Financiera Compartamos

Limpieza de Datos

```
# warnings
import warnings
warnings.filterwarnings("ignore")
# for graphics
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib notebook
# import data
import numpy as np
import pandas as pd
df = pd.read_excel('data/dataOriginal.xlsx')
# cast to str
df.codigoCuenta=df.codigoCuenta.apply(str)
df.codigoCliente=df.codigoCliente.apply(str)
df.codigoFuente=df.codigoFuente.apply(str)
# cast to date
df.fechaApertura = pd.to_datetime(df.fechaApertura)
df.fechaVigencia = pd.to_datetime(df.fechaVigencia)
df.fechaUltimoPago = pd.to_datetime(df.fechaUltimoPago)
df.fechaVencimiento = pd.to_datetime(df.fechaVencimiento)
df.dSalPag = pd.to_datetime(df.dSalPag)
# drop useless columns
df.drop(['codigoFuente', 'ctipcuo', 'fechaVigencia', 'fechaTermino', 'cestado',
         'ccodprd', 'ccodana', 'cCodPdc', 'capitalDesembolsado', 'diasDeAtraso',
         'Fuente de financiamiento', 'nPerLbr', 'cComApr', 'diasAtrasoMaximo',
         'cctapre', 'ntasint', 'npagcta', 'nmorpag', 'ngastos', 'ngaspag',
         'nintpen', 'nintcal', 'nintpag', 'nintmor', 'capitalCalculado', 'dSalPag'
        ], axis=1, inplace=True)
```

Datos Cualitativos

```
df.describe(include=['object'])
```

```
.dataframe thead th {
    text-align: left;
}

.dataframe tbody tr th {
    vertical-align: top;
}
```

	codigoCuenta	codigoCliente	tipoCredito	sectorEconomico	ctipcal	condicionCredito	cmarjud
count	30575	30575	30575	30575	30575	30575	30575
unique	30575	26412	3	8	2	2	3
top	152310097851	24082661	R	01	S	Z	S
freq	1	4	18526	14892	30471	30433	27708

Establecer como index a codigo cuenta y cambiar codigo cliente a valor si es unica cuenta

```
dftest = df['codigoCuenta'].groupby([df['codigoCliente']]).count()
md1c = dftest.index[dftest==1]
df['cuentaUnica']=df.codigoCliente.isin(md1c)
df.cuentaUnica=df.cuentaUnica.apply(int).apply(str)
df.drop(['codigoCliente'], axis=1, inplace=True)
df = df.set_index('codigoCuenta')
```

Verificar Datos de tipo objeto

```
df.describe(include=['object'])
```

```
.dataframe thead th {
    text-align: left;
}

.dataframe tbody tr th {
    vertical-align: top;
}
```

	tipoCredito	sectorEconomico	ctipcal	condicionCredito	cmarjud	cuentaUnica
count	30575	30575	30575	30575	30575	30575
unique	3	8	2	2	3	2
top	R	01	S	Z	S	1
freq	18526	14892	30471	30433	27708	22671

```
display('Tipo Credito', df.tipoCredito.unique())
display('Sector Económico', df.sectorEconomico.unique())#(01,02,03) (04,05) (06,07)
display('Tipo de Cálculo', df.ctipcal.unique())
display('Condición Crédito', df.condicionCredito.unique())
display('Mar Judicial', df.cmarjud.unique())
```

```
'Tipo Credito'
    array(['N', 'R', 'U'], dtype=object)
'Sector Económico'
    array(['01', '03', '02', '05', '07', '04', '  ', '06'], dtype=object)
'Tipo de Cálculo'
    array(['S', 'E'], dtype=object)
'Condición Crédito'
    array(['Z', 'J'], dtype=object)
'Mar Judicial'
    array(['S', 'J', 'P'], dtype=object)
```

Buscando Campos en blanco

```
df.sectorEconomico[df.sectorEconomico=='  ']=None
df.isnull().sum() # buscar campos sin datos
```

```
tipoCredito            0
Moneda                 0
```

```
modalidadCredito       0
sectorEconomico      438
destinoCredito         0
fechaApertura          0
ctipper                0
ctipcal                0
fechaUltimoPago     1361
montoApertura          0
capitalPagado          0
numeroDeCuotas         0
condicionCredito       0
cmarjud                0
fechaVencimiento       9
nTasEfe                0
cCodOfi                0
nPerGra                0
cuentaUnica            0
dtype: int64
```

Definimos a los que son deudores

```
df['esDeudor'] = ((df.montoApertura-df.capitalPagado)/df.montoApertura<0.5).apply(str)
df.esDeudor.describe()
```

```
count     30575
unique        2
top       False
freq      17159
Name: esDeudor, dtype: object
```

Datos de Tiempo

```
df.describe(include=['datetime'])
```

```
.dataframe thead th {
    text-align: left;
}

.dataframe tbody tr th {
    vertical-align: top;
}
```

	fechaApertura	fechaUltimoPago	fechaVencimiento
count	30575	29214	30566
unique	907	2037	1331
top	2012-08-31 00:00:00	2013-05-28 00:00:00	2013-12-02 00:00:00
freq	899	1049	326
first	2009-12-22 00:00:00	2010-02-10 00:00:00	2010-04-17 00:00:00
last	2012-12-31 00:00:00	2018-08-31 00:00:00	2018-01-08 00:00:00

Filtro de data por Fechas

```
df.dropna(subset=['fechaVencimiento'], inplace=True) # quitamos data sin fecha de vencimiento
diasxmes = 30
df['diasSinPagar'] = np.int32(((df.fechaVencimiento - df.fechaUltimoPago).dt.days
                               ).fillna((df.fechaVencimiento - df.fechaApertura).dt.days))
df['clienteMoroso'] = (df.diasSinPagar<0).apply(int).apply(str) # ultimo pago post fecha vencida
df['diasSinPagar'] = np.abs(df.diasSinPagar)
df.clienteMoroso.describe()
```

```
count     30566
unique        2
top           0
freq      25219
Name: clienteMoroso, dtype: object
```

Retirando data no útil

```
# quitamos data que pueda alterar nuestra predictividad
df.dropna(subset=['sectorEconomico'], inplace=True) # quitamos data sin sector economico definido
df.drop(['fechaApertura'], axis=1, inplace=True)
df.drop(['fechaUltimoPago'], axis=1, inplace=True)
df.drop(['fechaVencimiento'], axis=1, inplace=True)
df.isnull().sum() # buscar campos sin datos
```

```
tipoCredito         0
Moneda              0
modalidadCredito    0
sectorEconomico     0
destinoCredito      0
ctipper             0
ctipcal             0
montoApertura       0
capitalPagado       0
numeroDeCuotas      0
condicionCredito    0
cmarjud             0
nTasEfe             0
cCodOfi             0
nPerGra             0
cuentaUnica         0
esDeudor            0
diasSinPagar        0
clienteMoroso       0
dtype: int64
```

Datos Cuantitativos

```
df.describe(include=['int64'])
```

```
.dataframe thead th {
    text-align: left;
}

.dataframe tbody tr th {
```

```
    vertical-align: top;
}
```

	Moneda	modalidadCredito	destinoCredito	ctipper	numeroDeCuotas	cCodOfi	nPerGra
count	30128.000000	30128.000000	30128.000000	30128.000000	30128.000000	30128.000000	30128.000000
mean	1.001129	1.021044	3.217738	1.996448	13.244855	15.277549	0.000133
std	0.033575	0.238289	2.132137	0.059490	5.651643	9.512876	0.014112
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000
25%	1.000000	1.000000	2.000000	2.000000	12.000000	8.000000	0.000000
50%	1.000000	1.000000	2.000000	2.000000	12.000000	14.000000	0.000000
75%	1.000000	1.000000	4.000000	2.000000	12.000000	24.000000	0.000000
max	2.000000	5.000000	8.000000	2.000000	60.000000	59.000000	2.000000

Analizando los Campos

```
display('Moneda', df.Moneda.unique())
display('Modalidad Crédito', df.modalidadCredito.unique())
display('Destino Crédito', df.destinoCredito.unique())
display('Tipo Persona', df.ctipper.unique())
display('Numero de Cuotas', df.numeroDeCuotas.unique()) # numerico, cuotas (meses)
display('Oficinas', df.cCodOfi.unique()) # agrupar por pais, zona
display('Periodo de Gracia', df.nPerGra.unique()) # numerico, periodos de gracia (meses)
```

```
'Moneda'
    array([1, 2])
'Modalidad Crédito'
    array([1, 3, 5])
'Destino Crédito'
    array([2, 4, 7, 8, 1, 3, 6, 5])
'Tipo Persona'
    array([2, 1])
'Numero de Cuotas'
    array([12, 18, 24,  9,  6, 36, 30, 15,  1, 14, 10,  8, 16, 13, 20, 60, 48,
           7, 25, 22, 34, 21, 17,  4,  3, 40, 27, 19, 28, 42, 44, 33,  5, 11,
           32, 38, 29, 26, 35,  2])
'Oficinas'
    array([ 1, 24,  7, 14, 26,  2, 22, 29,  5,  3, 27, 30,  4, 36,  6, 28, 25,
            8, 37, 57,  9, 11, 31, 59, 13, 32, 10, 12, 15, 38, 16, 17, 35, 18,
           19, 20, 33, 34])
'Periodo de Gracia'
    array([0, 2, 1])
```

Definimos los que si serán numericos enteros

```
df.Moneda=df.Moneda.apply(str)
df.modalidadCredito=df.modalidadCredito.apply(str)
df.destinoCredito=df.destinoCredito.apply(str)
df.ctipper=df.ctipper.apply(str)
df.cCodOfi=df.cCodOfi.apply(str)
df.describe(include=['int64'])
```

```
.dataframe thead th {
    text-align: left;
}

.dataframe tbody tr th {
    vertical-align: top;
}
```

	numeroDeCuotas	nPerGra
count	30128.000000	30128.000000
mean	13.244855	0.000133
std	5.651643	0.014112
min	1.000000	0.000000
25%	12.000000	0.000000
50%	12.000000	0.000000
75%	12.000000	0.000000
max	60.000000	2.000000

Datos Cuantitativos no exactos, numéricos de punto flotante (Reales)

```
df.describe(include=['float'])
```

```
.dataframe thead th {
    text-align: left;
}

.dataframe tbody tr th {
    vertical-align: top;
}
```

	montoApertura	capitalPagado	nTasEfe
count	30128.000000	30128.000000	30128.000000
mean	4085.468884	1978.120835	3.670227
std	5718.899807	3500.031275	0.670471
min	138.490000	0.000000	1.090000
25%	1000.000000	364.037500	3.290000
50%	2000.000000	858.385000	3.490000
75%	5000.000000	1981.987500	3.990000
max	60000.000000	59840.580000	6.993000

```
montoMensual = (df.montoApertura/df.numeroDeCuotas)
df['numeroCuotasPagadas']=(df.capitalPagado/montoMensual).apply(int)
df.describe(include=['int64'])
```

```
.dataframe thead th {
    text-align: left;
}

.dataframe tbody tr th {
    vertical-align: top;
}
```

	numeroDeCuotas	nPerGra	numeroCuotasPagadas
count	30128.000000	30128.000000	30128.000000
mean	13.244855	0.000133	5.610462
std	5.651643	0.014112	4.833444
min	1.000000	0.000000	0.000000
25%	12.000000	0.000000	2.000000
50%	12.000000	0.000000	5.000000
75%	12.000000	0.000000	8.000000
max	60.000000	2.000000	59.000000

Resumen de Características

Tipo de Dato: int - numeroDeCuotas - nPerGra - numeroCuotasPagadas

Tipo de Dato: float - montoApertura - capitalPagado - nTasEfe

Tipo de Dato: object

1. tipoCredito:
    - N(Nuevo)
    - R(Recurrente)
    - U(Recuperado)
2. Moneda
    - 1(Soles)
    - 2(Dolar)
3. modalidadCredito
    - 1(Credito)
    - 3(Ordinario)
    - 5(Paralelo)
4. sectorEconomico
    - 01(comercio)
    - 02(Industria)
    - 03(Servicios)
    - 04(Agricola)
    - 05(Ganadero)
    - 06(Pesquero)
    - 07(Minero)
5. destinoCredito
    - 1(Capital Trabajo Formal)
    - 2(Capital Trabajo Informal)
    - 3(Activo Fijo Formal)
    - 4(Activo Fijo Informal)
    - 5(Mixto Formal)
    - 6(Mixto Informal)
    - 7(Consumo)
    - 8(Vivienda)

6. ctipper
    - 1(Persona)
    - 2(Empresa)
7. ctipcal
    - E(Exponencial)
    - S(Sin Calculo)
8. condicionCredito
    - J(Judicial)
    - Z(Castigado)
9. cmarjud
    - J(Judicial)
    - P(Pre Judicial)
    - S(Simple)
10. cCodOfi
    - 1,3,4,... (38 oficinas)
11. cuentaUnica
    - 1(Cuenta unica por cliente)
    - 0(Esta solo es 1 de las cuentas del cliente)
12. clienteMoroso
    - 1(Último pago que efectuó fue después de la fecha límite)
    - 0(Último pago efectuado antes de la fecha límite)

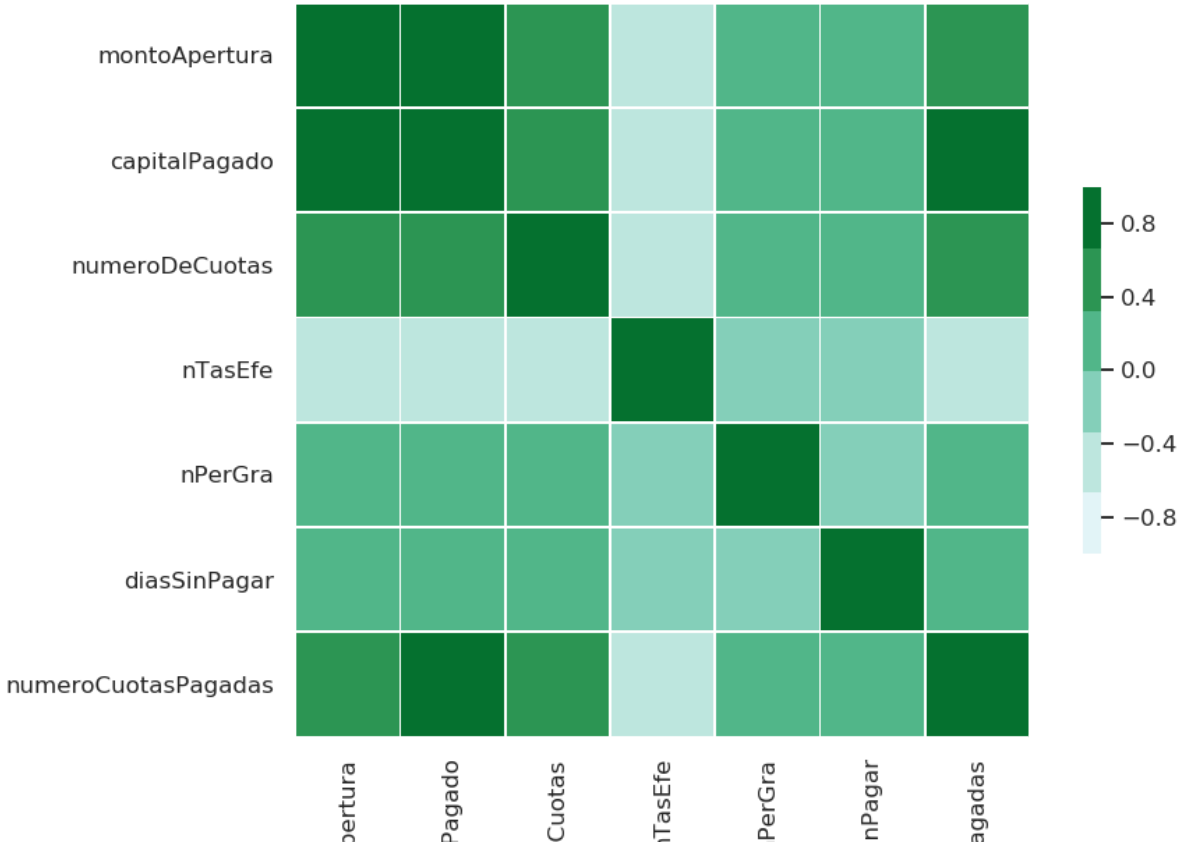
Graficos de Visualización

```
dfGraphics = df.copy()
dfGraphics.tipoCredito.replace(['N', 'R', 'U'], ['Nuevo', 'Recurrente', 'Recuperado'], inplace=True)
dfGraphics.Moneda.replace(['1', '2'], ['Soles', 'Dolares'], inplace=True)
dfGraphics.modalidadCredito.replace(['1', '3', '5'], ['Credito', 'Ordinario', 'Paralelo'], inplace=True)
dfGraphics.sectorEconomico.replace(['01', '02', '03', '04', '05', '06', '07'],
                ['Comercio', 'Industria', 'Servicios', 'Agricola', 'Ganadero', 'Pesquero', 'Minero'],
inplace=True)
dfGraphics.destinoCredito.replace(['1', '2', '3', '4', '5', '6', '7', '8'],
                ['Formal', 'Informal', 'Formal', 'Informal', 'Formal', 'Informal', 'Hogar', 'Hogar'],
inplace=True)
dfGraphics.ctipper.replace(['1', '2'], ['Persona', 'Empresa'], inplace=True)
dfGraphics.ctipcal.replace(['E', 'S'], ['Exponencial', 'Simple'], inplace=True)
dfGraphics.condicionCredito.replace(['J', 'Z'], ['Judicial', 'Castigado'], inplace=True)
dfGraphics.cmarjud.replace(['J', 'P', 'S'], ['Judicial', 'PreJudicial', 'Simple'], inplace=True)
"""
dfGraphics.cCodOfi.replace([
     '1',  '2',  '3',  '4',  '5',  '6',  '7',  '8',  '9',
    '10', '11', '12', '13', '14', '15', '16', '17', '18', '19',
    '20', '22', '24', '25', '26', '27', '28', '29', '57', '59',
    '30', '31', '32', '33', '34', '35', '36', '37', '38'
     ], [
    'Zona1', 'Zona1', 'Zona1', 'Zona1', 'Zona1', 'Zona1', 'Zona1', 'Zona1', 'Zona1',
    'Zona2', 'Zona2', 'Zona2', 'Zona2', 'Zona2', 'Zona2', 'Zona2', 'Zona2', 'Zona2', 'Zona2',
    'Zona3', 'Zona3', 'Zona3', 'Zona3', 'Zona3', 'Zona3', 'Zona3', 'Zona3', 'Zona3', 'Zona3',
    'Zona4', 'Zona4', 'Zona4', 'Zona4', 'Zona4', 'Zona4', 'Zona4', 'Zona4', 'Zona4'
    ], inplace=True)
"""
dfGraphics.cuentaUnica.replace(['0', '1'], ['Multi Cuenta', 'Cuenta Unica'], inplace=True)
dfGraphics.clienteMoroso.replace(['0', '1'], ['Cliente Moroso', 'Cliente Unica'], inplace=True)
dfGraphics.esDeudor.replace([0, 1], ['No', 'Si'], inplace=True)
```

Correlación entre los Datos

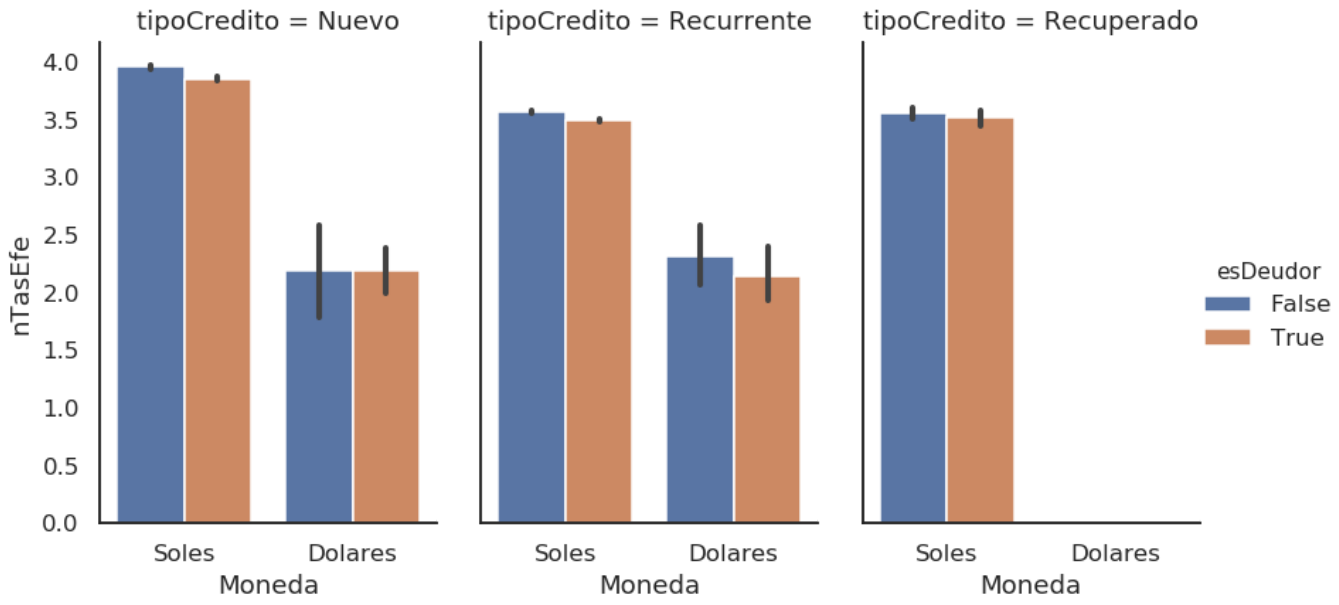
```
sns.set(style="white")
corr = dfGraphics.corr()
f, ax = plt.subplots()
```

```
cmap = sns.color_palette("BuGn")
sns.heatmap(corr, cmap=cmap, vmax=1.0, vmin=-1.0, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.show()
```



Influencia del Interes

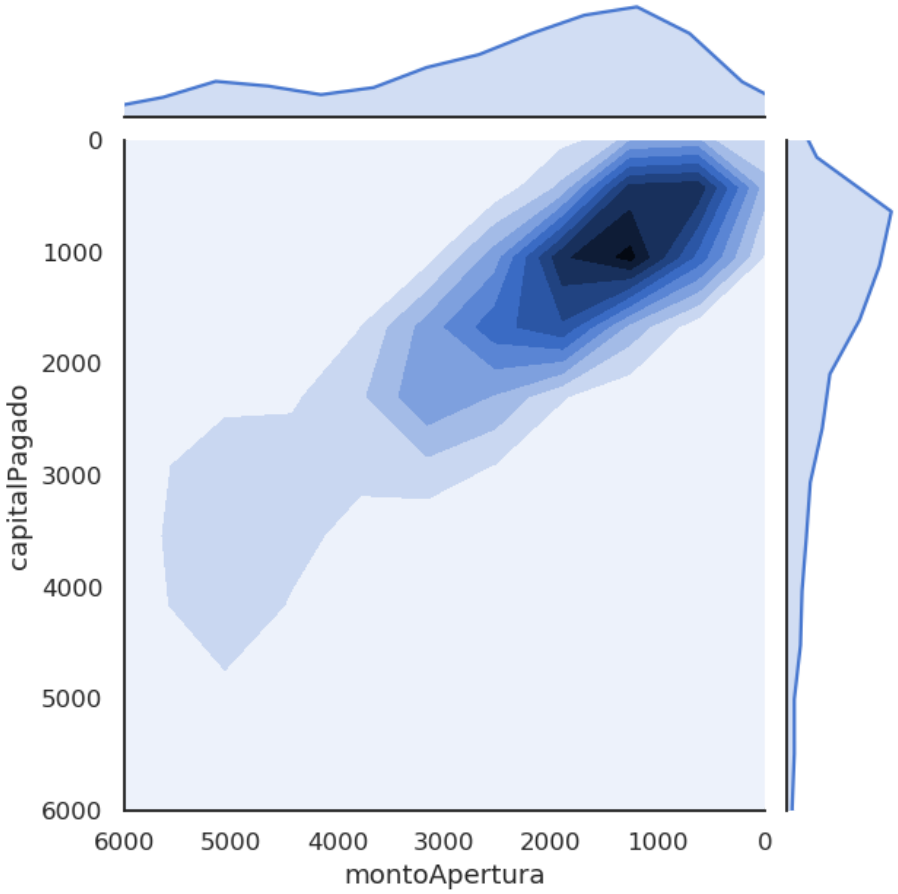
```
g = sns.catplot(x="Moneda", y="nTasEfe",
                hue="esDeudor", col="tipoCredito",
                data=dfGraphics, kind="bar",
                height=4, aspect=.7)
plt.show()
```



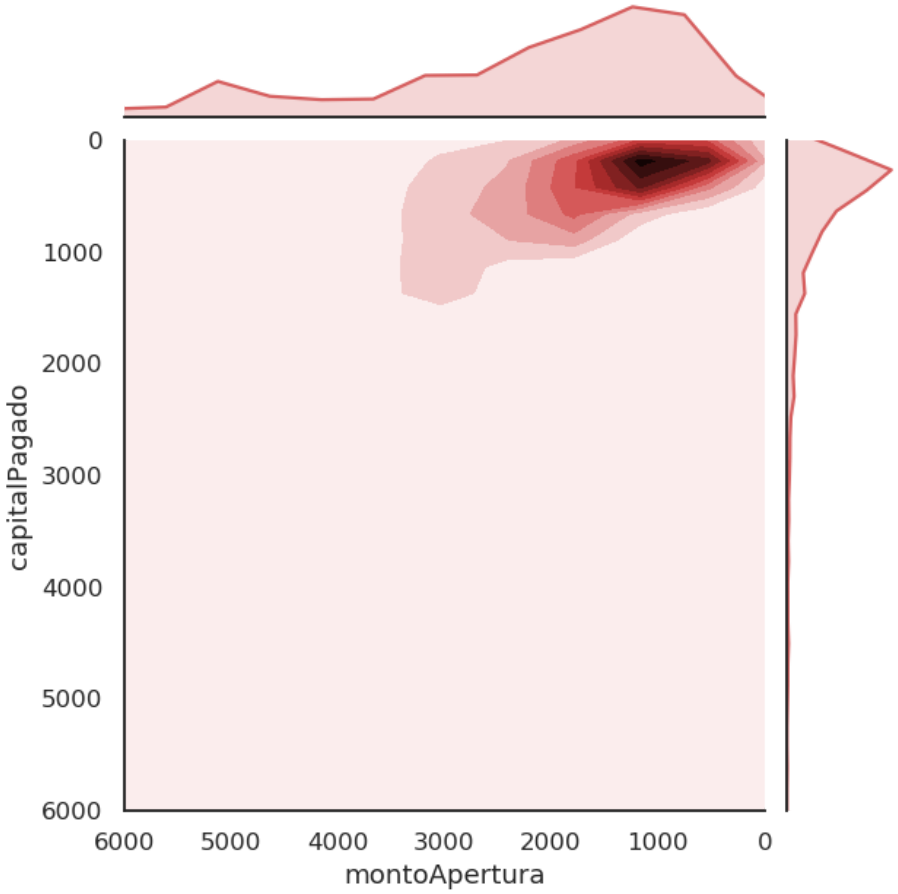
Distribución de Deudores según montoApertura y capitalPagado

```
dfDeudor = df.loc[df['esDeudor'] == 'True']
dfNoDeudor = df.loc[df['esDeudor'] == 'False']
print('Deudor')
sns.jointplot(dfDeudor.montoApertura, dfDeudor.capitalPagado, data=dfDeudor, kind="kde",
              xlim=6000, ylim=6000, color="b");
print('No Deudor')
sns.jointplot(x="montoApertura", y="capitalPagado", data=dfNoDeudor, kind="kde",
              xlim=6000, ylim=6000, color="r");
plt.show()
```

```
Deudor
```

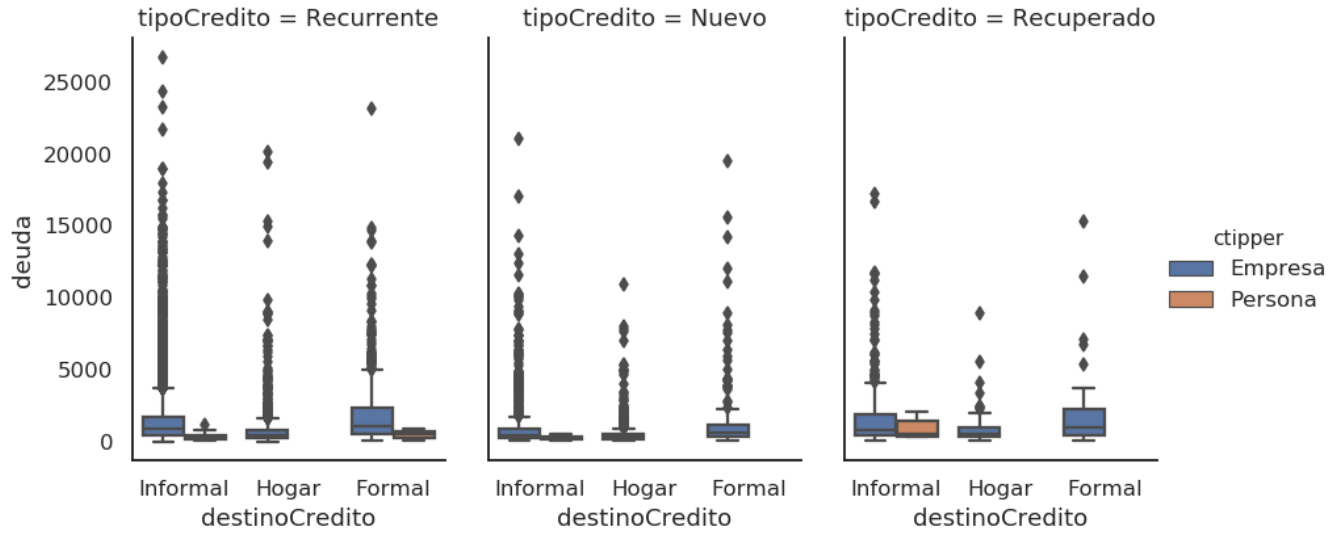

0
1000
2000
3000
4000
5000
6000
capitalPagado
6000
5000
4000
3000
2000
1000
0
montoApertura

```
No Deudor
```



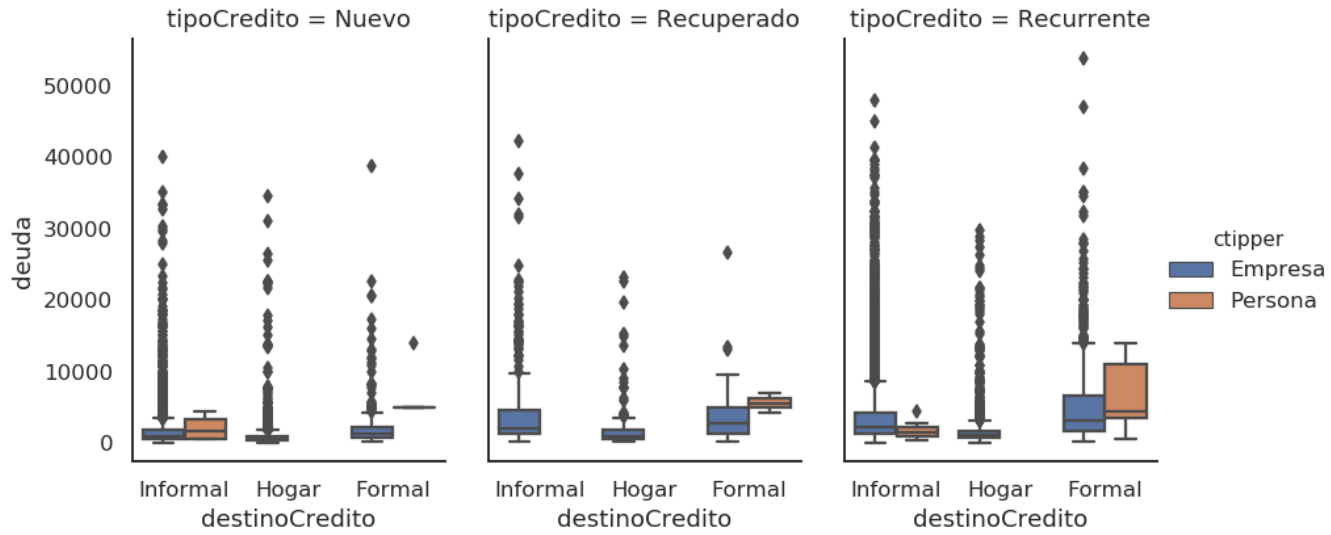
Boxplot para Deudores, según tipoCredito,tipoPersona y destinoCredito

```
dfDeudor.tipoCredito.replace(['N', 'R', 'U'], ['Nuevo', 'Recurrente', 'Recuperado'], inplace=True)
dfDeudor.destinoCredito.replace(['1', '2', '3', '4', '5', '6', '7', '8'],
        ['Formal', 'Informal', 'Formal', 'Informal', 'Formal', 'Informal', 'Hogar', 'Hogar'],
inplace=True)
dfDeudor.ctipper.replace(['1', '2'], ['Persona', 'Empresa'], inplace=True)
dfDeudor['deuda'] = dfDeudor.montoApertura-dfDeudor.capitalPagado
g = sns.catplot(x="destinoCredito", y="deuda",
                hue="ctipper", col="tipoCredito",
                data=dfDeudor, kind="box",
                height=4, aspect=.7)
plt.show()
```



Boxplot para No Deudores, según tipoCredito,tipoPersona y destinoCredito

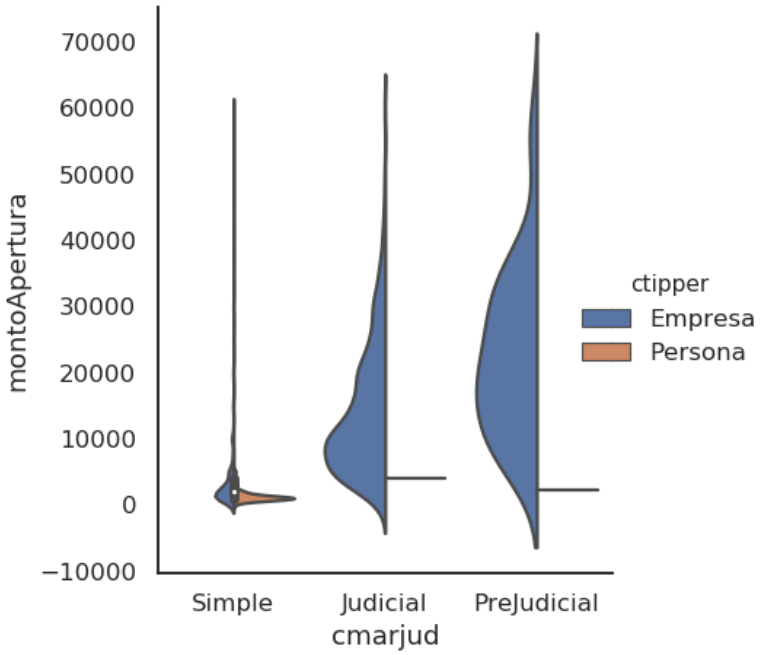
```
dfNoDeudor.tipoCredito.replace(['N', 'R', 'U'], ['Nuevo', 'Recurrente', 'Recuperado'], inplace=True)
dfNoDeudor.destinoCredito.replace(['1', '2', '3', '4', '5', '6', '7', '8'],
        ['Formal', 'Informal', 'Formal', 'Informal', 'Formal', 'Informal', 'Hogar', 'Hogar'],
inplace=True)
dfNoDeudor.ctipper.replace(['1', '2'], ['Persona', 'Empresa'], inplace=True)
dfNoDeudor['deuda'] = dfNoDeudor.montoApertura-dfNoDeudor.capitalPagado
g = sns.catplot(x="destinoCredito", y="deuda",
                hue="ctipper", col="tipoCredito",
                data=dfNoDeudor, kind="box",
                height=4, aspect=.7)
plt.show()
```



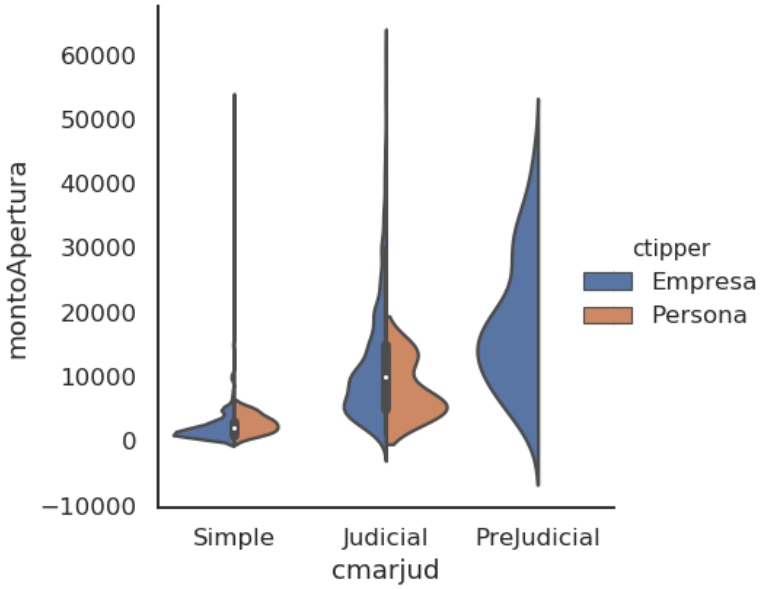
Comparativa del marco Judicial sobre las Personas

```
dfDeudor.cmarjud.replace(['J', 'P', 'S'], ['Judicial', 'PreJudicial', 'Simple'], inplace=True)
dfNoDeudor.cmarjud.replace(['J', 'P', 'S'], ['Judicial', 'PreJudicial', 'Simple'], inplace=True)
print('Deudores')
sns.catplot(x="cmarjud", y="montoApertura", hue="ctipper", kind="violin", split=True, data=dfDeudor);
print('No Deudores')
sns.catplot(x="cmarjud", y="montoApertura", hue="ctipper", kind="violin", split=True, data=dfNoDeudor);
plt.show()
```

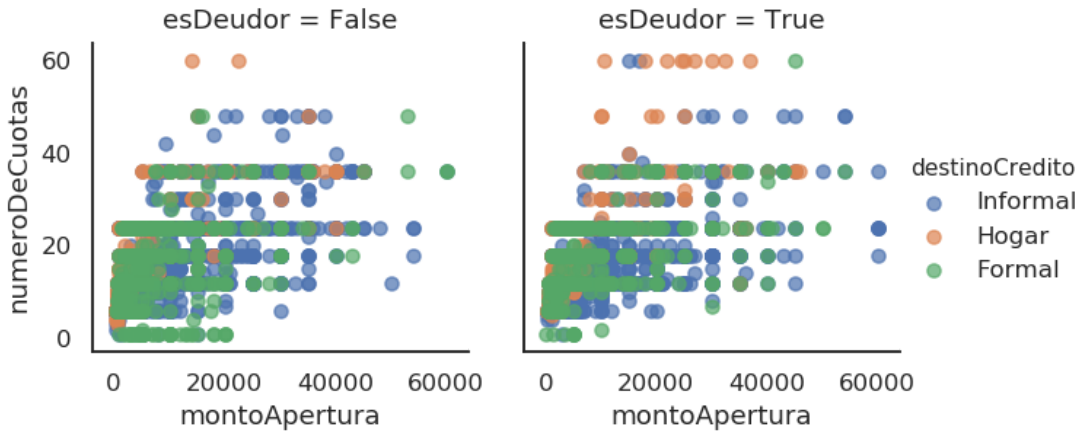
```
Deudores
```



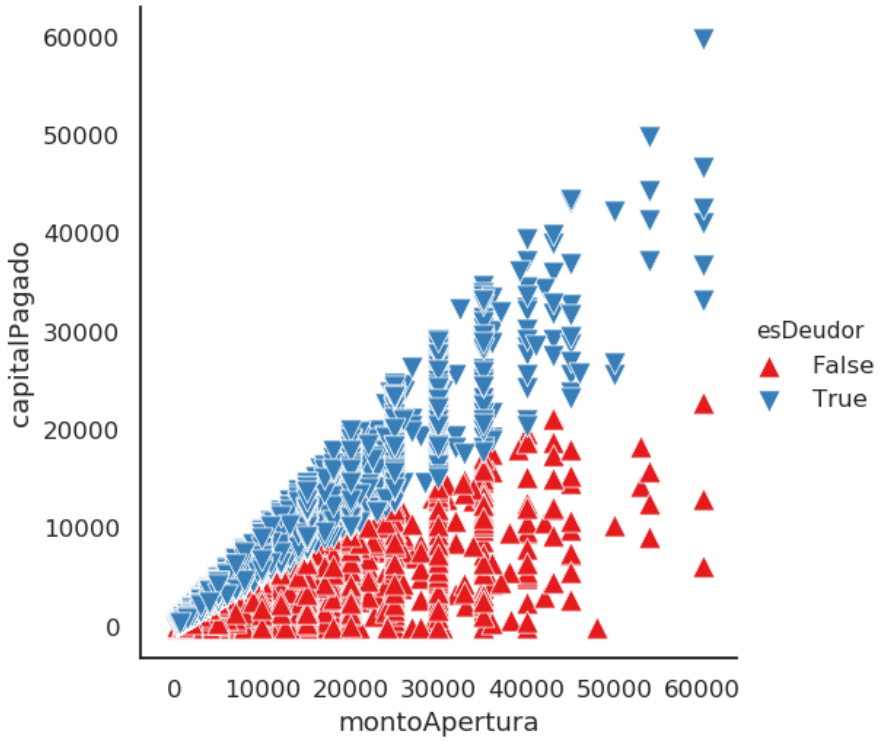

```
No Deudores
```




```
g = sns.FacetGrid(dfGraphics, col="esDeudor", hue="destinoCredito")
g.map(plt.scatter, "montoApertura", "numeroDeCuotas", alpha=.7)
g.add_legend();
```




```
g = sns.FacetGrid(dfGraphics, hue="esDeudor", palette="Set1", height=5, hue_kws={"marker": ["^", "v"]})
g.map(plt.scatter, "montoApertura", "capitalPagado", s=100, linewidth=.5, edgecolor="white")
g.add_legend();
plt.show()
```



Predicción y Evaluación del Modelo

```
dfTrain = df.copy()
dfTrain['tcNuevo']=(dfTrain.tipoCredito=='N').astype(int)
dfTrain['tcRecurrente']=(dfTrain.tipoCredito=='R').astype(int)
dfTrain['monedaSol']=(dfTrain.Moneda=='1').astype(int)
dfTrain['modCredito']=(dfTrain.modalidadCredito=='1').astype(int)
dfTrain['modOrdinario']=(dfTrain.modalidadCredito=='3').astype(int)
dfTrain['seBajo']=(dfTrain.sectorEconomico.isin(['01','02','03'])).astype(int)
dfTrain['seMedio']=(dfTrain.sectorEconomico.isin(['04','05'])).astype(int)
```

```
dfTrain['dcFormal']=(dfTrain.destinoCredito.isin(['1','3','5'])).astype(int)
dfTrain['dcInformal']=(dfTrain.destinoCredito.isin(['2','4','6'])).astype(int)
dfTrain['esPersona']=(dfTrain.ctipper=='1').astype(int)
dfTrain['esExponencial']=(dfTrain.ctipcal=='E').astype(int)
dfTrain['esCondJudicial']=(dfTrain.condicionCredito=='J').astype(int)
dfTrain['marcoJudicial']=(dfTrain.cmarjud=='J').astype(int)
dfTrain['marcoPreJudicial']=(dfTrain.cmarjud=='P').astype(int)
#display(dfTrain.cCodOfi.unique())
dfTrain['cuentaUnica']=(dfTrain.cuentaUnica=='1').astype(int)
dfTrain['clienteMoroso']=(dfTrain.clienteMoroso=='1').astype(int)
dfTrain.drop(['tipoCredito', 'Moneda', 'sectorEconomico', 'condicionCredito',
    'modalidadCredito', 'destinoCredito', 'ctipper', 'ctipcal', 'cmarjud'], axis=1, inplace=True)
dfTrain.esDeudor.replace(['True', 'False'], ['Si', 'No'], inplace=True)
```

Guardar los Datos Filtrados

```
writer = pd.ExcelWriter('dataFiltrada.xlsx')
dfTrain.to_excel(writer,'Hoja1')
writer.save()
```

Dividimos la Data aleatoriamente en 80% para entrenar y 20% para evaluar

```
from sklearn.model_selection import train_test_split
seed = 100
X = dfTrain.drop(['esDeudor'], axis=1, inplace=False)
y = dfTrain.esDeudor
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=seed)
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
clf = DecisionTreeClassifier(random_state=seed)
precision = cross_val_score(clf, X, y, cv=10)
display(precision)
```

```
array([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])
```

Entrenamiento y Predición con el modelo de Arbol de Clasificación

```
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

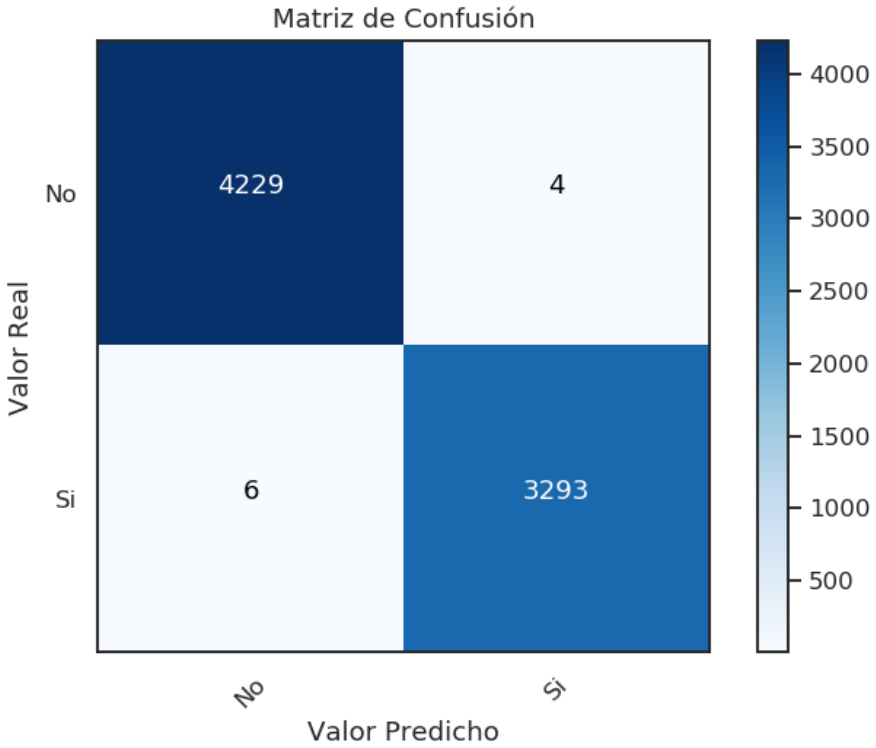
```
import itertools
from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Matriz de Confusión',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Matriz de Confusión Normalizada")
    else:
        print('Matriz de Confusión, sin normalizar')
    print(cm)
```

```
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('Valor Real')
    plt.xlabel('Valor Predicho')
```

```
cnf_matrix = confusion_matrix(y_test, y_pred)
np.set_printoptions(precision=2)
plot_confusion_matrix(cnf_matrix, classes=y.unique(), #normalize=True,
                      title='Matriz de Confusión')
plt.show()
```

```
Matriz de Confusión, sin normalizar
[[4229    4]
 [   6 3293]]
```




```
p_no_deudor = 3293/(6+3293)*100
p_deudor = 4229/(4+4229)*100
display('Probabilidad de identificar a cliente NO DEUDOR',p_no_deudor)
display('Probabilidad de identificar a cliente DEUDOR',p_deudor)
```

```
'Probabilidad de identificar a cliente NO DEUDOR'
    99.81812670506214
'Probabilidad de identificar a cliente DEUDOR'
    99.90550437042288
```

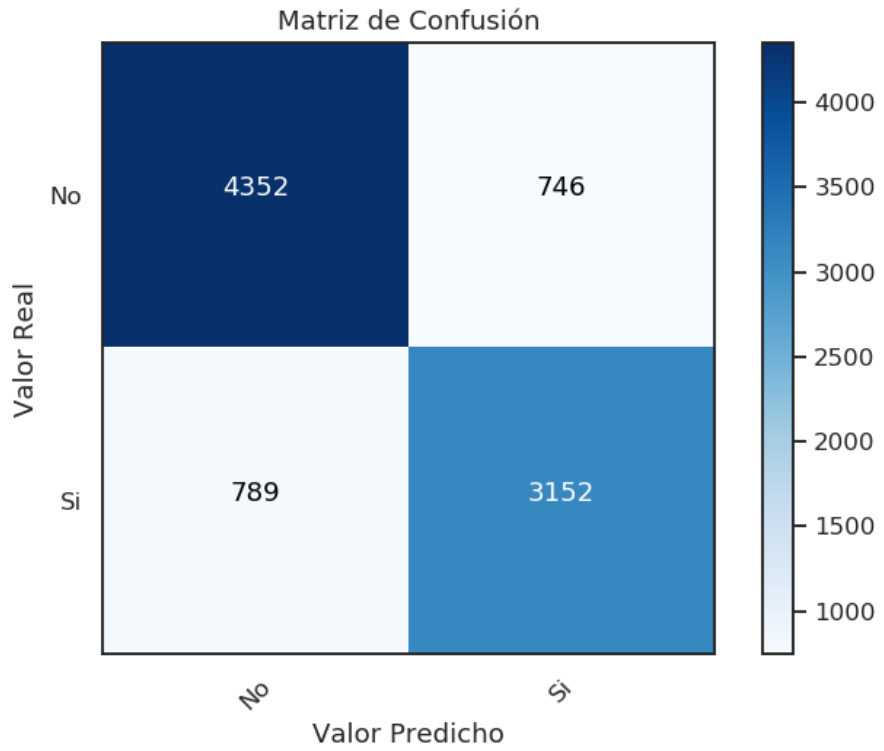
Segunda Prueba, quitando las columnas correlacionadas a esDeudor

```
X = dfTrain.drop(['esDeudor', 'capitalPagado', 'numeroCuotasPagadas'], axis=1, inplace=False)
y = dfTrain.esDeudor
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=seed)
clf = DecisionTreeClassifier(random_state=seed)
precision = cross_val_score(clf, X, y, cv=10)
display(precision)
```

```
array([ 0.78,  0.8 ,  0.81,  0.83,  0.81,  0.84,  0.84,  0.84,  0.82,  0.82])
```

```
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
cnf_matrix = confusion_matrix(y_test, y_pred)
np.set_printoptions(precision=2)
plot_confusion_matrix(cnf_matrix, classes=y.unique(), #normalize=True,
                      title='Matriz de Confusión')
plt.show()
```

```
Matriz de Confusión, sin normalizar
[[4352  746]
 [ 789 3152]]
```



```
p_no_deudor = 3152/(789+3152)*100
p_deudor = 4352/(746+4352)*100
display('Probabilidad de identificar a cliente NO DEUDOR',p_no_deudor)
display('Probabilidad de identificar a cliente DEUDOR',p_deudor)
```

```
'Probabilidad de identificar a cliente NO DEUDOR'
    79.97970058360822
'Probabilidad de identificar a cliente DEUDOR'
    85.36681051392702
```