Carlos Eduardo Beluzo

16/Jun/2021

LADDEM/UNICAMP - Laboratório de Análise de Dados em Demografia.

Análise de Dados (Hello World!)

Importação de bibliotecas

```
! pip install squarify
```

```
Collecting squarify
  Downloading https://files.pythonhosted.org/packages/0b/2b/2e77c35326efec1
Installing collected packages: squarify
Successfully installed squarify-0.4.3
```

```
%matplotlib inline

# Manipulação de Dados
import numpy
import pandas

# Dataviz
import matplotlib.pyplot as plt
import seaborn as sns
import squarify # TreeMap Graph

# Machine Learning
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
```

Carregando a base de dados

```
# Leitura de arquivo CSV e conversão da um objeto do tipo Data Frame
meu_data_frame_original = pandas.read_csv('sivep_sample_10.csv', sep=';')

# Dimensões da base
meu_data_frame_original.shape
```

```
(274524, 22)
```

```
# Exibir os primeiros registros da base para inspeção inicial
meu_data_frame_original.head()
```

	Unnamed: 0	tempo	trat	age_groups	year	grp_risc	importado	COD_OCUP	I
0	1556898	1	1	[60,Inf)	2007	Alto	Autóctone	Agricultura	I po M
1	2379030	1	1	[5,40)	2008	Alto	Autóctone	Agricultura	I viv o
2	2579468	4	1	[5,40)	2016	Medio	Importado	Agricultura	I viv o
3	1247069	0	1	[5,40)	2018	Alto	Autóctone	Outros	I fa con
4	2586983	5	0	[5,40)	2018	Baixo	Autóctone	Outros	

```
# Exibindo o nome de todas as colunas
colunas = meu_data_frame_original.columns
colunas
```

```
Index(['Unnamed: 0', 'tempo', 'trat', 'age_groups', 'year', 'grp_risc',
       'importado', 'COD_OCUP', 'ESQUEMA', 'EXAME', 'FALCIPARUM',
'GESTANTE.',
       'HEMOPARASI', 'ID_LVC', 'ID_PACIE', 'NIV_ESCO', 'QTD_CRUZ', 'RACA',
       'RES_EXAM', 'SEXO', 'TIPO_LAM', 'VIVAX'],
      dtype='object')
```

```
# Exibindo os valores de cada coluna (domínio de valores)
for col in colunas:
    print(col, ": ", meu_data_frame_original[col].dtype)
    print(meu_data_frame_original[col].unique(), "\n")
```

```
 'Infecções por Pf com Quinina em 7 dias'
 'Infecções pelo P. vivax, ou P. ovale com cloroquina em 3 dias e primaquin
 'Infecções não complicadas por P. falciparum no 1º trimestre da gestação e
 'Malária grave e complicada pelo P. falciparum em todas as faixas etárias'
 'Infecções por P. falciparum com a combinação fixa de artesunato+mefloquin
 'Infecções por Pv em crianças apresentando vômitos, com cápsulas retais de
 'Infecções mistas por Pv + Pf com Quinina em 3 dias, doxiciclina em 5 dias
 'Infecções por Pf de crianças com cápsulas retais de artesunato em 4 dias

EXAME :  object
[nan 'Gota espessa/esfregaço' 'Teste rápido']

FALCIPARUM :  object
[nan 'Não' 'Sim']

GESTANTE. :  object
[nan 'Não' '2º Trimestre' 'Idade gestacional ignorada' '3º Trimestre'
 '1º Trimestre']

HEMOPARASI :  object
[nan 'Não pesquisados' 'Negativo' 'Microfilária' 'Trypanosoma sp.'
 'Trypanosoma sp.+ Microfilária']

ID_LVC :  object
['Não LVC']

ID_PACIE :  int64
[ 75  18  28   5  23  48   9   6  24  29  45  32  13  25  22  20  37  16
  44  60  40  14   3  47   1  31  42  26  17  19  33  62  56  11   2  10
   8  27  15  35  63  66  21   4  77  50  39  54  30  43  38  69  55   7
  64  34  70  52  53  36  59  41  51   0  58  72  12  67  57  49  61  46
  85  73  65  68  78  80  74  84  76  83  94  96  88  79  87  71  81  82
  97 101  86  93  98 105 103  95  89  90  92  91  99 104 102 100]

NIV_ESCO :  object
['Analfabeto' '5ª a 8ª série incompleta do EF' nan 'Ensino médio completo'
 'Ensino médio incompleto' '4ª série completa do EF'
 'Educação superior incompleto' '1ª a 4ª série incompleta do EF'
 'Ensino fundamental completo' 'Educação superior completa']
```

```
QTD_CRUZ :  object
['< +/2' '++' '+/2' '+' '+++' '++++']

RACA :  object
[nan 'Parda' 'Indigena' 'Preta' 'Branca' 'Amarela']

RES_EXAM :  object
['F+Fg' 'Vivax' 'Falciparum' 'F+V' 'Não Falciparum' 'V+Fg' 'Fg' 'F+M'
 'Malariae']

SEXO :  object
['Masculino' 'Feminino' nan]

TIPO_LAM :  object
['Detecção Passiva' 'Detecção Ativa']

VIVAX :  object
[nan 'Não' 'Sim']
```

Análise inicial com gráficos

```
# Verificação valores nulos
meu_data_frame_original.isna().sum()
```

```
Unnamed: 0         0
tempo              0
trat               0
age_groups         0
year               0
grp_risc           0
importado          0
COD_OCUP       24774
ESQUEMA         4354
EXAME         127051
FALCIPARUM    127051
GESTANTE.     198275
HEMOPARASI    127051
ID_LVC             0
ID_PACIE           0
NIV_ESCO       32521
QTD_CRUZ           0
RACA          127051
RES_EXAM           0
SEXO              22
TIPO_LAM           0
VIVAX         127051
dtype: int64
```

```
# Calculando a proporção de nulos em cada coluna
porcentagens_de_nulos = (meu_data_frame_original.isna().sum() / 274524) * 10
porcentagens_de_nulos
```

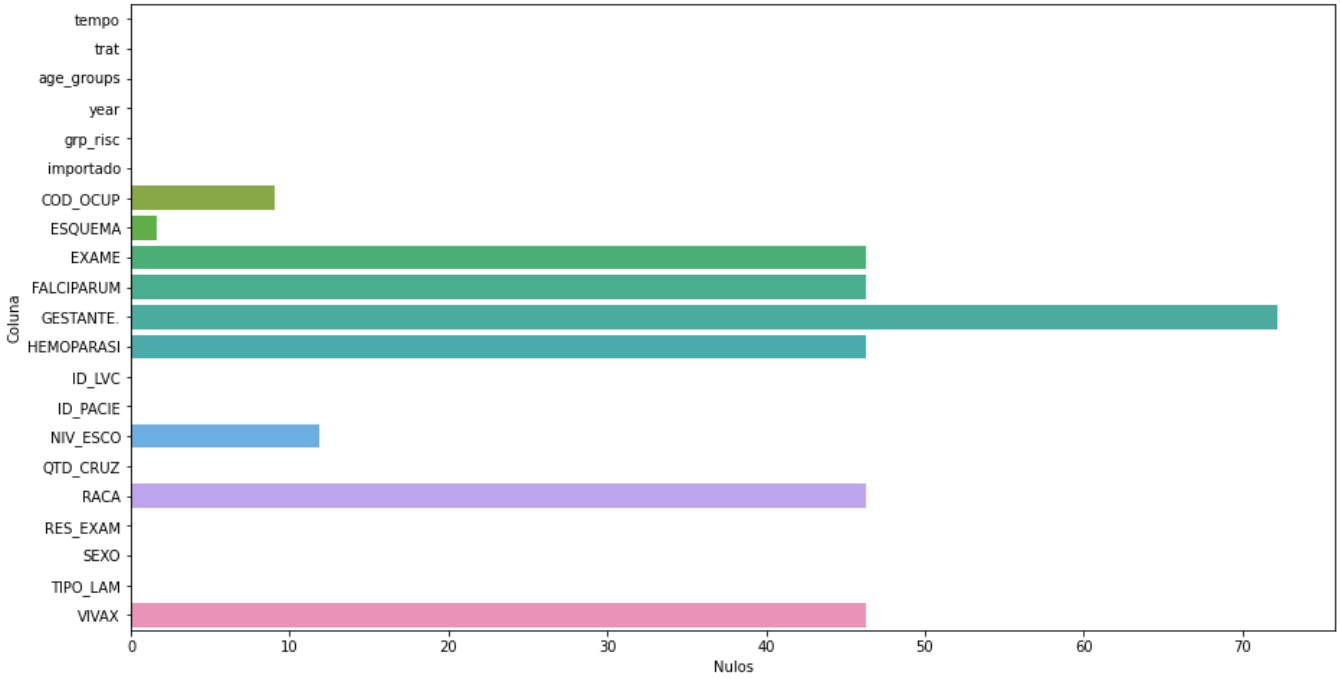
```
Unnamed: 0     0.000000
tempo          0.000000
trat           0.000000
age_groups     0.000000
year           0.000000
grp_risc       0.000000
importado      0.000000
COD_OCUP       9.024348
ESQUEMA        1.586018
EXAME         46.280471
FALCIPARUM    46.280471
GESTANTE.     72.225015
HEMOPARASI    46.280471
ID_LVC         0.000000
ID_PACIE       0.000000
NIV_ESCO      11.846323
QTD_CRUZ       0.000000
RACA          46.280471
RES_EXAM       0.000000
SEXO           0.008014
TIPO_LAM       0.000000
VIVAX         46.280471
dtype: float64
```

```
# Criar um DataFrame contendo as porcentagens de nulos por colunas
porcentagens_de_nulos_df = pandas.DataFrame(data = porcentagens_de_nulos)

# Removendo linhas indesejadas
porcentagens_de_nulos_df.drop('Unnamed: 0', inplace=True)

# Redefinindo indice
porcentagens_de_nulos_df.reset_index(inplace=True)

# Renomeando columnas
porcentagens_de_nulos_df.columns = ["Coluna", "Nulos"]

# Exibindo o DataFrame final
porcentagens_de_nulos_df
```

	Coluna	Nulos
0	tempo	0.000000
1	trat	0.000000
2	age_groups	0.000000
3	year	0.000000
4	grp_risc	0.000000
5	importado	0.000000
6	COD_OCUP	9.024348
7	ESQUEMA	1.586018
8	EXAME	46.280471
9	FALCIPARUM	46.280471
10	GESTANTE.	72.225015
11	HEMOPARASI	46.280471
12	ID_LVC	0.000000
13	ID_PACIE	0.000000
14	NIV_ESCO	11.846323
15	QTD_CRUZ	0.000000
16	RACA	46.280471
17	RES_EXAM	0.000000
18	SEXO	0.008014
19	TIPO_LAM	0.000000
20	VIVAX	46.280471

```
# Gráfico de barras dos valores nulos
plt.figure(figsize=(15,8))

sns.barplot(x = "Nulos", y = "Coluna", data = porcentagens_de_nulos_df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fda75187790>
```

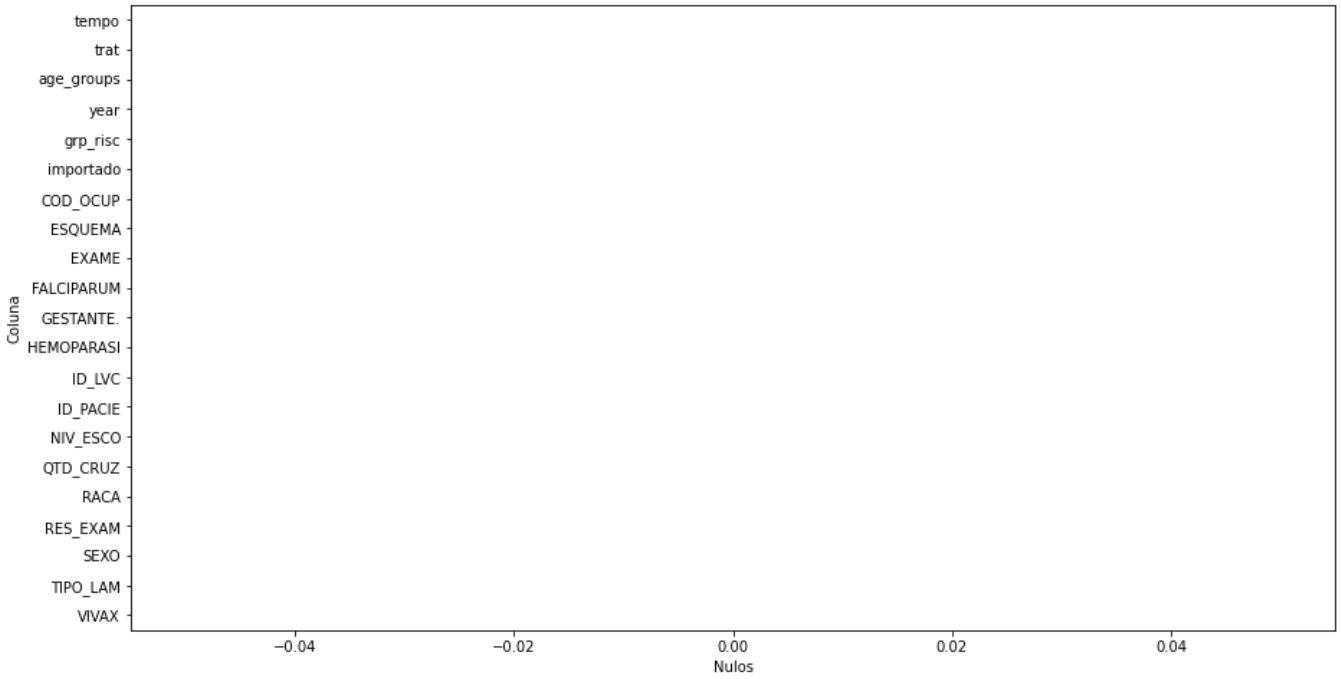


O que vazer com valores nulos?

- Excluir a linhas?
- Excluir a coluna?
- Colocar um valor "dummy"?

```
# Substituindo nulos por um valor "dummy"
meu_data_frame_original.fillna("-1", inplace=True)

porcentagens_de_nulos = (meu_data_frame_original.isna().sum() / 274524) * 10
porcentagens_de_nulos_df = pandas.DataFrame(data = porcentagens_de_nulos)

# Removendo linhas indesejadas
porcentagens_de_nulos_df.drop('Unnamed: 0', inplace=True)

# Redefinindo ínidices
porcentagens_de_nulos_df.reset_index(inplace=True)

# Renomeando Columnas
porcentagens_de_nulos_df.columns = ["Coluna", "Nulos"]

# Exibindo o DataFrame final
porcentagens_de_nulos_df

# Substituindo por um valor "dummy"
plt.figure(figsize=(15,8))
sns.barplot(x = "Nulos", y = "Coluna", data = porcentagens_de_nulos_df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fda6951ccd0>
```



Várias outras análise podem e devem ser realizadas, tipos de dados (numéricos ou categóricos), distribuição dos dados em cada variável.

Análise Exploratória

```
# Variáveis de interesse
variaveis_interesse = ["SEXO", "RACA", "ID_PACIE", "VIVAX", "ESQUEMA"]

# Fazendo o recorte com variáveis de interesse
meu_recorte = meu_data_frame_original[variaveis_interesse].copy()

# Variável que quero "prever" é a variável ESQUEMA, que tenha o valor abaixo
tratamento = 'Infecções pelo P. vivax, ou P. ovale com cloroquina em 3 dias

# Criando coluna 'Target' a partir dos valores da coluna ESQUEMA
meu_recorte['target'] = numpy.where(meu_recorte['ESQUEMA'] == tratamento, 1,

meu_recorte.head()
```

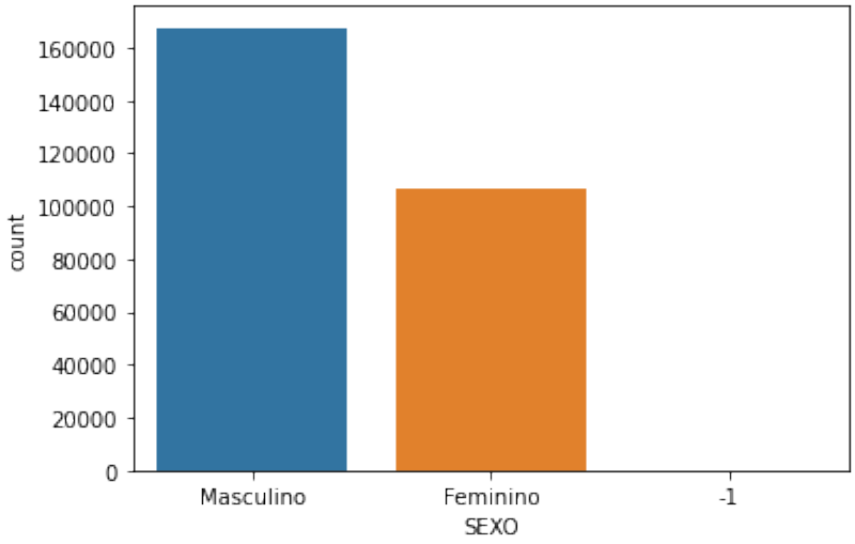
	SEXO	RACA	ID_PACIE	VIVAX	ESQUEMA	target
0	Masculino	-1	75	-1	Infecções por Pf com Mefloquina em dose única ...	0
1	Masculino	-1	18	-1	Infecções pelo P. vivax, ou P. ovale com cloro...	1
2	Masculino	Parda	28	Não	Infecções pelo P. vivax, ou P. ovale com cloro...	1
					Infecções por P. falciparum com a	

```
# Removendo coluna ESQUEMA
meu_recorte.drop(columns='ESQUEMA', inplace=True)
meu_recorte.head()
```

	SEXO	RACA	ID_PACIE	VIVAX	target
0	Masculino	-1	75	-1	0
1	Masculino	-1	18	-1	1
2	Masculino	Parda	28	Não	1
3	Feminino	Indigena	5	Não	0
4	Masculino	Parda	23	Não	0

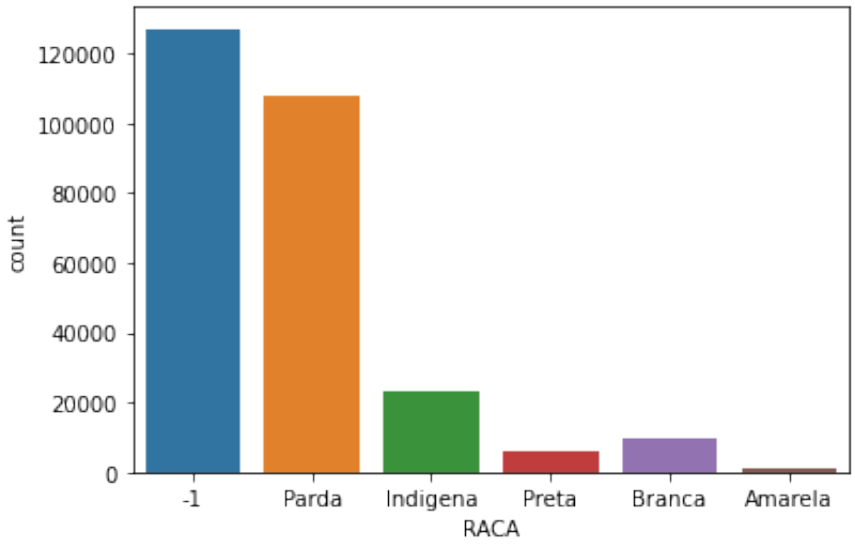
```
sns.countplot(x = 'SEXO', data = meu_recorte)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fda694c5910>
```



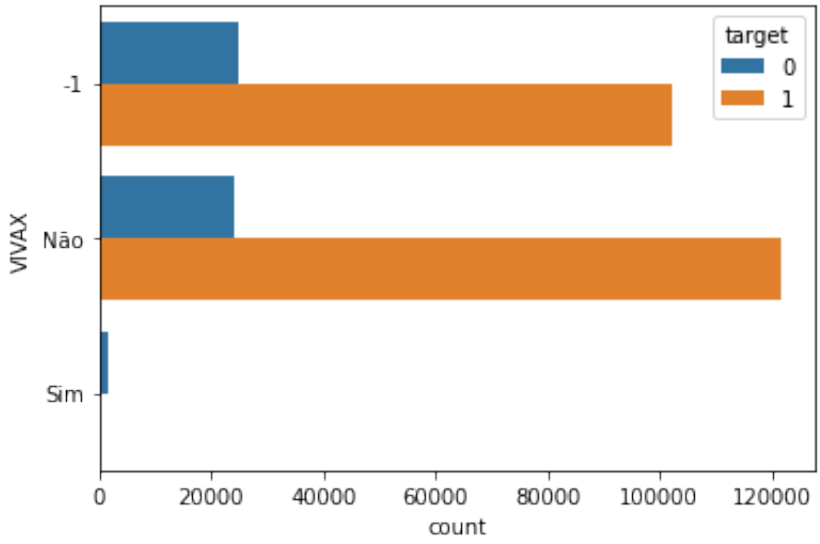

```
sns.countplot(x = 'RACA', data = meu_recorte)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fda6a0940d0>
```



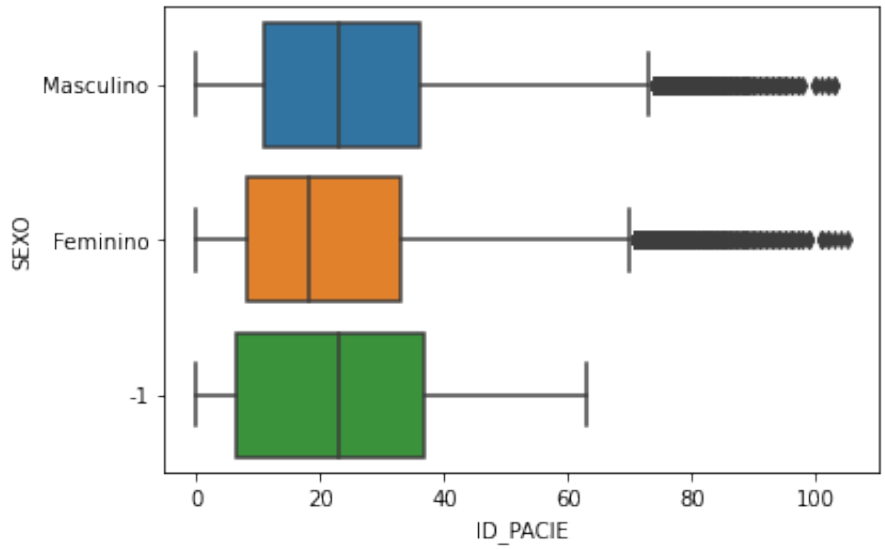
```
sns.countplot(y = 'VIVAX', hue="target", data = meu_recorte)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fda68e431d0>
```



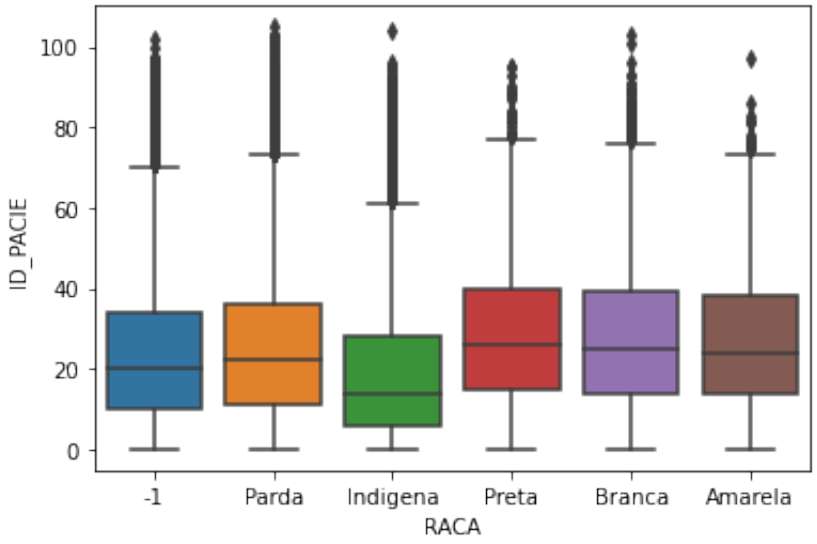

```
sns.boxplot(x="ID_PACIE", y="SEXO", data=meu_recorte)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fda68dc0b90>
```



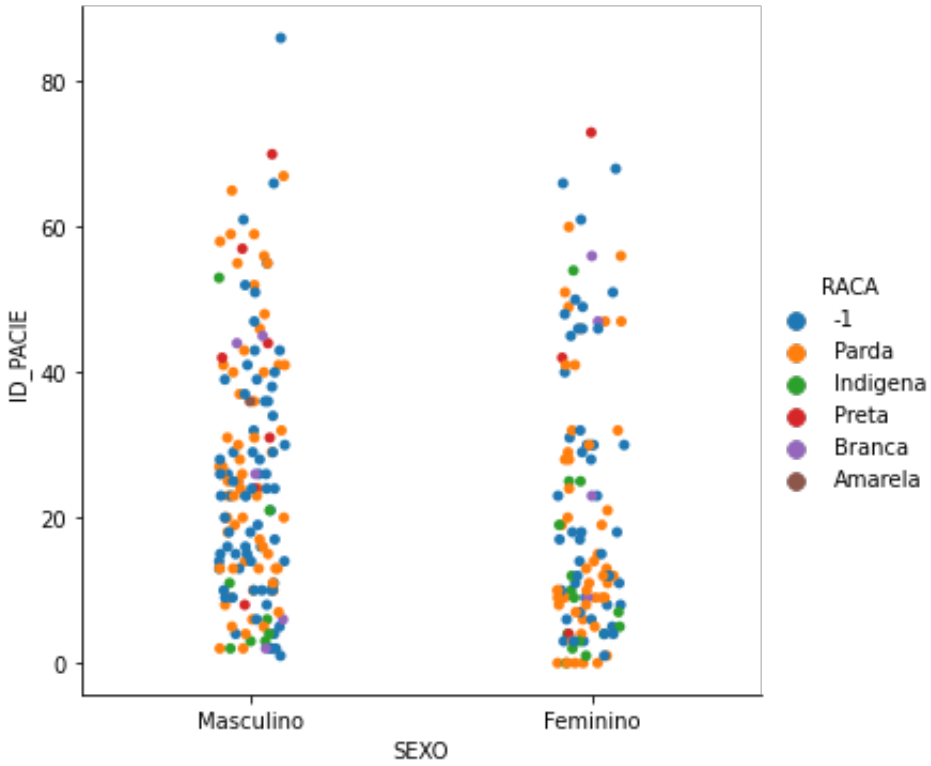
```
sns.boxplot(y="ID_PACIE", x="RACA", data=meu_recorte)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fda68ce5d90>
```

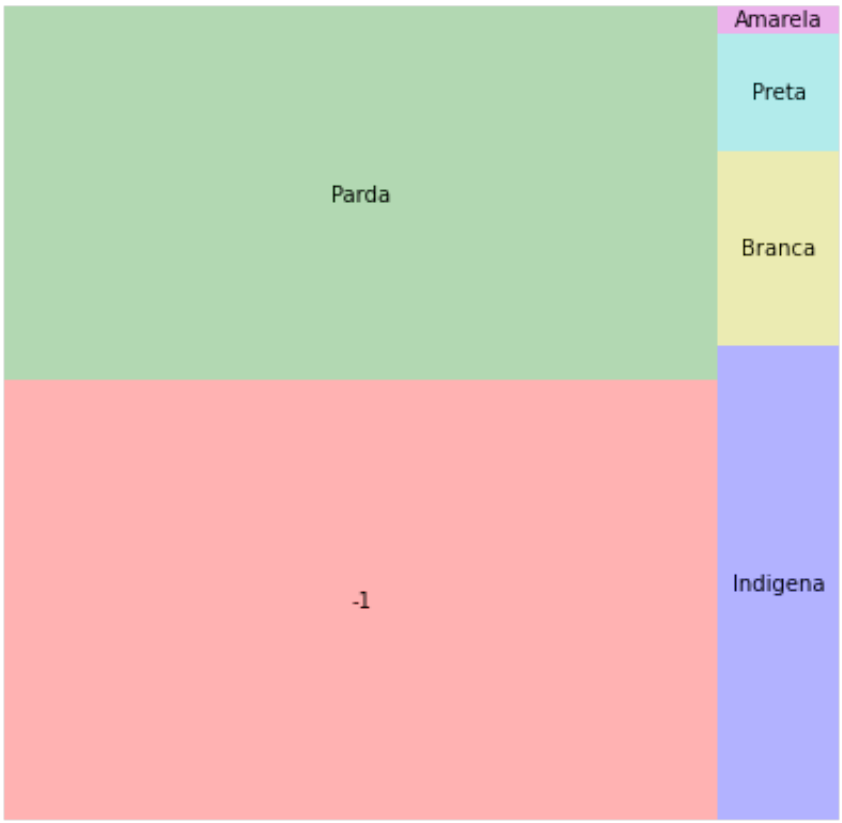



```
sns.catplot(x="SEXO", y="ID_PACIE", hue="RACA", data = meu_recorte.sample(fr
```

```
<seaborn.axisgrid.FacetGrid at 0x7fda68c13810>
```



```
# TreeMap
plt.figure(figsize=(7,7))

squarify.plot(sizes=meu_recorte.RACA.value_counts().values,
  label=meu_recorte.RACA.value_counts().index, alpha=.3,
  color=['r','g','b','y','c','m','tab:orange'])

plt.axis('off')
plt.show()
```



Machine Learning (Hello World!)

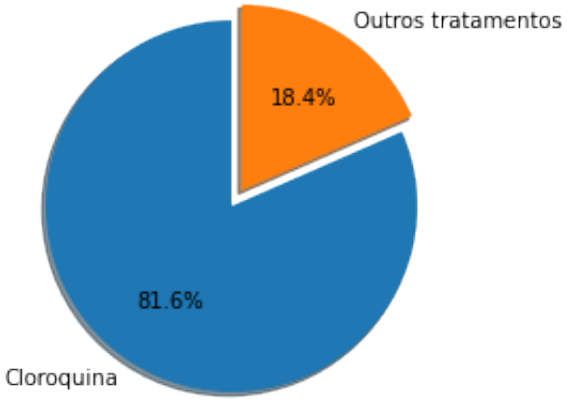
Objetivo

Prever se um paciente diagnosticado com Malária deve ser tratado com cloroquina, com base em dados sociodemográficos.

Distribuição da Base

Distribuição da base de dados entre os que foram tratados com cloroquina, e os que foram tratados com outros esquemas de tratamento.

```
plt.pie(meu_recorte.target.value_counts(),
        explode = (0, 0.1),
        labels = ['Cloroquina', 'Outros tratamentos'],
        autopct='%1.1f%%', shadow=True, startangle=90)

plt.show()
```



Preparação dos dados

```
# Removendo a coluna a ser prevista
X = meu_recorte[["SEXO", "RACA", "ID_PACIE", "VIVAX"]].copy()
Y = meu_recorte['target'].copy()

# Preparando dados para o algorítimo XGBoost especificamente
X.SEXO.replace({'Masculino':0, 'Feminino': 1}, inplace=True)
X.SEXO = X.SEXO.astype('int64')

X.VIVAX.replace({'Não':0, 'Sim': 1}, inplace=True)
X.VIVAX = X.VIVAX.astype('int64')

X.RACA.replace({'Parda':0, 'Indigena':1, 'Preta':2, 'Branca':3, 'Amarela':4}
X.RACA = X.RACA.astype('int64')

X.ID_PACIE = X.ID_PACIE.astype('int64')
X.head()
```

	SEXO	RACA	ID_PACIE	VIVAX
0	0	-1	75	-1
1	0	-1	18	-1
2	0	0	28	0
3	1	1	5	0
4	0	0	23	0

Construindo o modelo (treino/teste)

```
# Particionando os dados em dados em conjutos separados para treino e para t
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, ran

# Criando/Treinando o modelo
model = xgb.XGBClassifier()
model.fit(X_train, y_train)

  XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                colsample_bynode=1, colsample_bytree=1, gamma=0,
                learning_rate=0.1, max_delta_step=0, max_depth=3,
                min_child_weight=1, missing=None, n_estimators=100,
  n_jobs=1,
                nthread=None, objective='binary:logistic', random_state=0,
                reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                silent=None, subsample=1, verbosity=1)
```

```
# "Testando" o modelo, fazendo predições
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
```

Avaliação dos Resultados

Acurácia

Proximidade entre o valor obtido experimentalmente e o valor verdadeiro na medição de uma grandeza física.

Acurácia = Acertos/Total

```
# Avaliando o Modelo
accuracy = accuracy_score(y_test, predictions)
print("Acurácia do modelo: %.2f%%" % (accuracy * 100.0))
```

```
Acurácia do modelo: 82.01%
```

Matriz de Confusão

True Negative | False Positive

False Negatice | True Positive

```
confusion_matrix(y_test, predictions)
```

```
array([[  497, 14761],
       [   53, 67047]])
```

De um total de 16.797 indivíduos que deveriam ser classificado como outros tratamentos, o modelo acertou apenas 541.

Entretanto, de 73.796 indivíduos que deveriam ser classificados como tratado com cloroquina, o modelo errou apenas 57.

Produtos pagos do Colab - Cancelar contratos