Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

Prediction of Customer Satisfaction Level at Santander using Machine Learning

Defining the Business Problem

Customer satisfaction is a critical measure of success. Unsatisfied customers tend to cancel their services and rarely express their dissatisfaction before leaving. On the other hand, satisfied customers become brand advocates. The Santander Bank is seeking assistance in identifying dissatisfied customers early in the relationship. This would enable Santander to take proactive measures to improve the customer experience before it's too late.

In this machine learning project, we will work with hundreds of anonymous features to predict whether a customer is satisfied or dissatisfied with their banking experience. The challenge lies in the unfamiliarity with each variable and the large number of variables available. Our goal is to deliver a list of satisfied and dissatisfied customers to the decision-maker with a model accuracy of 70%. We aim to create a simpler and more generalizable probabilistic model, making it easier for the decision-maker to use the results with clear understanding.

We will utilize the Python programming language and a dataset available on Kaggle, accessible at the following address:

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

Please note that the data is provided in two separate files, train.csv and test.csv. Only the train.csv file contains the response variable. Therefore, we will work exclusively with train.csv during the model-building process. We will use the test.csv file to make predictions using the best model identified and provide the results to the decision-maker.

Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

Packages and Versions

```
# Imports
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import roc_auc_score
from imblearn.over_sampling import SMOTE
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore', category = FutureWarning)

# Versões dos Pacotes
%reload_ext watermark
%watermark --iversions
```

```
pandas    : 1.5.2
numpy     : 1.23.5
matplotlib: 3.6.2
sklearn   : 1.0.2
seaborn   : 0.12.2
```

Loading the Datasets

As explained in the problem definition, we will initially work only with the train.csv file.

```
# Carregando os dados em um Dataframe do Pandas
df = pd.read_csv('Dados/train.csv')
```

Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

```
# Visualizando os dados
df
```

	ID	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3	imp_op_var40_
0	1	2	23	0.0	0.0	0.0	
1	3	2	34	0.0	0.0	0.0	
2	4	2	23	0.0	0.0	0.0	
3	8	2	37	0.0	195.0	195.0	
4	10	2	39	0.0	0.0	0.0	
...	...	...	...	...	...	...	
76015	151829	2	48	0.0	0.0	0.0	
76016	151830	2	39	0.0	0.0	0.0	
76017	151835	2	23	0.0	0.0	0.0	
76018	151836	2	25	0.0	0.0	0.0	
76019	151838	2	46	0.0	0.0	0.0	

76020 rows × 371 columns

```
# Informações Gerais dos Dados
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76020 entries, 0 to 76019
Columns: 371 entries, ID to TARGET
dtypes: float64(111), int64(260)
memory usage: 215.2 MB
```

Data Cleaning and Exploratory Analysis

Upon initial inspection of our data, we notice that we have a vast number of variables (370) without descriptions, making it difficult to understand each variable and analyze their behaviors as they are.

Therefore, our initial steps will be to address missing data (NaN) by applying a technique to handle this issue. Subsequently, we will explore two alternatives for dimensionality reduction:

Selecting the Most Important Variables and Vectorizing Variables with PCA. Finally, we will create models and compare their performance.

Let's start by checking for missing data and applying a technique to handle it.

```
# Identificando dados nulos ou NaN
pd.isnull(df).sum()
```

```
ID                          0
var3                        0
var15                       0
imp_ent_var16_ult1          0
imp_op_var39_comer_ult1     0
                           ..
saldo_medio_var44_hace3     0
saldo_medio_var44_ult1      0
saldo_medio_var44_ult3      0
var38                       0
TARGET                      0
Length: 371, dtype: int64
```

We don't have any missing data, including NaN values, in our working dataset. Let's proceed by renaming the columns to facilitate our visualizations.

Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

```
# Função para Renomear as Variáveis
def ajusta_dados(df):

    # Reserva o Nome das Novas Colunas e Velhas Colunas
    NewCols = []
    for i in range(0, len(df.columns)):
        col = "Var" + str(i)
        NewCols.append(col)
    NewCols[0] = 'ID'
    if NewCols[-1] == 'TARGET':
        NewCols[-1] = 'RESP'
    else:
        pass
    OldCols = list(df.columns)

    # Aplica o De-Para
    DePara = {}
    for i,j in zip(OldCols, NewCols):
        DePara[i] = j
    df.rename(columns = DePara, inplace = True)

    return df
```

```
# Aplica a Função ajusta_dados()
df = ajusta_dados(df=df)
df
```

	ID	Var1	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9	...	Var361	Var362	Var363	Var364	Va
0	1	2	23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
1	3	2	34	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
2	4	2	23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
3	8	2	37	0.0	195.0	195.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
4	10	2	39	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
76015	151829	2	48	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
76016	151830	2	39	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
76017	151835	2	23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
76018	151836	2	25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
76019	151838	2	46	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	

76020 rows × 371 columns

To continue with our work, let's remove the 'ID' variable in a new version of our dataset, as it doesn't provide any statistical information gain.

```
# Vamos eliminar as variáveis RESP e ID em um novo Dataset
df1 = df.drop(['ID'], axis =1)
df1
```

	Var1	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9	Var10	...	Var361	Var362	Var363	Var364
0	2	23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1	2	34	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2	2	23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
3	2	37	0.0	195.0	195.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
4	2	39	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
76015	2	48	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
76016	2	39	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
76017	2	23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
76018	2	25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
76019	2	46	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

76020 rows × 370 columns

Our dataset is organized and clean. However, since we have a large number of variables and do not know the information that each one represents in relation to the business problem, it is not feasible to study the correlation between variables or analyze central and dispersion statistics. Our next step is to begin reducing the dimensionality before applying predictive modeling.

To conclude, let's examine the target variable and determine its distribution.

```
df1['RESP'].value_counts()
```

```
0    73012
1     3008
Name: RESP, dtype: int64
```

We can observe that we have a large number of samples of type 0 (satisfied customers) and very few samples of type 1 (unsatisfied customers). It is quite common to encounter such class imbalance in business problems such as fraud detection or sentiment analysis.

Certainly, addressing the class imbalance issue is an important step in the modeling process. We can explore various techniques such as oversampling the minority class, undersampling the

majority class, or using a combination of both. Additionally, we can consider using algorithms specifically designed to handle imbalanced data, such as ensemble methods or cost-sensitive learning.

We can further discuss and implement these techniques to improve the performance of our predictive models.

Work Strategy

As we have identified some challenges in our project, let's outline our strategy to address them:

Different Variable Scales: We will apply a scaling method, such as standardization or normalization, to bring the variables to a similar scale.

Imbalanced Target Variable: We will create two scenarios to tackle the imbalance:

Scenario 1: Initially, we will work with the unbalanced target variable RESP. We will use dimensionality reduction methods based on variable importance, such as feature selection or feature importance. Additionally, we will apply vectorization using PCA to further reduce the dimensionality. We will train different algorithms on the transformed dataset and evaluate their performance using appropriate metrics to determine the best model.

Scenario 2: We will address the class imbalance issue by applying an oversampling technique, specifically SMOTE, to balance the dataset. Then, we will use the same dimensionality reduction techniques as in Scenario 1 and train the algorithms on the transformed dataset. We will compare the performance of these models with those from Scenario 1.

High Number of Variables: Given that we have 369 variables, we will apply dimensionality reduction techniques, as mentioned in Scenarios 1 and 2, to reduce the number of variables and capture the most relevant information.

At the end of the project, we will compare the models developed in both scenarios and determine the best model based on their performance metrics.

By following this strategy, we aim to address the challenges posed by different variable scales, imbalanced target variable, and high dimensionality, and identify the most effective model for predicting customer satisfaction.

Scenario 1 - Imbalanced Dataset + Dimensionality Reduction Techniques

First, we will divide the data into training and testing sets. We will use the Sklearn package for this task.

```
# Separando os dados em Treino e Teste
Xtreino, Xteste, Ytreino, Yteste = train_test_split(df1.iloc[:, 0:369],
                                                    df1.iloc[:, -1],
                                                    test_size = 0.3,
                                                    random_state = 0)
```

After separating the datasets, we will apply normalization to the training and testing data to equalize the scales of the variables.

```
# Modelo para Normalização dos Dados
scaler = preprocessing.StandardScaler().fit(Xtreino)
```

```
# Normalizando os dados de Treino
XtreinoS = scaler.transform(Xtreino)
XtreinoS
```

```
array([[ 3.99807940e-02, -7.89237455e-01, -5.24513963e-02, ...,
        -1.91798836e-02, -2.00718862e-02, -4.84081375e-04],
       [ 3.99807940e-02,  1.12995969e+00, -5.24513963e-02, ...,
        -1.91798836e-02, -2.00718862e-02,  1.84784599e+00],
       [ 3.99807940e-02,  5.15816600e-01, -5.24513963e-02, ...,
        -1.91798836e-02, -2.00718862e-02, -4.20753830e-01],
       ...,
       [ 4.01560803e-02, -7.89237455e-01, -5.24513963e-02, ...,
        -1.91798836e-02, -2.00718862e-02, -2.66835804e-01],
       [ 3.99807940e-02, -2.51862256e-01, -1.50190114e-02, ...,
        -1.91798836e-02, -2.00718862e-02, -2.81950539e-01],
       [ 3.99807940e-02,  1.66733488e+00, -5.24513963e-02, ...,
        -1.91798836e-02, -2.00718862e-02, -2.28898997e-01]])
```

```
# Normalizando os dados de Teste
XtesteS = scaler.transform(Xteste)
XtesteS
```

```
array([[ 3.99807940e-02, -7.89237455e-01, -5.24513963e-02, ...,
        -1.91798836e-02, -2.00718862e-02, -1.24801003e-01],
       [ 3.99807940e-02, -7.89237455e-01, -5.24513963e-02, ...,
        -1.91798836e-02, -2.00718862e-02,  1.36030923e-02],
       [ 3.99807940e-02,  8.22888143e-01, -5.24513963e-02, ...,
        -1.91798836e-02, -2.00718862e-02, -4.84081375e-04],
       ...,
       [ 3.99807940e-02, -7.89237455e-01, -5.24513963e-02, ...,
        -1.91798836e-02, -2.00718862e-02,  2.12808818e-01],
       [ 3.99807940e-02, -1.75094370e-01, -5.24513963e-02, ...,
        -1.91798836e-02, -2.00718862e-02,  1.76104562e-01],
       [ 3.99807940e-02, -7.89237455e-01, -5.24513963e-02, ...,
        -1.91798836e-02, -2.00718862e-02, -4.15936973e-01]])
```

Model 00 - Logistic Regression + Most Important Variables (SelectFromModel - Sklearn)

Datasets with equivalent scales, we can apply our first method of dimensionality reduction. We will apply a Feature Selection based on a Logistic Regression model using SelectFromModel from the Sklearn package.

```
# Modelo para Seleção de Variáveis utilizando Regressão Logística e SelectFromModel do pacote Sklearn
LR = LogisticRegression('l2', random_state = 0, max_iter = 10000).fit(XtreinoS, Ytreino)
model = SelectFromModel(LR, prefit=True)
```

```
# Aplicando Redução de Dimensionalidade aos dados de Treino
Xtreino00 = model.transform(XtreinoS)
Xtreino00.shape
```

```
(53214, 105)
```

```
# Aplicando Redução de Dimensionalidade aos dados de Teste
Xteste00 = model.transform(XtesteS)
Xteste00.shape
```

```
(22806, 105)
```

We have arrived at 105 important variables for the classification of our customers. Although we still have a large number of variables, we will apply them to a predictive model and calculate the metrics.

```
# Modelo Preditivo 00 - Regressão Logística
modelo00 = LogisticRegression(random_state=0, max_iter=10000).fit(Xtreino00, Ytreino)
modelo00
```

Making predictions with the trained model and evaluating its performance.

```
# Aplicando Modelo aos Dados de Teste e Calculando as Métricas
prev00 = modelo00.predict(Xteste00)
Acc00 = accuracy_score(Yteste, prev00)
Prec00 = precision_score(Yteste, prev00, average='weighted')
AUC00 = roc_auc_score(Yteste, prev00)
```

```
# Visualizando as Métricas do Modelo
print('A Acurácia do Modelo é ', Acc00)
print('A Precisão do Modelo é ', Prec00)
print('A área sob a Curva ROC do Modelo é ', AUC00)
```

```
A Acurácia do Modelo é  0.9593966500043848
A Precisão do Modelo é  0.9214315812010174
A área sob a Curva ROC do Modelo é  0.4997259272793715
```

Let's store the results in a table for comparison later.

```
# Colocando as Métricas em um Dataframe de Resultado
GLM00 = pd.Series(data=[Acc00, Prec00, AUC00], index=['Acurácia', 'Precisão', 'ROC AUC'])
Resultado = pd.DataFrame(data={'GLM00': GLM00})
Resultado
```

	GLM00
Acurácia	0.959397
Precisão	0.921432
ROC AUC	0.499726

Model 01 - Logistic Regression + PCA

```
# Aplicando Redução de Dimensionalidade com Vetorização para os dados de Treino
Xtreino01 = PCA(n_components = 100).fit_transform(XtreinoS)
Xtreino01.shape
```

```
(53214, 100)
```

We will work with 100 variables for this model and evaluate its performance.

```
# Modelo Preditivo 01 - Regressão Logística
modelo01 = LogisticRegression(random_state=0, max_iter=10000).fit(Xtreino01, Ytreino)
modelo01
```

```
LogisticRegression(max_iter=10000, random_state=0)
```

```
# Aplicando Modelo Preditivo e Calculando as Métricas
prev01 = modelo01.predict(Xteste01)
Acc01 = accuracy_score(Yteste, prev01)
Prec01 = precision_score(Yteste, prev01, average='weighted')
AUC01 = roc_auc_score(Yteste, prev01)
```

```
# Visualizando as Métricas do Modelo
print('A Acurácia do Modelo é ', Acc01)
print('A Precisão do Modelo é ', Prec01)
print('A área sob a Curva ROC do Modelo é ', AUC01)
```

```
A Acurácia do Modelo é  0.040164868894150 66
A Precisão do Modelo é  0.961529147963921
A área sob a Curva ROC do Modelo é  0.5000456787867714
```

```
# Salvando os resultados na Tabela Resultado
GLM01 = pd.Series(data=[Acc01, Prec01, AUC01], index=['Acurácia', 'Precisão', 'ROC AUC'])
Resultado['GLM01'] = pd.DataFrame(GLM01)
Resultado
```

	GLM00	GLM01
Acurácia	0.959397	0.040165
Precisão	0.921432	0.961529
ROC AUC	0.499726	0.500046

Model 02 - Stochastic Gradient Descent (SGD) + SelectFromModel

For this model, we will use Stochastic Gradient Descent (SGD) and the variable selection we performed with SelectFromModel from Sklearn.

```
# Modelo 02 - Stochastic Gradient Descent
modelo02 = SGDClassifier().fit(Xtreino00, Ytreino)
modelo02
```

```
SGDClassifier()
```

```
# Aplicando Modelo Preditivo e Calculando as Métricas
prev02 = modelo02.predict(Xteste00)
Acc02 = accuracy_score(Yteste, prev02)
Prec02 = precision_score(Yteste, prev02, average='weighted')
AUC02 = roc_auc_score(Yteste, prev02)
```

```
# Visualizando as Métricas do Modelo
print('A Acurácia do Modelo é ', Acc02)
print('A Precisão do Modelo é ', Prec02)
print('A área sob a Curva ROC do Modelo é ', AUC02)
```

```
A Acurácia do Modelo é  0.95970358677541
A Precisão do Modelo é  0.9214433981705747
A área sob a Curva ROC do Modelo é  0.49988580303307145
```

```
# Salvando os resultados na Tabela Resultado
SGD02 = pd.Series(data=[Acc02, Prec02, AUC02], index=['Acurácia', 'Precisão', 'ROC AUC'])
Resultado['SGD02'] = SGD02
Resultado
```

	GLM00	GLM01	SGD02
Acurácia	0.959397	0.040165	0.959704
Precisão	0.921432	0.961529	0.921443
ROC AUC	0.499726	0.500046	0.499886

Model 03 - Stochastic Gradient Descent (SGD) + PCA

For this model, we will use Stochastic Gradient Descent (SGD) and apply PCA for dimensionality reduction.

```
# Modelo 03 - Stochastic Gradient Descent
modelo03 = SGDClassifier().fit(Xtreino01, Ytreino)
modelo03
```

```
SGDClassifier()
```

```
# Aplicando Modelo Preditivo e Calculando as Métricas
prev03 = modelo03.predict(Xteste01)
Acc03 = accuracy_score(Yteste, prev03)
Prec03 = precision_score(Yteste, prev03, average='weighted')
AUC03 = roc_auc_score(Yteste, prev03)
```

```
# Visualizando as Métricas do Modelo
print('A Acurácia do Modelo é ', Acc03)
print('A Precisão do Modelo é ', Prec03)
print('A área sob a Curva ROC do Modelo é ', AUC03)
```

```
A Acurácia do Modelo é  0.04016486889415066
A Precisão do Modelo é  0.961529147963921
A área sob a Curva ROC do Modelo é  0.5000456787867714
```

```
# Salvando os resultados na Tabela Resultado
SGD03 = pd.Series(data=[Acc03, Prec03, AUC03], index=['Acurácia', 'Precisão', 'ROC AUC'])
Resultado['SGD03'] = SGD03
Resultado
```

Scenario 2 - Dataset Balancing + Dimensionality Reduction Techniques

For this scenario, we will repeat the algorithms used in Scenario 1, but we will apply a technique to balance the target variable using synthetic data, called Oversampling.

```
# Criando um modelo de Oversampling
sm = SMOTE(random_state=0)
```

```
# Balanceando os dados de Treino
XtreinoB, YtreinoB = sm.fit_resample(XtreinoS, Ytreino)
YtreinoB.value_counts()
```

```
0    51120
1    51120
Name: RESP, dtype: int64
```

Model 04 - Logistic Regression + SelectFromModel

Now we will repeat the algorithms and techniques used previously, but with our balanced dataset.

```
# Modelo de Redução de Dimensionalidade com SelectFromModel()
LROS = LogisticRegression('l2', random_state = 0, max_iter = 10000).fit(XtreinoB, YtreinoB)
modelOS= SelectFromModel(LR, prefit=True)
```

```
# Aplicando Redução de Dimensionalidade aos Datasets de Treino e Teste
XtreinoB00 = modelOS.transform(XtreinoB)
XtesteB00 = modelOS.transform(XtesteS)
```

```
# Modelo 04 - Regressão Logística
modelo04 = LogisticRegression(random_state=0, max_iter=10000).fit(XtreinoB00, YtreinoB)
```

```
# Aplicando Modelo Preditivo e Calculando as Métricas
prev04 = modelo04.predict(XtesteB00)
Acc04 = accuracy_score(Yteste, prev04)
Prec04 = precision_score(Yteste, prev04, average='weighted')
AUC04 = roc_auc_score(Yteste, prev04)
```

```
# Visualizando as Métricas do Modelo
print('A Acurácia do Modelo é ', Acc04)
print('A Precisão do Modelo é ', Prec04)
print('A área sob a Curva ROC do Modelo é ', AUC04)
```

```
A Acurácia do Modelo é  0.6973164956590371
A Precisão do Modelo é  0.9494225971764026
A área sob a Curva ROC do Modelo é  0.7222963655678303
```

```
# Salvando os resultados na Tabela Resultado
GLMB04 = pd.Series(data=[Acc04, Prec04, AUC04], index=['Acurácia', 'Precisão', 'ROC AUC'])
Resultado['GLMB04'] = GLMB04
Resultado
```

	GLM00	GLM01	SGD02	SGD03	GLMB04
Acurácia	0.959397	0.040165	0.959704	0.040165	0.697316
Precisão	0.921432	0.961529	0.921443	0.961529	0.949423
ROC AUC	0.499726	0.500046	0.499886	0.500046	0.722296

Model 05 - Logistic Regression + PCA

```
# Modelo de Redução de Dimensionalidade com PCA
PCA01 = PCA(n_components = 100)
```

```
# Aplicando Redução de Dimensionalidade aos dados de Treino e Teste
XtreinoB01 = PCA01.fit_transform(XtreinoB)
XtesteB01 = PCA01.fit_transform(XtesteS)
```

```
# Modelo 05 -Regressão Logística
modelo05 = LogisticRegression(random_state=0, max_iter=10000).fit(XtreinoB01, YtreinoB)
```

```
# Aplicando Modelo Preditivo e Calculando as Métricas
prev05 = modelo05.predict(XtesteB01)
Acc05 = accuracy_score(Yteste, prev05)
Prec05 = precision_score(Yteste, prev05, average='weighted')
AUC05 = roc_auc_score(Yteste, prev05)
```

```
# Visualizando as Métricas do Modelo
print('A Acurácia do Modelo é ', Acc05)
print('A Precisão do Modelo é ', Prec05)
print('A área sob a Curva ROC do Modelo é ', AUC05)
```

```
A Acurácia do Modelo é  0.04012102078400421
A Precisão do Modelo é  0.9615290775267082
A área sob a Curva ROC do Modelo é  0.5000228393933857
```

```
# Salvando os resultados na Tabela Resultado
GLMB05 = pd.Series(data=[Acc05, Prec05, AUC05], index=['Acurácia', 'Precisão', 'ROC AUC'])
Resultado['GLMB05'] = GLMB05
Resultado
```

	GLM00	GLM01	SGD02	SGD03	GLMB04	GLMB05
Acurácia	0.959397	0.040165	0.959704	0.040165	0.697316	0.040121
Precisão	0.921432	0.961529	0.921443	0.961529	0.949423	0.961529
ROC AUC	0.499726	0.500046	0.499886	0.500046	0.722296	0.500023

Model 06 - Stochastic Gradient Descent (SGD) + SelectFromModel

```
# Modelo 06 - Stochastic Gradient Descent
modelo06 = SGDClassifier().fit(XtreinoB00, YtreinoB)
```

```
# Aplicando Modelo Preditivo e Calculando as Métricas
prev06 = modelo06.predict(XtesteB00)
Acc06 = accuracy_score(Yteste, prev06)
Prec06 = precision_score(Yteste, prev06, average='weighted')
AUC06 = roc_auc_score(Yteste, prev06)
```

```
# Visualiza as Metricas do Modelo
print('A Acurácia do Modelo é ', Acc06)
print('A Precisão do Modelo é ', Prec06)
print('A área sob a Curva ROC do Modelo é ', AUC06)
```

```
A Acurácia do Modelo é  0.7280540208717005
A Precisão do Modelo é  0.9489354547371738
A área sob a Curva ROC do Modelo é  0.7257258229278323
```

```
# Salvando os resultados na Tabela Resultado
SGDB06 = pd.Series(data=[Acc06, Prec06, AUC06], index=['Acurácia', 'Precisão', 'ROC AUC'])
Resultado['SGDB06'] = SGDB06
Resultado
```

	GLM00	GLM01	SGD02	SGD03	GLMB04	GLMB05	SGDB06
Acurácia	0.959397	0.040165	0.959704	0.040165	0.697316	0.040121	0.728054
Precisão	0.921432	0.961529	0.921443	0.961529	0.949423	0.961529	0.948935
ROC AUC	0.499726	0.500046	0.499886	0.500046	0.722296	0.500023	0.725726

Model 07 - Stochastic Gradient Descent (SGD) + PCA

```
# Modelo 08 - Stochastic Gradient Descent
modelo07 = SGDClassifier().fit(XtreinoB01, YtreinoB)
```

```
# Aplicando Modelo Preditivo e Calculando as Métricas
prev07 = modelo07.predict(XtesteB01)
Acc07 = accuracy_score(Yteste, prev07)
Prec07 = precision_score(Yteste, prev07, average='weighted')
AUC07 = roc_auc_score(Yteste, prev07)
```

```
# Visualizando as Métricas do Modelo
print('A Acurácia do Modelo é ', Acc07)
print('A Precisão do Modelo é ', Prec07)
print('A área sob a Curva ROC do Modelo é ', AUC07)
```

```
A Acurácia do Modelo é  0.9598351311058494
A Precisão do Modelo é  0.9214484603652541
A área sob a Curva ROC do Modelo é  0.4999543212132286
```

```
# Salvando os resultados na Tabela Resultados
SGDB07 = pd.Series(data=[Acc07, Prec07, AUC07], index=['Acurácia', 'Precisão', 'ROC AUC'])
Resultado['SGDB07'] = SGDB07
Resultado
```

	GLM00	GLM01	SGD02	SGD03	GLMB04	GLMB05	SGDB06	SGDB07
Acurácia	0.959397	0.040165	0.959704	0.040165	0.697316	0.040121	0.728054	0.959835
Precisão	0.921432	0.961529	0.921443	0.961529	0.949423	0.961529	0.948935	0.921448
ROC AUC	0.499726	0.500046	0.499886	0.500046	0.722296	0.500023	0.725726	0.499954

Developed by Thiago Bulgarelli

Contact: bugath36@gmail.com

Conclusion and Prediction Delivery

Analyzing the table of model results, we can see that the SGD model, using the balanced dataset with the oversampling technique and applying the dimensionality reduction technique from Sklearn (SelectFromModel), achieved the most balanced performance among the metrics. It obtained an Accuracy of 72.8%, Precision of 94.89%, and ROC AUC of 72.57%.

```
# Modelo de Performance mais Equilibrada
Resultado['SGDB06']
```

```
Acurácia    0.728054
Precisão    0.948935
ROC AUC     0.725726
Name: SGDB06, dtype: float64
```

```
# Carregando Novos Dados
ND = pd.read_csv('Dados/test.csv')
ND.shape
```

```
(75818, 370)
```

```
# Ajustanto os Novos Dados
ND1 = ajusta_dados(ND)
ND1 = ND1.drop('ID', axis=1)
ND1
```

	Var1	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9	Var10	...	Var360	Var361	Var362	Var363	Var36
0	2	32	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.
1	2	35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.
2	2	23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.
3	2	24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.
4	2	23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
75813	2	23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.
75814	2	26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.
75815	2	24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.
75816	2	40	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.
75817	2	23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.

75818 rows × 369 columns

```
# Normalizando os Novos Dados com Scaler()
NDNorm = scaler.transform(ND1)

# Aplicando Redução de Dimensionalidade com SelectFromModel()
NDNormR = modelOS.transform(NDNorm)
```

```
# Previsão de Classificação com o Modelo de Melhor Performance SGD06
PrevisaoFinal = modelo06.predict(NDNormR)
```

```
# Preparando Tabela Final
data = {'ID Cliente':ND['ID'], 'Status Felicidade': PrevisaoFinal}
TabelaFinal = pd.DataFrame(data=data)
TabelaFinal.head()
```

	ID Cliente	Status Felicidade
0	2	0
1	5	0
2	6	0
3	7	0
4	9	0

```
# Salando em Disco em extensão .CSV
TabelaFinal.to_csv('dados/TabelaFinal.csv', sep=',')
```

We have completed our work and delivered the PrevisaoFinal table to the client, containing the predictions for the requested examples.