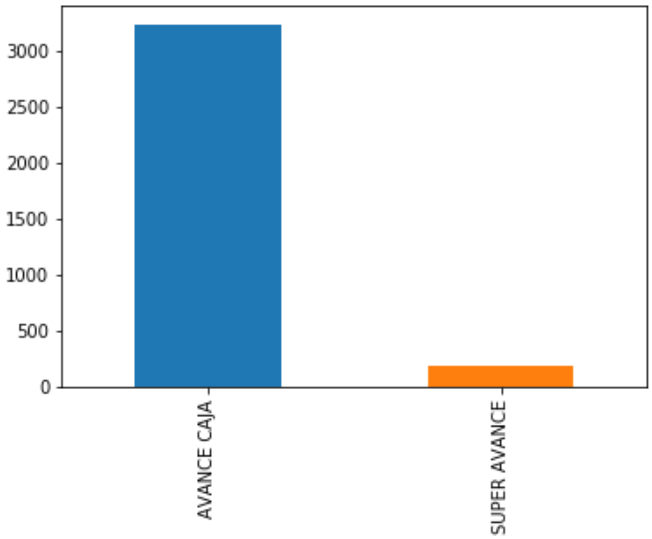
**ANÁLISIS DE CLIENTES.PARQUET, CREDITOS.PARQUET, CUENTAS.PARQUET Y DEMOGRAFICO.PARQUET PARA EL DESARROLLO DE DOS MODELOS DE MACHINE LEARNINIG, UNO BASADO EN UN ÁRBOL DE DECISIÓN UTILIZANDO SKLEARN Y OTRO BASADO EN REGRESIÓN NO LINEAL UTILIZANDO SVR (CHALLENGE).**

Aquiles Barreto

Comencemos conociendo la proporción de clientes beneficiados con ambos tipos de créditos:

```
import pandas as pd
creditos = pd.read_parquet('creditos.parquet', engine='pyarrow')
```




Es notable que:

```
creditos['nombre_transaccion'].value_counts(normalize=True)
AVANCE CAJA     0.946491
SUPER AVANCE    0.053509
Name: nombre_transaccion, dtype: float64
```

Luego quiero ver los 20 montos más altos otorgados a los primeros 20 clientes beneficiados con el tipo de crédito SUPER AVANCE:

```
df = spark.sql("SELECT customerid,monto_transaccion FROM parquet.`creditos.parquet`
WHERE nombre_transaccion = 'SUPER AVANCE' ORDER BY monto_transaccion desc")
df.show()
```

```
+----------+-----------------+------------+
|customerid|monto_transaccion|fecha_posteo|
+----------+-----------------+------------+
|     40214|       5370569.00|  2017-05-22|
|     40214|       5370569.00|  2017-05-18|
|      5120|       5370569.00|  2017-05-16|
|     87958|       5370569.00|  2017-05-15|
|     11271|       5005371.00|  2017-05-04|
|     81587|       5000000.00|  2017-05-12|
|     44542|       5000000.00|  2017-05-16|
|     16227|       5000000.00|  2017-05-04|
|     93189|       5000000.00|  2017-05-19|
|     44542|       5000000.00|  2017-05-16|
```

```
|     81587|       5000000.00|  2017-05-09|
|     10215|       5000000.00|  2017-05-22|
|     87109|       5000000.00|  2017-05-22|
|     90301|       5000000.00|  2017-05-02|
|     47981|       5000000.00|  2017-05-02|
|     54638|       5000000.00|  2017-05-16|
|     10215|       5000000.00|  2017-05-22|
|     60013|       5000000.00|  2017-05-04|
|     44542|       5000000.00|  2017-05-16|
|     10094|       5000000.00|  2017-05-02|
+----------+-----------------+------------+
only showing top 20 rows


+----------+-----------------+------------+
|customerid|monto_transaccion|fecha_posteo|
+----------+-----------------+------------+
|     40214|       5370569.00|  2017-05-18|
|     40214|       5370569.00|  2017-05-22|
+----------+-----------------+------------+
```

De igual forma quiero conocer los 20 montos más altos otorgados a los primeros 20 clientes beneficiados con el tipo de crédito AVANCE CAJA:

df = spark.sql("SELECT customerid,monto_transaccion,fecha_posteo FROM parquet.`creditos.parquet` WHERE nombre_transaccion = 'AVANCE CAJA' ORDER BY monto_transaccion DESC")
df.show()

```
+----------+-----------------+------------+
|customerid|monto_transaccion|fecha_posteo|
+----------+-----------------+------------+
|     23189|       5370569.00|  2017-05-05|
|      1559|       5000000.00|  2017-05-30|
|     90248|       5000000.00|  2017-05-05|
|     41946|       5000000.00|  2017-05-30|
|     90248|       4781240.00|  2017-05-05|
|     18968|       4748000.00|  2017-05-03|
|      1559|       4000000.00|  2017-05-30|
|     20945|       3840000.00|  2017-05-19|
|     58138|       3500000.00|  2017-05-18|
|     63466|       3500000.00|  2017-05-24|
|     33116|       3000000.00|  2017-05-03|
|     84322|       3000000.00|  2017-05-24|
|     59816|       3000000.00|  2017-05-31|
|     73189|       3000000.00|  2017-05-30|
|     76149|       3000000.00|  2017-05-24|
|     31945|       2903983.00|  2017-05-18|
|     27067|       2600000.00|  2017-05-15|
|     23189|       2597682.00|  2017-05-05|
|     51797|       2500000.00|  2017-05-26|
|     27656|       2400000.00|  2017-05-10|
+----------+-----------------+------------+
only showing top 20 rows

+----------+-----------------+------------+
|customerid|monto_transaccion|fecha_posteo|
```

```
+----------+-----------------+------------+
|     23189|       2597682.00|  2017-05-05|
|     23189|       5370569.00|  2017-05-05|
+----------+-----------------+------------+
```

A partir de las dos tablas puedo observar que pudiera no haber distinción en cuanto a la cantidad otorgada para ambos tipos de créditos, además pudieran solicitarse más de un avance el mismo día. Sigamos.

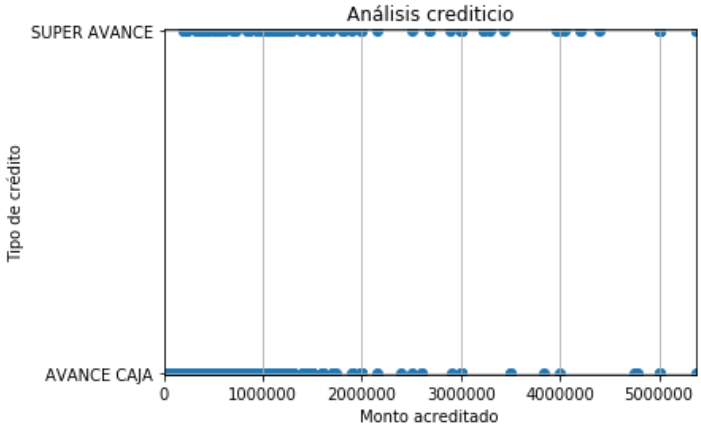
Veamos si el límite o cantidad de tarjetas influye en el monto otorgado al cliente. Preparé un INNER JOIN para ver los campos que me interesan de las tablas Créditos y Cuentas:

```
df = spark.sql("SELECT cr.customerid, cu.cant_plasticos, cu.cuponac, cu.totallimitnac, cr.nombre_transaccion, cr.monto_transaccion FROM parquet.`creditos.parquet` as cr INNER JOIN parquet.`cuentas.parquet` as cu ON cr.customerid == cu.customerid ORDER BY cr.monto_transaccion DESC")
df.show()
```

```
+----------+-------------+--------------------+--------------------+------------------+-----------------+
|customerid|cant_plasticos|            cuponac|       totallimitnac|nombre_transaccion|monto_transaccion|
+----------+-------------+--------------------+--------------------+------------------+-----------------+
|     23189|            1|4798000.000000000...|3667089.000000000...|       AVANCE CAJA|       5370569.00|
|      5120|            1|1700000.000000000...|1038636.000000000...|      SUPER AVANCE|       5370569.00|
|     40214|            2|2640000.000000000...|1056294.000000000...|      SUPER AVANCE|       5370569.00|
|     40214|            2|2640000.000000000...|1056294.000000000...|      SUPER AVANCE|       5370569.00|
|     87958|            1|3810000.000000000...|740582.0000000000...|      SUPER AVANCE|       5370569.00|
|     11271|            2|2000000.000000000...|517671.0000000000...|      SUPER AVANCE|       5005371.00|
|     60013|            2|11500000.00000000...|11004624.00000000...|      SUPER AVANCE|       5000000.00|
|     87109|            1|1266000.000000000...|319509.0000000000...|      SUPER AVANCE|       5000000.00|
|     10215|            1|6500000.000000000...|4952240.000000000...|      SUPER AVANCE|       5000000.00|
|     44542|            1|4000000.000000000...|4000000.000000000...|      SUPER AVANCE|       5000000.00|
|     47981|            1|6640000.000000000...|5991909.000000000...|      SUPER AVANCE|       5000000.00|
|      1559|            2|12100000.00000000...|489484.0000000000...|       AVANCE CAJA|       5000000.00|
|     81587|            2|2400000.000000000...|1268016.000000000...|      SUPER AVANCE|       5000000.00|
|     41946|            1|5780000.000000000...|2724586.000000000...|       AVANCE CAJA|       5000000.00|
|     44542|            1|4000000.000000000...|4000000.000000000...|      SUPER AVANCE|       5000000.00|
|     81587|            2|2400000.000000000...|1268016.000000000...|      SUPER AVANCE|       5000000.00|
|     73415|            2|1370000.000000000...|1253679.000000000...|      SUPER AVANCE|       5000000.00|
|     90301|            1|5180000.000000000...|-39673.0000000000...|      SUPER AVANCE|       5000000.00|
|     16227|            1|1200000.000000000...|-7668.00000000000...|      SUPER AVANCE|       5000000.00|
|     44542|            1|4000000.000000000...|4000000.000000000...|      SUPER AVANCE|       5000000.00|
+----------+-------------+--------------------+--------------------+------------------+-----------------+
only showing top 20 rows
```

Observamos acá que, dos clientes con un sólo plástico fueron beneficiados con el monto más alto y además la misma cantidad para préstamos incluso de diferentes tipos, esto reafirma un poco el caso de estudio de las dos tablas anteriores.

Luego dibujemos el comportamiento de los tipos de crédito versus los montos otorgados y vemos los siguiente:




Observamos acá que existe claramente un balance en cuanto a los montos otorgados para los dos tipos de crédito. Habiendo comprobado que parecieran indistinto los montos otorgados a los clientes a la hora de solicitar ambos tipos de créditos, probaremos otras variables que puedan ayuda a discriminar la decisión del otorgamiento del crédito, tales como el estado marital, el sexo y la edad.

Luego de unir las tablas de créditos y demográfico observamos:

df = spark.sql("SELECT cr.customerid, cr.nombre_transaccion, cr.monto_transaccion, de.maritalstatus, de.gender, de.edad FROM parquet.`creditos.parquet` as cr INNER JOIN parquet.`demografico.parquet` as de ON cr.customerid == de.customerid ORDER BY cr.monto_transaccion DESC")
df.show()

```
+----------+------------------+-----------------+-------------+------+----+
|customerid|nombre_transaccion|monto_transaccion|maritalstatus|gender|edad|
+----------+------------------+-----------------+-------------+------+----+
|     40214|      SUPER AVANCE|       5370569.00|       CASADO|     M|74.0|
|     40214|      SUPER AVANCE|       5370569.00|       CASADO|     M|74.0|
|      5120|      SUPER AVANCE|       5370569.00|       CASADO|     M|72.0|
|     23189|       AVANCE CAJA|       5370569.00|       CASADO|     F|71.0|
|     87958|      SUPER AVANCE|       5370569.00|      SOLTERO|     F|72.0|
|     11271|      SUPER AVANCE|       5005371.00|       CASADO|     F|75.0|
|     90301|      SUPER AVANCE|       5000000.00|       CASADO|     M|69.0|
|     93189|      SUPER AVANCE|       5000000.00|       CASADO|     M|73.0|
|     60013|      SUPER AVANCE|       5000000.00|       CASADO|     M|74.0|
|     90248|       AVANCE CAJA|       5000000.00|       CASADO|     M|75.0|
|     47981|      SUPER AVANCE|       5000000.00|      SOLTERO|     M|69.0|
|     10094|      SUPER AVANCE|       5000000.00|      SOLTERO|     M|75.0|
|     16227|      SUPER AVANCE|       5000000.00|       CASADO|     F|71.0|
|     73415|      SUPER AVANCE|       5000000.00|      SOLTERO|     M|73.0|
|     81587|      SUPER AVANCE|       5000000.00|       CASADO|     F|75.0|
|     81587|      SUPER AVANCE|       5000000.00|       CASADO|     F|75.0|
|     10215|      SUPER AVANCE|       5000000.00|       CASADO|     F|71.0|
|     10215|      SUPER AVANCE|       5000000.00|       CASADO|     F|71.0|
|     54638|      SUPER AVANCE|       5000000.00|       CASADO|     M|67.0|
|     41946|       AVANCE CAJA|       5000000.00|       CASADO|     M|75.0|
+----------+------------------+-----------------+-------------+------+----+
only showing top 20 rows
```

El máximo monto 5.370.569,00 fue otorgado indistintamente para ambos tipos de crédito a 4 personas casadas y también a 1 soltera en edades comprendidas entre la media de la muestra:

df.describe(['edad']).show()

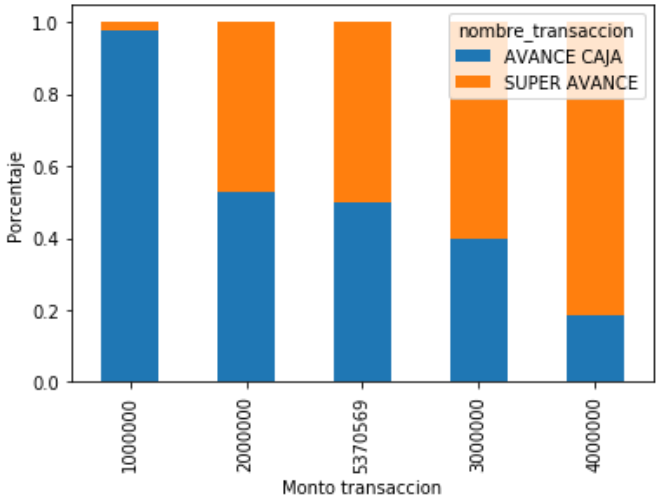
```
+-------+-----------------+
|summary|             edad|
+-------+-----------------+
|  count|             3420|
|   mean|75.30087719298245|
| stddev|7.719705764048321|
|    min|             32.0|
|    max|             96.0|
+-------+-----------------+
```

Hasta ahora pudiera inferir quizás erróneamente que el banco está otorgando ambos tipos de crédito de forma indistinta en cuanto a:

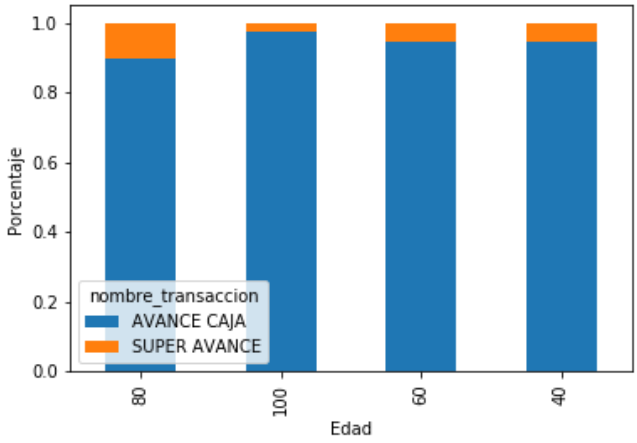
- monto_transaccion (Pareciera haber un tope diario de 5.370.569,00 pero hay un solo caso en todos los datos donde ese monto fue otorgado junto a otra cantidad en un pago aparte el mismo día a un mismo cliente, puede ser que se haya tratado de un caso especial o una excepción, el préstamo fue del tipo AVANCE CAJA y documentado anteriormente.
- cuponac
- totallimitnac
- nombre_transaccion
- maritalstatus
- gender
- edad

Al realizar un análisis multivariado del tipo de crédito a solicitar con respecto al monto de la transacción, edad, estado marital y el sexo del solicitante tenemos:

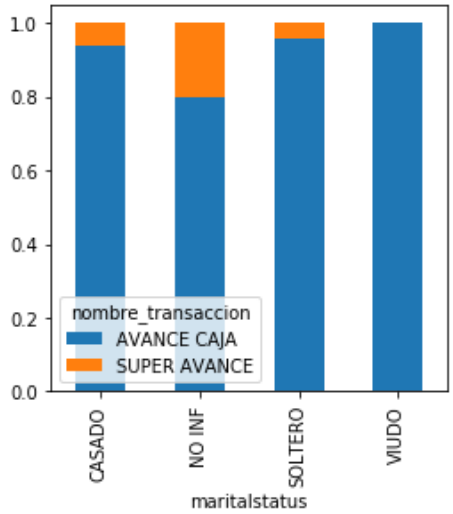
```
bins=[0,1000000,2000000,3000000,4000000,5370569]
group=['1000000','2000000','3000000','4000000','5370569']
creditos['monto_transaccion_bin']=pd.cut(creditos['monto_transaccion'],bins,labels=group)
monto_transaccion_bin=pd.crosstab(creditos['monto_transaccion_bin'],creditos['nombre_tra
nsaccion'])
monto_transaccion_bin.div(monto_transaccion_bin.sum(1).astype(float),
axis=0).plot(kind="bar", stacked=True)
plt.xlabel('Monto transaccion')
P = plt.ylabel('Porcentaje')
```



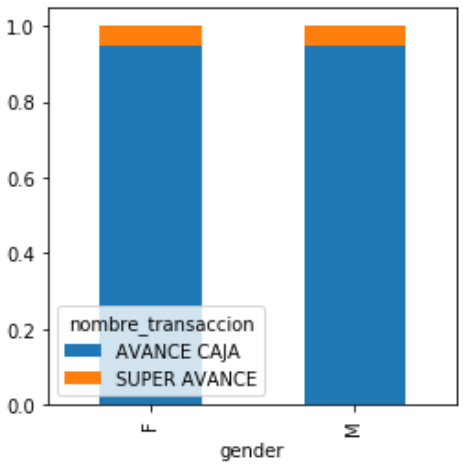

```
bins=[0,20,40,60,80,100]
group=['20','40','60','80','100']
demografico['edad_bin']=pd.cut(demografico['edad'],bins,labels=group)
monto_transaccion_bin=pd.crosstab(demografico['edad_bin'],nt)
monto_transaccion_bin.div(monto_transaccion_bin.sum(1).astype(float),
axis=0).plot(kind="bar", stacked=True)
plt.xlabel('Edad')
P = plt.ylabel('Porcentaje')
```




```
marital=pd.crosstab(ms,nt)
marital.div(marital.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(4,4))
```




```
gender=pd.crosstab(ge,nt)
gender.div(gender.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(4,4))
```




A partir del análisis multivariado previo, un modelo de Machine Learning utilizando la biblioteca de Python SKLearn basado en Tree-Based Algorithms representa una opción aceptable para construir una solución para este caso de estudio, en posteriores análisis pudieran probarse otras alternativas de Machine Learning basadas en Clasificación, Regresión o Clustering. El modelo plantea construir un árbol de decisión como se muestra a continuación:

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt

creditos = pd.read_parquet('creditos.parquet', engine='pyarrow')
demografico = pd.read_parquet('demografico.parquet', engine='pyarrow')
from pyspark.sql import SparkSession
spark = SparkSession \
    .builder \
    .appName("Challenge") \
```

```
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

df = spark.sql("SELECT cr.customerid, cr.nombre_transaccion, cr.monto_transaccion,
de.maritalstatus, de.gender, de.edad FROM parquet.`creditos.parquet` as cr INNER JOIN
parquet.`demografico.parquet` as de ON cr.customerid == de.customerid ORDER BY
cr.monto_transaccion DESC")
df.show()
```

```
+----------+-----------------+-----------------+-------------+------+----+
|customerid|nombre_transaccion|monto_transaccion|maritalstatus|gender|edad|
+----------+-----------------+-----------------+-------------+------+----+
|     40214|      SUPER AVANCE|       5370569.00|       CASADO|     M|74.0|
|      5120|      SUPER AVANCE|       5370569.00|       CASADO|     M|72.0|
|     40214|      SUPER AVANCE|       5370569.00|       CASADO|     M|74.0|
|     87958|      SUPER AVANCE|       5370569.00|      SOLTERO|     F|72.0|
|     23189|       AVANCE CAJA|       5370569.00|       CASADO|     F|71.0|
|     11271|      SUPER AVANCE|       5005371.00|       CASADO|     F|75.0|
|     90301|      SUPER AVANCE|       5000000.00|       CASADO|     M|69.0|
|     93189|      SUPER AVANCE|       5000000.00|       CASADO|     M|73.0|
|     81587|      SUPER AVANCE|       5000000.00|       CASADO|     F|75.0|
|     81587|      SUPER AVANCE|       5000000.00|       CASADO|     F|75.0|
|     47981|      SUPER AVANCE|       5000000.00|      SOLTERO|     M|69.0|
|     54638|      SUPER AVANCE|       5000000.00|       CASADO|     M|67.0|
|     41946|       AVANCE CAJA|       5000000.00|       CASADO|     M|75.0|
|     92959|      SUPER AVANCE|       5000000.00|       CASADO|     M|75.0|
|     10215|      SUPER AVANCE|       5000000.00|       CASADO|     F|71.0|
|     60013|      SUPER AVANCE|       5000000.00|       CASADO|     M|74.0|
|     73415|      SUPER AVANCE|       5000000.00|      SOLTERO|     M|73.0|
|     10094|      SUPER AVANCE|       5000000.00|      SOLTERO|     M|75.0|
|     16227|      SUPER AVANCE|       5000000.00|       CASADO|     F|71.0|
|     90248|       AVANCE CAJA|       5000000.00|       CASADO|     M|75.0|
+----------+-----------------+-----------------+-------------+------+----+
only showing top 20 rows
```

```
pdf = df.toPandas()
pdf.shape
(3420, 6)
```

Categorizamos las variables de la tabla anterior y mapeamos dichos valore a una tabla con sólo las columnas que nos interesen, eso es:

```
pdf.loc[pdf['nombre_transaccion'] == 'AVANCE CAJA', 'nombre_transaccion_map'] = 0
pdf.loc[pdf['nombre_transaccion'] == 'SUPER AVANCE', 'nombre_transaccion_map'] = 1

pdf.loc[pdf['monto_transaccion'] <= 1000000, 'monto_transaccion_map'] = 0
pdf.loc[(pdf['monto_transaccion'] > 1000000) & (pdf['monto_transaccion'] <= 2000000),
'monto_transaccion_map'] = 1
pdf.loc[(pdf['monto_transaccion'] > 2000000) & (pdf['monto_transaccion'] <= 3000000),
'monto_transaccion_map'] = 2
pdf.loc[(pdf['monto_transaccion'] > 3000000) & (pdf['monto_transaccion'] <= 4000000),
'monto_transaccion_map'] = 3
pdf.loc[(pdf['monto_transaccion'] > 4000000) & (pdf['monto_transaccion'] <= 5000000),
'monto_transaccion_map'] = 4
pdf.loc[pdf['monto_transaccion'] > 5000000, 'monto_transaccion_map'] = 5

pdf.loc[pdf['edad'] <= 40, 'edad_map'] = 0
```

```
pdf.loc[(pdf['edad'] > 41) & (pdf['edad'] <= 60), 'edad_map'] = 1
pdf.loc[(pdf['edad'] > 61) & (pdf['edad'] <= 80), 'edad_map'] = 2
pdf.loc[(pdf['edad'] > 81) & (pdf['edad'] <= 100), 'edad_map'] = 3
pdf.loc[pdf['edad'] > 100, 'edad_map'] = 4

pdf.loc[pdf['maritalstatus'] == 'SOLTERO', 'maritalstatus_map'] = 0
pdf.loc[pdf['maritalstatus'] == 'CASADO', 'maritalstatus_map'] = 1
pdf.loc[pdf['maritalstatus'] == 'VIUDO', 'maritalstatus_map'] = 2
pdf.loc[pdf['maritalstatus'] == 'NO INF', 'maritalstatus_map'] = 3

pdf.loc[pdf['gender'] == 'F', 'gender_map'] = 0
pdf.loc[pdf['gender'] == 'M', 'gender_map'] = 1
```

```
pdf.columns
```

```
Index(['customerid', 'nombre_transaccion', 'monto_transaccion',
       'maritalstatus', 'gender', 'edad', 'nombre_transaccion_map',
       'monto_transaccion_map', 'edad_map', 'maritalstatus_map',
'gender_map'],
      dtype='object')
```

```
drop_elements = ['customerid', 'monto_transaccion', 'nombre_transaccion',
    'maritalstatus', 'gender', 'edad']

pdf_map = pdf.drop(drop_elements, axis = 1)

pdf_map
```

| | nombre_transaccion_map | monto_transaccion_map | edad_map | maritalstatus_map | gender_map |
|---|---|---|---|---|---|
| **0** | 0.0 | 5.0 | 2.0 | 1.0 | 0.0 |
| **1** | 1.0 | 5.0 | 2.0 | 0.0 | 0.0 |
| **2** | 1.0 | 5.0 | 2.0 | 1.0 | 1.0 |
| **3** | 1.0 | 5.0 | 2.0 | 1.0 | 1.0 |
| **4** | 1.0 | 5.0 | 2.0 | 1.0 | 1.0 |
| **5** | 1.0 | 5.0 | 2.0 | 1.0 | 0.0 |
| **6** | 1.0 | 4.0 | 2.0 | 1.0 | 0.0 |
| **7** | 0.0 | 4.0 | 2.0 | 1.0 | 0.0 |
| **8** | 0.0 | 4.0 | 2.0 | 1.0 | 1.0 |
| **9** | 1.0 | 4.0 | 2.0 | 1.0 | 0.0 |
| **10** | 1.0 | 4.0 | 2.0 | 1.0 | 1.0 |

Verificamos la existencia de datos nulos:

```
pdf_map.isnull().sum()
```

```
nombre_transaccion_map      0
monto_transaccion_map       0
edad_map                  172
```

```
maritalstatus_map            0
gender_map                   0
dtype: int64
```

Completaré los 172 datos nulos por un valor que pudiera ser la mediana de ellos.

pdf_map['edad_map'].fillna(pdf_map['edad_map'].median(), inplace=True)

pdf_map.isnull().sum()

```
nombre_transaccion_map    0
monto_transaccion_map     0
edad_map                  0
maritalstatus_map         0
gender_map                0
dtype: int64
```

Nuestras variables deben ser enteras pasa su manejo.

```
pdf_map['nombre_transaccion_map'] = pdf_map.nombre_transaccion_map.astype(int)
pdf_map['monto_transaccion_map'] = pdf_map.monto_transaccion_map.astype(int)
pdf_map['edad_map'] = pdf_map.edad_map.astype(int)
pdf_map['maritalstatus_map'] = pdf_map.maritalstatus_map.astype(int)
pdf_map['gender_map'] = pdf_map.gender_map.astype(int)
```

Los datos ahora deben verse así:

pdf_map

| | nombre_transaccion_map | monto_transaccion_map | edad_map | maritalstatus_map | gender_map |
|---|---|---|---|---|---|
| **0** | 0 | 5 | 2 | 1 | 0 |
| **1** | 1 | 5 | 2 | 0 | 0 |
| **2** | 1 | 5 | 2 | 1 | 1 |
| **3** | 1 | 5 | 2 | 1 | 1 |
| **4** | 1 | 5 | 2 | 1 | 1 |
| **5** | 1 | 5 | 2 | 1 | 0 |
| **6** | 1 | 4 | 2 | 1 | 0 |
| **7** | 0 | 4 | 2 | 1 | 0 |
| **8** | 0 | 4 | 2 | 1 | 1 |
| **9** | 1 | 4 | 2 | 1 | 0 |
| **10** | 1 | 4 | 2 | 1 | 1 |

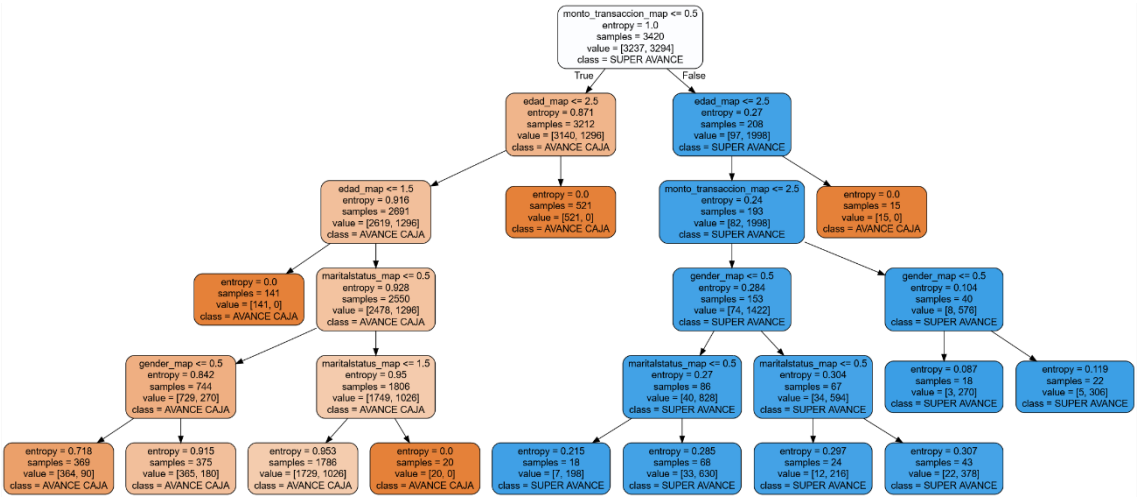
Ahora:

```
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from IPython.display import Image as PImage
from subprocess import check_call
from PIL import Image, ImageDraw, ImageFont

kf = KFold(n_splits=10)
accuracies = list()
max_attributes = len(list(pdf_map))
depth_range = range(1, max_attributes + 1)

for depth in depth_range:
   fold_accuracy = []
   tree_model                                                                                    =
tree.DecisionTreeClassifier(criterion='entropy',min_samples_split=20,min_samples_leaf=5,ma
x_depth = depth,class_weight={1:18})
   for train_fold, valid_fold in kf.split(pdf_map):
      f_train = pdf_map.loc[train_fold]
      f_valid = pdf_map.loc[valid_fold]
      model = tree_model.fit(X = f_train.drop(['nombre_transaccion_map'], axis=1),y =
f_train['nombre_transaccion_map'])
      valid_acc = model.score(X = f_valid.drop(['nombre_transaccion_map'], axis=1),y =
f_valid['nombre_transaccion_map'])
      fold_accuracy.append(valid_acc)
   avg = sum(fold_accuracy)/len(fold_accuracy)
   accuracies.append(avg)

dfr = pd.DataFrame({"Max Depth": depth_range, "Average Accuracy": accuracies})
dfr = dfr[["Max Depth", "Average Accuracy"]]
print(dfr.to_string(index=False))
```

```
Max Depth  Average Accuracy
        1          0.946491
        2          0.946491
        3          0.946491
        4          0.946491
        5          0.946491
```

```
from subprocess import check_call
import pydot
y_train = pdf_map['nombre_transaccion_map']
x_train = pdf_map.drop(['nombre_transaccion_map'], axis=1).values
decision_tree = tree.DecisionTreeClassifier(criterion='entropy', min_samples_split=20,
min_samples_leaf=5,max_depth = depth,class_weight={1:18})
decision_tree.fit(x_train, y_train)
with open(r"tree1.dot", 'w') as f:
```

```
f = tree.export_graphviz(decision_tree,out_file=f,max_depth = 7,impurity = True,feature_names = list(pdf_map.drop(['nombre_transaccion_map'], axis=1)), class_names = ['AVANCE CAJA', 'SUPER AVANCE'], rounded = True,filled= True)
check_call(['dot','-Tpng',r'tree1.dot','-o',r'tree1.png'])
PImage("tree1.png")
```




Para no romper el modelo inicial, y que este segundo análisis pueda comprobarse con el primero de ellos, construyo nuevamente 2 casos de prueba. A cada cliente se le solicitará la edad, su estado marital y el sexo a fin de determinar el crédito que se le puede otorgar y el monto del mismo.

Ahora 2 casos de prueba:

A) Un cliente con las siguientes características, edad = 72 años, maritalstatus = 'SOLTERO' y gender = 'M'.

```
if(edad <= 40):
    edad_map = 0
elif ((edad > 41) and (edad <= 60)):
    edad_map = 1
elif ((edad > 61) and (edad <= 80)):
    edad_map = 2
elif ((edad > 81) and (edad <= 100)):
    edad_map = 3
else:
    edad_map = 4

if(maritalstatus == 'SOLTERO'):
    maritalstatus_map = 0
elif (maritalstatus == 'CASADO'):
    maritalstatus_map = 1
elif (maritalstatus == 'VIUDO'):
    maritalstatus_map = 2
```

```
else:
   maritalstatus_map = 3

if(gender == 'F'):
   gender_map = 0
else:
   gender_map = 1

nt_map = pdf_map[(pdf_map['edad_map'] == edad_map) & (pdf_map['maritalstatus_map'] ==
maritalstatus_map) & (pdf_map['gender_map'] ==
gender_map)].groupby('nombre_transaccion_map').size()
mt_map = pdf_map[(pdf_map['edad_map'] == edad_map) & (pdf_map['maritalstatus_map'] ==
maritalstatus_map) & (pdf_map['gender_map'] ==
gender_map)].groupby('monto_transaccion_map').size()
nt_map = nt_map.index[len(nt_map.index)-1]
mt_map = mt_map.index[len(mt_map.index)-1]

if(nt_map == 0):
   tipo_de_credito = "AVANCE CAJA"
else:
   tipo_de_credito = "SUPER AVANCE"

if(mt_map == 0):
   monto = 1000000
elif(mt_map == 1):
   monto = 2000000
elif(mt_map == 2):
   monto = 3000000
elif(mt_map == 3):
   monto = 4000000
elif(mt_map == 4):
   monto = 5000000
elif(mt_map == 5):
   monto = 5370569

print("Tipo de crédito aprobado: " + str(tipo_de_credito) + ", Monto: " + str(monto))
x_test = pd.DataFrame(columns=('nombre_transaccion_map','monto_transaccion_map',
'edad_map', 'maritalstatus_map','gender_map'))
print(str(nt_map) + ", " + str(mt_map) + ", " + str(edad_map) + ", " + str(maritalstatus_map) + ",
" + str(gender_map))
x_test.loc[0] = (nt_map,mt_map,edad_map,maritalstatus_map,gender_map)
y_pred = decision_tree.predict(x_test.drop(['nombre_transaccion_map'], axis = 1))
if(y_pred[0]==1):
   res = "Aprobado"
else:
   res = "Rechazado"
print("Decisión: " + res)
y_proba = decision_tree.predict_proba(x_test.drop(['nombre_transaccion_map'], axis = 1))
```

```
print("Con probabilidad: " + str(round(y_proba[0][y_pred][0]* 100, 2))+"%")
```

Respuesta del modelo:

```
Tipo de crédito aprobado: SUPER AVANCE, Monto: 5000000
1, 4, 2, 0, 1
Decisión: Aprobado
Con probabilidad: 98.39%
```

B) Un cliente con las siguientes características, edad = 55 años, maritalstatus = 'CASADO' y gender = 'F'.

```
if(edad <= 40):
   edad_map = 0
elif ((edad > 41) and (edad <= 60)):
   edad_map = 1
elif ((edad > 61) and (edad <= 80)):
   edad_map = 2
elif ((edad > 81) and (edad <= 100)):
   edad_map = 3
else:
   edad_map = 4

if(maritalstatus == 'SOLTERO'):
   maritalstatus_map = 0
elif (maritalstatus == 'CASADO'):
   maritalstatus_map = 1
elif (maritalstatus == 'VIUDO'):
   maritalstatus_map = 2
else:
   maritalstatus_map = 3

if(gender == 'F'):
   gender_map = 0
else:
   gender_map = 1

nt_map = pdf_map[(pdf_map['edad_map'] == edad_map) & (pdf_map['maritalstatus_map'] == maritalstatus_map) & (pdf_map['gender_map'] == gender_map)].groupby('nombre_transaccion_map').size()
mt_map = pdf_map[(pdf_map['edad_map'] == edad_map) & (pdf_map['maritalstatus_map'] == maritalstatus_map) & (pdf_map['gender_map'] == gender_map)].groupby('monto_transaccion_map').size()
nt_map = nt_map.index[len(nt_map.index)-1]
mt_map = mt_map.index[len(mt_map.index)-1]

if(nt_map == 0):
   tipo_de_credito = "AVANCE CAJA"
else:
   tipo_de_credito = "SUPER AVANCE"
```

```
if(mt_map == 0):
   monto = 1000000
elif(mt_map == 1):
   monto = 2000000
elif(mt_map == 2):
   monto = 3000000
elif(mt_map == 3):
   monto = 4000000
elif(mt_map == 4):
   monto = 5000000
elif(mt_map == 5):
   monto = 5370569

print("Tipo de crédito aprobado: " + str(tipo_de_credito) + ", Monto: " + str(monto))

x_test = pd.DataFrame(columns=('nombre_transaccion_map','monto_transaccion_map',
'edad_map', 'maritalstatus_map','gender_map'))
print(str(nt_map) + ", " + str(mt_map) + ", " + str(edad_map) + ", " + str(maritalstatus_map) + ",
" + str(gender_map))
x_test.loc[0] = (nt_map,mt_map,edad_map,maritalstatus_map,gender_map)
y_pred = decision_tree.predict(x_test.drop(['nombre_transaccion_map'], axis = 1))
if(y_pred[0]==1):
   res = "Aprobado"
else:
   res = "Rechazado"
print("Decisión: " + res)
y_proba = decision_tree.predict_proba(x_test.drop(['nombre_transaccion_map'], axis = 1))
print("Con probabilidad: " + str(round(y_proba[0][y_pred][0]* 100, 2))+"%")
```

Respuesta del modelo:

```
Tipo de crédito aprobado: SUPER AVANCE, Monto: 2000000
1, 1, 1, 1, 0
Decisión: Aprobado
Con probabilidad: 95.02%
```

**Probemos ahora utilizando un esquema basado en Regresión No Lineal utilizando SVR (Support Vector Regression).**

```
import pandas as pd
from pyspark.sql import SparkSession
from sklearn import svm

creditos = pd.read_parquet('creditos.parquet', engine='pyarrow')
demografico = pd.read_parquet('demografico.parquet', engine='pyarrow')

spark = SparkSession \
   .builder \
```

```
    .appName("Challenge") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

df = spark.sql("SELECT cr.customerid, cr.nombre_transaccion, cr.monto_transaccion, de.maritalstatus, de.gender, de.edad FROM parquet.`creditos.parquet` as cr INNER JOIN parquet.`demografico.parquet` as de ON cr.customerid == de.customerid ORDER BY cr.monto_transaccion DESC")
df.show()
```

```
+----------+------------------+-----------------+-------------+------+----+
|customerid|nombre_transaccion|monto_transaccion|maritalstatus|gender|edad|
+----------+------------------+-----------------+-------------+------+----+
|      5120|      SUPER AVANCE|       5370569.00|       CASADO|     M|72.0|
|     40214|      SUPER AVANCE|       5370569.00|       CASADO|     M|74.0|
|     87958|      SUPER AVANCE|       5370569.00|      SOLTERO|     F|72.0|
|     40214|      SUPER AVANCE|       5370569.00|       CASADO|     M|74.0|
|     23189|       AVANCE CAJA|       5370569.00|       CASADO|     F|71.0|
|     11271|      SUPER AVANCE|       5005371.00|       CASADO|     F|75.0|
|     92959|      SUPER AVANCE|       5000000.00|       CASADO|     M|75.0|
|     81587|      SUPER AVANCE|       5000000.00|       CASADO|     F|75.0|
|     93189|      SUPER AVANCE|       5000000.00|       CASADO|     M|73.0|
|     16227|      SUPER AVANCE|       5000000.00|       CASADO|     F|71.0|
|     60013|      SUPER AVANCE|       5000000.00|       CASADO|     M|74.0|
|     41946|       AVANCE CAJA|       5000000.00|       CASADO|     M|75.0|
|     81587|      SUPER AVANCE|       5000000.00|       CASADO|     F|75.0|
|     44542|      SUPER AVANCE|       5000000.00|       CASADO|     M|70.0|
|     47981|      SUPER AVANCE|       5000000.00|      SOLTERO|     M|69.0|
|     10094|      SUPER AVANCE|       5000000.00|      SOLTERO|     M|75.0|
|     44542|      SUPER AVANCE|       5000000.00|       CASADO|     M|70.0|
|     73415|      SUPER AVANCE|       5000000.00|      SOLTERO|     M|73.0|
|     90301|      SUPER AVANCE|       5000000.00|       CASADO|     M|69.0|
|     90248|       AVANCE CAJA|       5000000.00|       CASADO|     M|75.0|
+----------+------------------+-----------------+-------------+------+----+
only showing top 20 rows
```

```
pdf = df.toPandas()

pdf.loc[pdf['maritalstatus'] == 'SOLTERO', 'maritalstatus_map'] = 0
pdf.loc[pdf['maritalstatus'] == 'CASADO', 'maritalstatus_map'] = 1
pdf.loc[pdf['maritalstatus'] == 'VIUDO', 'maritalstatus_map'] = 2
pdf.loc[pdf['maritalstatus'] == 'NO INF', 'maritalstatus_map'] = 3

pdf.loc[pdf['gender'] == 'F', 'gender_map'] = 0
pdf.loc[pdf['gender'] == 'M', 'gender_map'] = 1

drop_elements = ['customerid', 'nombre_transaccion', 'maritalstatus', 'gender']

pdf.loc[pdf['nombre_transaccion'] == 'AVANCE CAJA', 'nombre_transaccion_map'] = 0
pdf.loc[pdf['nombre_transaccion'] == 'SUPER AVANCE', 'nombre_transaccion_map'] = 1

pdf_map = pdf.drop(drop_elements, axis = 1)
pdf_map
```

| | monto_transaccion | edad | maritalstatus_map | gender_map | nombre_transaccion_map |
|---|---|---|---|---|---|
| **0** | 5370569.00 | 71.0 | 1.0 | 0.0 | 0.0 |
| **1** | 5370569.00 | 72.0 | 0.0 | 0.0 | 1.0 |
| **2** | 5370569.00 | 72.0 | 1.0 | 1.0 | 1.0 |
| **3** | 5370569.00 | 74.0 | 1.0 | 1.0 | 1.0 |
| **4** | 5370569.00 | 74.0 | 1.0 | 1.0 | 1.0 |
| **5** | 5005371.00 | 75.0 | 1.0 | 0.0 | 1.0 |
| **6** | 5000000.00 | 70.0 | 1.0 | 0.0 | 1.0 |
| **7** | 5000000.00 | 71.0 | 1.0 | 0.0 | 0.0 |
| **8** | 5000000.00 | 75.0 | 1.0 | 1.0 | 0.0 |
| **9** | 5000000.00 | 71.0 | 1.0 | 0.0 | 1.0 |
| **10** | 5000000.00 | 69.0 | 1.0 | 1.0 | 1.0 |

```
only showing top 10 rows
```

```
pdf_map['monto_transaccion'] = pdf_map.monto_transaccion.astype(int)
pdf_map['edad'] = pdf_map.edad.astype(int)
pdf_map['maritalstatus_map'] = pdf_map.maritalstatus_map.astype(int)
pdf_map['gender_map'] = pdf_map.gender_map.astype(int)
pdf_map['nombre_transaccion_map'] = pdf_map.nombre_transaccion_map.astype(int)
pdf_map
```

| | monto_transaccion | edad | maritalstatus_map | gender_map | nombre_transaccion_map |
|---|---|---|---|---|---|
| **0** | 5370569 | 71 | 1 | 0 | 0 |
| **1** | 5370569 | 72 | 0 | 0 | 1 |
| **2** | 5370569 | 72 | 1 | 1 | 1 |
| **3** | 5370569 | 74 | 1 | 1 | 1 |
| **4** | 5370569 | 74 | 1 | 1 | 1 |
| **5** | 5005371 | 75 | 1 | 0 | 1 |
| **6** | 5000000 | 70 | 1 | 0 | 1 |
| **7** | 5000000 | 71 | 1 | 0 | 0 |
| **8** | 5000000 | 75 | 1 | 1 | 0 |
| **9** | 5000000 | 71 | 1 | 0 | 1 |
| **10** | 5000000 | 69 | 1 | 1 | 1 |

```
only showing top 10 rows
```

```
drop_elements = ['monto_transaccion']
```

**Datos de entrenamiento (aprendizaje supervisado):**

**Features:**

X = pdf_map.drop(drop_elements, axis = 1)

**Label**:

y = pdf_map['monto_transaccion']

clf = svm.SVR()
clf.fit(X, y)

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
gamma='auto',kernel='rbf', max_iter=-1, shrinking=True, tol=0.001,
verbose=False)
```

""" Para un cliente con: (EDAD = 66, CASADO(1), MASCULINO(1)), el monto a otrogar es: 1.000.005 para un crédito AC(0) """

**Predictor:**

r = clf.predict([[66, 1, 1, 1]])
int(r[0])

```
100005
```

""" Para un cliente con: (EDAD = 66, CASADO(1), FEMENINO(0)), el monto a otrogar es: 1.000.000 para un crédito AC(0) """

**Predictor:**

r = clf.predict([[66, 1, 1, 0]])
int(r[0])

```
100000
```

```
|      8158|       AVANCE CAJA|        100000.00|       CASADO|     M|66.0|  SANTIAGO|
|      1841|       AVANCE CAJA|        100000.00|       CASADO|     F|66.0|  SANTIAGO|
```