Introdução e Avaliação de Algoritmos de Classificação de Dados

Hugo Fernandes[1], Flavius Gorgonio[1]

[1]Depto de Computação e Tecnologia - Universidade Federal do Rio Grande do Norte (UFRN)
Caixa Postal: 59.300-000 - Caicó - RN - Brazil

hugo.medeiros.fernandes@gmail.com, flavius@ufrnet.br

Abstract. *This paper seeks to show the essential concepts about the data classification techniques and perform a comparison of some of the main algorithms of data mining area. To assist in this task, a diagnostic database of Parkinson's disease will be used.*

Resumo. *Este trabalho busca mostrar os conceitos essenciais a respeito das técnicas de classificação de dados e realizar um comparativo de alguns dos principais algoritmos da área de mineração de dados. Para auxiliar nesta tarefa, será utilizada uma base de dados de diagnósticos do mal de Parkinson.*

1. Base de Dados

Essa base foi criada por Max Little da Universidade de Oxford, em colaboração com o Centro Nacional da Voz e Fala, de Denver/Colorado, onde foram gravados os sinais de voz. O estudo foi originalmente publicado como um recurso no diagnostico de distúrbios da voz.

A base de dados é composta por medições da voz de 31 pessoas, sendo 23 diagnosticadas com mal de Parkinson (PD). Cada coluna da base é uma medição particular da voz do paciente, e cada linha representa uma nova gravação. Para cada paciente foram coletadas 6 medições de discursos diferentes, onde os pesquisadores orientavam aos pacientes que reproduzissem frases selecionadas para a gravação e posteriormente medições dos áudios.

Em cada gravação é atribuído um identificador do paciente, junto com o número do discurso selecionado. Esses valores estão presentes na coluna '*name*'. Este é o único dado do tipo texto na base de dados, sendo todos os outros valores numéricos. Da mesma maneira da coluna '*name*', também é inserido o estado do paciente, se possui ou não o mal de Parkinson. Está informação pode ser encontrada na coluna '*status*' e assume dois valores, 0 ou 1, onde o primeiro mostra que o paciente não possui o mal de Parkinson e o segundo que possui. Essa última informação será utilizada como a classe para os algoritmos de classificação a seguir. Os outros atributos da base podem ser vistos na tabela 1 no item Apêndices.

2. Preprocessamento

As principais alterações feitas na base de dados foram em relação a coluna '*name*', pois como se tratava de um atributo texto, teve que ser convertido para numérico, tendo em vista que, os algoritmos não poderiam trabalhar com valores do tipo texto. Outra mudança aconteceu nessa mesma coluna, o identificador do paciente e o código da gravação (número do discurso selecionado), estavam unidos por um caractere ('_'). Para solucionar esse problema, foram criadas duas novas colunas, '*patient*' e '*record*' onde foi inserido os atributos identificador do paciente e código de gravação respectivamente. Finalmente a coluna '*name*' foi excluída.

3. Algoritmos de Classificação

- RandomForestClassifier
- SVC
- GaussianNB
- KNeighborsClassifier
- DecisionTreeClassifier

4. Aglomeração dos Elementos da Base

Uma forma interessante de visualizar se a base de dados pode ter um bom índice de classificação, é gerar gráficos que possam mostrar a distribuição dos elementos.

Na figura 1, podemos identificar os elementos que possuem o mal de Parkinson na cor verde e os que são saudáveis em azul. É possível visualizar que os pacientes saudáveis estão mais aglomerados em uma parte do gráfico.

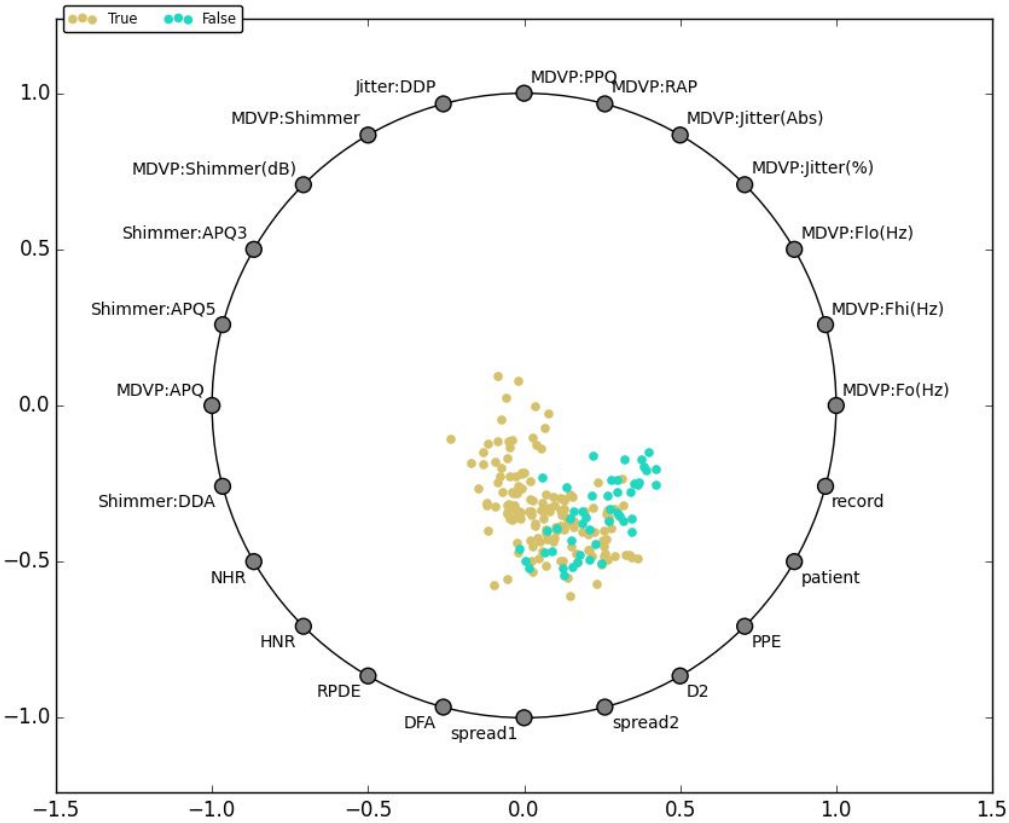

Figura 1. Representação Gráfica da Distribuição dos Elementos (Método Radviz)

5. Acurácia dos Algoritmos de Classificação

Uma maneira de verificar o grau de confiabilidade de um algoritmo de classificação é medindo sua acurácia média. O processo consiste basicamente em verificar o número de acertos na classificação dos elementos. Para tornar o processos percentual, basta utilizar a relação 1:

$$\frac{Elementos\ classificados\ corretamente}{Elementos\ classificados} \qquad (1)$$

Neste trabalho, para se calcular a acurácia média dos algoritmos, foi determinado que cada algoritmo deveria ser executado 1000 vezes para assim, se calcular a acurácia média, que pode ser vista na relação 2.

$$\frac{\sum_{i}^{Execuções} \frac{Elementos\ classificados\ corretamente_i}{Elementos\ classificados_i}}{Execuções} \quad (2)$$

Outra característica que vale ser mencionada, é que cada um dos algoritmos utilizou 70% da base de dados para realização de treinamento e 30% para testes de predição.

Dessa forma, na tabela 2 segue os valores da acurácia média da execução dos algoritmos de classificação para está base de dados.

Algoritmo de Classificação	**Acurácia Média (%)**
RandomForestClassifier	91.1559322034 %
SVC	88.4372881356 %
GaussianNB	70.1881355932 %
KNeighborsClassifier	89.3254237288 %
DecisionTreeClassifier	93.3203389831 %

Tabela 2. Acurácia Média dos Algoritmos de Classificação

6. Referências

- A principal referência deste trabalho foi a biblioteca Sklearn do Python.
- Little MA, McSharry PE, Roberts SJ, Costello DAE, Moroz IM. **Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection**. BioMedical Engineering OnLine 2007, 6:23 (26 June 2007).
- Little MA. **Parkinsons Data Set**. <https://archive.ics.uci.edu/ml/datasets/Parkinsons>. Acessado em: 05/12/2015.

7. Apêndices

Atributos	**Descrição**
MDVP:Fo(Hz)	Average vocal fundamental frequency
MDVP:Fhi(Hz)	Maximum vocal fundamental frequency
MDVP:Flo(Hz)	Minimum vocal fundamental frequency
MDVP:Jitter(%)	Several measures of variation in fundamental frequency
MDVP:Jitter(Abs)	
MDVP:RAP	
MDVP:PPQ	
Jitter:DDP	
MDVP:Shimmer	Several measures of variation in amplitude
MDVP:Shimmer(dB)	
Shimmer:APQ3	
Shimmer:APQ5	
MDVP:APQ	
Shimmer:DDA	
NHR	Two measures of ratio of noise to tonal components in the voice
HNR	
RPDE	Two nonlinear dynamical complexity measures
D2	
DFA	Signal fractal scaling exponent
spread1	Three nonlinear measures of fundamental frequency variation
spread2	
PPE	

Tabela 1. Outros Atributos da Base de Dados

```
# -*- coding: utf-8 -*-
import warnings
warnings.filterwarnings('ignore')

from pandas.tools.plotting import parallel_coordinates
from pandas.tools.plotting import radviz
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
import datetime
import time
import glob

def radvizPlot(base, classe):
    plt.figure(figsize=(10,8))
    ax = radviz(base, classe)
    ax.legend(loc='center left', bbox_to_anchor=(0, 1), fancybox=True, ncol=2,
fontsize='x-small')
    plt.ylim([-2,2])
    plt.show()

def accuracy(base, classe, n, classificador):
    media = 0
    for i in range(n):
        x = base.drop([classe], axis=1)
        y = base[classe]
        x = StandardScaler().fit_transform(x)
        X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=.3)
        classificador.fit(X_train, y_train)
        z = classificador.predict(X_test)
        media += accuracy_score(y_test, z) * 100
    return media/n
```

```
def determineClassific(train, test, classe, classificador):
	trainBase = train.drop([classe], axis=1)
	trainClasse = train[classe]
	testBase = test.drop([classe], axis=1)
	testClasse = test[classe]
	trainBase = StandardScaler().fit_transform(trainBase)
	testBase = StandardScaler().fit_transform(testBase)
	classificador.fit(trainBase, trainClasse)
	return classificador.predict(testBase)

def categoricalConversion(data):
	return pd.Categorical.from_array(data).codes

parkinsons = pd.read_csv('parkinsons.csv',sep=',')
parkinsons['status'] = parkinsons['status'] > 0
parkinsons['patient'] = categoricalConversion(parkinsons['name'].str.rsplit('_', expand=True,
n=1)[0])
parkinsons['record'] = categoricalConversion(parkinsons['name'].str.rsplit('_', expand=True,
n=1)[1])
parkinsons = parkinsons.drop('name', axis=1)
exemploParkinsons = pd.read_csv('exemploParkinsons.csv',sep=',')
exemploParkinsons['status'] = exemploParkinsons['status'] > 0
exemploParkinsons['patient'] =
categoricalConversion(exemploParkinsons['name'].str.rsplit('_', expand=True, n=1)[0])
exemploParkinsons['record'] =
categoricalConversion(exemploParkinsons['name'].str.rsplit('_', expand=True, n=1)[1])
exemploParkinsons = exemploParkinsons.drop('name', axis=1)
print (accuracy(parkinsons, 'status', 1000, RandomForestClassifier()))
print (accuracy(parkinsons, 'status', 1000, SVC()))
print (accuracy(parkinsons, 'status', 1000, GaussianNB()))
print (accuracy(parkinsons, 'status', 1000, KNeighborsClassifier()))
print (accuracy(parkinsons, 'status', 1000, DecisionTreeClassifier()))
#print (determineClassific(parkinsons, exemploParkinsons, 'status', DecisionTreeClassifier()))
radvizPlot(parkinsons, 'status')
```

Código Fonte 1. Implementação em Python