```
# -*- coding: utf-8 -*-
"""
TCC_CaioMota_ArvoreDecisao_SPNeoDeath_2012_2018_FINAL.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/11WavsUi1N0M-HTOFKWiYbJUU92NadjRa

**Implementação de modelo utilizando algoritmo de Árvore de Decisão**
<br>
<br>
**Autor**: Caio Augusto de Souza Mota (*caiomota802@gmail.com*)

Data: 09/06/2021

**Revisor**: Carlos Eduardo Beluzo (*cbeluzo@gmail.com*)
<br>
<br>
*Codigo adaptado de Diogo Cortiz disponivel em: https://github.com/diogocortiz/Curso-IA-para-todos/tree/master/ArvoreDecis%C3%A3o*

---
Este código é parte do Trabalho de Conclusão de Curso  apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo Câmpus Campinas.

# 1. Importação de Bibliotecas, carga de dados e funções
"""

# Instalando o Synapse Client
```

```
28 ! pip install synapseclient
29
30 # Util
31 import itertools
32 import synapseclient as syna
33 from getpass import getpass
34
35 # Data
36 import numpy as np
37 import pandas as pd
38
39 from sklearn.tree import DecisionTreeClassifier,
   export_graphviz
40 from sklearn import preprocessing
41 from sklearn.model_selection import train_test_split
   , cross_val_score, cross_validate, cross_val_predict
42 from sklearn.externals.six import StringIO
43 from sklearn.metrics import accuracy_score,
   classification_report, precision_score, recall_score
   , confusion_matrix, precision_recall_curve, roc_curve
   , auc, log_loss
44
45 # Images
46 import matplotlib.pyplot as plt
47 import pydotplus
48 import matplotlib.image as mpimg
49 from IPython.display import Image
50
51 """## 1.1 Carregando base de dados disponícel no
   Synapse"""
52
53 # SPNeodeath
54 # Recuperando a base de dados do repositório de dados
    Synapse
55 syn = syna.Synapse()
56 syn.login(input('Sybapse User: '), getpass('Passwd:'
   ))
57
58 # Obtendo um ponteiro e baixando os dados
59 dataset = syn.get(entity='syn22240290') # ID do
   dataset SPNeodeath
```

```
60
61 df_ori = pd.read_csv(dataset.path)
62 df_ori.shape
63
64 # Amostra para testes rápidos
65 df = df_ori.sample(frac=1)
66
67 # REMOVENDO OS REGISTROS NOS QUAIS PELO MENOS UM
   CAMPO ESTÁ EM BRANCO (NAN)
68 df = df.dropna()
69
70 # Exibindo as 5 primeiras linhas
71 print(df.head(5))
72
73 count_row = df.shape[0]  # Número de linhas
74 count_col = df.shape[1]  # Número de colunas
75
76 print(count_row)
77 print(count_col)
78
79 # Disbribuição entre registros de vivos (negativo =
   0) e mortos (positivo = 1)
80 print ('Total de registros negativos: ', df[df['
   neonatal_death'] == 0 ].shape[0])
81 print ('Total de registros positivos: ', df[df['
   neonatal_death'] == 1 ].shape[0])
82
83 """# 2. Aplicação de modelos de Machine Learning:
   Decision Tree
84
85 ---
86
87 Precisamos converter o Dataframe para um Array Numpy
   , que é o tipo de dados que iremos usar no
   treinamento.
88
89 Um com as features de entrada, e outro com os labels
    (etiquetas, rótulos do registro).
90 """
91
92 # "Features" do modelo
```

```
 93 nome_features = ['tp_birth_place', 'maternal_age', '
    tp_marital_status',
 94                 'tp_maternal_education_years', '
    num_live_births','num_fetal_losses',
 95                 'tp_pregnancy_duration', '
    tp_pregnancy', 'tp_labor',
 96                 'tp_prenatal_appointments', '
    cd_apgar1', 'cd_apgar5',
 97                 'newborn_weight', '
    has_congenital_malformation', '
    tp_maternal_skin_color',
 98                 'num_gestations', 'num_normal_labors'
    , 'num_cesarean_labors',
 99                 'num_gestational_weeks', '
    tp_presentation_newborn', 'tp_childbirth_assistance'
    ,
100                 'tp_fill_form_responsible','
    cd_robson_group']
101
102 target = ['neonatal_death']
103
104 # Array de features
105 X = df[nome_features].values
106
107 Y = df[target].values
108
109 """### Criando modelo usando divisão da base em  90
    %/10% para treino e teste."""
110
111 # "Split" do dataset para Treino e Teste do modelo
112 # Usando treino com 90% dos dados e 10% dos dados
    para teste
113 xTreino, xTeste, yTreino, yTeste = train_test_split(
    X, Y,
114
    test_size=0.1,
115
    shuffle=False,
116
    random_state=7)
117
```

```
118 # Instanciando algortimo
119 algortimo_arvore = DecisionTreeClassifier(criterion=
    'entropy', max_depth=5)
120
121 # "Treinamento"/Construção do modelo
122 modelo_90_10 = algortimo_arvore.fit(xTreino, yTreino
    )
123
124 """### Exibindo a árvore que o modelo montou
125
126 A árvore de decisão pode ser considerada um modelo
    White Box, ou seja, um modelo que podemos entender
    melhor o que ele aprendeu e como ele decide. Podemos
     mostrar a árvore para isso.
127 """
128
129 importances = modelo_90_10.feature_importances_
130 indices = np.argsort(importances)[::-1]
131 print("Importância de cada variável para o modelo:")
132
133 for f in range(X.shape[1]):
134     print("%d. feature %d (%f)" % (f + 1, indices[f
    ], importances[indices[f]]))
135 f, ax = plt.subplots(figsize=(10, 5))
136 plt.title("Importância de cada variável para o
    modelo", fontsize = 14)
137 plt.bar(range(X.shape[1]), importances[indices],
138     color="b",
139     align="center")
140 plt.xticks(range(X.shape[1]), indices)
141 plt.xlim([-1, X.shape[1]])
142 plt.ylabel("Importância", fontsize = 12)
143 plt.xlabel("Features (índice)", fontsize = 12)
144
145 plt.savefig("importance_features_tree.png", dpi=300)
146
147 plt.show()
148
149
150 # 0 - 'maternal_age'
151 # 1 - 'tp_maternal_schooling'
```

```
152 # 2 - 'tp_marital_status'
153 # 3 - 'tp_maternal_race'
154 # 4 - 'num_live_births'
155 # 5 - 'num_fetal_losses'
156 # 6 - 'num_previous_gestations'
157 # 7 - 'num_normal_labors'
158 # 8 - 'num_cesarean_labor'
159 # 9 - 'tp_pregnancy'
160 # 10 - 'newborn_weight'
161 # 11 - 'gestaional_week'
162 # 12 - 'cd_apgar1',
163 # 13 - 'cd_apgar5'
164 # 14 - 'has_congenital_malformation'
165 # 15 - 'tp_newborn_presentation'
166 # 16 - 'num_prenatal_appointments'
167 # 17 - 'tp_labor'
168 # 18 -'was_cesarean_before_labor'
169 # 19 -'was_labor_induced'
170 # 20 -'tp_childbirth_care'
171 # 21 -'tp_robson_group'
172
173 # Montando imagem da árvore de decisão
174 dot_data = StringIO()
175 export_graphviz(modelo_90_10, out_file=dot_data,
    filled=True, feature_names=nome_features,
176                 class_names=['Vivo','Morto'],
    rounded=True, special_characters=True,
177                 node_ids=True, proportion=True,
    max_depth=5,rotate=True)
178
179 graph = pydotplus.graph_from_dot_data(dot_data.
    getvalue())
180 Image(graph.create_png())
181 graph.write_png("arvore.png")
182 Image('arvore.png',)
183
184 # Montando imagem da árvore de decisão
185 dot_data = StringIO()
186 export_graphviz(modelo_90_10, out_file=dot_data,
    filled=True, feature_names=nome_features,
187                 class_names=['Vivo','Morto'],
```

```
187 rounded=True, special_characters=True,
188                 node_ids=True, proportion=True,
    max_depth=2)
189
190 graph = pydotplus.graph_from_dot_data(dot_data.
    getvalue())
191 Image(graph.create_png())
192 graph.write_png("arvore_short.png")
193 Image('arvore_short.png',)
194
195 """### Curva ROC
196
197 """
198
199 y_pred = modelo_90_10.predict(xTeste)
200 y_pred_proba = modelo_90_10.predict_proba(xTeste
    )[:, 1]
201 [fpr, tpr, thr] = roc_curve(yTeste, y_pred_proba)
202
203 print('Treino/Teste resultados para %s:' %
    modelo_90_10.__class__.__name__)
204 print("Accuracy = %2.3f" % accuracy_score(yTeste,
    y_pred))
205 print("Log_loss = %2.3f" % log_loss(yTeste,
    y_pred_proba))
206 print("AUC = %2.3f" % auc(fpr, tpr))
207
208 idx = np.min(np.where(tpr > 0.95))
209
210 #plt.figure(figsize=(13, 13))
211 plt.plot(fpr, tpr, color='coral', label='Curva ROC (
    area = %0.3f)' % auc(fpr, tpr))
212 plt.plot([0, 1], [0, 1], 'k--')
213 plt.plot([0,fpr[idx]], [tpr[idx],tpr[idx]], 'k--',
    color='blue')
214 plt.plot([fpr[idx],fpr[idx]], [0,tpr[idx]], 'k--',
    color='blue')
215 plt.xlim([0.0, 1.0])
216 plt.ylim([0.0, 1.05])
217 plt.xlabel('Taxa de falso positivo (1 -
    especificidade)', fontsize=14)
```

```
218 plt.ylabel('Taxa de verdadeiro positivo (recall)',
    fontsize=14)
219 plt.title('Curva ROC')
220 plt.legend(loc="lower right")
221
222 plt.savefig("ROC.png", dpi=300)
223 print("\n*Nota: Usando um limite de %.3f " % thr[idx
    ] +
224       "garante uma sensibilidade de %.3f " % tpr[idx
    ] +
225       "\ne uma especificidade de %.3f" % (1-fpr[idx
    ]) +
226       ", ou seja, uma taxa de falsos positivos de %.
    2f%%.\n" % (np.array(fpr[idx])*100))
227
228 plt.show()
229
230 """Métricas do modelo usando divisão 90/10
231
232 - PRECISÃO: DAS CLASSIFICAÇÕES QUE O MODELO FEZ PARA
     UMA DETERMINADA CLASSE
233
234 - RECALL: DOS POSSÍVEIS DATAPOINTS PERTECENTES A UMA
     DETERMINADA CLASSE
235 """
236
237 Y_predicoes = modelo_90_10.predict(xTeste)
238
239 print("ACURÁCIA DA ÁRVORE: ", accuracy_score(yTeste
    , Y_predicoes))
240 print (classification_report(yTeste, Y_predicoes))
241
242 """ # 3. Criando modelo usando validação cruzada"""
243
244 # k = 10 na divisão da base para treino e teste
245 algortimo_arvore = DecisionTreeClassifier(criterion=
    'entropy', max_depth=3)
246 scoring = {'accuracy': 'accuracy', 'log_loss': '
    neg_log_loss', 'auc': 'roc_auc'}
247
248 modelo2 = cross_validate(algortimo_arvore, X, Y, cv=
```

```
248 10,
249                             scoring=list(scoring.values
    ()),
250                             return_train_score=False)
251
252 print('Resultado usando K-fold = 10 no cross-
    validation:\n')
253 for sc in range(len(scoring)):
254     print("%s: %.3f (+/-%.3f)" % (list(scoring.keys
    ())[sc], -modelo2['test_%s' % list(scoring.values
    ())[sc]].mean()
255                                    if list(scoring.
    values())[sc]=='neg_log_loss'
256                                    else modelo2['test_%s
    ' % list(scoring.values())[sc]].mean(),
257                                    modelo2['test_%s' %
    list(scoring.values())[sc]].std()))
258
259 """Matriz de confusão
260
261 """
262
263 algortimo_arvore = DecisionTreeClassifier(criterion=
    'entropy', max_depth=3)
264 modelo2_predicts = cross_val_predict(
    algortimo_arvore, X, Y, cv=10)
265 cm_modelo2 = confusion_matrix(Y, modelo2_predicts)
266 print(cm_modelo2)
267
268 # generate report
269 print(classification_report(Y, modelo2_predicts))
```