Análise dataset filtrado

In [1]:

```
# Imports

import os
import subprocess
import stat
import numpy as np
from numpy.random import randn
import pandas as pd
from pandas import Series, DataFrame
import seaborn as sns
#sns.set(style='white')
import matplotlib.pyplot as plt

%matplotlib inline

import datetime
from datetime import datetime
from datetime import time
from datetime import date
```

In [2]:

```
testedf = pd.read_excel('Plancopy.xlsx')
testedf.head()
```

Out[2]:

	id	qt_hit	diasnav	notlidas	visita_capa	usou_app	perfil	genero	dt_nasc	rei
0	3321	0	0	0	0	NAO	ASSINANTE	F	23.04.1981 00:00:00	4 / 8
1	1459	1	23	0	362	SIM	ASSINANTE	M	01.01.1900 00:00:00	3 / 4
2	1630	5	16	11	4	NAO	ASSINANTE	M	01.01.1900 00:00:00	pos
3	905	9	13	8	25	SIM	ASSINANTE	F	01.01.1900 00:00:00	ACI 25
4	1219	1	1	0	9	SIM	ASSINANTE	M	16.08.1977 00:00:00	4 / 8

In [3]:

```
testedf.dtypes
```

Out[3]:

```
id              int64
qt_hit          int64
diasnav         int64
notlidas        int64
visita_capa     int64
usou_app       object
perfil         object
genero         object
dt_nasc        object
renda          object
dtype: object
```

In [4]:

```
testedf['nasc'] = pd.to_datetime(testedf['dt_nasc'], errors='coerce')
```

In [5]:

```
testedf['idade'] = date.today().year - testedf['nasc'].dt.year
```

In [6]:

```
testedf.head(2)
```

Out[6]:

	id	qt_hit	diasnav	notlidas	visita_capa	usou_app	perfil	genero	dt_nasc	ren
0	3321	0	0	0	0	NAO	ASSINANTE	F	23.04.1981 00:00:00	[49 A 89
1	1459	1	23	0	362	SIM	ASSINANTE	M	01.01.1900 00:00:00	[39 A 49

In [7]:

```
df1 = testedf[['id', 'qt_hit', 'diasnav','notlidas', 'visita_capa', 'idade', 'genero', 'usc
```

In [8]:

```
df1.head(2)
```

Out[8]:

	id	qt_hit	diasnav	notlidas	visita_capa	idade	genero	usou_app	renda	perfil
0	3321	0	0	0	0	38.0	F	NAO	DE 4SM ATE 8SM	ASSINANTE
1	1459	1	23	0	362	119.0	M	SIM	DE 3SM ATE 4SM	ASSINANTE

In [9]:

```
df1.shape
```

Out[9]:

```
(5600, 10)
```

In [10]:

```
# Verificando se existem valores nulos
df1.isnull().values.any()
```

Out[10]:

```
True
```

In [11]:

```
df10 = df1
df10.shape
```

Out[11]:

```
(5600, 10)
```

In [12]:

```
df10[df10['idade'].isnull()]
```

Out[12]:

	id	qt_hit	diasnav	notlidas	visita_capa	idade	genero	usou_app	renda	pe
877	168369	0	33	104	0	NaN	F	NAO	DE 8SM ATE 14SM	PROSPE
1101	23046	0	3	8	0	NaN	F	NAO	não possui	PROSPE
1317	108761	0	56	328	389	NaN	M	SIM	DE 4SM ATE 8SM	PROSPE
1563	183562	0	1	0	5	NaN	M	SIM	DE 4SM ATE 8SM	PROSPE
2087	111528	0	1	0	0	NaN	M	NAO	não possui	PROSPE
3419	80176	0	24	7	0	NaN	F	NAO	DE 3SM ATE 4SM	PROSPE
4120	194220	0	37	178	62	NaN	M	NAO	não possui	PROSPE
4406	183390	0	4	3	0	NaN	M	NAO	DE 3SM ATE 4SM	PROSPE
4737	161324	0	20	41	79	NaN	M	SIM	DE 4SM ATE 8SM	PROSPE
4951	187037	8	12	20	0	NaN	M	NAO	DE 3SM ATE 4SM	PROSPE

In [13]:

```
df11 = df10.fillna({'idade': 0})
```

In [14]:

```
df11.isnull().values.any()
```

Out[14]:

```
False
```

In [15]:

```
df11.shape
```

Out[15]:

```
(5600, 10)
```

In [16]:

```
df11.head()
```

Out[16]:

	id	qt_hit	diasnav	notlidas	visita_capa	idade	genero	usou_app	renda	perfil
0	3321	0	0	0	0	38.0	F	NAO	DE 4SM ATE 8SM	ASSINANTE
1	1459	1	23	0	362	119.0	M	SIM	DE 3SM ATE 4SM	ASSINANTE
2	1630	5	16	11	4	119.0	M	NAO	não possui	ASSINANTE
3	905	9	13	8	25	119.0	F	SIM	ACIMA DE 25SM	ASSINANTE
4	1219	1	1	0	9	42.0	M	SIM	DE 4SM ATE 8SM	ASSINANTE

In [17]:

```
# Trannsformando a coluna perfil em booleano
p = {'ASSINANTE': True, 'PROSPECT': False}
g = {'M': True, 'F': False}
u = {'SIM': True, 'NAO': False}
```

In [18]:

```
df11['perfil'] = df11['perfil'].map(p)
df11['genero'] = df11['genero'].map(g)
df11['usou_app'] = df11['usou_app'].map(u)
```

In [19]:

```
df11.isnull().values.any()
```

Out[19]:

```
True
```

In [20]:

```
df11.isnull().sum()
```

Out[20]:

```
id                0
qt_hit            0
diasnav           0
notlidas          0
visita_capa       0
idade             0
genero          136
usou_app          0
renda             0
perfil         5000
dtype: int64
```

In [21]:

```
df11copy = df11.copy()
df12 = df11copy.fillna({
    'genero': False,
    'perfil': False
})
df12.head(2)
```

Out[21]:

	id	qt_hit	diasnav	notlidas	visita_capa	idade	genero	usou_app	renda	perfil
0	3321	0	0	0	0	38.0	False	False	DE 4SM ATE 8SM	True
1	1459	1	23	0	362	119.0	True	True	DE 3SM ATE 4SM	True

In [22]:

```
df12.shape
```

Out[22]:

```
(5600, 10)
```

In [23]:

```
df12.isnull().values.any()
```

Out[23]:

```
False
```

In [24]:

```
df12.head(2)
```

Out[24]:

	id	qt_hit	diasnav	notlidas	visita_capa	idade	genero	usou_app	renda	perfil
0	3321	0	0	0	0	38.0	False	False	DE 4SM ATE 8SM	True
1	1459	1	23	0	362	119.0	True	True	DE 3SM ATE 4SM	True

In [25]:

```
df12[df12['idade'] < 18]
```

Out[25]:

	id	qt_hit	diasnav	notlidas	visita_capa	idade	genero	usou_app	renda	perfil
288	3390	0	0	0	0	12.0	True	False	não possui	True
877	168369	0	33	104	0	0.0	False	False	DE 8SM ATE 14SM	False
1101	23046	0	3	8	0	0.0	False	False	não possui	False
1317	108761	0	56	328	389	0.0	True	True	DE 4SM ATE 8SM	False
1563	183562	0	1	0	5	0.0	True	True	DE 4SM ATE 8SM	False
1959	115944	20	41	36	132	8.0	True	True	não possui	False
2087	111528	0	1	0	0	0.0	True	False	não possui	False
3005	224318	0	19	23	5	15.0	False	True	DE 3SM ATE 4SM	False
3419	80176	0	24	7	0	0.0	False	False	DE 3SM ATE 4SM	False
4120	194220	0	37	178	62	0.0	True	False	não possui	False
4406	183390	0	4	3	0	0.0	True	False	DE 3SM ATE 4SM	False
4737	161324	0	20	41	79	0.0	True	True	DE 4SM ATE 8SM	False
4951	187037	8	12	20	0	0.0	True	False	DE 3SM ATE 4SM	False
5366	286385	0	12	7	0	14.0	True	False	não possui	False
5532	272057	0	3	3	0	-19.0	False	False	DE 14SM ATE 25SM	False

In [26]:

```
df12[df12['idade'] > 100].count()
```

Out[26]:

```
id             556
qt_hit         556
diasnav        556
notlidas       556
visita_capa    556
idade          556
genero         556
usou_app       556
renda          556
perfil         556
dtype: int64
```

In [27]:

```
df13 = df12[(df12['idade'] > 18) & (df12['idade'] < 100)]
```

In [28]:

```
df13.shape
```

Out[28]:

```
(5028, 10)
```

In [29]:

```
len(df13.loc[df13['perfil'] == True])
```

Out[29]:

```
88
```

In [30]:

```
len(df13.loc[df13['perfil'] == False])
```

Out[30]:

```
4940
```

Verificando a correlação entre os atributos

In [31]:

```
## Dataset filtrado
```

In [32]:

```
# Lembrar de verificar o df13 com o range de idades corretas
```

In [33]:

```
# Visualizando a correlação em tabela
# Coeficiente de correlação:
# +1  = forte correlação positiva
# 0   = não há correlação
# -1  = forte correlação negativa
df13.corr()
```

Out[33]:

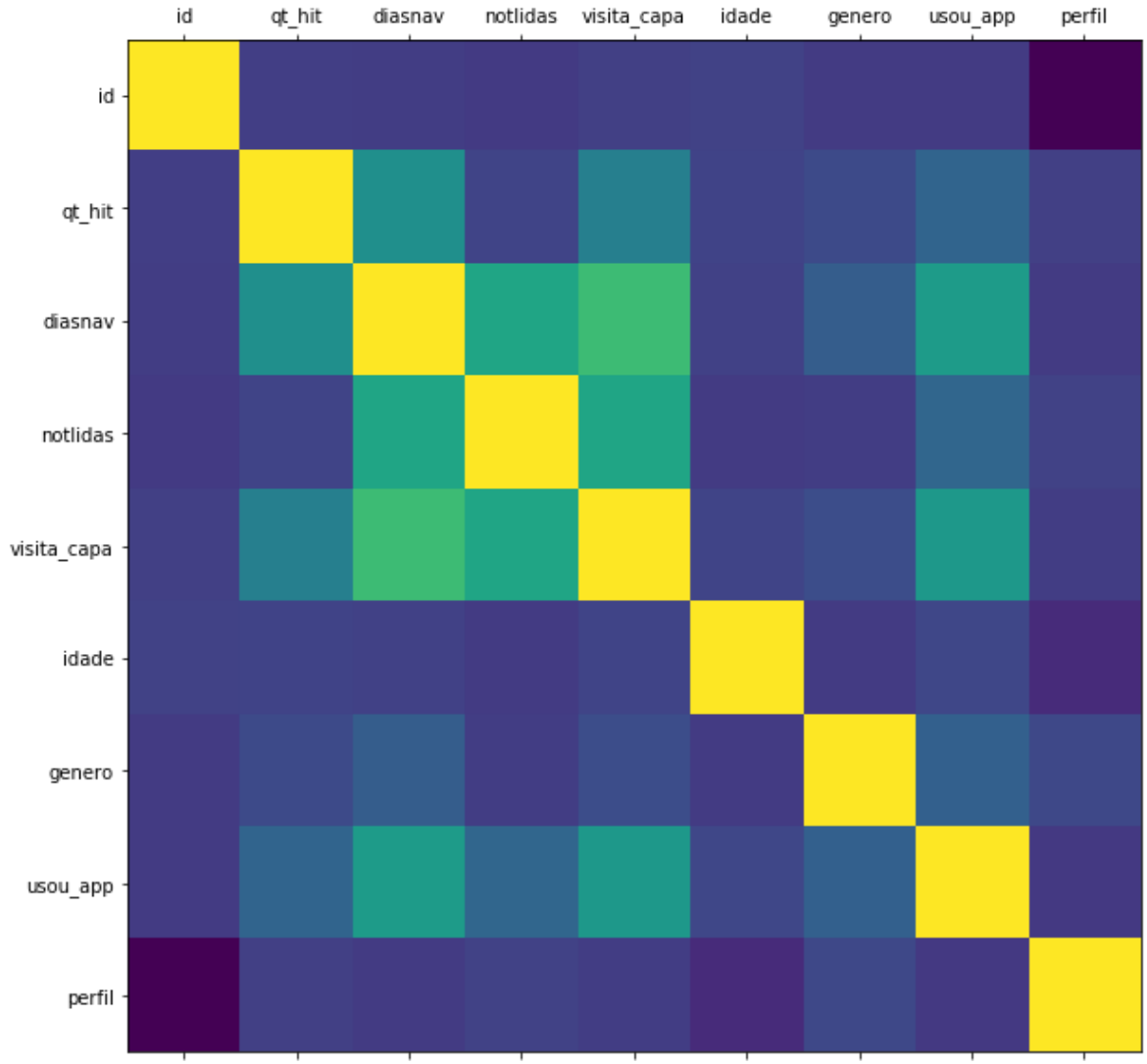
	id	qt_hit	diasnav	notlidas	visita_capa	idade	genero	usou_a
id	1.000000	0.004051	0.001443	-0.014344	0.008686	0.021035	-0.010445	-0.006
qt_hit	0.004051	1.000000	0.388475	0.030716	0.304910	0.024269	0.054554	0.174
diasnav	0.001443	0.388475	1.000000	0.498112	0.617039	0.016166	0.138345	0.451
notlidas	-0.014344	0.030716	0.498112	1.000000	0.498644	-0.009484	0.002881	0.184
visita_capa	0.008686	0.304910	0.617039	0.498644	1.000000	0.030587	0.068448	0.431
idade	0.021035	0.024269	0.016166	-0.009484	0.030587	1.000000	-0.006430	0.037
genero	-0.010445	0.054554	0.138345	0.002881	0.068448	-0.006430	1.000000	0.152
usou_app	-0.006107	0.174427	0.451407	0.184075	0.431478	0.037767	0.152975	1.000
perfil	-0.220215	0.012570	-0.007940	0.020045	0.001234	-0.069708	0.046431	-0.016

In [34]:

```
# Identificando a correlação entre as variáveis
# Correlação não implica causalidade
def plot_corr(df13, size=10):
    corr = df13.corr()
    fig, ax = plt.subplots(figsize = (size, size))
    ax.matshow(corr)
    plt.xticks(range(len(corr.columns)), corr.columns)
    plt.yticks(range(len(corr.columns)), corr.columns)
```

In [35]:

```
# Criando o gráfico
plot_corr(df13)
```



In [36]:

```
# Definindo as classes
perf_map = {True : 1, False : 0}
usou_app_map = {True : 1, False : 0}
genero_map = {True : 1, False : 0}
```

In [37]:

```
# Aplicando o mapeamento ao dataset
df13['perfil'] = df13['perfil'].map(perf_map)
df13['usou_app'] = df13['usou_app'].map(usou_app_map)
df13['genero'] = df13['genero'].map(genero_map)
```

```
C:\Users\Resende\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)

C:\Users\Resende\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  This is separate from the ipykernel package so we can avoid doing imports
 until
C:\Users\Resende\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  after removing the cwd from sys.path.
```

In [38]:

```
# Verificando como os dados estão distribuídos
num13_true = len(df13.loc[df13['perfil'] == True])
num13_false = len(df13.loc[df13['perfil'] == False])
print("Número de assinantes: {0} ({1:2.2f}%)".format(num13_true, (num13_true/ (num13_true +
print("Número de não assinantes: {0} ({1:2.2f}%)".format(num13_false, (num13_false/ (num13_
```

```
Número de assinantes: 88 (1.75%)
Número de não assinantes: 4940 (98.25%)
```

Spliting (dataset filtrado)

In [39]:

```
#Achoq que deve ser 98% para treino e 2% para teste
```

In [40]:

```
from sklearn.model_selection import train_test_split
```

In [41]:

```
# Seleção de variáveis preditoras (Feature Selection)
atributos13 = ['qt_hit', 'diasnav', 'notlidas', 'visita_capa', 'idade', 'genero', 'usou_app
```

In [42]:

```
# Variável a ser prevista
atrib_prev13 = ['perfil']
```

In [43]:

```
# Criando objetos
X13 = df13[atributos13].values
Y13 = df13[atrib_prev13].values
```

In [44]:

```
X13
```

Out[44]:

```
array([[ 0.,  0.,  0., ...,  0.,  0.,  1.],
       [ 1.,  1.,  0., ...,  1.,  1.,  1.],
       [ 1.,  6.,  3., ...,  1.,  0.,  1.],
       ...,
       [ 0.,  1.,  1., ...,  0.,  0.,  0.],
       [ 0.,  1.,  1., ...,  0.,  0.,  0.],
       [25., 18., 34., ...,  0.,  0.,  0.]])
```

In [45]:

```
Y13
```

Out[45]:

```
array([[1],
       [1],
       [1],
       ...,
       [0],
       [0],
       [0]], dtype=int64)
```

In [46]:

```
# Definindo a taxa de split
split_test_size = 0.02
```

In [47]:

```
# Criando dados de treino e de teste
X_treino13, X_teste13, Y_treino13, Y_teste13 = train_test_split(X13, Y13, test_size = split
```

In [48]:

```
# Imprimindo os resultados
print("{0:0.2f}% nos dados de treino".format((len(X_treino13)/len(df13.index)) * 100))
print("{0:0.2f}% nos dados de teste".format((len(X_teste13)/len(df13.index)) * 100))
```

```
97.99% nos dados de treino
2.01% nos dados de teste
```

In [49]:

```
X_treino13
```

Out[49]:

```
array([[ 0.,  7.,  2., ...,  1.,  0.,  0.],
       [ 1., 10.,  7., ...,  1.,  0.,  0.],
       [ 0.,  1.,  0., ...,  1.,  0.,  0.],
       ...,
       [ 0.,  3.,  1., ...,  1.,  0.,  0.],
       [ 1.,  3.,  3., ...,  0.,  0.,  0.],
       [ 1., 45., 50., ...,  1.,  1.,  0.]])
```

In [50]:

```
print("Original True : {0} ({1:0.2f}%)".format(len(df13.loc[df13['perfil'] == 1]),
                                               (len(df13.loc[df13['perfil'] ==1])/len(df13.

print("Original False : {0} ({1:0.2f}%)".format(len(df13.loc[df13['perfil'] == 0]),
                                               (len(df13.loc[df13['perfil'] == 0])/len(df13
print("")
print("Training True : {0} ({1:0.2f}%)".format(len(Y_treino13[Y_treino13[:] == 1]),
                                               (len(Y_treino13[Y_treino13[:] == 1])/len(Y_t

print("Training False : {0} ({1:0.2f}%)".format(len(Y_treino13[Y_treino13[:] == 0]),
                                               (len(Y_treino13[Y_treino13[:] == 0])/len(Y_t
print("")
print("Test True : {0} ({1:0.2f}%)".format(len(Y_teste13[Y_teste13[:] == 1]),
                                               (len(Y_teste13[Y_teste13[:] == 1])/len(Y_tes

print("Test False : {0} ({1:0.2f}%)".format(len(Y_teste13[Y_teste13[:] == 0]),
                                               (len(Y_teste13[Y_teste13[:] == 0])/len(Y_tes
```

```
Original True : 88 (1.75%)
Original False : 4940 (98.25%)

Training True : 83 (1.68%)
Training False : 4844 (98.32%)

Test True : 5 (4.95%)
Test False : 96 (95.05%)
```

In [51]:

```
# Valores missing ou ocultosnão serão tratados
```

Construindo e treinando o modelo

In [52]:

```
# Utilizando um classificador Naive Bayes
from sklearn.naive_bayes import GaussianNB
```

In [53]:

```
# Criando o modelo preditivo
modelo_v13 = GaussianNB()
```

In [54]:

```
# Treinando o modelo
modelo_v13.fit(X_treino13, Y_treino13.ravel())
```

Out[54]:

```
GaussianNB(priors=None)
```

Verificando a exatidão no modelo nos dados de treino

In [55]:

```
from sklearn import metrics
```

In [56]:

```
nb_predict_train = modelo_v13.predict(X_treino13)
```

In [57]:

```
print("Exatidão (Accuracy): {0:.4f}".format(metrics.accuracy_score(Y_treino13, nb_predict_t
print()
```

```
Exatidão (Accuracy): 1.0000
```

Verificando a exatidão no modelo nos dados de teste

In [58]:

```
nb_predict_test = modelo_v13.predict(X_teste13)
```

In [59]:

```
print("Exatidão (Accuracy): {0:.4f}".format(metrics.accuracy_score(Y_teste13, nb_predict_te
print()
```

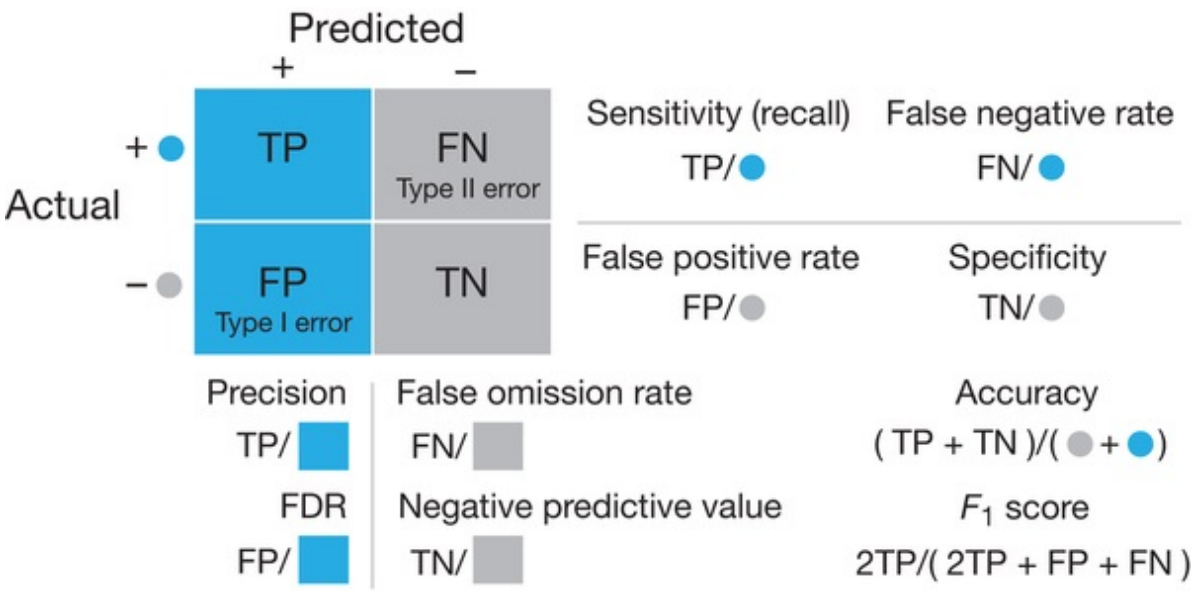
```
Exatidão (Accuracy): 1.0000
```

Métricas

In [60]:

```
from IPython.display import Image
Image('ConfusionMatrix.jpg')
```

Out[60]:



In [61]:

```
# Criando uma Confusion Matrix
print("Confusion Matrix")

print("{0}".format(metrics.confusion_matrix(Y_teste13, nb_predict_test, labels = [1, 0])))
print("")

print("Classification Report")
print(metrics.classification_report(Y_teste13, nb_predict_test, labels = [1, 0]))
```

```
Confusion Matrix
[[ 5  0]
 [ 0 96]]

Classification Report
             precision    recall  f1-score   support

          1       1.00      1.00      1.00         5
          0       1.00      1.00      1.00        96

avg / total       1.00      1.00      1.00       101
```

In [ ]:

```

```

Testando outro algoritmo

In [62]:

```
# conjunto de árvorede decisão = random forest
```

Otimizando o modelo com RandomForest

In [63]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [64]:

```
modelo_v213 = RandomForestClassifier(random_state = 42)
modelo_v213.fit(X_treino13, Y_treino13.ravel())
```

Out[64]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=42, verbose=0, warm_start=False)
```

In [65]:

```
# Verificando os dados de treino
rf_predict_train = modelo_v213.predict(X_treino13)
print("Exatidão (Accuracy): {0:.4f}".format(metrics.accuracy_score(Y_treino13, rf_predict_t
```

```
Exatidão (Accuracy): 1.0000
```

In [66]:

```
# Verificando nos dados de teste
rf_predict_test = modelo_v213.predict(X_teste13)
print("Exatidão (Accuracy): {0:.4f}".format(metrics.accuracy_score(Y_teste13, rf_predict_te
print()
```

```
Exatidão (Accuracy): 1.0000
```

In [67]:

```
print("Confusion Matrix")

print("{0}".format(metrics.confusion_matrix(Y_teste13, rf_predict_test, labels = [1, 0])))
print("")

print("Classification Report")
print(metrics.classification_report(Y_teste13, rf_predict_test, labels = [1, 0]))
```

```
Confusion Matrix
[[ 5  0]
 [ 0 96]]

Classification Report
             precision    recall  f1-score   support

          1       1.00      1.00      1.00         5
          0       1.00      1.00      1.00        96

avg / total       1.00      1.00      1.00       101
```

Testando outro algoritmo

In [68]:

```
# Regressão logística
# é um algoritmo de classificação diferente de regressão linear simples
```

Regressão logística

In [69]:

```
from sklearn.linear_model import LogisticRegression
```

In [70]:

```
# Terceira versão do modelo usando Regressão Logística
modelo_v313 = LogisticRegression(C = 0.7, random_state = 42)
modelo_v313.fit(X_treino13, Y_treino13.ravel())
lr_predict_test = modelo_v313.predict(X_teste13)
```

In [71]:

```
print("Exatidão (Accuracy): {0:.4f}".format(metrics.accuracy_score(Y_teste13, lr_predict_te
print()
print("Classification Report")
print(metrics.classification_report(Y_teste13, lr_predict_test, labels = [1, 0]))
```

```
Exatidão (Accuracy): 1.0000

Classification Report
             precision    recall  f1-score   support

          1       1.00      1.00      1.00         5
          0       1.00      1.00      1.00        96

avg / total       1.00      1.00      1.00       101
```

In [72]:

```
### Resumindo
## Exatidão nos dados de teste

# Modelo usando algoritmo Naive Bayes         = 0.93
# Modelo usando algoritmo Random Forest       = 0.94
# Modelo usando algoritmo Regressão Logística = 0.94
```

In [73]:

```
## Escolheu-se a regressão logística para realizar previsões
```

Fazendo previsões com o modelo treinado

In [74]:

```
import pickle
```

In [75]:

```
# Salvando o modelo para usar mais tarde
filename = 'modelo_treinado_v3.sav'
pickle.dump(modelo_v313, open(filename, 'wb'))
```

In [76]:

```
X_teste13
```

Out[76]:

```
array([[  1.,   8.,   0.,  42.,  49.,   1.,   1.,   0.],
       [  0.,   2.,   2.,   0.,  49.,   0.,   0.,   0.],
       [  0.,   8.,   7.,   0.,  49.,   0.,   0.,   0.],
       [ 14.,   8.,  10.,   0.,  43.,   0.,   0.,   0.],
       [  0.,   1.,   1.,   0.,  49.,   0.,   0.,   0.],
       [  0.,   1.,   1.,   0.,  49.,   1.,   0.,   0.],
       [  0.,   5.,   7.,   0.,  33.,   0.,   0.,   0.],
       [  0.,   7.,   7.,   0.,  36.,   0.,   0.,   0.],
       [  2.,   4.,   0.,  11.,  49.,   1.,   1.,   0.],
       [  0.,   1.,   1.,   0.,  49.,   0.,   0.,   0.],
       [  0.,   6.,   4.,   0.,  49.,   0.,   0.,   0.],
       [  0.,   5.,   1.,   0.,  66.,   1.,   0.,   0.],
       [  9.,   0.,   0.,   0.,  52.,   1.,   0.,   1.],
       [  0.,   3.,   2.,   0.,  49.,   1.,   0.,   0.],
       [  6.,  47.,   0., 188.,  49.,   0.,   1.,   0.],
       [  9.,  58., 191.,   3.,  31.,   0.,   0.,   1.],
       [  0.,   1.,   1.,   0.,  49.,   1.,   0.,   0.],
       [  1.,  16.,  21.,   0.,  49.,   1.,   0.,   0.],
       [  0.,   9.,   5.,  10.,  49.,   0.,   1.,   0.],
       [ 24.,  15.,  22.,   0.,  49.,   1.,   0.,   0.],
       [  6.,   2.,   0.,   1.,  37.,   0.,   0.,   0.],
       [  0.,   6.,   5.,   1.,  37.,   0.,   0.,   0.],
       [  0.,   6.,   2.,   0.,  49.,   1.,   0.,   0.],
       [ 14.,  56.,   0., 258.,  49.,   1.,   1.,   0.],
       [  0.,   1.,   1.,   0.,  49.,   0.,   0.,   0.],
       [  6.,   1.,   3.,   0.,  42.,   0.,   0.,   0.],
       [  0.,   2.,   1.,   0.,  72.,   0.,   0.,   0.],
       [  0.,   2.,   2.,   0.,  29.,   1.,   0.,   0.],
       [ 20.,  15.,   7.,   0.,  49.,   1.,   0.,   0.],
       [  0.,   2.,   3.,   0.,  49.,   1.,   0.,   0.],
       [  0.,  18.,  37.,   0.,  38.,   1.,   0.,   0.],
       [172.,  58.,  81.,   0.,  49.,   1.,   0.,   1.],
       [  0.,   6.,   4.,   0.,  49.,   0.,   0.,   0.],
       [  1.,   9.,  14.,   0.,  49.,   0.,   0.,   0.],
       [  0.,   7.,   4.,   1.,  49.,   1.,   0.,   0.],
       [  0.,   2.,   4.,   0.,  49.,   1.,   0.,   0.],
       [ 10.,  12.,   9.,  10.,  29.,   1.,   0.,   0.],
       [  0.,   3.,   1.,   0.,  49.,   1.,   0.,   0.],
       [  0.,  10.,  10.,   0.,  31.,   0.,   0.,   0.],
       [  0.,   1.,   7.,   0.,  49.,   0.,   0.,   0.],
       [  0.,   5.,   3.,   1.,  49.,   1.,   0.,   0.],
       [  0.,   5.,   4.,   0.,  49.,   0.,   0.,   0.],
       [  0.,   2.,   2.,   0.,  49.,   1.,   0.,   0.],
       [  0.,   1.,   0.,   0.,  49.,   1.,   0.,   0.],
       [ 11.,  20.,  30.,   0.,  49.,   1.,   0.,   0.],
       [  0.,  14.,  23.,  34.,  49.,   1.,   0.,   0.],
       [  1.,   1.,   0.,   0.,  49.,   1.,   0.,   0.],
       [  0.,   1.,   0.,   0.,  49.,   1.,   0.,   0.],
       [ 14.,  19.,   0.,  94.,  49.,   1.,   1.,   0.],
       [  0.,   0.,   0.,   0.,  40.,   1.,   0.,   1.],
       [  0.,   3.,   4.,   0.,  49.,   0.,   0.,   0.],
       [  0.,   1.,   2.,   0.,  49.,   1.,   0.,   0.],
       [  0.,  23.,  30.,  52.,  49.,   0.,   1.,   0.],
       [  0.,   2.,   2.,   0.,  49.,   0.,   0.,   0.],
       [  0.,   2.,   0.,   0.,  49.,   1.,   0.,   0.],
```

```
       [ 10.,  19.,  13.,   8.,  49.,   1.,   0.,   0.],
       [  0.,   2.,   0.,   0.,  49.,   0.,   0.,   0.],
       [ 80.,  34.,  12.,   0.,  45.,   0.,   0.,   0.],
       [ 25.,  23.,  12.,   0.,  49.,   1.,   0.,   0.],
       [ 22.,  19.,  12.,   0.,  49.,   1.,   0.,   0.],
       [  0.,   2.,   0.,   0.,  49.,   1.,   0.,   0.],
       [  0.,  43.,  37.,  88.,  49.,   1.,   0.,   0.],
       [  0.,  26.,  54.,   0.,  49.,   1.,   0.,   0.],
       [  0.,   2.,   2.,   0.,  49.,   0.,   0.,   0.],
       [ 12.,  21.,  13.,   5.,  49.,   0.,   0.,   0.],
       [  2.,  27.,  17.,  42.,  49.,   1.,   1.,   0.],
       [  1.,   7.,   4.,   6.,  49.,   1.,   0.,   0.],
       [  0.,   3.,   4.,   4.,  49.,   1.,   0.,   0.],
       [  0.,   7.,   3.,   0.,  49.,   0.,   0.,   0.],
       [  1.,   7.,  17.,   0.,  49.,   0.,   0.,   0.],
       [  0.,   1.,   6.,   0.,  47.,   0.,   0.,   0.],
       [  0.,   1.,   1.,   0.,  49.,   0.,   0.,   0.],
       [ 65.,  43.,   0., 416.,  49.,   0.,   1.,   0.],
       [  0.,  12.,   5.,  13.,  49.,   1.,   0.,   0.],
       [  0.,   5.,   4.,   0.,  49.,   0.,   0.,   0.],
       [  2.,  15.,  13.,   0.,  42.,   1.,   0.,   0.],
       [ 10.,  27.,  17.,  34.,  49.,   1.,   0.,   0.],
       [  0.,   1.,   1.,   0.,  49.,   1.,   0.,   0.],
       [  4.,   8.,   0.,  13.,  39.,   1.,   1.,   0.],
       [  0.,   3.,   3.,   0.,  49.,   1.,   0.,   0.],
       [  0.,   2.,   1.,   0.,  49.,   1.,   0.,   0.],
       [ 48.,  40.,   7., 179.,  49.,   0.,   1.,   0.],
       [  1.,   2.,   3.,   0.,  49.,   1.,   0.,   0.],
       [  1.,   4.,   9.,   0.,  49.,   0.,   0.,   0.],
       [  0.,  24.,  39.,   0.,  49.,   1.,   0.,   0.],
       [  0.,   1.,   1.,   0.,  49.,   1.,   0.,   0.],
       [  0.,   2.,   3.,   0.,  49.,   1.,   0.,   0.],
       [  3.,   4.,   3.,   0.,  43.,   1.,   0.,   1.],
       [  3.,  16.,   0.,  19.,  49.,   1.,   1.,   0.],
       [ 10.,  10.,   0.,   6.,  49.,   1.,   1.,   0.],
       [  0.,  10.,   4.,   0.,  49.,   0.,   0.,   0.],
       [  0.,  10.,  11.,   0.,  49.,   0.,   0.,   0.],
       [  0.,   1.,   1.,   0.,  49.,   1.,   0.,   0.],
       [  0.,   1.,   0.,   0.,  49.,   1.,   0.,   0.],
       [  0.,   1.,   0.,   1.,  49.,   0.,   0.,   0.],
       [  0.,   2.,   0.,   0.,  49.,   1.,   0.,   0.],
       [  0.,  11.,  12.,   0.,  29.,   1.,   0.,   0.],
       [  2.,   5.,   5.,   1.,  49.,   1.,   0.,   0.],
       [  0.,   2.,   2.,   0.,  49.,   0.,   0.,   0.],
       [  0.,   8.,   4.,   1.,  49.,   0.,   0.,   0.],
       [  0.,   1.,   1.,   0.,  35.,   0.,   0.,   0.]])
```

In [80]:

```
len(X_teste13)
```

Out[80]:

```
101
```

In [81]:

```
# Carregando o modelo e fazendo previsão com novos conjuntos de dados
# (X_teste, Y_teste devem ser novos conjuntos de dados preparados com o procedimento de lim
loaded_model = pickle.load(open(filename, 'rb'))
resultado1 = loaded_model.predict(X_teste13[25].reshape(1, -1))
resultado2 = loaded_model.predict(X_teste13[30].reshape(1, -1))
resultado3 = loaded_model.predict(X_teste13[35].reshape(1, -1))
resultado4 = loaded_model.predict(X_teste13[40].reshape(1, -1))
resultado5 = loaded_model.predict(X_teste13[45].reshape(1, -1))
resultado6 = loaded_model.predict(X_teste13[55].reshape(1, -1))
resultado7 = loaded_model.predict(X_teste13[60].reshape(1, -1))
resultado8 = loaded_model.predict(X_teste13[65].reshape(1, -1))
resultado9 = loaded_model.predict(X_teste13[75].reshape(1, -1))
resultado10 = loaded_model.predict(X_teste13[85].reshape(1, -1))
print(resultado1)
print(resultado2)
print(resultado3)
print(resultado4)
print(resultado5)
print(resultado6)
print(resultado7)
print(resultado8)
print(resultado9)
print(resultado10)
```

```
[0]
[0]
[0]
[0]
[0]
[0]
[0]
[0]
[0]
[0]
```

In [82]:

```
# Carregando o modelo e fazendo previsão com novos conjuntos de dados
# (X_teste, Y_teste devem ser novos conjuntos de dados preparados com o procedimento de lim
loaded_model = pickle.load(open(filename, 'rb'))
resultado11 = loaded_model.predict(X_teste13[10].reshape(1, -1))
resultado12 = loaded_model.predict(X_teste13[15].reshape(1, -1))
resultado13 = loaded_model.predict(X_teste13[20].reshape(1, -1))
resultado14 = loaded_model.predict(X_teste13[29].reshape(1, -1))
resultado15 = loaded_model.predict(X_teste13[39].reshape(1, -1))
resultado16 = loaded_model.predict(X_teste13[49].reshape(1, -1))
resultado17 = loaded_model.predict(X_teste13[59].reshape(1, -1))
resultado18 = loaded_model.predict(X_teste13[80].reshape(1, -1))
resultado19 = loaded_model.predict(X_teste13[90].reshape(1, -1))
resultado20 = loaded_model.predict(X_teste13[95].reshape(1, -1))
print(resultado11)
print(resultado12)
print(resultado13)
print(resultado14)
print(resultado15)
print(resultado16)
print(resultado17)
print(resultado18)
print(resultado19)
print(resultado20)
```

```
[0]
[1]
[0]
[0]
[0]
[1]
[0]
[0]
[0]
[0]
```

In [86]:

```
print(X_teste13[10]) # Não assinante
```

```
[ 0.  6.  4.  0. 49.  0.  0.  0.]
```

In [87]:

```
print(X_teste13[15]) # Assinante
```

```
[  9.  58. 191.   3.  31.   0.   0.   1.]
```

In [88]:

```
print(X_teste13[49]) # Assinante
```

```
[ 0.  0.  0.  0. 40.  1.  0.  1.]
```

In [83]:

```
# Carregando o modelo e fazendo previsão com novos conjuntos de dados
# (X_teste, Y_teste devem ser novos conjuntos de dados preparados com o procedimento de lim
loaded_model = pickle.load(open(filename, 'rb'))
resultado21 = loaded_model.predict(X_teste13[91].reshape(1, -1))
resultado22 = loaded_model.predict(X_teste13[92].reshape(1, -1))
resultado23 = loaded_model.predict(X_teste13[93].reshape(1, -1))
resultado24 = loaded_model.predict(X_teste13[94].reshape(1, -1))
resultado25 = loaded_model.predict(X_teste13[95].reshape(1, -1))
resultado26 = loaded_model.predict(X_teste13[96].reshape(1, -1))
resultado27 = loaded_model.predict(X_teste13[97].reshape(1, -1))
resultado28 = loaded_model.predict(X_teste13[98].reshape(1, -1))
resultado29 = loaded_model.predict(X_teste13[99].reshape(1, -1))
resultado30 = loaded_model.predict(X_teste13[100].reshape(1, -1))
print(resultado21)
print(resultado22)
print(resultado23)
print(resultado24)
print(resultado25)
print(resultado26)
print(resultado27)
print(resultado28)
print(resultado29)
print(resultado30)
```

```
[0]
[0]
[0]
[0]
[0]
[0]
[0]
[0]
[0]
[0]
```

In [89]:

```
print(X_teste13[91]) # Não assinante
```

```
[ 0. 10. 11.  0. 49.  0.  0.  0.]
```

In [90]:

```
print(X_teste13[97]) # Não assinante
```

```
[ 2.  5.  5.  1. 49.  1.  0.  0.]
```

In [84]:

```
# Carregando o modelo e fazendo previsão com novos conjuntos de dados
# (X_teste, Y_teste devem ser novos conjuntos de dados preparados com o procedimento de lim
loaded_model = pickle.load(open(filename, 'rb'))
resultado31 = loaded_model.predict(X_teste13[1].reshape(1, -1))
resultado32 = loaded_model.predict(X_teste13[2].reshape(1, -1))
resultado33 = loaded_model.predict(X_teste13[3].reshape(1, -1))
resultado34 = loaded_model.predict(X_teste13[4].reshape(1, -1))
resultado35 = loaded_model.predict(X_teste13[5].reshape(1, -1))
resultado36 = loaded_model.predict(X_teste13[6].reshape(1, -1))
resultado37 = loaded_model.predict(X_teste13[7].reshape(1, -1))
resultado38 = loaded_model.predict(X_teste13[8].reshape(1, -1))
resultado39 = loaded_model.predict(X_teste13[8].reshape(1, -1))
resultado40 = loaded_model.predict(X_teste13[10].reshape(1, -1))
print(resultado31)
print(resultado32)
print(resultado33)
print(resultado34)
print(resultado35)
print(resultado36)
print(resultado37)
print(resultado38)
print(resultado39)
print(resultado40)
```

```
[0]
[0]
[0]
[0]
[0]
[0]
[0]
[0]
[0]
[0]
```

In [91]:

```
print(X_teste13[3]) # Não assinante
```

```
[14.  8. 10.  0. 43.  0.  0.  0.]
```

In [92]:

```
print(X_teste13[6]) # Não assinante
```

```
[ 0.  5.  7.  0. 33.  0.  0.  0.]
```

In [93]:

```
print(X_teste13[8]) # Não assinante
```

```
[ 2.  4.  0. 11. 49.  1.  1.  0.]
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```