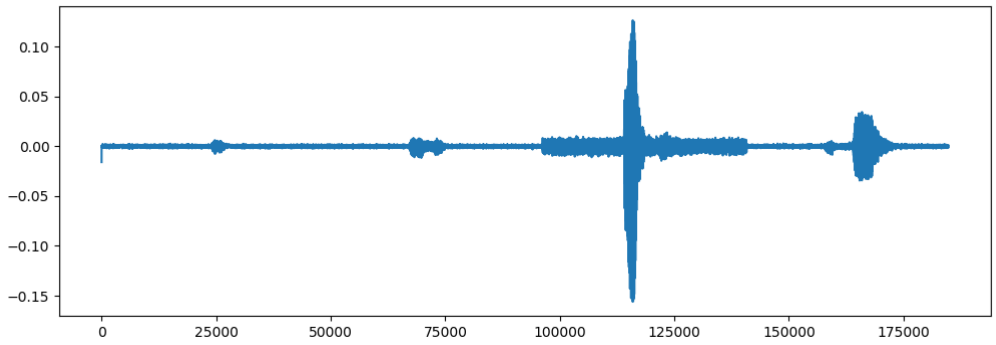

Figura 5: Onda Normalizada

Fonte: Produzido pelos autores.

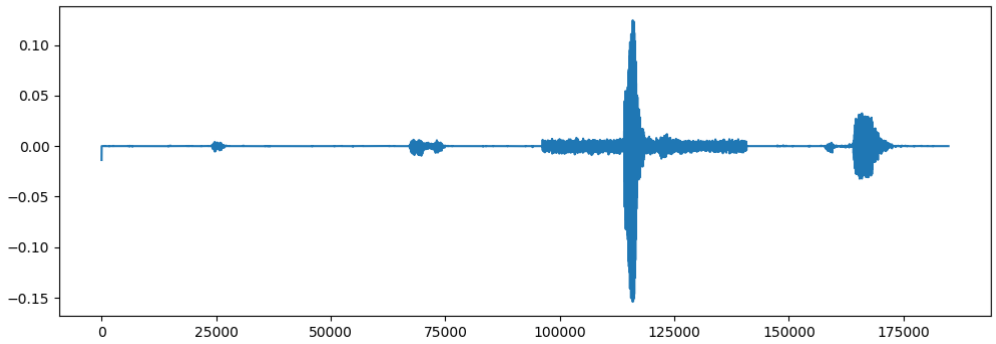

Figura 6: Onda Obtida Após Cancelamento do Ruído

Fonte: Produzido pelos autores.

```
Predizendo: RANDOM FOREST
Caracteres corretos:  298 de 596
Captchas corretos:  10 de 149
Matriz de confusao
['a', 'b', 'c', 'd', 'h', 'm', 'n', 'x', '6', '7']
[[58  0  0  0  6  0  0  0  0  0]
 [ 0 18  5  4  0  3  5  5 10  3]
 [ 0  8 24  6  0  1  6  5 10  6]
```

```
 [ 0 14  6  5  0  4  8  3 15  0]
 [ 4  0  0  0 48  0  1  0  0  3]
 m[ 0  3  6  1  0 16 13  2 11  5]
 [ 0  6  3  0  0  6 18  1  8  3]
 [ 0  7  5  1  0  5  9 34  3  3]
 [ 0 10 11  6  0  3  6 10 23  1]
 [ 1  0  0  0  3  0  1  0  4 54]]

Predizendo: EXTRA TREE
Caracteres corretos:  284 de 596
Captchas corretos:  10 de 149
Matriz de confusao
['a', 'b', 'c', 'd', 'h', 'm', 'n', 'x', '6', '7']
[[55  0  0  0  7  1  0  0  0  1]
 [ 0 20  6  2  0  1  2  3 14  5]
 [ 1  7 14  5  1  3  9  5 16  5]
 [ 0 15  4  7  0  7 11  1 10  0]
 [ 5  0  0  0 47  0  0  0  0  4]
 m[ 1  6  4  3  0 13 10  4  7  9]
 [ 0  6  4  0  0  8 20  2  2  3]
 [ 0  8  8  1  0  7  6 30  3  4]
 [ 0  8 10  6  0  2  6  6 30  2]
 [ 0  0  0  0  6  5  2  0  2 48]]

Predizendo: KNN - distance
Caracteres corretos:  236 de 596
Captchas corretos:  5 de 149
Matriz de confusao
['a', 'b', 'c', 'd', 'h', 'm', 'n', 'x', '6', '7']
[[40  0  0  0 24  0  0  0  0  0]
 [ 0  5  6  0  0 10  6  7 14  5]
 [ 0  3 13  7  0  9 17  1 14  2]
 [ 0  4  2  4  0 15 19  6  5  0]
 [ 3  0  0  0 52  0  1  0  0  0]
 m[ 0  0  3  1  0 22 17  3  8  3]
 [ 0  1  1  0  0 14 19  4  3  3]
 [ 0  2  2  3  0 10  8 29  8  5]
 [ 0  5 10  2  0 10 15  4 23  1]
 [ 0  0  1  2 10  7  7  1  6 29]]

Predizendo: KNN - uniform
Caracteres corretos:  203 de 596
Captchas corretos:  3 de 149
Matriz de confusao
['a', 'b', 'c', 'd', 'h', 'm', 'n', 'x', '6', '7']
[[39  1  0  0 24  0  0  0  0  0]
 [ 0  4  6  0  0  9  2  7 17  8]
 [ 0  4 13  6  0 11 15  1 13  3]
 [ 0  4  2  2  0 16 16  8  7  0]
 [11  0  0  0 41  0  1  0  0  3]
 [ 0  2  3  2  0 22 14  3  8  3]
 [ 0  3  4  0  0 12 12  4  8  2]
 [ 0  3  8  2  0  9  9 18 11  7]
 [ 0  5 12  2  0  9 10  5 26  1]
 [ 0  0  1  3  9  7  6  1 10 26]]

Predizendo: Naive Bayes
Caracteres corretos:  211 de 596
Captchas corretos:  4 de 149
Matriz de confusao
['a', 'b', 'c', 'd', 'h', 'm', 'n', 'x', '6', '7']
```

```
[[44  0  0  1 14  1  2  2  0  0]
 [ 0  5  0  7  0  3  1 28  4  5]
 [ 0  2  6  2  0  1  1 41  9  4]
 [ 0  3  1  9  0  1  1 33  7  0]
 m->h[ 2  0  0  0 40 10  0  3  0  1]
 n->o[ 0  1  1  2  0 10  2 32  6  3]
 [ 0  0  0  1  0  6  7 28  2  1]
 [ 0  0  0  0  0  5  0 59  1  2]
 [ 0  1  2  4  0 10  2 43  8  0]
 [ 0  0  5  0  4 16  2 13  0 23]]

Predizendo: SVM
Caracteres corretos:  331 de 596
Captchas corretos:  16 de 149
Matriz de confusao
['a', 'b', 'c', 'd', 'h', 'm', 'n', 'x', '6', '7']
[[60  0  0  0  4  0  0  0  0  0]
 [ 1 26  3  8  0  3  1  2  5  4]
 [ 0  9 21 13  0  2  5  3 11  2]
 [ 0 15  9 11  0  4  6  1  8  1]
 [ 2  0  0  0 53  0  0  0  0  1]
 m[ 1  4  7  1  0 20 17  1  3  3]
 [ 1  3  3  2  3  6 20  0  4  3]
 [ 4  5  6  1  0  4  5 33  5  4]
 [ 0  4  6  7  0  6  8  5 34  0]
 [ 1  1  0  0  4  1  1  1  1 53]]

Predizendo: Gaussian Process
Caracteres corretos:  294 de 596
Captchas corretos:  11 de 149
Matriz de confusao
['a', 'b', 'c', 'd', 'h', 'm', 'n', 'x', '6', '7']
[[54  0  0  0 10  0  0  0  0  0]
 [ 6 15  3  2  0  2  1 15  8  1]
 [ 1 14 16  2  1  5  6  8  8  5]
 [ 0  8  6  1  1  7 10 11 10  1]
 [ 1  0  0  0 54  0  1  0  0  0]
 m[ 1  2  4  0  0 21 18  4  3  4]
 n[ 1  2  3  1  3 12 15  1  3  4]
 [ 4  2  3  0  1  5  6 42  1  3]
 [ 0  6  9  3  0 14  5 11 22  0]
 [ 0  0  0  0  5  3  1  0  0 54]]
```

Tivemos uma surpresa, para estes dados, o modelo que mais se adaptou ao problema foi o SVM.

O acerto de cada modelo é mostrado na figura 7.

4.1 Regra

4.1.1 Stacking

Tentamos utilizar stacking de modelos. PS: olhar aquivo raciocinio.py.

Como o SVM apresentou melhor resultado, aplicamos o peso relativo maior para ele:

`relative_wheigths = [2,2,1,4,3,2]` `(mesma sequencia da matriz de confusao)`

Utilizando esses pesos relativos, treinamos outro modelo com a saída dos classificadores escolhidos (Utilizando a base de treino). Não obtivemos resultados melhores que o SVM, pelo contrário, obtivemos resultados piores: Acertou 8 de 149 enquanto SVM acertou 16 de 149.

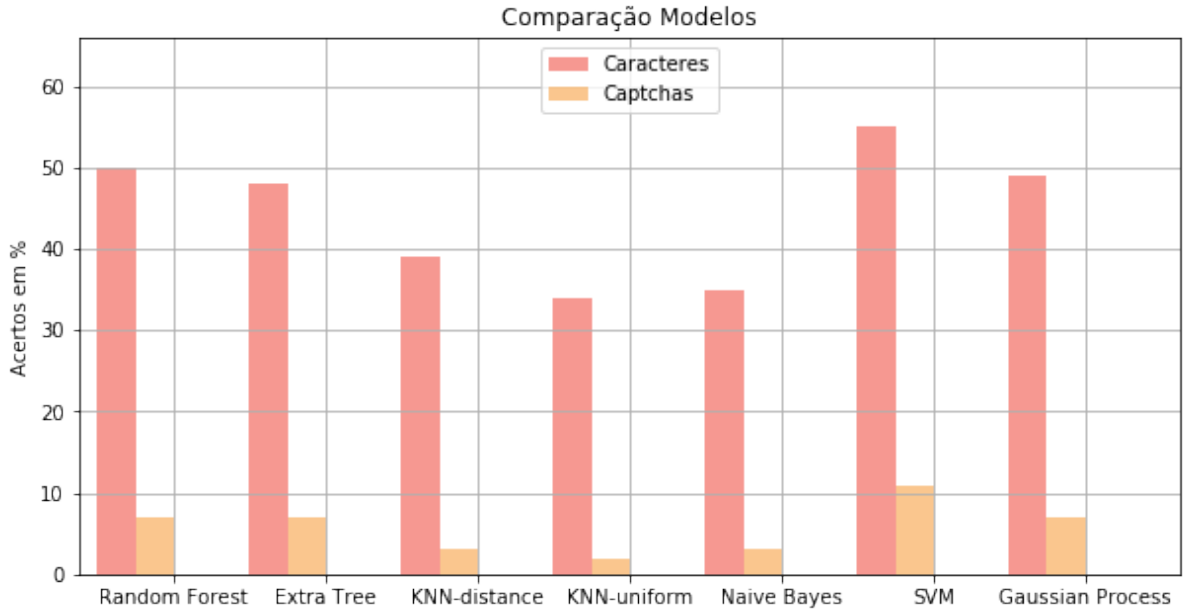

Figura 7: Onda Obtida Após Cancelamento do Ruído

Fonte: Produzido pelos autores.

Utilizamos 4 modelos no stacking diferentes e todos eles não apresentaram resultados melhores que o SVM sozinho Métodos testados:

```
mStack = GaussianNB()
mStack = RandomForestClassifier(n_jobs=4, random_state=1)
mStack = MLPClassifier(random_state=1,hidden_layer_sizes=(32,16),activation='relu')
mStack = GridSearchCV(SVC(kernel='rbf', class_weight='balanced'), param_grid)
```

4.1.2 Modelo misto

Tentamos utilizar uma heurística mista, levando em conta o SVM como principal classificador e para os caracteres 'm', 'n' e 'b' que ele apresenta bastante colisão, utilizarmos outros classificadores. Também não obtivemos bons resultados: 10 de 149

4.1.3 Composição do modelo misto

Utilizamos os outros classificadores que foram selecionados: *KNN , Random Forest, Extra Tree, Gaussian Process, Naive Bayes*. Estes competiam entre si levando em conta o peso relativo determinado acima.

Forçamos também para o caractere 'm', aumentar o peso relativo do *NaiveBayes* por ter uma colisão muito alta com 'h', desta forma, ao classificar o caractere como 'h', tínhamos uma certeza maior que o caractere correspondente era 'm'.

5 Comentários Finais

Para não utilizarmos um modelo puro e que poderia estar preso em mínimo local, optamos por testar outro método, desta vez um pronto do sklearn. utilizamos um Ensemble por voto de classificador: primeiro utilizamos a votação por pesos, o que não conseguiu ganhar ainda do SVM com 16 acertos. Então, fizemos modificações "empíricas"nos pesos sem sucesso. Então decidimos utilizar a votação "hard", onde ele escolhe o classificador que teve mais certeza para aquela predição. Chegamos ao mesmo resultado em captchas acertados que o SVM (16), mas perdemos 2 caracteres acertados - de 331 para 329 Mas optamos por deixar assim, pois acreditamos que mais de um classificador possa resolver algum ótimo local possível.

Referências

[1] André Jucovsky Bianchi and Marcelo Queiroz. Real time digital audio processing using arduino. In *Proceedings of the Sound and Music Computing Conference, Stockholm, Sweden*, volume 30, pages 538–545, 2012.

[2] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.

[3] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.