```
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/)
that gets preserved as output when you create a version using "Save &
Run All"
# You can also write temporary files to /kaggle/temp/, but they won't
be saved outside of the current session
```

```
/kaggle/input/santander-customer-satisfaction/sample_submission.csv
/kaggle/input/santander-customer-satisfaction/train.csv
/kaggle/input/santander-customer-satisfaction/test.csv
```

Santander Customer Satisfaction

Description

Santander Bank, headquartered in Spain, is one of the largest banks in the world with a presence in 10 core markets. With over 190,000 employees serving 150 million customers, Santander is committed to customer satisfaction and experience.

Traditionally, banks have been reactive to customer dissatisfaction, often only finding out about issues after customers leave. To improve retention and build loyalty, Santander has partnered with Kaggle to gain insights into customer satisfaction in a proactive manner.

The goal of this project is to develop a classification model that can predict whether a Santander banking customer is satisfied or dissatisfied based on hundreds of anonymized customer attributes and behaviors. By identifying potentially dissatisfied customers early, the bank can address issues, resolve complaints, and improve customer experience before loyalty is lost.

The findings will provide Santander strategic guidance on areas of customer experience that most impact satisfaction levels. Frontline and customer-facing teams will also benefit from clearer direction on satisfaction drivers to help retain and nurture long-term, profitable

customer relationships. Overall, the aim is to enhance Santander's customer-centric culture and competitive positioning through a better understanding of satisfaction dynamics.

Import Libraries

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use("Agg") #Needed to save figures
from sklearn.model_selection import train_test_split, cross_val_score
import xgboost as xgb
from sklearn.metrics import roc_auc_score
import seaborn as sns
```

Import Dataset

```
training = pd.read_csv("/kaggle/input/santander-customer-
satisfaction/train.csv", index_col=0)
test =
pd.read_csv("/kaggle/input/santander-customer-satisfaction/test.csv",
index_col=0)

print(training.shape)
print(test.shape)
```

```
(76020, 370)
(75818, 369)
```

Exploratory Data Analysis

We will begin our analysis of the Santander customer data by conducting exploratory data exploration and analysis. This will involve checking for missing data, getting a high-level overview of the feature distributions and types, and identifying any outliers. We aim to explore univariate relationships as well as correlations between variables to see how features may be related to satisfaction levels. From there, we will do some preliminary feature engineering like converting categorical variables and creating derived attributes. This initial exploration phase will help us gain important insights into the data to guide subsequent modeling steps, highlight variables likely to be predictive, and identify any data cleaning tasks that need addressing before building predictive models.

```
training.describe().T
```

```
                              count           mean             std
min  \
var3                        76020.0   -1523.199277   39033.462364 -
999999.00
var15                       76020.0      33.212865      12.956486
5.00
```

```
imp_ent_var16_ult1       76020.0      86.208265    1614.757313
0.00
imp_op_var39_comer_ult1  76020.0      72.363067     339.315831
0.00
imp_op_var39_comer_ult3  76020.0     119.529632     546.266294
0.00
...                          ...            ...            ...
...
saldo_medio_var44_hace3  76020.0       1.858575     147.786584
0.00
saldo_medio_var44_ult1   76020.0      76.026165    4040.337842
0.00
saldo_medio_var44_ult3   76020.0      56.614351    2852.579397
0.00
var38                    76020.0  117235.809430  182664.598503
5163.75
TARGET                   76020.0       0.039569       0.194945
0.00

                                25%         50%           75%
max
var3                       2.0000        2.00        2.0000
238.00
var15                     23.0000       28.00       40.0000
105.00
imp_ent_var16_ult1         0.0000        0.00        0.0000
210000.00
imp_op_var39_comer_ult1    0.0000        0.00        0.0000
12888.03
imp_op_var39_comer_ult3    0.0000        0.00        0.0000
21024.81
...                           ...         ...           ...           .
..
saldo_medio_var44_hace3    0.0000        0.00        0.0000
24650.01
saldo_medio_var44_ult1     0.0000        0.00        0.0000
681462.90
saldo_medio_var44_ult3     0.0000        0.00        0.0000
397884.30
var38                  67870.6125   106409.16   118756.2525
22034738.76
TARGET                     0.0000        0.00        0.0000
1.00

[370 rows x 8 columns]
```

```
from IPython.display import display_html
import io
#buffer = io.StringIO()
```

```
# save the describe as a df
desc = training.describe().T

# Display per 50 columns and convert them to an HTML
desc1 = desc.iloc[:50].to_html()
desc2 = desc.iloc[50:100].to_html()
desc3 = desc.iloc[100:150].to_html()

#Display using html
display_html(f"""
            <div style="display: flex; justify-content: space-around;
gap: 20px;">
            <div style="flex: 1;">{desc1}</div>
            <div style="flex: 1;">{desc2}</div>
            <div style="flex: 1;">{desc3}</div>
            </div>
            """, raw=True
            )

# Display per 50 columns and convert them to an HTML
desc4 = desc.iloc[150:200].to_html()
desc5 = desc.iloc[200:250].to_html()
desc6 = desc.iloc[250:300].to_html()

display_html(f"""
            <div style="display: flex; justify-content: space-around;
gap: 20px;">
            <div style="flex: 1;">{desc4}</div>
            <div style="flex: 1;">{desc5}</div>
            <div style="flex: 1;">{desc6}</div>
            </div>
            """, raw=True
            )

training.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 76020 entries, 1 to 151838
Columns: 370 entries, var3 to TARGET
dtypes: float64(111), int64(259)
memory usage: 215.2 MB
```

```
training.dtypes.value_counts()
```

```
int64      259
float64    111
Name: count, dtype: int64
```

All of our outputs are either Integer or Float

```
# Assuming 'training' is your DataFrame
z_scores = (training - training.mean()) / training.std()
outliers = training[(z_scores > 3) | (z_scores < -3)]



missing_values = training.isnull().sum().sort_index()
has_missing_values = missing_values.any()  # Check if any column has
missing values
has_missing_values
```

```
False
```

There are no missing values as seen on the above output.

Data Cleaning

Before modeling the data, we will need to implement some cleaning and preprocessing steps. We will start by checking for any missing values in key features and either drop rows/columns or fill them in using mean/mode imputation depending on the extent and distribution of missingness. We will also identify and address any outliers that may skew results by winsorizing or capping extreme values. Additional checks involve recoding inconsistent data types or formats and fixing any logical inconsistencies. Categorical variables with high cardinality may need to be grouped. Once cleaned, we will do a final check to ensure issues are resolved before splitting the data for training and testing models. Thorough data cleaning is essential to ensure high quality inputs for downstream analysis and accurate predictive modeling.

```
rows_with_value = training[training == -999999]

# Filter out rows with NaN values
rows_with_value = rows_with_value.dropna()

# Get the index of the rows
index_of_rows = rows_with_value.index

# Get the rows themselves
rows_with_value_df = training.loc[index_of_rows]
rows_with_value_df
```

```
Empty DataFrame
Columns: [var3, var15, imp_ent_var16_ult1, imp_op_var39_comer_ult1,
imp_op_var39_comer_ult3, imp_op_var40_comer_ult1,
imp_op_var40_comer_ult3, imp_op_var40_efect_ult1,
imp_op_var40_efect_ult3, imp_op_var40_ult1, imp_op_var41_comer_ult1,
imp_op_var41_comer_ult3, imp_op_var41_efect_ult1,
imp_op_var41_efect_ult3, imp_op_var41_ult1, imp_op_var39_efect_ult1,
imp_op_var39_efect_ult3, imp_op_var39_ult1, imp_sal_var16_ult1,
ind_var1_0, ind_var1, ind_var2_0, ind_var2, ind_var5_0, ind_var5,
ind_var6_0, ind_var6, ind_var8_0, ind_var8, ind_var12_0, ind_var12,
ind_var13_0, ind_var13_corto_0, ind_var13_corto, ind_var13_largo_0,
```

```
ind_var13_largo, ind_var13_medio_0, ind_var13_medio, ind_var13,
ind_var14_0, ind_var14, ind_var17_0, ind_var17, ind_var18_0,
ind_var18, ind_var19, ind_var20_0, ind_var20, ind_var24_0, ind_var24,
ind_var25_cte, ind_var26_0, ind_var26_cte, ind_var26, ind_var25_0,
ind_var25, ind_var27_0, ind_var28_0, ind_var28, ind_var27,
ind_var29_0, ind_var29, ind_var30_0, ind_var30, ind_var31_0,
ind_var31, ind_var32_cte, ind_var32_0, ind_var32, ind_var33_0,
ind_var33, ind_var34_0, ind_var34, ind_var37_cte, ind_var37_0,
ind_var37, ind_var39_0, ind_var40_0, ind_var40, ind_var41_0,
ind_var41, ind_var39, ind_var44_0, ind_var44, ind_var46_0, ind_var46,
num_var1_0, num_var1, num_var4, num_var5_0, num_var5, num_var6_0,
num_var6, num_var8_0, num_var8, num_var12_0, num_var12, num_var13_0,
num_var13_corto_0, num_var13_corto, ...]
Index: []

[0 rows x 370 columns]
```

```
training['var3'].value_counts()
```

```
var3
 2         74165
 8           138
-999999      116
 9           110
 3           108
            ...
 231           1
 188           1
 168           1
 135           1
 87            1
Name: count, Length: 208, dtype: int64
```

```
# Several values in the var3 column have the value -999999
# We shall replace
training = training.replace(-999999,2)
```

```
from sklearn.feature_selection import SelectPercentile
from sklearn.feature_selection import f_classif,chi2
from sklearn.preprocessing import Binarizer, scale
#for finding features with constant features
from sklearn.feature_selection import VarianceThreshold
```

Feature Selection

We have a toal of 370 columns or features. Some of which are very constant in nature. Meaning that a large or all of their values are the same. This could be unnecessary to our model. In order to select and decide on those features let's look at them all together.

```
sel = VarianceThreshold(threshold=0)
sel.fit(training)
sum(sel.get_support())
```

```
336
```

```
# Print the constant features
print(
    len([
        x for x in training.columns
        if x not in training.columns[sel.get_support()]
    ]))
```

```
[x for x in training.columns if x not in
training.columns[sel.get_support()]]
```

```
34
```

```
['ind_var2_0',
 'ind_var2',
 'ind_var27_0',
 'ind_var28_0',
 'ind_var28',
 'ind_var27',
 'ind_var41',
 'ind_var46_0',
 'ind_var46',
 'num_var27_0',
 'num_var28_0',
 'num_var28',
 'num_var27',
 'num_var41',
 'num_var46_0',
 'num_var46',
 'saldo_var28',
 'saldo_var27',
 'saldo_var41',
 'saldo_var46',
 'imp_amort_var18_hace3',
 'imp_amort_var34_hace3',
 'imp_reemb_var13_hace3',
 'imp_reemb_var33_hace3',
 'imp_trasp_var17_out_hace3',
 'imp_trasp_var33_out_hace3',
 'num_var2_0_ult1',
 'num_var2_ult1',
 'num_reemb_var13_hace3',
 'num_reemb_var33_hace3',
 'num_trasp_var17_out_hace3',
 'num_trasp_var33_out_hace3',
```

```
 'saldo_var2_ult1',
 'saldo_medio_var13_medio_hace3']
```

```
sel2 = VarianceThreshold(threshold=0.01) #0.1 of the values are
differents the others are all the same
sel2.fit(training)
```

```
VarianceThreshold(threshold=0.01)
```

```
print(
    len([
        x for x in training.columns
        if x not in training.columns[sel2.get_support()]
    ]))
```

```
[x for x in training.columns if x not in
training.columns[sel2.get_support()]]
```

```
97
```

```
['ind_var1',
 'ind_var2_0',
 'ind_var2',
 'ind_var6_0',
 'ind_var6',
 'ind_var13_largo',
 'ind_var13_medio_0',
 'ind_var13_medio',
 'ind_var14',
 'ind_var17_0',
 'ind_var17',
 'ind_var18_0',
 'ind_var18',
 'ind_var19',
 'ind_var20_0',
 'ind_var20',
 'ind_var27_0',
 'ind_var28_0',
 'ind_var28',
 'ind_var27',
 'ind_var29_0',
 'ind_var29',
 'ind_var30_0',
 'ind_var31_0',
 'ind_var31',
 'ind_var32_cte',
 'ind_var32_0',
 'ind_var32',
 'ind_var33_0',
 'ind_var33',
```

```
 'ind_var34_0',
 'ind_var34',
 'ind_var40',
 'ind_var41',
 'ind_var39',
 'ind_var44_0',
 'ind_var44',
 'ind_var46_0',
 'ind_var46',
 'num_var6_0',
 'num_var6',
 'num_var13_medio_0',
 'num_var13_medio',
 'num_var18_0',
 'num_var18',
 'num_var27_0',
 'num_var28_0',
 'num_var28',
 'num_var27',
 'num_var29_0',
 'num_var29',
 'num_var33',
 'num_var34_0',
 'num_var34',
 'num_var41',
 'num_var46_0',
 'num_var46',
 'saldo_var28',
 'saldo_var27',
 'saldo_var41',
 'saldo_var46',
 'imp_amort_var18_hace3',
 'imp_amort_var34_hace3',
 'imp_reemb_var13_hace3',
 'imp_reemb_var33_hace3',
 'imp_trasp_var17_out_hace3',
 'imp_trasp_var33_out_hace3',
 'ind_var7_emit_ult1',
 'ind_var7_recib_ult1',
 'num_var2_0_ult1',
 'num_var2_ult1',
 'num_aport_var33_hace3',
 'num_aport_var33_ult1',
 'num_var7_emit_ult1',
 'num_compra_var44_hace3',
 'num_meses_var13_medio_ult3',
 'num_meses_var17_ult3',
 'num_meses_var29_ult3',
 'num_meses_var33_ult3',
```

```
 'num_meses_var44_ult3',
 'num_reemb_var13_hace3',
 'num_reemb_var13_ult1',
 'num_reemb_var17_hace3',
 'num_reemb_var17_ult1',
 'num_reemb_var33_hace3',
 'num_reemb_var33_ult1',
 'num_trasp_var17_in_hace3',
 'num_trasp_var17_in_ult1',
 'num_trasp_var17_out_hace3',
 'num_trasp_var17_out_ult1',
 'num_trasp_var33_in_hace3',
 'num_trasp_var33_in_ult1',
 'num_trasp_var33_out_hace3',
 'num_trasp_var33_out_ult1',
 'num_venta_var44_hace3',
 'saldo_var2_ult1',
 'saldo_medio_var13_medio_hace3']

# for prooving purposes
training['num_trasp_var17_in_ult1'].value_counts()

num_trasp_var17_in_ult1
0    76016
3        4
Name: count, dtype: int64
```

we can then drop these columns from the train and test sets

X_train = sel.transform(X_train) X_test = sel.transform(X_test)

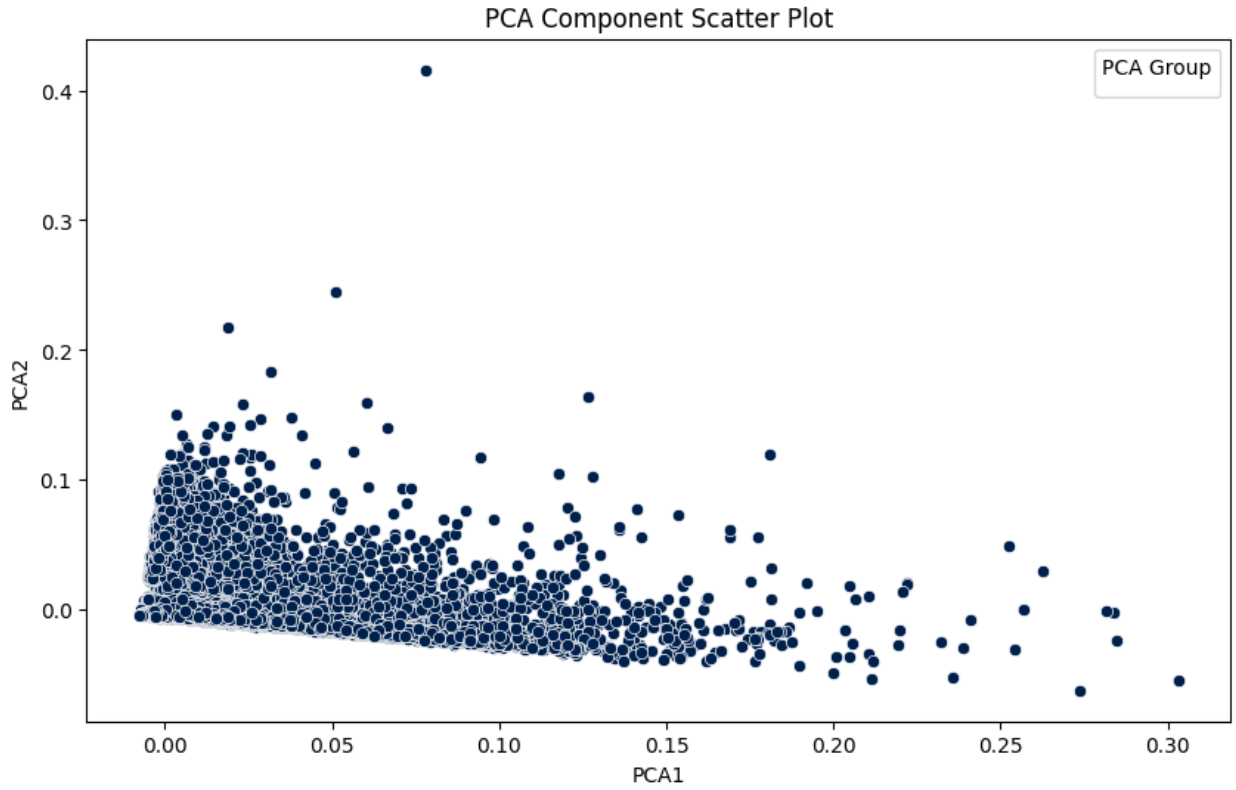
```
# Store the constant features in a dictionary
constant_features_dict = {
    'constant_features': [x for x in training.columns if x not in
training.columns[sel.get_support()]]
                         }

print(f"Number of constant features:
{len(constant_features_dict['constant_features'])}")
print(constant_features_dict['constant_features'])

Number of constant features: 34
['ind_var2_0', 'ind_var2', 'ind_var27_0', 'ind_var28_0', 'ind_var28',
'ind_var27', 'ind_var41', 'ind_var46_0', 'ind_var46', 'num_var27_0',
'num_var28_0', 'num_var28', 'num_var27', 'num_var41', 'num_var46_0',
'num_var46', 'saldo_var28', 'saldo_var27', 'saldo_var41',
'saldo_var46', 'imp_amort_var18_hace3', 'imp_amort_var34_hace3',
'imp_reemb_var13_hace3', 'imp_reemb_var33_hace3',
'imp_trasp_var17_out_hace3', 'imp_trasp_var33_out_hace3',
'num_var2_0_ult1', 'num_var2_ult1', 'num_reemb_var13_hace3',
```

```
'num_reemb_var33_hace3', 'num_trasp_var17_out_hace3',
'num_trasp_var33_out_hace3', 'saldo_var2_ult1',
'saldo_medio_var13_medio_hace3']

# We can now drop colulmns/ features that are completely the same.
Which will not help in training our model.
constant_features_todrop = constant_features_dict['constant_features']
= constant_features_dict['constant_features']
# Drop the columns
training = training.drop(columns=constant_features_todrop)

training.info()

<class 'pandas.core.frame.DataFrame'>
Index: 76020 entries, 1 to 151838
Columns: 336 entries, var3 to TARGET
dtypes: float64(111), int64(225)
memory usage: 197.5 MB
```

Split Dataset

```
# Add PCA components as features
from sklearn.preprocessing import normalize
from sklearn.decomposition import PCA

X = training.iloc[:,:-1]
y = training.TARGET

X_normalized = normalize(X, axis=0)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_normalized)

# Add the paca to our dataframe frame for analysis.
X['PCA1'] = X_pca[:,0]
X['PCA2'] = X_pca[:,1]

print(X[['PCA1', 'PCA2']].isnull().sum())

PCA1    0
PCA2    0
dtype: int64

%matplotlib inline
plt.figure(figsize=(10, 6))

col_pal = ['#00204C', '#31446B', '#782170', '#958F78', '#00B050',
'#FFE945']
# Scatter plot with color based on 'Cluster'
sns.scatterplot(x='PCA1', y='PCA2', data=X, marker='o',color=
col_pal[0])
```

```
plt.title('PCA Component Scatter Plot')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.legend(title='PCA Group')

plt.show()
```




```
# Create a calss for XGB classifier model
clf = xgb.XGBClassifier()

from sklearn.feature_selection import SelectPercentile, chi2,
f_classif
from sklearn.preprocessing import Binarizer, scale
from sklearn.model_selection import cross_val_score
import numpy as np

# List of percentiles to test
percentiles = [75,76, 77, 78, 80, 85, 90, 95, 96]

# Placeholder for tracking the best percentile and its score
best_percentile = 0
best_score = 0

# Iterate over the percentiles
for p in percentiles:
```

```
    X_bin = Binarizer().fit_transform(scale(X))

    # Chi-square selection
    selectChi2 = SelectPercentile(chi2, percentile=p).fit(X_bin, y)
    X_selected = selectChi2.transform(X_bin)

    # Evaluate model performance (example using cross-validation)
    model = clf
    scores = cross_val_score(model, X_selected, y, cv=5,
scoring='accuracy')  # Adjust scoring method as needed
    avg_score = np.mean(scores)

    print(f"Percentile: {p}, Score: {avg_score}")

    # Track the best score
    if avg_score > best_score:
        best_score = avg_score
        best_percentile = p

print(f"Best percentile: {best_percentile} with score: {best_score}")
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/preprocessing/
_data.py:240: UserWarning: Numerical issues were encountered when
centering the data and might not be solved. Dataset may contain too
large values. You may need to prescale your features.
  warnings.warn(

Percentile: 75, Score: 0.9600894501446989

/opt/conda/lib/python3.10/site-packages/sklearn/preprocessing/
_data.py:240: UserWarning: Numerical issues were encountered when
centering the data and might not be solved. Dataset may contain too
large values. You may need to prescale your features.
  warnings.warn(

Percentile: 76, Score: 0.960049986845567

/opt/conda/lib/python3.10/site-packages/sklearn/preprocessing/
_data.py:240: UserWarning: Numerical issues were encountered when
centering the data and might not be solved. Dataset may contain too
large values. You may need to prescale your features.
  warnings.warn(

Percentile: 77, Score: 0.9601946856090503

/opt/conda/lib/python3.10/site-packages/sklearn/preprocessing/
_data.py:240: UserWarning: Numerical issues were encountered when
centering the data and might not be solved. Dataset may contain too
large values. You may need to prescale your features.
  warnings.warn(
```

```
Percentile: 78, Score: 0.9601946856090503

/opt/conda/lib/python3.10/site-packages/sklearn/preprocessing/
_data.py:240: UserWarning: Numerical issues were encountered when
centering the data and might not be solved. Dataset may contain too
large values. You may need to prescale your features.
  warnings.warn(

Percentile: 80, Score: 0.9601552223099186

/opt/conda/lib/python3.10/site-packages/sklearn/preprocessing/
_data.py:240: UserWarning: Numerical issues were encountered when
centering the data and might not be solved. Dataset may contain too
large values. You may need to prescale your features.
  warnings.warn(

Percentile: 85, Score: 0.9599842146803473

/opt/conda/lib/python3.10/site-packages/sklearn/preprocessing/
_data.py:240: UserWarning: Numerical issues were encountered when
centering the data and might not be solved. Dataset may contain too
large values. You may need to prescale your features.
  warnings.warn(

Percentile: 90, Score: 0.9599842146803473

/opt/conda/lib/python3.10/site-packages/sklearn/preprocessing/
_data.py:240: UserWarning: Numerical issues were encountered when
centering the data and might not be solved. Dataset may contain too
large values. You may need to prescale your features.
  warnings.warn(

Percentile: 95, Score: 0.9601552223099186

/opt/conda/lib/python3.10/site-packages/sklearn/preprocessing/
_data.py:240: UserWarning: Numerical issues were encountered when
centering the data and might not be solved. Dataset may contain too
large values. You may need to prescale your features.
  warnings.warn(

Percentile: 96, Score: 0.9601552223099186
Best percentile: 77 with score: 0.9601946856090503
```

The feature best performs by using the selected 95% of the data.

```
# Percentile for the feature selection
p = 95
X_bin = Binarizer().fit_transform(scale(X))
selectChi2 = SelectPercentile(chi2, percentile=p).fit(X_bin, y)
selectF_classif = SelectPercentile(f_classif, percentile=p).fit(X, y)
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/preprocessing/
_data.py:240: UserWarning: Numerical issues were encountered when
centering the data and might not be solved. Dataset may contain too
large values. You may need to prescale your features.
  warnings.warn(

chi2_selected = selectChi2.get_support()
chi2_selected_features = [ f for i,f in enumerate(X.columns) if
chi2_selected[i]]


print('Chi2 selected {} features
{}.'.format(chi2_selected.sum(),chi2_selected_features))

Chi2 selected 320 features ['var3', 'var15', 'imp_ent_var16_ult1',
'imp_op_var39_comer_ult1', 'imp_op_var39_comer_ult3',
'imp_op_var40_comer_ult1', 'imp_op_var40_comer_ult3',
'imp_op_var40_efect_ult1', 'imp_op_var40_efect_ult3',
'imp_op_var40_ult1', 'imp_op_var41_comer_ult1',
'imp_op_var41_comer_ult3', 'imp_op_var41_efect_ult1',
'imp_op_var41_efect_ult3', 'imp_op_var41_ult1',
'imp_op_var39_efect_ult1', 'imp_op_var39_efect_ult3',
'imp_op_var39_ult1', 'imp_sal_var16_ult1', 'ind_var1_0', 'ind_var1',
'ind_var5_0', 'ind_var5', 'ind_var6_0', 'ind_var6', 'ind_var8_0',
'ind_var8', 'ind_var12_0', 'ind_var12', 'ind_var13_0',
'ind_var13_corto_0', 'ind_var13_corto', 'ind_var13_largo_0',
'ind_var13_largo', 'ind_var13_medio_0', 'ind_var13_medio',
'ind_var13', 'ind_var14_0', 'ind_var14', 'ind_var17_0', 'ind_var17',
'ind_var18_0', 'ind_var18', 'ind_var19', 'ind_var20_0', 'ind_var20',
'ind_var24_0', 'ind_var24', 'ind_var25_cte', 'ind_var26_0',
'ind_var26_cte', 'ind_var26', 'ind_var25_0', 'ind_var25',
'ind_var29_0', 'ind_var29', 'ind_var30', 'ind_var31_0', 'ind_var31',
'ind_var32_cte', 'ind_var33_0', 'ind_var33', 'ind_var34_0',
'ind_var34', 'ind_var37_cte', 'ind_var37_0', 'ind_var37',
'ind_var39_0', 'ind_var40_0', 'ind_var40', 'ind_var41_0', 'ind_var39',
'ind_var44_0', 'ind_var44', 'num_var1_0', 'num_var1', 'num_var4',
'num_var5_0', 'num_var5', 'num_var6_0', 'num_var6', 'num_var8_0',
'num_var8', 'num_var12_0', 'num_var12', 'num_var13_0',
'num_var13_corto_0', 'num_var13_corto', 'num_var13_largo_0',
'num_var13_largo', 'num_var13_medio_0', 'num_var13_medio',
'num_var13', 'num_var14_0', 'num_var14', 'num_var17_0', 'num_var17',
'num_var18_0', 'num_var18', 'num_var20_0', 'num_var20', 'num_var24_0',
'num_var24', 'num_var26_0', 'num_var26', 'num_var25_0', 'num_var25',
'num_op_var40_hace2', 'num_op_var40_hace3', 'num_op_var40_ult1',
'num_op_var40_ult3', 'num_op_var41_hace2', 'num_op_var41_hace3',
'num_op_var41_ult1', 'num_op_var41_ult3', 'num_op_var39_hace2',
'num_op_var39_hace3', 'num_op_var39_ult1', 'num_op_var39_ult3',
'num_var29_0', 'num_var29', 'num_var30_0', 'num_var30', 'num_var31_0',
'num_var31', 'num_var33_0', 'num_var33', 'num_var34_0', 'num_var34',
'num_var35', 'num_var37_med_ult2', 'num_var37_0', 'num_var37',
```

```
'num_var39_0', 'num_var40_0', 'num_var40', 'num_var41_0', 'num_var39',
'num_var42_0', 'num_var42', 'num_var44_0', 'num_var44', 'saldo_var1',
'saldo_var5', 'saldo_var6', 'saldo_var8', 'saldo_var12',
'saldo_var13_corto', 'saldo_var13_largo', 'saldo_var13_medio',
'saldo_var13', 'saldo_var14', 'saldo_var17', 'saldo_var18',
'saldo_var20', 'saldo_var24', 'saldo_var26', 'saldo_var25',
'saldo_var29', 'saldo_var30', 'saldo_var31', 'saldo_var33',
'saldo_var34', 'saldo_var37', 'saldo_var40', 'saldo_var42',
'saldo_var44', 'var36', 'delta_imp_amort_var18_1y3',
'delta_imp_amort_var34_1y3', 'delta_imp_aport_var13_1y3',
'delta_imp_aport_var17_1y3', 'delta_imp_aport_var33_1y3',
'delta_imp_compra_var44_1y3', 'delta_imp_reemb_var13_1y3',
'delta_imp_reemb_var17_1y3', 'delta_imp_reemb_var33_1y3',
'delta_imp_trasp_var17_in_1y3', 'delta_imp_trasp_var17_out_1y3',
'delta_imp_trasp_var33_in_1y3', 'delta_imp_trasp_var33_out_1y3',
'delta_imp_venta_var44_1y3', 'delta_num_aport_var13_1y3',
'delta_num_aport_var17_1y3', 'delta_num_compra_var44_1y3',
'delta_num_reemb_var13_1y3', 'delta_num_reemb_var17_1y3',
'delta_num_trasp_var17_in_1y3', 'delta_num_trasp_var17_out_1y3',
'delta_num_trasp_var33_in_1y3', 'delta_num_venta_var44_1y3',
'imp_amort_var18_ult1', 'imp_amort_var34_ult1',
'imp_aport_var13_hace3', 'imp_aport_var13_ult1',
'imp_aport_var17_hace3', 'imp_aport_var17_ult1',
'imp_aport_var33_hace3', 'imp_aport_var33_ult1', 'imp_var7_emit_ult1',
'imp_var7_recib_ult1', 'imp_compra_var44_hace3',
'imp_compra_var44_ult1', 'imp_reemb_var13_ult1',
'imp_reemb_var17_ult1', 'imp_var43_emit_ult1', 'imp_trans_var37_ult1',
'imp_trasp_var17_in_hace3', 'imp_trasp_var17_in_ult1',
'imp_trasp_var17_out_ult1', 'imp_trasp_var33_in_hace3',
'imp_trasp_var33_in_ult1', 'imp_venta_var44_hace3',
'imp_venta_var44_ult1', 'ind_var7_emit_ult1', 'ind_var7_recib_ult1',
'ind_var10_ult1', 'ind_var10cte_ult1', 'ind_var9_cte_ult1',
'ind_var9_ult1', 'ind_var43_emit_ult1', 'ind_var43_recib_ult1',
'var21', 'num_aport_var13_hace3', 'num_aport_var13_ult1',
'num_aport_var17_hace3', 'num_aport_var17_ult1',
'num_aport_var33_hace3', 'num_aport_var33_ult1', 'num_var7_emit_ult1',
'num_var7_recib_ult1', 'num_compra_var44_hace3',
'num_compra_var44_ult1', 'num_ent_var16_ult1', 'num_var22_hace2',
'num_var22_hace3', 'num_var22_ult1', 'num_var22_ult3',
'num_med_var22_ult3', 'num_med_var45_ult3', 'num_meses_var5_ult3',
'num_meses_var8_ult3', 'num_meses_var12_ult3',
'num_meses_var13_corto_ult3', 'num_meses_var13_largo_ult3',
'num_meses_var13_medio_ult3', 'num_meses_var17_ult3',
'num_meses_var29_ult3', 'num_meses_var33_ult3',
'num_meses_var39_vig_ult3', 'num_meses_var44_ult3',
'num_op_var39_comer_ult1', 'num_op_var40_comer_ult1',
'num_op_var40_comer_ult3', 'num_op_var40_efect_ult1',
'num_op_var40_efect_ult3', 'num_op_var41_comer_ult1',
'num_op_var41_comer_ult3', 'num_op_var41_efect_ult1',
```

```
'num_op_var41_efect_ult3', 'num_op_var39_efect_ult1',
'num_op_var39_efect_ult3', 'num_reemb_var13_ult1',
'num_reemb_var17_ult1', 'num_sal_var16_ult1', 'num_var43_emit_ult1',
'num_var43_recib_ult1', 'num_trasp_var11_ult1',
'num_trasp_var17_in_hace3', 'num_trasp_var17_in_ult1',
'num_trasp_var17_out_ult1', 'num_trasp_var33_in_hace3',
'num_trasp_var33_in_ult1', 'num_venta_var44_hace3',
'num_venta_var44_ult1', 'num_var45_hace2', 'num_var45_hace3',
'num_var45_ult1', 'num_var45_ult3', 'saldo_medio_var5_hace2',
'saldo_medio_var5_hace3', 'saldo_medio_var5_ult1',
'saldo_medio_var5_ult3', 'saldo_medio_var8_hace2',
'saldo_medio_var8_hace3', 'saldo_medio_var8_ult1',
'saldo_medio_var8_ult3', 'saldo_medio_var12_hace2',
'saldo_medio_var12_hace3', 'saldo_medio_var12_ult1',
'saldo_medio_var12_ult3', 'saldo_medio_var13_corto_hace2',
'saldo_medio_var13_corto_hace3', 'saldo_medio_var13_corto_ult1',
'saldo_medio_var13_corto_ult3', 'saldo_medio_var13_largo_hace2',
'saldo_medio_var13_largo_hace3', 'saldo_medio_var13_largo_ult1',
'saldo_medio_var13_largo_ult3', 'saldo_medio_var13_medio_hace2',
'saldo_medio_var13_medio_ult1', 'saldo_medio_var13_medio_ult3',
'saldo_medio_var17_hace2', 'saldo_medio_var17_hace3',
'saldo_medio_var17_ult1', 'saldo_medio_var17_ult3',
'saldo_medio_var29_hace2', 'saldo_medio_var29_ult1',
'saldo_medio_var29_ult3', 'saldo_medio_var33_hace2',
'saldo_medio_var33_hace3', 'saldo_medio_var33_ult1',
'saldo_medio_var33_ult3', 'saldo_medio_var44_hace2',
'saldo_medio_var44_hace3', 'saldo_medio_var44_ult1',
'saldo_medio_var44_ult3', 'var38', 'PCA1', 'PCA2'].

f_classif_selected = selectF_classif.get_support()
f_classif_selected_features = [ f for i,f in enumerate(X.columns) if
f_classif_selected[i]]
print('F_classif selected {} features
{}.'.format(f_classif_selected.sum(),f_classif_selected_features))

F_classif selected 320 features ['var3', 'var15',
'imp_op_var39_comer_ult1', 'imp_op_var39_comer_ult3',
'imp_op_var40_comer_ult1', 'imp_op_var40_efect_ult1',
'imp_op_var40_efect_ult3', 'imp_op_var40_ult1',
'imp_op_var41_comer_ult1', 'imp_op_var41_comer_ult3',
'imp_op_var41_efect_ult1', 'imp_op_var41_efect_ult3',
'imp_op_var41_ult1', 'imp_op_var39_efect_ult1',
'imp_op_var39_efect_ult3', 'imp_op_var39_ult1', 'ind_var1_0',
'ind_var1', 'ind_var5_0', 'ind_var5', 'ind_var6_0', 'ind_var6',
'ind_var8_0', 'ind_var8', 'ind_var12_0', 'ind_var12', 'ind_var13_0',
'ind_var13_corto_0', 'ind_var13_corto', 'ind_var13_largo_0',
'ind_var13_largo', 'ind_var13_medio_0', 'ind_var13_medio',
'ind_var13', 'ind_var14_0', 'ind_var14', 'ind_var17_0', 'ind_var17',
'ind_var18_0', 'ind_var18', 'ind_var19', 'ind_var20_0', 'ind_var20',
'ind_var24_0', 'ind_var24', 'ind_var25_cte', 'ind_var26_0',
```

```
'ind_var26_cte', 'ind_var26', 'ind_var25_0', 'ind_var25',
'ind_var29_0', 'ind_var29', 'ind_var30_0', 'ind_var30', 'ind_var31_0',
'ind_var31', 'ind_var32_cte', 'ind_var33_0', 'ind_var33',
'ind_var34_0', 'ind_var34', 'ind_var37_cte', 'ind_var37_0',
'ind_var37', 'ind_var39_0', 'ind_var40_0', 'ind_var40', 'ind_var41_0',
'ind_var39', 'ind_var44_0', 'ind_var44', 'num_var1_0', 'num_var1',
'num_var4', 'num_var5_0', 'num_var5', 'num_var6_0', 'num_var6',
'num_var8_0', 'num_var8', 'num_var12_0', 'num_var12', 'num_var13_0',
'num_var13_corto_0', 'num_var13_corto', 'num_var13_largo_0',
'num_var13_largo', 'num_var13_medio_0', 'num_var13_medio',
'num_var13', 'num_var14_0', 'num_var14', 'num_var17_0', 'num_var17',
'num_var18_0', 'num_var18', 'num_var20_0', 'num_var20', 'num_var24_0',
'num_var24', 'num_var26_0', 'num_var26', 'num_var25_0', 'num_var25',
'num_op_var40_hace2', 'num_op_var40_hace3', 'num_op_var40_ult1',
'num_op_var41_hace2', 'num_op_var41_hace3', 'num_op_var41_ult1',
'num_op_var41_ult3', 'num_op_var39_hace2', 'num_op_var39_hace3',
'num_op_var39_ult1', 'num_op_var39_ult3', 'num_var29_0', 'num_var29',
'num_var30_0', 'num_var30', 'num_var31_0', 'num_var31', 'num_var33_0',
'num_var33', 'num_var34_0', 'num_var34', 'num_var35', 'num_var37_0',
'num_var37', 'num_var39_0', 'num_var40_0', 'num_var40', 'num_var41_0',
'num_var39', 'num_var42_0', 'num_var42', 'num_var44_0', 'num_var44',
'saldo_var5', 'saldo_var6', 'saldo_var8', 'saldo_var12',
'saldo_var13_corto', 'saldo_var13_largo', 'saldo_var13_medio',
'saldo_var13', 'saldo_var14', 'saldo_var17', 'saldo_var18',
'saldo_var20', 'saldo_var24', 'saldo_var26', 'saldo_var25',
'saldo_var29', 'saldo_var30', 'saldo_var31', 'saldo_var33',
'saldo_var34', 'saldo_var37', 'saldo_var40', 'saldo_var42',
'saldo_var44', 'var36', 'delta_imp_amort_var18_1y3',
'delta_imp_amort_var34_1y3', 'delta_imp_aport_var13_1y3',
'delta_imp_aport_var17_1y3', 'delta_imp_compra_var44_1y3',
'delta_imp_reemb_var13_1y3', 'delta_imp_reemb_var17_1y3',
'delta_imp_reemb_var33_1y3', 'delta_imp_trasp_var17_in_1y3',
'delta_imp_trasp_var17_out_1y3', 'delta_imp_trasp_var33_in_1y3',
'delta_imp_trasp_var33_out_1y3', 'delta_imp_venta_var44_1y3',
'delta_num_aport_var13_1y3', 'delta_num_aport_var17_1y3',
'delta_num_compra_var44_1y3', 'delta_num_reemb_var13_1y3',
'delta_num_reemb_var17_1y3', 'delta_num_reemb_var33_1y3',
'delta_num_trasp_var17_in_1y3', 'delta_num_trasp_var17_out_1y3',
'delta_num_trasp_var33_in_1y3', 'delta_num_trasp_var33_out_1y3',
'delta_num_venta_var44_1y3', 'imp_amort_var18_ult1',
'imp_amort_var34_ult1', 'imp_aport_var13_hace3',
'imp_aport_var13_ult1', 'imp_aport_var17_hace3',
'imp_aport_var17_ult1', 'imp_aport_var33_hace3',
'imp_aport_var33_ult1', 'imp_var7_emit_ult1', 'imp_var7_recib_ult1',
'imp_compra_var44_hace3', 'imp_compra_var44_ult1',
'imp_reemb_var13_ult1', 'imp_reemb_var17_hace3',
'imp_reemb_var17_ult1', 'imp_reemb_var33_ult1', 'imp_var43_emit_ult1',
'imp_trans_var37_ult1', 'imp_trasp_var17_in_hace3',
'imp_trasp_var17_in_ult1', 'imp_trasp_var17_out_ult1',
```

```
'imp_trasp_var33_in_hace3', 'imp_trasp_var33_in_ult1',
'imp_trasp_var33_out_ult1', 'imp_venta_var44_hace3',
'imp_venta_var44_ult1', 'ind_var7_emit_ult1', 'ind_var7_recib_ult1',
'ind_var10_ult1', 'ind_var10cte_ult1', 'ind_var9_cte_ult1',
'ind_var9_ult1', 'ind_var43_emit_ult1', 'ind_var43_recib_ult1',
'var21', 'num_aport_var13_hace3', 'num_aport_var13_ult1',
'num_aport_var17_hace3', 'num_aport_var17_ult1',
'num_aport_var33_hace3', 'num_aport_var33_ult1', 'num_var7_emit_ult1',
'num_compra_var44_hace3', 'num_compra_var44_ult1',
'num_ent_var16_ult1', 'num_var22_hace2', 'num_var22_hace3',
'num_var22_ult1', 'num_var22_ult3', 'num_med_var22_ult3',
'num_med_var45_ult3', 'num_meses_var5_ult3', 'num_meses_var8_ult3',
'num_meses_var12_ult3', 'num_meses_var13_corto_ult3',
'num_meses_var13_largo_ult3', 'num_meses_var13_medio_ult3',
'num_meses_var17_ult3', 'num_meses_var29_ult3',
'num_meses_var33_ult3', 'num_meses_var39_vig_ult3',
'num_meses_var44_ult3', 'num_op_var39_comer_ult1',
'num_op_var39_comer_ult3', 'num_op_var40_comer_ult1',
'num_op_var40_comer_ult3', 'num_op_var40_efect_ult1',
'num_op_var40_efect_ult3', 'num_op_var41_comer_ult1',
'num_op_var41_comer_ult3', 'num_op_var41_efect_ult1',
'num_op_var41_efect_ult3', 'num_op_var39_efect_ult1',
'num_op_var39_efect_ult3', 'num_reemb_var13_ult1',
'num_reemb_var17_hace3', 'num_reemb_var17_ult1',
'num_reemb_var33_ult1', 'num_sal_var16_ult1', 'num_var43_emit_ult1',
'num_var43_recib_ult1', 'num_trasp_var11_ult1',
'num_trasp_var17_in_hace3', 'num_trasp_var17_in_ult1',
'num_trasp_var17_out_ult1', 'num_trasp_var33_in_hace3',
'num_trasp_var33_in_ult1', 'num_trasp_var33_out_ult1',
'num_venta_var44_hace3', 'num_venta_var44_ult1', 'num_var45_hace2',
'num_var45_hace3', 'num_var45_ult1', 'num_var45_ult3',
'saldo_medio_var5_hace2', 'saldo_medio_var5_hace3',
'saldo_medio_var5_ult1', 'saldo_medio_var5_ult3',
'saldo_medio_var8_hace2', 'saldo_medio_var8_hace3',
'saldo_medio_var8_ult1', 'saldo_medio_var8_ult3',
'saldo_medio_var12_hace2', 'saldo_medio_var12_hace3',
'saldo_medio_var12_ult1', 'saldo_medio_var12_ult3',
'saldo_medio_var13_corto_hace2', 'saldo_medio_var13_corto_hace3',
'saldo_medio_var13_corto_ult1', 'saldo_medio_var13_corto_ult3',
'saldo_medio_var13_largo_hace2', 'saldo_medio_var13_largo_hace3',
'saldo_medio_var13_largo_ult1', 'saldo_medio_var13_largo_ult3',
'saldo_medio_var13_medio_hace2', 'saldo_medio_var13_medio_ult1',
'saldo_medio_var13_medio_ult3', 'saldo_medio_var17_hace2',
'saldo_medio_var17_hace3', 'saldo_medio_var29_hace2',
'saldo_medio_var29_ult1', 'saldo_medio_var29_ult3',
'saldo_medio_var33_hace2', 'saldo_medio_var33_hace3',
'saldo_medio_var33_ult1', 'saldo_medio_var33_ult3',
'saldo_medio_var44_hace2', 'saldo_medio_var44_hace3',
```

```
'saldo_medio_var44_ult1', 'saldo_medio_var44_ult3', 'var38', 'PCA1',
'PCA2'].

selected = chi2_selected & f_classif_selected
print('Chi2 & F_classif selected {} features'.format(selected.sum()))
features = [ f for f,s in zip(X.columns, selected) if s]
print (features)

Chi2 & F_classif selected 310 features
['var3', 'var15', 'imp_op_var39_comer_ult1',
'imp_op_var39_comer_ult3', 'imp_op_var40_comer_ult1',
'imp_op_var40_efect_ult1', 'imp_op_var40_efect_ult3',
'imp_op_var40_ult1', 'imp_op_var41_comer_ult1',
'imp_op_var41_comer_ult3', 'imp_op_var41_efect_ult1',
'imp_op_var41_efect_ult3', 'imp_op_var41_ult1',
'imp_op_var39_efect_ult1', 'imp_op_var39_efect_ult3',
'imp_op_var39_ult1', 'ind_var1_0', 'ind_var1', 'ind_var5_0',
'ind_var5', 'ind_var6_0', 'ind_var6', 'ind_var8_0', 'ind_var8',
'ind_var12_0', 'ind_var12', 'ind_var13_0', 'ind_var13_corto_0',
'ind_var13_corto', 'ind_var13_largo_0', 'ind_var13_largo',
'ind_var13_medio_0', 'ind_var13_medio', 'ind_var13', 'ind_var14_0',
'ind_var14', 'ind_var17_0', 'ind_var17', 'ind_var18_0', 'ind_var18',
'ind_var19', 'ind_var20_0', 'ind_var20', 'ind_var24_0', 'ind_var24',
'ind_var25_cte', 'ind_var26_0', 'ind_var26_cte', 'ind_var26',
'ind_var25_0', 'ind_var25', 'ind_var29_0', 'ind_var29', 'ind_var30',
'ind_var31_0', 'ind_var31', 'ind_var32_cte', 'ind_var33_0',
'ind_var33', 'ind_var34_0', 'ind_var34', 'ind_var37_cte',
'ind_var37_0', 'ind_var37', 'ind_var39_0', 'ind_var40_0', 'ind_var40',
'ind_var41_0', 'ind_var39', 'ind_var44_0', 'ind_var44', 'num_var1_0',
'num_var1', 'num_var4', 'num_var5_0', 'num_var5', 'num_var6_0',
'num_var6', 'num_var8_0', 'num_var8', 'num_var12_0', 'num_var12',
'num_var13_0', 'num_var13_corto_0', 'num_var13_corto',
'num_var13_largo_0', 'num_var13_largo', 'num_var13_medio_0',
'num_var13_medio', 'num_var13', 'num_var14_0', 'num_var14',
'num_var17_0', 'num_var17', 'num_var18_0', 'num_var18', 'num_var20_0',
'num_var20', 'num_var24_0', 'num_var24', 'num_var26_0', 'num_var26',
'num_var25_0', 'num_var25', 'num_op_var40_hace2',
'num_op_var40_hace3', 'num_op_var40_ult1', 'num_op_var41_hace2',
'num_op_var41_hace3', 'num_op_var41_ult1', 'num_op_var41_ult3',
'num_op_var39_hace2', 'num_op_var39_hace3', 'num_op_var39_ult1',
'num_op_var39_ult3', 'num_var29_0', 'num_var29', 'num_var30_0',
'num_var30', 'num_var31_0', 'num_var31', 'num_var33_0', 'num_var33',
'num_var34_0', 'num_var34', 'num_var35', 'num_var37_0', 'num_var37',
'num_var39_0', 'num_var40_0', 'num_var40', 'num_var41_0', 'num_var39',
'num_var42_0', 'num_var42', 'num_var44_0', 'num_var44', 'saldo_var5',
'saldo_var6', 'saldo_var8', 'saldo_var12', 'saldo_var13_corto',
'saldo_var13_largo', 'saldo_var13_medio', 'saldo_var13',
'saldo_var14', 'saldo_var17', 'saldo_var18', 'saldo_var20',
'saldo_var24', 'saldo_var26', 'saldo_var25', 'saldo_var29',
'saldo_var30', 'saldo_var31', 'saldo_var33', 'saldo_var34',
```

```
'saldo_var37', 'saldo_var40', 'saldo_var42', 'saldo_var44', 'var36',
'delta_imp_amort_var18_1y3', 'delta_imp_amort_var34_1y3',
'delta_imp_aport_var13_1y3', 'delta_imp_aport_var17_1y3',
'delta_imp_compra_var44_1y3', 'delta_imp_reemb_var13_1y3',
'delta_imp_reemb_var17_1y3', 'delta_imp_reemb_var33_1y3',
'delta_imp_trasp_var17_in_1y3', 'delta_imp_trasp_var17_out_1y3',
'delta_imp_trasp_var33_in_1y3', 'delta_imp_trasp_var33_out_1y3',
'delta_imp_venta_var44_1y3', 'delta_num_aport_var13_1y3',
'delta_num_aport_var17_1y3', 'delta_num_compra_var44_1y3',
'delta_num_reemb_var13_1y3', 'delta_num_reemb_var17_1y3',
'delta_num_trasp_var17_in_1y3', 'delta_num_trasp_var17_out_1y3',
'delta_num_trasp_var33_in_1y3', 'delta_num_venta_var44_1y3',
'imp_amort_var18_ult1', 'imp_amort_var34_ult1',
'imp_aport_var13_hace3', 'imp_aport_var13_ult1',
'imp_aport_var17_hace3', 'imp_aport_var17_ult1',
'imp_aport_var33_hace3', 'imp_aport_var33_ult1', 'imp_var7_emit_ult1',
'imp_var7_recib_ult1', 'imp_compra_var44_hace3',
'imp_compra_var44_ult1', 'imp_reemb_var13_ult1',
'imp_reemb_var17_ult1', 'imp_var43_emit_ult1', 'imp_trans_var37_ult1',
'imp_trasp_var17_in_hace3', 'imp_trasp_var17_in_ult1',
'imp_trasp_var17_out_ult1', 'imp_trasp_var33_in_hace3',
'imp_trasp_var33_in_ult1', 'imp_venta_var44_hace3',
'imp_venta_var44_ult1', 'ind_var7_emit_ult1', 'ind_var7_recib_ult1',
'ind_var10_ult1', 'ind_var10cte_ult1', 'ind_var9_cte_ult1',
'ind_var9_ult1', 'ind_var43_emit_ult1', 'ind_var43_recib_ult1',
'var21', 'num_aport_var13_hace3', 'num_aport_var13_ult1',
'num_aport_var17_hace3', 'num_aport_var17_ult1',
'num_aport_var33_hace3', 'num_aport_var33_ult1', 'num_var7_emit_ult1',
'num_compra_var44_hace3', 'num_compra_var44_ult1',
'num_ent_var16_ult1', 'num_var22_hace2', 'num_var22_hace3',
'num_var22_ult1', 'num_var22_ult3', 'num_med_var22_ult3',
'num_med_var45_ult3', 'num_meses_var5_ult3', 'num_meses_var8_ult3',
'num_meses_var12_ult3', 'num_meses_var13_corto_ult3',
'num_meses_var13_largo_ult3', 'num_meses_var13_medio_ult3',
'num_meses_var17_ult3', 'num_meses_var29_ult3',
'num_meses_var33_ult3', 'num_meses_var39_vig_ult3',
'num_meses_var44_ult3', 'num_op_var39_comer_ult1',
'num_op_var40_comer_ult1', 'num_op_var40_comer_ult3',
'num_op_var40_efect_ult1', 'num_op_var40_efect_ult3',
'num_op_var41_comer_ult1', 'num_op_var41_comer_ult3',
'num_op_var41_efect_ult1', 'num_op_var41_efect_ult3',
'num_op_var39_efect_ult1', 'num_op_var39_efect_ult3',
'num_reemb_var13_ult1', 'num_reemb_var17_ult1', 'num_sal_var16_ult1',
'num_var43_emit_ult1', 'num_var43_recib_ult1', 'num_trasp_var11_ult1',
'num_trasp_var17_in_hace3', 'num_trasp_var17_in_ult1',
'num_trasp_var17_out_ult1', 'num_trasp_var33_in_hace3',
'num_trasp_var33_in_ult1', 'num_venta_var44_hace3',
'num_venta_var44_ult1', 'num_var45_hace2', 'num_var45_hace3',
'num_var45_ult1', 'num_var45_ult3', 'saldo_medio_var5_hace2',
```

```
'saldo_medio_var5_hace3', 'saldo_medio_var5_ult1',
'saldo_medio_var5_ult3', 'saldo_medio_var8_hace2',
'saldo_medio_var8_hace3', 'saldo_medio_var8_ult1',
'saldo_medio_var8_ult3', 'saldo_medio_var12_hace2',
'saldo_medio_var12_hace3', 'saldo_medio_var12_ult1',
'saldo_medio_var12_ult3', 'saldo_medio_var13_corto_hace2',
'saldo_medio_var13_corto_hace3', 'saldo_medio_var13_corto_ult1',
'saldo_medio_var13_corto_ult3', 'saldo_medio_var13_largo_hace2',
'saldo_medio_var13_largo_hace3', 'saldo_medio_var13_largo_ult1',
'saldo_medio_var13_largo_ult3', 'saldo_medio_var13_medio_hace2',
'saldo_medio_var13_medio_ult1', 'saldo_medio_var13_medio_ult3',
'saldo_medio_var17_hace2', 'saldo_medio_var17_hace3',
'saldo_medio_var29_hace2', 'saldo_medio_var29_ult1',
'saldo_medio_var29_ult3', 'saldo_medio_var33_hace2',
'saldo_medio_var33_hace3', 'saldo_medio_var33_ult1',
'saldo_medio_var33_ult3', 'saldo_medio_var44_hace2',
'saldo_medio_var44_hace3', 'saldo_medio_var44_ult1',
'saldo_medio_var44_ult3', 'var38', 'PCA1', 'PCA2']
```

selected

```
array([ True,  True, False,  True,  True,  True, False,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
       False,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True, False,  True,  True,  True,  True, False, False,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True, False,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True, False, False,  True,  True,  True,  True,  True,
       False,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True, False,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True, False,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True, False,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True, False,  True,  True,  True, False,  True,  True,  True,
       False,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True, False,  True, False,
        True,  True,  True,  True,  True,  True,  True, False,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True, False,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
```

```
       True,  True,  True, False,  True,  True,  True,  True,  True,
       True,  True,  True,  True,  True,  True, False,  True, False,
       True,  True,  True,  True,  True,  True,  True,  True,  True,
      False,  True,  True,  True,  True,  True,  True,  True,  True,
       True,  True,  True,  True,  True,  True,  True,  True,  True,
       True,  True,  True,  True,  True,  True,  True,  True,  True,
       True,  True,  True,  True,  True, False, False,  True, False,
       True,  True,  True,  True,  True,  True,  True,  True,  True,
       True,  True,  True,  True])
```

```
from sklearn import model_selection
X_sel = X[features]
X_train, X_test, y_train, y_test =
model_selection.train_test_split(X_sel, y, random_state=1301,
stratify=y, test_size=0.4)

clf = xgb.XGBClassifier()

clf.fit(X_train, y_train, early_stopping_rounds=50,
eval_metric="auc",eval_set=[(X_train, y_train), (X_test, y_test)])
```

```
/opt/conda/lib/python3.10/site-packages/xgboost/sklearn.py:889:
UserWarning: `eval_metric` in `fit` method is deprecated for better
compatibility with scikit-learn, use `eval_metric` in constructor
or`set_params` instead.
  warnings.warn(
/opt/conda/lib/python3.10/site-packages/xgboost/sklearn.py:889:
UserWarning: `early_stopping_rounds` in `fit` method is deprecated for
better compatibility with scikit-learn, use `early_stopping_rounds` in
constructor or`set_params` instead.
  warnings.warn(
```

```
[0]  validation_0-auc:0.83404   validation_1-auc:0.82401
[1]  validation_0-auc:0.84094   validation_1-auc:0.82576
[2]  validation_0-auc:0.84970   validation_1-auc:0.82845
[3]  validation_0-auc:0.85291   validation_1-auc:0.82921
[4]  validation_0-auc:0.85909   validation_1-auc:0.83163
[5]  validation_0-auc:0.86336   validation_1-auc:0.83380
[6]  validation_0-auc:0.86744   validation_1-auc:0.83425
[7]  validation_0-auc:0.87098   validation_1-auc:0.83449
[8]  validation_0-auc:0.87331   validation_1-auc:0.83352
[9]  validation_0-auc:0.87698   validation_1-auc:0.83400
[10] validation_0-auc:0.88073   validation_1-auc:0.83354
[11] validation_0-auc:0.88266   validation_1-auc:0.83438
[12] validation_0-auc:0.88468   validation_1-auc:0.83437
[13] validation_0-auc:0.88791   validation_1-auc:0.83471
[14] validation_0-auc:0.88941   validation_1-auc:0.83459
[15] validation_0-auc:0.89104   validation_1-auc:0.83375
[16] validation_0-auc:0.89241   validation_1-auc:0.83374
[17] validation_0-auc:0.89431   validation_1-auc:0.83336
```

```
[18]  validation_0-auc:0.89497   validation_1-auc:0.83340
[19]  validation_0-auc:0.89631   validation_1-auc:0.83289
[20]  validation_0-auc:0.89852   validation_1-auc:0.83213
[21]  validation_0-auc:0.90104   validation_1-auc:0.83124
[22]  validation_0-auc:0.90178   validation_1-auc:0.83083
[23]  validation_0-auc:0.90249   validation_1-auc:0.83100
[24]  validation_0-auc:0.90319   validation_1-auc:0.83062
[25]  validation_0-auc:0.90477   validation_1-auc:0.83005
[26]  validation_0-auc:0.90577   validation_1-auc:0.82989
[27]  validation_0-auc:0.90622   validation_1-auc:0.83014
[28]  validation_0-auc:0.90655   validation_1-auc:0.82997
[29]  validation_0-auc:0.90708   validation_1-auc:0.83020
[30]  validation_0-auc:0.90783   validation_1-auc:0.83018
[31]  validation_0-auc:0.90820   validation_1-auc:0.83031
[32]  validation_0-auc:0.90846   validation_1-auc:0.83012
[33]  validation_0-auc:0.90970   validation_1-auc:0.83036
[34]  validation_0-auc:0.91061   validation_1-auc:0.82990
[35]  validation_0-auc:0.91123   validation_1-auc:0.82939
[36]  validation_0-auc:0.91152   validation_1-auc:0.82935
[37]  validation_0-auc:0.91278   validation_1-auc:0.82848
[38]  validation_0-auc:0.91293   validation_1-auc:0.82826
[39]  validation_0-auc:0.91327   validation_1-auc:0.82848
[40]  validation_0-auc:0.91349   validation_1-auc:0.82848
[41]  validation_0-auc:0.91536   validation_1-auc:0.82824
[42]  validation_0-auc:0.91719   validation_1-auc:0.82753
[43]  validation_0-auc:0.91818   validation_1-auc:0.82729
[44]  validation_0-auc:0.91925   validation_1-auc:0.82689
[45]  validation_0-auc:0.92046   validation_1-auc:0.82703
[46]  validation_0-auc:0.92069   validation_1-auc:0.82726
[47]  validation_0-auc:0.92132   validation_1-auc:0.82707
[48]  validation_0-auc:0.92287   validation_1-auc:0.82712
[49]  validation_0-auc:0.92414   validation_1-auc:0.82642
[50]  validation_0-auc:0.92488   validation_1-auc:0.82564
[51]  validation_0-auc:0.92599   validation_1-auc:0.82538
[52]  validation_0-auc:0.92633   validation_1-auc:0.82507
[53]  validation_0-auc:0.92724   validation_1-auc:0.82513
[54]  validation_0-auc:0.92753   validation_1-auc:0.82521
[55]  validation_0-auc:0.92799   validation_1-auc:0.82486
[56]  validation_0-auc:0.92892   validation_1-auc:0.82459
[57]  validation_0-auc:0.92925   validation_1-auc:0.82417
[58]  validation_0-auc:0.93071   validation_1-auc:0.82369
[59]  validation_0-auc:0.93219   validation_1-auc:0.82311
[60]  validation_0-auc:0.93311   validation_1-auc:0.82285
[61]  validation_0-auc:0.93418   validation_1-auc:0.82280
[62]  validation_0-auc:0.93462   validation_1-auc:0.82256

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None,
early_stopping_rounds=None,
```

```
              enable_categorical=False, eval_metric=None,
feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None,
max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan,
monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)

from sklearn.metrics import roc_auc_score

# Make predictions with a specific number of trees (up to
best_iteration)
probs = clf.predict_proba(X_sel, iteration_range=(0,
clf.best_iteration))[:, 1]

# Calculate AUC
print('Overall AUC with best_iteration:', roc_auc_score(y, probs))

Overall AUC with best_iteration: 0.8645303729400065
```

Test Predictions

Now that we have trained our model on the training dataset, we can use it to make predictions on the unseen test data.

This section covers the basic process of using a trained ML model to generate predictions on new, previously unseen data and save them for submission/evaluation.

```
#Let's normalize the test dataset
test_normalized = normalize(test, axis=0)
#Create a class for PCA
pca = PCA(n_components=2)
#Let's fit the data to PCA
test_pca = pca.fit_transform(test_normalized)

#Create two pca components
test['PCA1'] = test_pca[:,0]
test['PCA2'] = test_pca[:,1]
sel_test = test[features]
#Use previous predictions based on the model
y_pred = clf.predict_proba(sel_test)

submission = pd.DataFrame({"ID":test.index, "TARGET":y_pred[:,1]})
submission.to_csv("submission.csv", index=False)
```

```
mapFeat = dict(zip(["f"+str(i) for i in
range(len(features))],features))
# Get the feature importance scores from the model
ts = pd.Series(clf.get_booster().get_fscore())

# The top 25 most important features
plt.figure(figsize=(15,6))
ts.sort_values()[-25:].plot(kind="barh", title="XGBoost Feature
Importance",color='green',fontsize=8)
plt.xlabel('Feature',fontsize=10,color =  col_pal[0])
plt.ylabel('F Score',fontsize=10,color =  col_pal[0]);
```