Curso: Ciência de Dados e Big Data | Disciplina: Machine Learning

Aplicação de Machine Learning para Auxílio no Diagnóstico do Diabetes

Professor: Hugo de Paula

Alunos:

- Anderson Carvalho
- Angela Mendonça
- Camila Coutinho
- Marina Oliveira
- Thiago Silva

Introdução

O diabetes é uma síndrome metabólica de origem múltipla, decorrente da falta de insulina e/ou da incapacidade de a insulina exercer adequadamente seus efeitos, causando um aumento da glicose (açúcar) no sangue. Ao longo do tempo, a diabetes pode provocar danos ao coração, vasos sanguíneos, olhos, rins e nervos.

Para elaboração do trabalho utilizamos um conjunto de dados sobre diabetes extraida do site: http://storm.cis.fordham.edu (http://storm.cis.fordham.edu). A base de dados é do ano de 1988 e retratava dados reais de uma população que vivia perto de Phoenix, Arizona, EUA. Possui um total de 767 registros e 8 classes com atributos relacionados a saúde e idade dos pacientes.

Problema

Segundo a OMC (Organização mundial de saúde) 1 em cada 11 pessoas no mundo possui diabetes. Esse número representa um total de 420 milhões de pessoas (dados de 2014). Ainda segunda a OMC, a taxa de incidência de diabetes cresceu mais de 61% nos ultimos 10 anos. Só no Brasil, os números apontam que 16 milhões de pessoas sofrem de diabetes.

Objetivo

Com base nas informações do conjunto de dados, o objetivo do trabalho é tentar prever com maior assertividade se uma possoa possui ou não pré disposição para o diabetes. A identificação precoce de diabetes é extremamente importante pois, auxilia o diabético a manter um nível bom de glicose podendo evitar consequências graves como, infarto, derrame cerebral ou cegueira.

Desenvolvimento

1. Carga e tratamento da base de dados

In [4]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.feature_extraction import DictVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import BernoulliNB

dados = pd.read_csv('./diabetes.txt')
```

In [5]:

```
#Renomeia as colunas do data frame
#colunas = ['Numero vezes grávida', 'Glicose', 'Pressão arterial', 'Espessura da dobra
 da pele', 'Insulina sérica','IMC','Função de Pedigree diabetes','Idade','Teste para di
abetes']

#Criação data frame diabetes
diabetes_df = pd.DataFrame(dados)

#Renomeia as colunas do data frame
diabetes_df.columns = ['Numero vezes grávida', 'Glicose', 'Pressão arterial', 'Espessur
a da dobra da pele', 'Insulina sérica','IMC','Função de Pedigree diabetes','Idade','Tes
te para diabetes']

#Lista Top 5 dados do data set
diabetes_df.head()
```

Out[5]:

	Numero vezes grávida	Glicose	Pressão arterial	Espessura da dobra da pele	Insulina sérica	IMC	Função de Pedigree diabetes	Idade	Test dia
0	1	85	66	29	0	26.6	0.351	31	tested_ne
1	8	183	64	0	0	23.3	0.672	32	tested_po
2	1	89	66	23	94	28.1	0.167	21	tested_ne
3	0	137	40	35	168	43.1	2.288	33	tested_po
4	5	116	74	0	0	25.6	0.201	30	tested_ne

In [6]:

```
#verificando as opções da variável resposta Possui diabes
#Identificamos que realmente há apenas duas possíveis opções de resposta ("tested_negat
ive","tested_positive")

print(diabetes_df["Teste para diabetes"].unique())
```

```
['tested_negative' 'tested_positive']
```

In [7]:

```
#Substituição valores tested_negative e tested_positive para valores binários (0 e 1).

df_diabetes_novo = diabetes_df.replace({'Teste para diabetes': {"tested_negative": 0,
"tested_positive": 1}})
```

In [8]:

```
#Lista Top 5 dados para verificar as alterações

df_diabetes_novo.head().head()
```

Out[8]:

	Numero vezes grávida	Glicose	Pressão arterial	Espessura da dobra da pele	Insulina sérica	IMC	Função de Pedigree diabetes	Idade	Teste para diabetes
0	1	85	66	29	0	26.6	0.351	31	0
1	8	183	64	0	0	23.3	0.672	32	1
2	1	89	66	23	94	28.1	0.167	21	0
3	0	137	40	35	168	43.1	2.288	33	1
4	5	116	74	0	0	25.6	0.201	30	0

In [9]:

```
#Verificacao valores de tipos de dados e existencia de valores nulos
#Identificamos que não há valores nulos

df_diabetes_novo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 767 entries, 0 to 766
Data columns (total 9 columns):
Numero vezes grávida           767 non-null int64
Glicose                        767 non-null int64
Pressão arterial               767 non-null int64
Espessura da dobra da pele     767 non-null int64
Insulina sérica                767 non-null int64
IMC                            767 non-null float64
Função de Pedigree diabetes    767 non-null float64
Idade                          767 non-null int64
Teste para diabetes            767 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.0 KB
```

2. Estatística descritiva sobre a base de dados

In [10]:

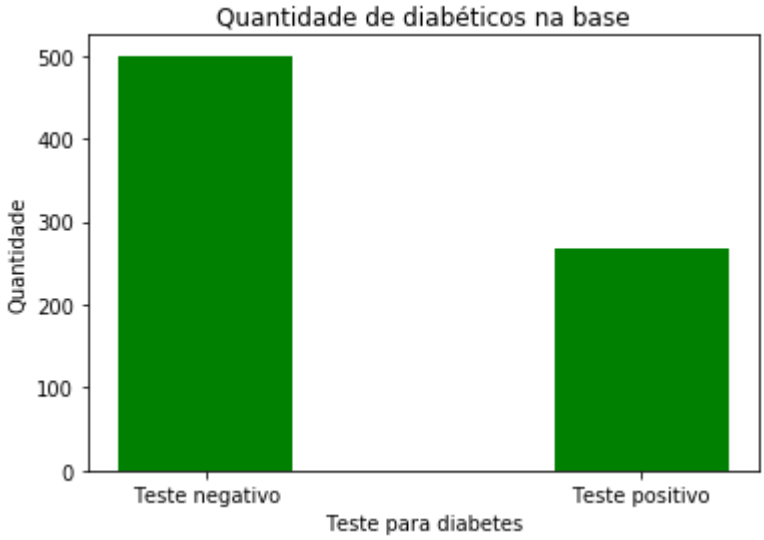
```
#Estatisticas descritivas

df_diabetes_novo.describe()

#Algumas observações do conjunto de dados:
#Verificamos que não existem valores ausentes no conjunto de dados.
#O numero máximo de vezes que uma pessoa engravidou foi 17.
#A pessoa com maior idade possui 81 anos enquanto a de menor idade possui 21 anos
```

Out[10]:

	Numero vezes grávida	Glicose	Pressão arterial	Espessura da dobra da pele	Insulina sérica	IMC	F
count	767.000000	767.000000	767.000000	767.000000	767.000000	767.000000	7(
mean	3.842243	120.859192	69.101695	20.517601	79.903520	31.990482	0.
std	3.370877	31.978468	19.368155	15.954059	115.283105	7.889091	0.
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.
50%	3.000000	117.000000	72.000000	23.000000	32.000000	32.000000	0.
75%	6.000000	140.000000	80.000000	32.000000	127.500000	36.600000	0.
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.

In [11]:

```
#Verificação da variável resposta Teste para diabetes
#Verificamos que do total de 767 pessoas, 500 tiveram o teste negativo para o diabetes
 e 267 teste possitivo para o diabetes

df_teste_diabetes = df_diabetes_novo.groupby(["Teste para diabetes"]).size().reset_inde
x(name='Qtd')

#renomeia as colunas
df_teste_diabetes.columns = ["Teste para diabetes","Qtd"]

df_grafico_diabetes = df_teste_diabetes.replace({'Teste para diabetes': {0: "Teste nega
tivo", 1: "Teste positivo"}})

plt.bar(df_grafico_diabetes["Teste para diabetes"], df_grafico_diabetes["Qtd"], width =
 0.4, color='green', align='center')

plt.title('Quantidade de diabéticos na base')
plt.xlabel('Teste para diabetes')
plt.ylabel('Quantidade')
plt.show()
```



3. Separação dos dados para Predição

In [12]:

```
#Armazena os dados da variável resposta "Teste para diabetes" na variável y_train
le = LabelEncoder()
y_train = le.fit_transform(df_diabetes_novo.iloc[:,(df_diabetes_novo.shape[1] - 1)])

#Cria um novo data frame sem a variável resposta que será utilizado no treinamento do m
odelo

X_dict = df_diabetes_novo.iloc[:,0:8].T.to_dict().values()

vect = DictVectorizer(sparse=False)

X_train = vect.fit_transform(X_dict)
```

4. Árvore de decisão

In [13]:

```
#Árvore de decisão

diabetes_tree = DecisionTreeClassifier(random_state=0)
diabetes_tree = diabetes_tree.fit(X_train, y_train)
print("Acurácia:", diabetes_tree.score(X_train, y_train))

Train_predict = diabetes_tree.predict(X_train)
print("Acurácia de previsão:", accuracy_score(y_train, Train_predict))
print(classification_report(y_train, Train_predict))

with open("diabetes_tree.dot", 'w') as f:
     f = tree.export_graphviz(diabetes_tree, out_file=f,
                              feature_names=vect.feature_names_,
                              class_names=["Teste para diabetes=Nao", "Teste para diabe
tes=Sim"])
```

```
Acurácia: 1.0
Acurácia de previsão: 1.0
             precision    recall  f1-score   support

          0       1.00      1.00      1.00       500
          1       1.00      1.00      1.00       267

avg / total       1.00      1.00      1.00       767
```

5. Naive Bayes

In [14]:

```
nb_nominal = BernoulliNB()
#Aplica o algoritmo de naive bayes
nb_nominal = nb_nominal.fit(X_train, y_train)
# print da acuracia do algoritmo
print("Acurácia:", nb_nominal.score(X_train, y_train))

#cria um y_pred para o modelo Naive Bayes (Outro nome para não sobrepor o criado na arv
ore de decisão)
y_pred_BNB = nb_nominal.predict(X_train)
#print da acurácia da previsão
print("Acurácia de previsão:", accuracy_score(y_train, y_pred_BNB))
print(classification_report(y_train, y_pred_BNB))
```

```
Acurácia: 0.6544980443285529
Acurácia de previsão: 0.6544980443285529
             precision    recall  f1-score   support

          0       0.66      0.98      0.79       500
          1       0.54      0.05      0.10       267

avg / total       0.62      0.65      0.55       767
```

6. Justificativa para uso dos algorítimos

A árvore de decisão é uma técnica estatística de treinamento supervisionado utilizada na classificação e previsão dos dados. É um modelo que usa a ideia de dividir para conquistar, em outras palavras, decompõe um problema maior em sub-problemas de ordem mais simples, para de forma recursiva se consiga a resolução do problema como um todo. É uma técnica bastante utilizada em diagnóstico médico e, por isso foi escolhida para ser utilizada nesse trabalho.

O Naive Bayes também é um algorítimo classificador probabilístico muito eficiente e de simples implementação. Apenas com uma pequena quantidade de dados é possível obter classificações com uma boa previsão. Essa técnica desconsidera completamente a correlação entre variáveis e portanto, bem diferente da árvore de decisão. Por esse motivo foi escolhido para ser utilizada nesse trabalho.

7. Análise dos resultados

Neste exercício foram utilizados dois algoritmos, a arvore de decisão e o algoritmo de Naive Bayes. Para o problema de diagnóstico da diabete a arvore de decisão apresentou um melhor resultado alcançando 100% de acurácia contra 65,4% do Naive Bayes. Um ponto importante, é que a tratativa dos dados para ambos os algoritmos foi exatamente a mesma, como vimos ao longo da disciplina se forem aplicados métodos diferentes para cada algoritmo o resultado pode ser afetado, porém isso não foi realizado no trabalho em questão.

Referências Bibliográficas

http://www.prontosaude.com.br/post/diabetes-a-importancia-do-diagnostico-precoce (http://www.prontosaude.com.br/post/diabetes-a-importancia-do-diagnostico-precoce)

http://agenciabrasil.ebc.com.br/geral/noticia/2016-11/uma-em-cada-11-pessoas-no-mundo-tem-diabetes-alerta-oms (http://agenciabrasil.ebc.com.br/geral/noticia/2016-11/uma-em-cada-11-pessoas-no-mundo-tem-diabetes-alerta-oms)

https://portal.fiocruz.br/noticia/taxa-de-incidencia-de-diabetes-cresceu-618-nos-ultimos-10-anos (https://portal.fiocruz.br/noticia/taxa-de-incidencia-de-diabetes-cresceu-618-nos-ultimos-10-anos)

https://pt.wikipedia.org/wiki/Diabetes_mellitus (https://pt.wikipedia.org/wiki/Diabetes_mellitus)

http://storm.cis.fordham.edu (http://storm.cis.fordham.edu)

https://www.organicadigital.com/seeds/algoritmo-de-classificacao-naive-bayes/ (https://www.organicadigital.com/seeds/algoritmo-de-classificacao-naive-bayes/)