Projeto_03_v2

November 19, 2019

1 Formação Cientista de Dados - DSA

1.0.1 Big Data Real-Time Analytics com Python e Spark

1.1 Projeto com Feedback 3 - Prevendo o Nível de Satisfação dos Clientes do Santander

https://www.kaggle.com/c/santander-customer-satisfaction

1.1.1 Leonardo Molero

2 Análise Exploratória

[1]:
```
# Importação pacotes iniciais
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

[2]:
```
# Faz ajustes para não exibir warnings
warnings.filterwarnings("ignore")

# Parametriza impressão dos gráficos dentro do notebook
%matplotlib inline
```

[3]:
```
# Carrega o dados de treino colocando a coluna ID como index
df = pd.read_csv('dados/train.csv',index_col='ID')
```

[4]:
```
# Checa o tamanho do dataframe
print(df.shape)
```

```
(76020, 370)
```

[5]:
```
# Visualiza os dados treino
df.head(10)
```

[5]:

ID	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	\
1	2	23	0.0	0.0	
3	2	34	0.0	0.0	
4	2	23	0.0	0.0	
8	2	37	0.0	195.0	
10	2	39	0.0	0.0	
13	2	23	0.0	0.0	
14	2	27	0.0	0.0	
18	2	26	0.0	0.0	
20	2	45	0.0	0.0	
23	2	25	0.0	0.0	

ID	imp_op_var39_comer_ult3	imp_op_var40_comer_ult1	imp_op_var40_comer_ult3	\
1	0.0	0.0	0.0	
3	0.0	0.0	0.0	
4	0.0	0.0	0.0	
8	195.0	0.0	0.0	
10	0.0	0.0	0.0	
13	0.0	0.0	0.0	
14	0.0	0.0	0.0	
18	0.0	0.0	0.0	
20	0.0	0.0	0.0	
23	0.0	0.0	0.0	

ID	imp_op_var40_efect_ult1	imp_op_var40_efect_ult3	imp_op_var40_ult1	…	\
				…	
1	0.0	0.0	0.0	…	
3	0.0	0.0	0.0	…	
4	0.0	0.0	0.0	…	
8	0.0	0.0	0.0	…	
10	0.0	0.0	0.0	…	
13	0.0	0.0	0.0	…	
14	0.0	0.0	0.0	…	
18	0.0	0.0	0.0	…	
20	0.0	0.0	0.0	…	
23	0.0	0.0	0.0	…	

ID	saldo_medio_var33_hace2	saldo_medio_var33_hace3	saldo_medio_var33_ult1	\
1	0.0	0.0	0.0	
3	0.0	0.0	0.0	
4	0.0	0.0	0.0	
8	0.0	0.0	0.0	
10	0.0	0.0	0.0	
13	0.0	0.0	0.0	

```
14                     0.0                     0.0                     0.0
18                     0.0                     0.0                     0.0
20                     0.0                     0.0                     0.0
23                     0.0                     0.0                     0.0

    saldo_medio_var33_ult3  saldo_medio_var44_hace2  saldo_medio_var44_hace3  \
ID
1                      0.0                      0.0                      0.0
3                      0.0                      0.0                      0.0
4                      0.0                      0.0                      0.0
8                      0.0                      0.0                      0.0
10                     0.0                      0.0                      0.0
13                     0.0                      0.0                      0.0
14                     0.0                      0.0                      0.0
18                     0.0                      0.0                      0.0
20                     0.0                      0.0                      0.0
23                     0.0                      0.0                      0.0

    saldo_medio_var44_ult1  saldo_medio_var44_ult3          var38  TARGET
ID
1                      0.0                     0.0   39205.170000       0
3                      0.0                     0.0   49278.030000       0
4                      0.0                     0.0   67333.770000       0
8                      0.0                     0.0   64007.970000       0
10                     0.0                     0.0  117310.979016       0
13                     0.0                     0.0   87975.750000       0
14                     0.0                     0.0   94956.660000       0
18                     0.0                     0.0  251638.950000       0
20                     0.0                     0.0  101962.020000       0
23                     0.0                     0.0  356463.060000       0

[10 rows x 370 columns]
```

[6]:
```
# Verifica os tipos das colunas
df.dtypes
```

[6]:
```
var3                        int64
var15                       int64
imp_ent_var16_ult1        float64
imp_op_var39_comer_ult1   float64
imp_op_var39_comer_ult3   float64
                           ...
saldo_medio_var44_hace3   float64
saldo_medio_var44_ult1    float64
saldo_medio_var44_ult3    float64
var38                     float64
TARGET                      int64
```

```
Length: 370, dtype: object
```

[7]:
```
# Verifica os tipos de colunas agrupados (devido a quantidade de colunas)
df.dtypes.value_counts()
```

[7]:
```
int64      259
float64    111
dtype: int64
```

[8]:
```
# Verifica estatísticas dos dados
df.describe()
```

[8]:

	var3	var15	imp_ent_var16_ult1	\
count	76020.000000	76020.000000	76020.000000	
mean	-1523.199277	33.212865	86.208265	
std	39033.462364	12.956486	1614.757313	
min	-999999.000000	5.000000	0.000000	
25%	2.000000	23.000000	0.000000	
50%	2.000000	28.000000	0.000000	
75%	2.000000	40.000000	0.000000	
max	238.000000	105.000000	210000.000000	

	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3	\
count	76020.000000	76020.000000	
mean	72.363067	119.529632	
std	339.315831	546.266294	
min	0.000000	0.000000	
25%	0.000000	0.000000	
50%	0.000000	0.000000	
75%	0.000000	0.000000	
max	12888.030000	21024.810000	

	imp_op_var40_comer_ult1	imp_op_var40_comer_ult3	\
count	76020.000000	76020.000000	
mean	3.559130	6.472698	
std	93.155749	153.737066	
min	0.000000	0.000000	
25%	0.000000	0.000000	
50%	0.000000	0.000000	
75%	0.000000	0.000000	
max	8237.820000	11073.570000	

	imp_op_var40_efect_ult1	imp_op_var40_efect_ult3	imp_op_var40_ult1	\
count	76020.000000	76020.000000	76020.000000	
mean	0.412946	0.567352	3.160715	
std	30.604864	36.513513	95.268204	
min	0.000000	0.000000	0.000000	

```
25%                  0.000000                 0.000000          0.000000
50%                  0.000000                 0.000000          0.000000
75%                  0.000000                 0.000000          0.000000
max               6600.000000              6600.000000       8237.820000

       …  saldo_medio_var33_hace2  saldo_medio_var33_hace3  \
count  …             76020.000000             76020.000000
mean   …                 7.935824                 1.365146
std    …               455.887218               113.959637
min    …                 0.000000                 0.000000
25%    …                 0.000000                 0.000000
50%    …                 0.000000                 0.000000
75%    …                 0.000000                 0.000000
max    …             50003.880000             20385.720000

       saldo_medio_var33_ult1  saldo_medio_var33_ult3  \
count            76020.000000            76020.000000
mean                12.215580                8.784074
std                783.207399              538.439211
min                  0.000000                0.000000
25%                  0.000000                0.000000
50%                  0.000000                0.000000
75%                  0.000000                0.000000
max             138831.630000            91778.730000

       saldo_medio_var44_hace2  saldo_medio_var44_hace3  \
count             76020.000000             76020.000000
mean                 31.505324                 1.858575
std                2013.125393               147.786584
min                   0.000000                 0.000000
25%                   0.000000                 0.000000
50%                   0.000000                 0.000000
75%                   0.000000                 0.000000
max              438329.220000             24650.010000

       saldo_medio_var44_ult1  saldo_medio_var44_ult3         var38  \
count            76020.000000            76020.000000  7.602000e+04
mean                76.026165               56.614351  1.172358e+05
std               4040.337842             2852.579397  1.826646e+05
min                  0.000000                0.000000  5.163750e+03
25%                  0.000000                0.000000  6.787061e+04
50%                  0.000000                0.000000  1.064092e+05
75%                  0.000000                0.000000  1.187563e+05
max             681462.900000           397884.300000  2.203474e+07

             TARGET
count  76020.000000
```

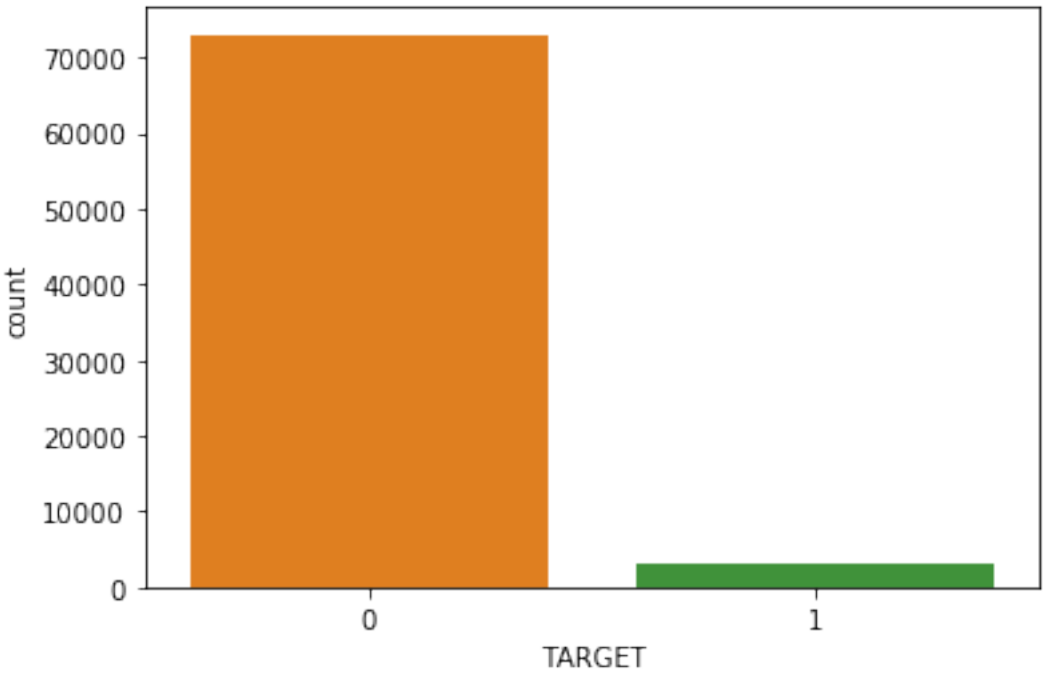
```
mean        0.039569
std         0.194945
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 370 columns]
```

[9]:
```
# Distribuição da variável alvo
df.groupby('TARGET').size()
```

[9]:
```
TARGET
0    73012
1     3008
dtype: int64
```

[10]:
```
# Plota a distribuição da variável alvo
sns.countplot(x='TARGET',data=df,palette="Paired_r")
```

[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d4096899b0>
```



[11]:
```
# Procura por valores nulos
df.isnull().values.any()
```

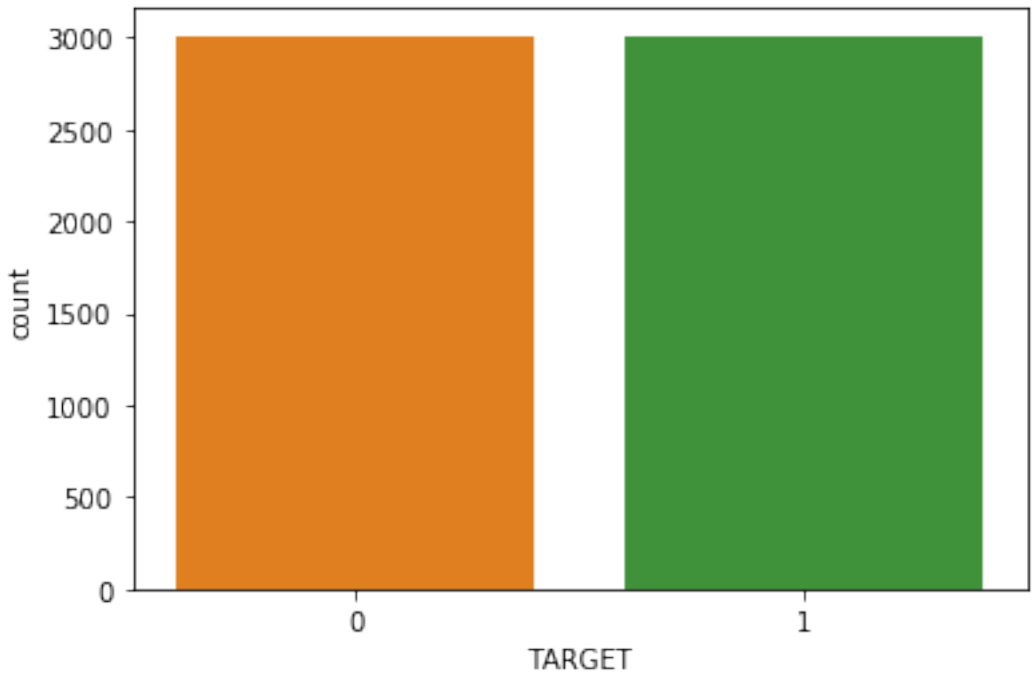
[11]:
```
False
```

[12]:
```
# Cria função para balancear os dados (undersampling) devido a diferença da␣
 ↪distribuição da variável alvo

# Importação dos pacotes
import math

# Criação da função de balanceamento
def undersample(df, target_col, r=1):
    falses = df[target_col].value_counts()[0]
    trues = df[target_col].value_counts()[1]
    relation = float(trues)/float(falses)

    if trues >= r*falses:
        df_drop = df[df[target_col] == True]
        drop_size = int(math.fabs(int((relation - r) * (falses))))
    else:
        df_drop = df[df[target_col] == False]
        drop_size = int(math.fabs(int((r-relation) * (falses))))

    df_drop = df_drop.sample(drop_size)
    df = df.drop(labels=df_drop.index, axis=0)
    return df

# Verifica os dados balanceados
df = undersample(df, 'TARGET')
df.groupby('TARGET').size()
```

[12]:
```
TARGET
0    3008
1    3008
dtype: int64
```

[13]:
```
# Plota a nova distribuição da variável alvo
sns.countplot(x='TARGET',data=df,palette="Paired_r")
```

[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d4095a4470>
```



[14]:
```
# Utiliza PCA para redução de dimensionalidade devido a grande quantidade de colunas

# Importação do módulo
from sklearn.decomposition import PCA

# Separa as varíavéis preditoras

var = df.drop('TARGET', axis=1)
y = df['TARGET']

# Seleção de atributos
pca = PCA(n_components = 5)
fit = pca.fit(var)
var_reduzido = pca.fit_transform(var)

X = pd.DataFrame(var_reduzido)
```

[15]:
```
# Separação dados de treino e de teste

#Importação dos módulos
from sklearn.model_selection import train_test_split

# Separa 67% dos dados para treino
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

[16]:
```
# Treina Modelo 01 Regressão Logística

# Importação dos módulos
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

# Definindo os valores para o número de folds
num_folds = 10
seed = 7

# Separando os dados em folds
kfold = KFold(num_folds, True)

# Criando o modelo
modelo_1 = LogisticRegression()

# Cross Validation
resultado = cross_val_score(modelo_1, X_train, y_train, cv = kfold)

# Print do resultado
print("Acurácia Modelo 1: %.3f%%" % (resultado.mean() * 100))
```

```
Acurácia Modelo 1: 47.643%
```

[17]:
```
# Normaliza os dados para tentar melhor o modelo

# Importação do módulo
from sklearn.preprocessing import Normalizer

# Separando o array em componentes de input e output
var = df.drop('TARGET',axis=1)
y = df['TARGET']

# Gerando os dados normalizados
scaler = Normalizer().fit(var)
normalizedVar = scaler.transform(var)
```

[18]:
```
# Utiliza PCA para redução de dimensionalidade nos dados normalizados

# Importação do módulo
from sklearn.decomposition import PCA

# Seleção de atributos
pca = PCA(n_components = 5)
```

```
fit = pca.fit(normalizedVar)
var_reduzido = pca.fit_transform(normalizedVar)

X = pd.DataFrame(var_reduzido)
```

[19]:
```
# Separação dados de treino e de teste

#Importação dos módulos
from sklearn.model_selection import train_test_split

# Separa 67% dos dados para treino
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

[20]:
```
# Treina Modelo 02 Regressão Logística

# Definindo os valores para o número de folds
num_folds = 10
seed = 7

# Separando os dados em folds
kfold = KFold(num_folds, True)

# Criando o modelo
modelo_2 = LogisticRegression()

# Cross Validation
resultado = cross_val_score(modelo_2, X_train, y_train, cv = kfold)

# Print do resultado
print("Acurácia Modelo 02: %.3f%%" % (resultado.mean() * 100))
```

```
Acurácia Modelo 02: 55.484%
```

[21]:
```
# Utiliza PCA para redução de dimensionalidade nos dados normalizados com mais componentes

# Importação do módulo
from sklearn.decomposition import PCA

# Seleção de atributos
pca = PCA(n_components = 50)
fit = pca.fit(normalizedVar)
var_reduzido = pca.fit_transform(normalizedVar)

X = pd.DataFrame(var_reduzido)
```

[22]:
```
# Separação dados de treino e de teste

#Importação dos módulos
from sklearn.model_selection import train_test_split

# Separa 67% dos dados para treino
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

[23]:
```
# Treina Modelo 03 com Regressão Logística

# Definindo os valores para o número de folds
num_folds = 10
seed = 7

# Separando os dados em folds
kfold = KFold(num_folds, True)

# Criando o modelo
modelo_3 = LogisticRegression()

# Cross Validation
resultado = cross_val_score(modelo_3, X_train, y_train, cv = kfold)

# Print do resultado
print("Acurácia Modelo 03: %.3f%%" % (resultado.mean() * 100))
```

```
Acurácia Modelo 03: 56.079%
```

[24]:
```
# Treina Modelo 04 com XGBoost

# Importação dos módulos
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier

# Criando o modelo
modelo_4 = XGBClassifier()

# Treinando o modelo
modelo_4.fit(X_train, y_train)

# Fazendo previsões
y_pred = modelo_4.predict(X_test)
previsoes = [round(value) for value in y_pred]

# Avaliando as previsões
accuracy = accuracy_score(y_test, previsoes)
print("Acurácia Modelo 04: %.2f%%" % (accuracy * 100.0))
```

```
Acurácia Modelo 04: 74.87%
```

2.0.1 Modelo 04 com XGBoost apresentou acurácia melhor que os modelos de Regressão Logística

[25]:
```
# Importa dados de teste

teste = pd.read_csv('dados/test.csv',index_col='ID')
```

[26]:
```
print(teste.shape)
```

```
(75818, 369)
```

[27]:
```
# Normaliza os dados de teste

# Gerando os dados normalizados
scaler = Normalizer().fit(teste)
normalizedTeste = scaler.transform(teste)


# Redução de dimensionalidade com PCA

pca = PCA(n_components = 50)
fit = pca.fit(normalizedTeste)
teste_reduzido = pca.fit_transform(normalizedTeste)

Z = pd.DataFrame(teste_reduzido)
```

[28]:
```
teste_pred = modelo_4.predict(Z)
```

[29]:
```
sample_submission = pd.DataFrame(teste.index)
```

[30]:
```
sample_submission['TARGET'] = teste_pred
```

[31]:
```
sample_submission.head()
```

[31]:

	ID	TARGET
0	2	1
1	5	1
2	6	0
3	7	1
4	9	1

[32]:
```
sample_submission.groupby('TARGET').size()
```

[32]:
```
TARGET
0    19787
```

```
1    56031
dtype: int64
```

[33]:
```
pd.DataFrame.to_csv(sample_submission,'sample_submission.csv',index=False)
```