Equipe 16

- Ana Beatriz Kindinger
- Daniel Victor Andrade
- Igor Buess Atala Y Mansour
- Marlon Mateus Prudente de Oliveira
- Ronaldo Santana da Silva Moco

Atividade 02 - melhorar o desempenho de RP em conjunto de dados existentes

A atividade 02 visa trabalhar com um conjunto de dados pré-construído, onde as opções que o desenvolvedor tem, são de aplicar as técnicas de pré-processamento abaixo relacionadas:

- Seleção
- Limpeza
- Codificação
- Enriquecimento
- Normalização
- Construção de Atributos
- Correção de Prevalência
- Partição do Conjunto de Dados

Busque uma base de dados na UCI Machine Learning que seja indicada para problemas de classificação. (https://archive.ics.uci.edu/datasets)

Para esse exemplo, vamos usar a base Secondary Mushroom (https://archive.ics.uci.edu/static/public/848/data.csv)

Opção 01 - carregando o arquivo de dados da pasta local para o colab.

```
from sklearn.preprocessing import scale
from sklearn.preprocessing import minmax_scale
import pandas as pd

from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

from sklearn.model_selection import train_test_split
import numpy as np

```

```
url = "https://archive.ics.uci.edu/static/public/848/data.csv"
colunas = ["class", "cap-diameter", "cap-shape", "cap-surface", "cap-color", "does-bruise-or-bleed",
           "gill-attachment", "gill-spacing", "gill-color", "stem-height",  "stem-width", "stem-root",
           "stem-surface",  "stem-color",   "veil-type",    "veil-color",   "has-ring", "ring-type",
           "spore-print-color", "habitat",  "season"]

colunas_categoricas = [ "cap-shape", "cap-surface", "cap-color", "does-bruise-or-bleed",
           "gill-attachment", "gill-spacing", "gill-color", "stem-root",
           "stem-surface",  "stem-color",   "veil-type",    "veil-color",   "has-ring", "ring-type",
           "spore-print-color", "habitat",  "season"]

colunas_numericas = ["cap-diameter", "stem-height", "stem-width"]

sm = pd.read_csv(url, header=None, low_memory=False, names=colunas)
sm = sm.drop(index=0)

print(type(sm))
print(sm.head())
```

```
<class 'pandas.core.frame.DataFrame'>
  class cap-diameter cap-shape cap-surface cap-color does-bruise-or-bleed  \
1     p        15.26         x           g         o                    f
2     p         16.6         x           g         o                    f
3     p        14.07         x           g         o                    f
4     p        14.17         f           h         e                    f
5     p        14.64         x           h         o                    f

  gill-attachment gill-spacing gill-color stem-height  ... stem-root  \
1               e          NaN          w       16.95  ...         s
2               e          NaN          w       17.99  ...         s
3               e          NaN          w        17.8  ...         s
4               e          NaN          w       15.77  ...         s
```

```
5              e          NaN          w       16.53  ...         s

  stem-surface stem-color veil-type veil-color has-ring ring-type  \
1            y          w         u          w        t         g
2            y          w         u          w        t         g
3            y          w         u          w        t         g
4            y          w         u          w        t         p
5            y          w         u          w        t         p

  spore-print-color habitat season
1               NaN       d      w
2               NaN       d      u
3               NaN       d      w
4               NaN       d      w
5               NaN       d      w

[5 rows x 21 columns]
```

Hora de realizar os tratamentos

no exemplo, iremos normalizar as colunas, remover a coluna de identificação e separar a classe dos atributos.

```
sm_dummies = pd.get_dummies(sm[colunas_categoricas])
X = pd.concat([sm[colunas_numericas], sm_dummies], axis=1)


#X = sm.iloc[:,1:]
cols = sm[0:]
print(X.head())
Y = sm['class']
Y_orig = sm['class']
print(Y.unique())
```

```
  cap-diameter stem-height stem-width  cap-shape_b  cap-shape_c  cap-shape_f  \
1        15.26       16.95      17.09        False        False        False
2         16.6       17.99      18.19        False        False        False
3        14.07        17.8      17.74        False        False        False
4        14.17       15.77      15.98        False        False         True
5        14.64       16.53       17.2        False        False        False

   cap-shape_o  cap-shape_p  cap-shape_s  cap-shape_x  ...  habitat_h  \
1        False        False        False         True  ...      False
2        False        False        False         True  ...      False
3        False        False        False         True  ...      False
4        False        False        False        False  ...      False
5        False        False        False         True  ...      False

   habitat_l  habitat_m  habitat_p  habitat_u  habitat_w  season_a  season_s  \
1      False      False      False      False      False     False     False
2      False      False      False      False      False     False     False
3      False      False      False      False      False     False     False
4      False      False      False      False      False     False     False
5      False      False      False      False      False     False     False

   season_u  season_w
1     False      True
2      True     False
3     False      True
4     False      True
5     False      True

[5 rows x 119 columns]
['p' 'e']
```

Na próxima seção que deverão ser realizada as tentativas de tratamento de dados, visando a melhoria no desempenho do classificador (SVM).

```
X_orig =  X.copy()
print(X_orig.head())

print(Y_orig.unique() )

# normalização min-max
X = pd.DataFrame( minmax_scale(X) )

#retirada de dados faltantes

X.dropna(axis = 1, how ='any')

print(X_orig.head())
print(X.head())
```

```
2      False      False      False      False      False     False     False
3      False      False      False      False      False     False     False
4      False      False      False      False      False     False     False
5      False      False      False      False      False     False     False

   season_u  season_w
1     False      True
2      True     False
3     False      True
4     False      True
5     False      True

[5 rows x 119 columns]
['p' 'e']
  cap-diameter stem-height stem-width  cap-shape_b  cap-shape_c  cap-shape_f  \
1        15.26       16.95      17.09        False        False        False
2         16.6       17.99      18.19        False        False        False
3        14.07        17.8      17.74        False        False        False
4        14.17       15.77      15.98        False        False         True
5        14.64       16.53       17.2        False        False        False

   cap-shape_o  cap-shape_p  cap-shape_s  cap-shape_x  ...  habitat_h  \
1        False        False        False         True  ...      False
2        False        False        False         True  ...      False
3        False        False        False         True  ...      False
4        False        False        False        False  ...      False
5        False        False        False         True  ...      False

   habitat_l  habitat_m  habitat_p  habitat_u  habitat_w  season_a  season_s  \
1      False      False      False      False      False     False     False
2      False      False      False      False      False     False     False
3      False      False      False      False      False     False     False
4      False      False      False      False      False     False     False
5      False      False      False      False      False     False     False

   season_u  season_w
1     False      True
2      True     False
3     False      True
4     False      True
5     False      True

[5 rows x 119 columns]
        0         1         2    3    4    5    6    7    8    9    ...  109  \
0  0.240155  0.499705  0.164469  0.0  0.0  0.0  0.0  0.0  0.0  1.0  ...  0.0
1  0.261782  0.530366  0.175055  0.0  0.0  0.0  0.0  0.0  0.0  1.0  ...  0.0
2  0.220949  0.524764  0.170725  0.0  0.0  0.0  0.0  0.0  0.0  1.0  ...  0.0
3  0.222563  0.464917  0.153787  0.0  0.0  1.0  0.0  0.0  0.0  0.0  ...  0.0
4  0.230148  0.487323  0.165528  0.0  0.0  0.0  0.0  0.0  0.0  1.0  ...  0.0

   110  111  112  113  114  115  116  117  118
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0

[5 rows x 119 columns]
```

A próxima seção trata da construção do modelo, dos testes e das métricas da matriz de confusão.

```
# com os dados originais
X_oring_train, X_orig_test, y_orig_train, y_orig_test = train_test_split(X_orig,
                     Y_orig, test_size=0.25, stratify=Y_orig,random_state=10)

# com os dados tratados
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25,
                                                    stratify=Y,random_state=10)
```

Treina o modelo com base nos dados originais (SVM).

```
treinador = svm.SVC()  #algoritmo escolhido

modelo_orig = treinador.fit(X_oring_train, y_orig_train)

# score com os dados de treinamento
acuracia_orig = modelo_orig.score(X_oring_train, y_orig_train)
print("Acurácia nos dados de treinamento ORIGINAIS: {:.2f}%".format(acuracia_orig * 100))

# predição com os mesmos dados usados para treinar
y_orig_pred = modelo_orig.predict(X_oring_train)
cm_orig_train = confusion_matrix(y_orig_train, y_orig_pred)
print('Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO')
print(cm_orig_train)
print(classification_report(y_orig_train, y_orig_pred))
```

```
print(classification_report(y_orig_train, y_orig_pred))

# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para TESTES')
y2_orig_pred = modelo_orig.predict(X_orig_test)
cm_orig_test = confusion_matrix(y_orig_test, y2_orig_pred)
print(cm_orig_test)
print(classification_report(y_orig_test, y2_orig_pred))
```

```
Acurácia nos dados de treinamento ORIGINAIS: 95.03%
Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO
[[19080  1305]
 [  970 24446]]
              precision    recall  f1-score   support

           e       0.95      0.94      0.94     20385
           p       0.95      0.96      0.96     25416

    accuracy                           0.95     45801
   macro avg       0.95      0.95      0.95     45801
weighted avg       0.95      0.95      0.95     45801

Matriz de confusão - com os dados ORIGINAIS usados para TESTES
[[6274  522]
 [ 327 8145]]
              precision    recall  f1-score   support

           e       0.95      0.92      0.94      6796
           p       0.94      0.96      0.95      8472

    accuracy                           0.94     15268
   macro avg       0.95      0.94      0.94     15268
weighted avg       0.94      0.94      0.94     15268
```

Como os dados ficam após os processos de tratamento dos dados?

```
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

treinador = svm.SVC()  #algoritmo escolhido

modelo = treinador.fit(X_train, y_train)

# score com os dados de treinamento
acuracia = modelo.score(X_train, y_train)
print("Acurácia nos dados de treinamento TRATADOS: {:.2f}%".format(acuracia * 100))

# predição com os mesmos dados usados para treinar
y_pred = modelo.predict(X_train)
cm_train = confusion_matrix(y_train, y_pred)
print('Matriz de confusão - com os dados TRATADOS usados no TREINAMENTO')
print(cm_train)
print(classification_report(y_train, y_pred))

# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados TRATADOS usados para TESTES')
y2_pred = modelo.predict(X_test)
cm_test = confusion_matrix(y_test, y2_pred)
print(cm_test)
print(classification_report(y_test, y2_pred))
```

```
Acurácia nos dados de treinamento TRATADOS: 99.96%
Matriz de confusão - com os dados TRATADOS usados no TREINAMENTO
[[20368    17]
 [    3 25413]]
              precision    recall  f1-score   support

           e       1.00      1.00      1.00     20385
           p       1.00      1.00      1.00     25416

    accuracy                           1.00     45801
   macro avg       1.00      1.00      1.00     45801
weighted avg       1.00      1.00      1.00     45801

Matriz de confusão - com os dados TRATADOS usados para TESTES
[[6795    1]
 [   3 8469]]
              precision    recall  f1-score   support

           e       1.00      1.00      1.00      6796
           p       1.00      1.00      1.00      8472
```

```
              precision    recall  f1-score   support

    accuracy                         1.00     15268
   macro avg       1.00      1.00      1.00     15268
weighted avg       1.00      1.00      1.00     15268
```