Modelos avanzados en ciencia de datos

Proyecto Integrador

American Express - Default Prediction

"Predict if a customer will default in the future"

Interantes del equipo:

Mat. Dulce María Reyes Lucas
Act. Pablo Gómez Garcia
Mat. Francisco Eduardo Zavala Rodríguez

Introducción

Los modelos de inteligencia artificial son en la actualidad muy utilizados en distintas áreas, no solo para su investigación, si bien es cierto, son herramientas que durante mucho tiempo han facilitado el manejo y comprensión de grandes volúmenes de información, compactándola y permitiendo realizar análisis cada vez más refinados. La importancia de este tema dentro de las áreas financieras ha sido de vital importancia ya que constantemente las instituciones financieras generan cantidades exorbitantes de información sobre sus clientes y productos bancarios, debido a esta ardua tarea en seleccionar siempre a los mejores clientes se han planteado modelos cada vez más complejos para este tipo de tareas. Con el paso del tiempo los modelos se vuelven obsoletos y las empresas llegan a presentar ciertas complicaciones para seguir llevando a cabos sus tareas diarias, es por eso que con frecuencia se actualizan estos modelos, la mejora más rapida a estos problemas es pormedio de los ajustes a sus variables objetivos. Con esto las empresas generan tiempo para poder realizar pruebas piloto y al mismo tiempo realizan una mejora al modelo.

Trabajos Previos

Al día de hoy y por tratarse de un tema con demasiada relevancia en el sector Financiero, que además es una obligación regulatoria en diversos países para quienes realizan prestamos y ofrecen créditos personas físicas o morales, y como apoyo a estas instituciones existen diversas investigaciones, regulaciones y trabajos referentes a la elaboración de modelos de riesgo de crédito o mas conocido como "credit risk", en México por ejemplo el mismo Buró de crédito cuenta con un modelo predictivo de incumplimiento en sus obligaciones de pago (impago), que considera la información de los clientes dentro de todo el sistema Financiero Mexicano con

el cual genera un score de crédito el cuál indica si la persona cumple o no cumple con sus pagos, al mismo tiempo existe una metodología que se ocupa comunmente tanto en regulaciones internaciones como nacionales; El modelo al que comúnmente se le atribuye esta tarea es la "regresión logística" debido a su capacidad predictiva y la sencilles del modelo.

Las razones son que es un algoritmo que logra capturar el patrón y aporta estabilidad entes los cambios de comportamiento y adicionalmente permite explicar los resultados de una manera sencilla a las personas que no tienen conocimiento del tema. Por otro lado en distintos trabajos se proponen Algoritmos como el Randomforest, XGBoosting, Redes Neuronales o distintos tipos de Árboles de Decisión, que son algoritmos que tienen un costo computacional alto pero también un nivel predictivo mayor que ele de la regresión Logística.

Desde nuestra perspectiva consideramos que para este proyecto es posible probar distintos algoritmos y evaluar sus desempeños. Por otro lado es importante mencionar que se deberán tomar en cuenta los diversos aspectos de negocio en este giro así como regulatorios. Consideramos que el mayor esfuerzo estará en la preparación de los datos de entrada y las variables propuestas. Lo que se espera de este trabajo es poner en practica todos los conocimientos adquiridos hasta el momento para la preparación y limpieza de los datos, la modelación en conjunto con distintas propuestas de algoritmos para finalmente dar una conclusión de porque creemos que el modelo propuesta cumple con el objetivo de la competencia o en su caso por qué no lo cumple.

Planteamiento del problema

El problema que se plantea en esta competición es acerca del incumplimiento de pago en temas crediticios. Se pide obtener un modelo predictivo (supervisado) cuya variable objetivo será el evento de si un cliente incumplió en su pago o no incumplió. Básicamente se trata de un modelo de riesgo de crédito cuya finalidad es dar certidumbre de que el cliente va a devolver lo que se le preste más ciertos intereses antes de otorgarle un crédito. Al día de hoy la empresa American Express ya cuenta con un modelo predictivo de impago con cierto Recall y Precisión en su día a día, sin embargo lo que pretende esta competencia es obtener un modelo con mejor desempeño (mayor Recall y mayor Precisión). La relevancia que tiene este proyecto a nuestra línea de trabajo es que nos permitirá probar distintos algoritmos supervisados para obtener un mejor modelo, así como poder proponer distintas variables de entrada al modelo que desde un sentido de negoción puedan aportar mayor potencia a la predicción. Por otro lado nos ayudará a ganar experiencia en temas de riesgo de Crédito lo cual es un campo muy amplio laboralmente hablando dentro del sistema Financiero.

Ya mencionado anteriormente que las instituciones financieras constantemente buscan mejorar y seleccionar a las personas que acuden a sus instalaciones para solicitar creditos de nomina, tarjetas, créditos de PYMES etc. Si bien existe gran variedad de trabajos de investigación sobre la probabilidad de impago,en un informe financiero elaborado por @BBVA 2010, muestra como las Tasas de incumplimiento para distinta segmentación de las herramientas en función de la antigüedad, se ven menos afectadas ya que estas están separadas por 3 grandes grupos: 1.no cliente 2.Cliente no vinculado 3.cliente vinculado Al segmentar de esta manera diversifica las

pérdidas y puede identificar claramente como las tasas de inclumplimiento son más bajas en las carteras de las personas o empresas que son clientes, por el contrario para las personas que no tienen ningún vinculo con la institución tienen en promedio 10% más de incumplir. Una vez analizado esto nos da una idea de como poder solucionar este problema, sin emebargo al contar con tanto detalle dentro de la información nos concentraremos únicamente en las varibles de gasto pago y variables de morosidad, y por medio de clusters segmentar para ver que variables son las más significativas para el modelo y posteriormente por medio del modelo de regresión logística identificar las variables que son significativas para la predicción del modelo de Default. *Nota: se esta trabajando con una muestra del 20% de los datos, ya que la información tiene un peso de 50 gb y es imposible manejarla dentro de una computadora con instancia local.

Descripción de variables del conjunto de datos

El conjunto de datos contiene características de perfil agregadas para cada cliente en cada fecha de estado de cuenta. Las funciones se anonimizan y normalizan, y se clasifican en las siguientes categorías generales:

D_* = Variables de morosidad

S_* = Variables de gasto

P_* = Variables de pago

B_* = Variables de balance

R_* = Variables de riesgo

Nota: Todas la variables son de tipo intervalo (continuas)

Desarrollo

Solución propuesta para el problema planteado en este proyecto.

* Análisis Exploratorio de Datos

Dentro de esta etapa se incluye lo siguiente:

1. Se explora y se gráfica la variable target para ver la distribución de los casos de impago.
2. Por medio de un análisis descriptivo de las variables observamos casos nulos, valores atípicos, media, desviación estándar, entre otros, se eliminan los campos con más del 70% con valores nulos.
3. Por medio de un análisis de correlación se busca eliminar la multicolinealidad entre las variables explicativas y también la relación con nuestra variable objetivo de cada uno de los grupos de variables, por ejemplo, morosidad, balance, étc.

4. Por medio de un análisis de clustering entre variables y manipulación un tanto manual por parte del analista se busca eliminar la redundancia entre variables y dejar aquellas más importantes y que explican mejor la variable objetivo.

* Modelación

1. Acercamiento 1: Modelación con variables estandarizadas e imputación de valores nulos con la mediana.

- Modelo 1:

  ```
  En esta parte se intentará obtener un primer resultado a través de
  una regresión logística, primero se estandarizarán las variables, se
  propone una proporción de 80% - 20% para los conjuntos de datos
  entrenamiento y prueba, se ejecuta el ajuste y validación del modelo
  y se obtiene el desempeño del modelo a través de la matriz de
  confusión y las métricas del ajuste precision, recall, f1 y el
  accuracy.
  ```

- Modelo 2:

  ```
  En esta parte se probará un segundo resultado solo con las variables
  estandarizadas a través de un árbol de decisión, tambiém se obtendrán
  los valores de desempeño del modelo a través de matríz y métricas.
  ```

- Modelo 3:

  ```
  En esta parte se probará un tercer resultado a través de un bosques
  aleatorios(Random Forest) con 100 estimadores, se entrenará y
  validará el modelo obteniendo nuevamente matriz de confusión y
  métricas de desempeño.
  ```

1. Acercamiento 2: Aplicando ingeniería de variables.

Ahora se eliminaran los registros con valores pérdidos a diferencia del primer approah en donde se decidió imputar valores con la mediana. Se aplica ingeniería de variables a través de Información Mutua cuya ventaja es que puede detectar cualquier tipo de relación, mientras que la correlación solo detecta relaciones lineales.

- Modelo 1:

  ```
  Para poder realizar una comparación entre acercamientos y para
  obtener un mejor desempeño el conjunto de datos se ajustará y
  validará con una distribución de 80% - 20%  a través de una regresión
  ```

logística y se obtendrám las métricas de desempeño.

- Modelo 2:

  Nuevamente con una conjunto de datos de entrenamiento del 80% y prueba de 20% se ajustará y validará un árbol de decisión y se obtendrá el desempeño del modelo.

- Modelo 3:

  Se ajustará y validará con la tercara opción propuesta, se ajustará y validará un modelo de bosques aleatorios con 100 estimadores.

Al final en las conclusiones se hará una comparativa con los resultados obtenidos con los 2 acercamientos aplicado a 3 modelos cada uno.

** Librerías a utilizar **

Dentro de este bloque exportamos las librerías que vamos a utilizar a lo largo de este trabajo

In [1]:
```
#Importamos los paquetes que vamos a utilizar

# lectura y manipulación de archivos
import json as js
import pandas as pd
import io
import numpy as np

# gráficas
import matplotlib.pyplot as plt
import seaborn as sns

# herramientas estadísitcas
import random as rd
import math as mt
from time import time
from statistics import mean
```

** Importar el conjunto de datos **

En este bloque realizamos la lectura de los Datos desde un formato .pkl para obtimizar el tamaño.

In [3]:
```
#Cargamos los datos
import pickle as pk
df_orig=pd.read_pickle('df_sample_amex.pkl')
df=pd.read_pickle('df_sample_amex.pkl')
```

```
# Visualizamos los datos
df
```

Out[3]:

	customer_ID	S_2	P_2	D_39	B_1	
5230464	f1e881682c6eac031984d20d3620b30514dcdf3ec9ce0d...	2018-01-12	0.875387	0.000703	0.010017	0.8
35131	01a4df870c9a606a9463f6daa20fcfe6e9fd1d089a3369...	2017-12-08	0.825674	0.008442	0.048399	0.8
306853	0e592593c5c1cb6c02bb96928533c56885fffdfb431514...	2017-11-27	0.668637	0.006712	0.011058	1.0
671170	1f37fd3ba4c0996f965bf3ffe951a0fcc1fd0f822c5a96...	2017-06-19	0.295480	0.004250	0.221267	0.0
4395008	cb1b3179f5f237ffa5fe647f169b38f05072d8580aa7e9...	2017-10-05	0.215187	0.242586	0.060069	0.6
...	...	...	...	...	...	
5280720	f44e58840e0c3e4db14ec219c3d122156e76564ffbfde4...	2018-03-14	0.941024	0.179749	0.036868	1.0
795121	24dc0b7083c1d4dbfabb25d04c4a0f87bc6db4c712f38a...	2018-03-06	0.958159	0.000013	0.013432	1.0
3620268	a73e67ceea453acf0bcb42b46945ce47042be704a9326f...	2018-01-19	0.297357	0.035479	0.873243	0.8
2154673	638905c4d5fd5368d0140c32ce66c60bbd78cea311e024...	2018-03-08	0.773871	0.008020	0.007429	0.8
4102465	bda308aba0a7fffface7780ee366508620c05c9ce896569...	2017-10-24	0.508677	0.004288	0.105060	0.8

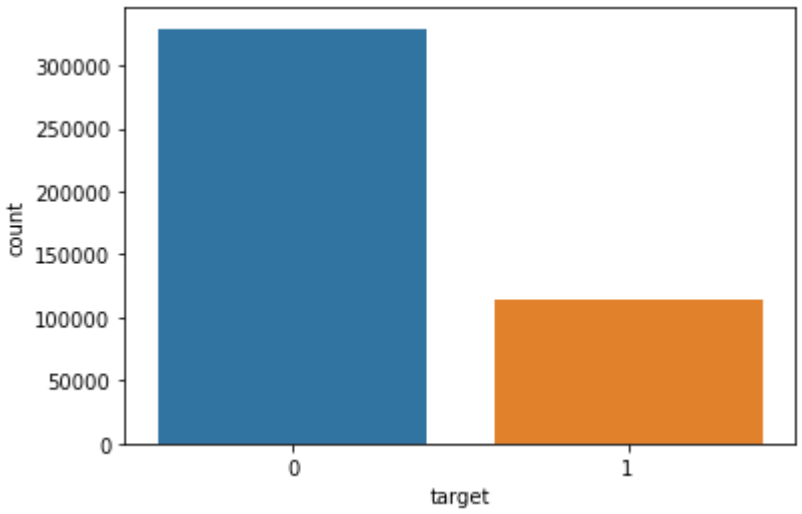
442542 rows × 191 columns

** Análisis exploratorio de datos **

Dentro de esta sección vamos a llevar a cabo el análisis exploratorio de datos, este consiste en realizar exploraciones visuales de las variables, los tipos de dato que tienen, estadísticas descripivas, indicadores de tendencia central y de disperción. De igual manera se analizan los valores perdidos, se realiza la imputaciónde datos, se realiza un análisis de correlación de las variables de entrada contra la variable objetivo y finalmente se analiza la correlación del total de las variables sin la target con el fin de realizar una reducción de la dimensionalidad por medio de un clustering de variables.

In [4]:
```
#Visualizamos la distribución de la Target por cada nivel
import seaborn as sns
sns.countplot(x = 'target',data = df )
```

Out[4]:
```
<AxesSubplot:xlabel='target', ylabel='count'>
```



Comentario

Se aprecia la siguiente proporción en la variable objetivo:

74%--> 0

26%--> 1

En un total de 442,542 con lo que se concluye que se puede modelar la regularidad estadística del patrón de 1's (Clientes que caen en impago)

In [5]:
```
# Visualizamos las columnas que contiene el dataset
df.columns
```

Out[5]:
```
Index(['customer_ID', 'S_2', 'P_2', 'D_39', 'B_1', 'B_2', 'R_1', 'S_3', 'D_41',
       'B_3',
       ...
       'D_137', 'D_138', 'D_139', 'D_140', 'D_141', 'D_142', 'D_143', 'D_144',
       'D_145', 'target'],
      dtype='object', length=191)
```

In [6]:
```
# Visualizamos los tipos de datos de las variables.
df.dtypes
```

Out[6]:
```
customer_ID     object
S_2             object
P_2            float64
D_39           float64
B_1            float64
                ...
D_142          float64
D_143          float64
D_144          float64
D_145          float64
target           int64
Length: 191, dtype: object
```

* Análisis Descriptivo *

Para poder entender de que manera están estrucutrados los datos analizaremos cada una de las variables por medio de estadística descriptiva

In [7]:
```
#Visualizamos los Estadigrafos de la variables
df_describe=df.describe().T
df_describe
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
P_2	437787.0	0.651225	0.244644	-3.726237e-01	0.473785	0.685655	0.861152	1.010000
D_39	442542.0	0.152065	0.270830	5.957676e-09	0.004514	0.009005	0.235317	5.362794
B_1	442542.0	0.125312	0.213382	-5.092473e-01	0.008956	0.032129	0.128480	1.324059
B_2	442294.0	0.620304	0.400589	4.737755e-07	0.105084	0.814252	1.002250	1.010000
R_1	442542.0	0.080174	0.227983	7.579088e-08	0.002897	0.005788	0.008686	3.004918
...	...	...	...	...	...	...	...	...
D_142	75867.0	0.388836	0.236186	-1.095431e-02	0.197536	0.378413	0.557781	2.217040
D_143	431569.0	0.180673	0.380557	7.740806e-08	0.003028	0.006066	0.009097	1.010000
D_144	438261.0	0.051939	0.181504	4.080035e-08	0.002760	0.005501	0.008256	1.343316
D_145	431569.0	0.063197	0.195072	5.794007e-08	0.003035	0.006069	0.009095	4.736494
target	442542.0	0.256123	0.436491	0.000000e+00	0.000000	0.000000	1.000000	1.000000

187 rows × 8 columns

In [8]:
```
# Observamos que variables tienen una cantidad fuerte de valores perdidos (más del 30%
# 303,169 representa el 70% aproximadamente de los registros, es decir quien tenga mer
# cuenta con más del 30% de valores perdidos en esa variable.
df_describe[df_describe['count']<303169].shape
```

Out[8]:
```
(31, 8)
```

In [9]:
```
# De la celda anterior tenemos 31 variables con más del 30% de valores perdidos, se vi
df_describe[df_describe['count']<303169].T.columns
```

Out[9]:
```
Index(['D_42', 'D_49', 'D_50', 'D_53', 'D_56', 'S_9', 'B_17', 'D_66', 'D_73',
       'D_76', 'D_77', 'R_9', 'D_82', 'B_29', 'D_87', 'D_88', 'D_105', 'D_106',
       'R_26', 'D_108', 'D_110', 'D_111', 'B_39', 'B_42', 'D_132', 'D_134',
       'D_135', 'D_136', 'D_137', 'D_138', 'D_142'],
      dtype='object')
```

In [10]:
```
# Visualizamos los estadigrafos de las variables con más del 30% de valores perdidos
df_describe[df_describe['count']<303169]
```

Out[10]:

	count	mean	std	min	25%	50%	75%	max
D_42	76948.0	0.185427	0.240634	-3.099458e-04	0.035656	0.118317	0.247200	4.186470
D_49	43930.0	0.191369	0.334243	3.886374e-06	0.061275	0.129370	0.245495	40.444644
D_50	188397.0	0.172939	0.555797	-3.127269e+00	0.064290	0.108622	0.185829	82.245252
D_53	113105.0	0.076461	0.202608	2.382584e-07	0.006153	0.013512	0.050590	5.465084
D_56	197911.0	0.203441	0.216372	-1.687785e-02	0.086601	0.148851	0.249321	10.950100
S_9	205049.0	0.074567	0.197249	1.219565e-06	0.009720	0.019394	0.053169	2.839586
B_17	187694.0	0.716171	0.386906	1.823382e-07	0.481305	0.933233	1.002399	1.010000
D_66	48794.0	0.989671	0.101107	0.000000e+00	1.000000	1.000000	1.000000	1.000000
D_73	5436.0	0.161995	0.223062	-4.043599e-02	0.058689	0.104513	0.175934	3.676230
D_76	48595.0	0.139674	0.279400	4.135793e-06	0.015380	0.058855	0.166777	15.132615
D_77	242811.0	0.250375	0.231461	7.095482e-08	0.069168	0.203138	0.366952	10.009876
R_9	25826.0	0.229530	0.189864	3.121485e-07	0.169100	0.172710	0.176324	1.509810
D_82	114943.0	0.457305	0.189899	3.400617e-07	0.501509	0.504414	0.507354	3.008526
B_29	30977.0	0.032056	0.272230	3.091880e-07	0.002546	0.005089	0.007611	9.110773
D_87	299.0	1.000000	0.000000	1.000000e+00	1.000000	1.000000	1.000000	1.000000
D_88	486.0	0.178258	0.235302	1.280797e-03	0.024479	0.078019	0.236751	1.477829
D_105	200065.0	0.368731	0.250632	-2.415681e-02	0.168672	0.332206	0.524443	5.475551
D_106	43593.0	0.205001	1.383892	1.945481e-07	0.051696	0.136031	0.262200	278.267250
R_26	49267.0	0.085723	0.243908	1.994525e-07	0.005315	0.037100	0.077308	8.682504
D_108	2331.0	0.081877	0.315079	6.726847e-06	0.002748	0.005430	0.008180	4.003198
D_110	2394.0	0.743788	0.306450	-2.329955e-02	0.527155	0.892722	1.003839	1.009979
D_111	2394.0	0.874180	0.272013	3.091680e-06	1.000540	1.003549	1.006696	1.010000
B_39	2571.0	0.253961	0.296193	-4.241231e-01	0.057346	0.146453	0.292805	1.053430
B_42	5548.0	0.108457	0.315585	5.134449e-05	0.007266	0.021330	0.103246	8.976747
D_132	43695.0	0.207932	0.249124	-1.297553e-02	0.071162	0.157474	0.275496	5.947635
D_134	15815.0	0.333040	0.298201	-1.120956e-02	0.114240	0.220150	0.464010	1.009990
D_135	15815.0	0.030824	0.158594	7.917598e-07	0.002605	0.005157	0.007750	1.009996
D_136	15815.0	0.244268	0.208292	3.217928e-06	0.009575	0.253995	0.258222	1.756223
D_137	15815.0	0.015245	0.100763	3.478927e-07	0.002529	0.005053	0.007566	1.009868
D_138	15815.0	0.162310	0.268002	3.527743e-08	0.003532	0.007001	0.501421	2.509801
D_142	75867.0	0.388836	0.236186	-1.095431e-02	0.197536	0.378413	0.557781	2.217040

Observación

Como podemos ver, tenemos 31 variables que no cuentan con más del casi 70% de observaciones en el conjunto de datos.

Entre ellas encontramos que 24 son variables de morosidad, 2 variables de riesgo, 1 de gasto y 4 de balance (saldo), por lo que se decide eliminarlas del análisis.

In [11]:
```
#Eliminamos la Fecha de transacción ya que no nos es util, así como las variables con
col_drop = ['S_2','customer_ID','D_42','D_49','D_50','D_53','D_56','S_9','B_17','D_66'
            'B_29','D_87','D_88','D_105','D_106','R_26','D_108','D_110','D_111','B_39'
            'D_136','D_137','D_138','D_142','D_63','D_64','B_31']
df.drop(col_drop, inplace=True, axis=1)
```

In [12]:
```
# Obtenemos el % total de valores perdidos que aún quedan en la base
df.isnull().values.mean() * 100
```

Out[12]:
```
2.077886975845267
```

In [13]:
```
# Obtenemos las columnas del conjunto de datos que contienen parte del 2% de valores p
cols_with_missing = [col for col in df.columns
                     if df[col].isnull().any()]
cols_with_missing.sort()
# Imprimimos el número de variables con valores missing
len(cols_with_missing)
```

Out[13]:
```
88
```

In [14]:
```
# Visualizamos la dimensión del data frame original
df.shape
```

Out[14]:
```
(442542, 155)
```

Observación

Visualizamos que son 88 de las 154 (sin la variable fecha) aún cuentan con valores perdidos, sin embargo, vemos que en 41 de ellas es menor al 1%, 2 de ellas sí tienen un porcentaje mayor al 25% de valores perdidos y el resto esta entre 5% y 10%.

In [15]:
```
# Imprimimos las variables con más del 10% de valores perdidos y su porcentaje exacto
for i in range(len(cols_with_missing)):
    if (df[cols_with_missing[i]].isnull().values.mean() * 100)>10:
        print(cols_with_missing[i],df[cols_with_missing[i]].isnull().values.mean() * 1
```

```
D_43 31.49373392807914
D_46 24.182789430155783
D_48 13.138413981045865
D_61 10.701583126573297
D_62 13.882298177348137
S_27 25.766367937958428
S_3 18.233975532265863
S_7 18.233975532265863
```

Missing Values: Imputación de Valores

En esta sección vamos a imputal los valores perdidos con un metodo simple, este consiste en asignar la mediana de los datos de la variable a aquellos cuyo valor sea missing.

Este proceso es opcional para algunos algoritmos como los árboles de decisión pues estos aceptan los valores perdidos, sin en bargo en nuestro caso utilizaremos más adelante una regresión logística la cual es sencible a los valores perdidos.

In [16]:
```
# Relizaremos un proceso de imputación simple para el aún 2% de varoles del total de l
from sklearn.impute import SimpleImputer
col_names = df.columns
imputer = SimpleImputer(missing_values=np.nan, strategy='median', fill_value=None)
df = pd.DataFrame(imputer.fit_transform(df))
df.columns = col_names
```

Análisis de correlación

En esta sección continuaremos con el Análisis exploratorio de datos. Antes de comenzar a modelar es importante conocer la relación que existe entre las variables de entrada con la variable objetivo, esto con el fin de ver si existe una relación directa de alguna o varias variables y a que grado con nuestra target. Otra parte de este análisis es conocer la correlación que tienen entre sí las variables de entrada para poder quitar la redeundancia entre ellas a cambio de sacrificar un porcentaje de la varianza explicada.

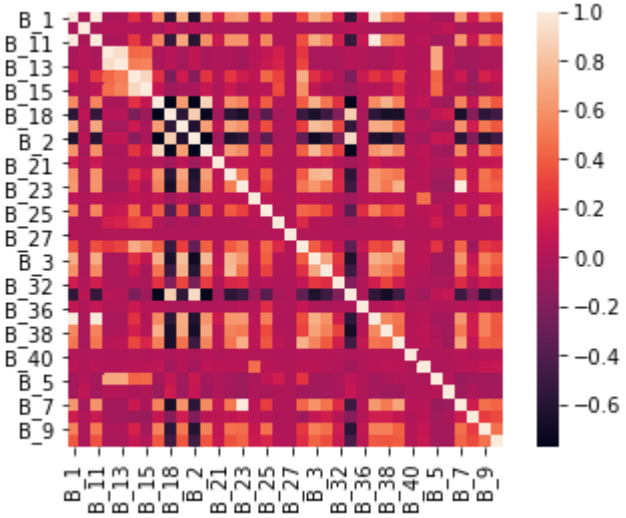
In [17]:
```
# Comenzamos el ánalisis partiendo las variables por tipo (morosidad, gasto, pago, bal

# Correlación variables de balance

# El siguiente código obtiene las correlaciones por el método de pearson y construye u
corr_B=df[['B_1', 'B_10', 'B_11', 'B_12', 'B_13', 'B_14', 'B_15', 'B_16', 'B_18', 'B_1
 'B_24', 'B_25', 'B_26', 'B_27', 'B_28', 'B_3', 'B_30', 'B_32', 'B_33', 'B_36', 'B_37'
 'B_6', 'B_7', 'B_8', 'B_9','target']].corr(method='pearson')

upper_corr_B=corr_B.where(np.triu(np.ones(corr_B.shape),k=1).astype(bool))
unique_corr_B=upper_corr_B.unstack().dropna()
sorted_corr_B=unique_corr_B.sort_values()
#sorted_corr_B=corr_B.unstack().sort_values()
sorted_corr_B
a=pd.DataFrame(sorted_corr_B)
a.reset_index(inplace=True)
#a[a["level_0"]=='target'].sort_values(by=0,ascending=False)
b=a[a["level_0"]=='target'].sort_values(by=0,ascending=False)
b[np.abs(b[0])>.3] # Por convención visualizamos aquellas con mas del 30% de correlaci
```

Out[17]:

	level_0	level_1	0
527	target	B_9	0.467484
514	target	B_7	0.419232
512	target	B_3	0.412367
511	target	B_23	0.411280
505	target	B_4	0.389986
500	target	B_1	0.378933
499	target	B_37	0.376236
498	target	B_19	0.374011
497	target	B_16	0.373834
496	target	B_38	0.369393
494	target	B_20	0.364909
493	target	B_11	0.358099
492	target	B_22	0.355531
489	target	B_30	0.337646
484	target	B_8	0.322482
45	target	B_33	-0.440924
44	target	B_2	-0.472300
43	target	B_18	-0.476918

In [18]:
```
# Imprimimos una gráfica de la matriz de correlación de todas las variables de Balance
sns.heatmap(df[['B_1', 'B_10', 'B_11', 'B_12', 'B_13', 'B_14', 'B_15', 'B_16', 'B_18',
                'B_22', 'B_23', 'B_24', 'B_25', 'B_26', 'B_27', 'B_28', 'B_3', 'B_30',
                'B_38', 'B_4', 'B_40', 'B_41', 'B_5', 'B_6', 'B_7', 'B_8', 'B_9','targ
```

Out[18]:
```
<AxesSubplot:>
```



Comentario

Con respecto a la correlación entre la variable target y las variables de Balance las que mayormente correlacionan tienen máximo un 47% de relación negativa con la target y 46% de relación positiva, de las 35 variables de Balance solo 18 cumplen con tener una relación de entre el 30% y 50% con la target.

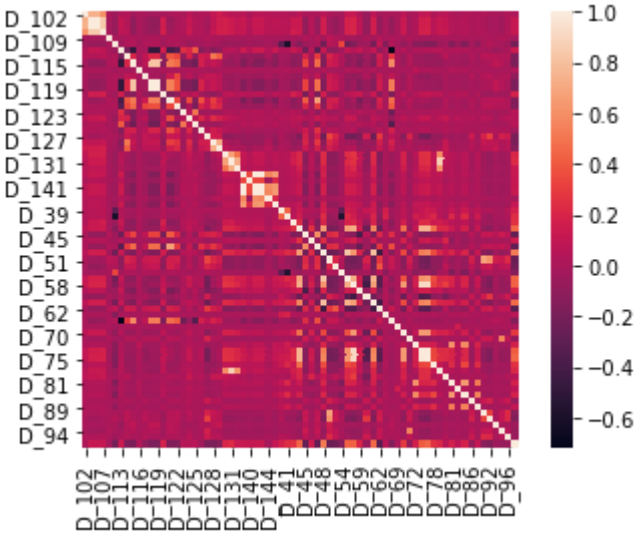
In [19]:
```
#Correlación variables de morosidad

# El siguiente código obtiene las correlaciones por el método de pearson y construye u
corr_D=df[['D_102', 'D_103', 'D_104', 'D_107', 'D_109', 'D_112', 'D_113', 'D_114', 'D_
 'D_120', 'D_121', 'D_122', 'D_123', 'D_124', 'D_125', 'D_126', 'D_127', 'D_128', 'D_1
 'D_140', 'D_141', 'D_143', 'D_144', 'D_145', 'D_39', 'D_41', 'D_43', 'D_44', 'D_45',
 'D_54', 'D_55', 'D_58', 'D_59', 'D_60', 'D_61', 'D_62', 'D_65', 'D_68', 'D_69', 'D_70
 'D_79', 'D_80', 'D_81', 'D_83', 'D_84', 'D_86', 'D_89', 'D_91', 'D_92', 'D_93', 'D_94
          'target']].corr(method='pearson')
upper_corr_D=corr_D.where(np.triu(np.ones(corr_D.shape),k=1).astype(bool))
unique_corr_D=upper_corr_D.unstack().dropna()
sorted_corr_D=unique_corr_D.sort_values()
#sorted_corr_B=corr_B.unstack().sort_values()
sorted_corr_D
a=pd.DataFrame(sorted_corr_D)
a.reset_index(inplace=True)
#(a[a["level_0"]=='target'].sort_values(by=0,ascending=False)).where(np.abs(a[0])>.3)
b=a[a["level_0"]=='target'].sort_values(by=0,ascending=False)
b[np.abs(b[0])>.3] # Por convención visualizamos aquellas con mas del 30% de correlaci
```

Out[19]:

	level_0	level_1	0
2421	target	D_48	0.527277
2410	target	D_61	0.467580
2408	target	D_44	0.448552
2407	target	D_75	0.446553
2405	target	D_58	0.435624
2404	target	D_55	0.431538
2401	target	D_74	0.410502
19	target	D_62	-0.314648

In [20]:
```
# Imprimimos una gráfica de la matriz de correlación de todas las variables de Morosid
sns.heatmap(df[['D_102', 'D_103', 'D_104', 'D_107', 'D_109', 'D_112', 'D_113', 'D_114'
                'D_119', 'D_120', 'D_121', 'D_122', 'D_123', 'D_124', 'D_125', 'D_126'
                'D_131', 'D_133', 'D_139', 'D_140', 'D_141', 'D_143', 'D_144', 'D_145'
                'D_45', 'D_46', 'D_47', 'D_48', 'D_51', 'D_52', 'D_54', 'D_55', 'D_58'
                'D_65', 'D_68', 'D_69', 'D_70', 'D_71', 'D_72', 'D_74', 'D_75', 'D_78'
                'D_84', 'D_86', 'D_89', 'D_91', 'D_92', 'D_93', 'D_94', 'D_96','target
```

Out[20]:
```
<AxesSubplot:>
```



Comentario

Solo hay 8 variables de morosidad que correlacionan en más del 30% con la variable target.

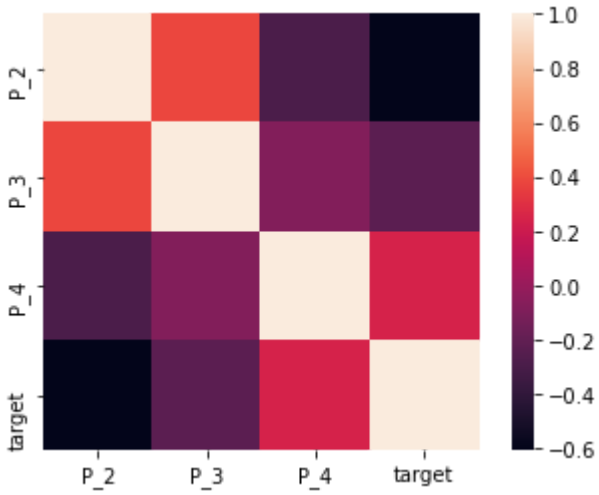
In [21]:
```
#Correlación variables de pago
# El siguiente código obtiene las correlaciones por el método de pearson y construye u
corr_D=df[['P_2', 'P_3', 'P_4','target']].corr(method='pearson')
upper_corr_D=corr_D.where(np.triu(np.ones(corr_D.shape),k=1).astype(bool))
unique_corr_D=upper_corr_D.unstack().dropna()
sorted_corr_D=unique_corr_D.sort_values()
#sorted_corr_B=corr_B.unstack().sort_values()
sorted_corr_D
a=pd.DataFrame(sorted_corr_D)
a.reset_index(inplace=True)
#(a[a["level_0"]=='target'].sort_values(by=0,ascending=False)).where(np.abs(a[0])>.3)
b=a[a["level_0"]=='target'].sort_values(by=0,ascending=False)
#b[np.abs(b[0])>.3]
b
```

Out[21]:

	level_0	level_1	0
4	target	P_4	0.242053
2	target	P_3	-0.228331
0	target	P_2	-0.604900

In [22]:
```
# Imprimimos una gráfica de la matriz de correlación de todas las variables de Pago, e
sns.heatmap(df[['P_2', 'P_3', 'P_4','target']].corr(method='pearson'),square=True)
```

Out[22]:
```
<AxesSubplot:>
```



Comentario

Solo hay una variable de Pago que correlaciona significativamente con la variable target.

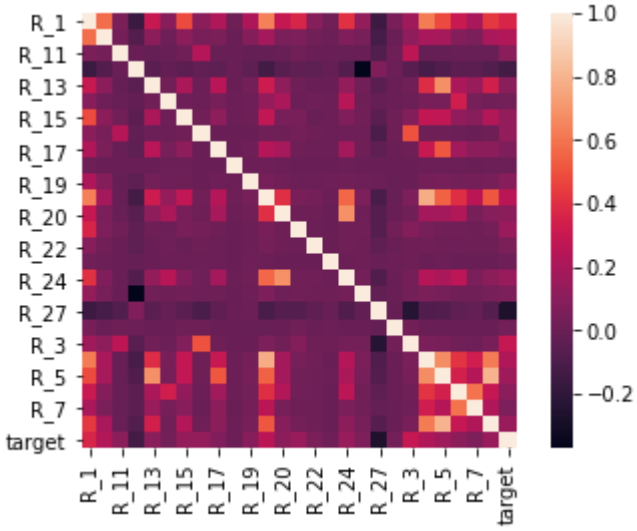
In [23]:
```
#Correlación variables de riesgo
# El siguiente código obtiene las correlaciones por el método de pearson y construye u
corr_D=df[['R_1', 'R_10', 'R_11', 'R_12', 'R_13', 'R_14', 'R_15', 'R_16', 'R_17', 'R_1
 'R_23', 'R_24', 'R_25', 'R_27', 'R_28', 'R_3', 'R_4', 'R_5', 'R_6', 'R_7', 'R_8','tar
upper_corr_D=corr_D.where(np.triu(np.ones(corr_D.shape),k=1).astype(bool))
unique_corr_D=upper_corr_D.unstack().dropna()
sorted_corr_D=unique_corr_D.sort_values()
#sorted_corr_B=corr_B.unstack().sort_values()
sorted_corr_D
a=pd.DataFrame(sorted_corr_D)
a.reset_index(inplace=True)
#(a[a["level_0"]=='target'].sort_values(by=0,ascending=False)).where(np.abs(a[0])>.3)
b=a[a["level_0"]=='target'].sort_values(by=0,ascending=False)
b[np.abs(b[0])>.25] # Por convención visualizamos aquellas con mas del 25% de correlac
```

Out[23]:

	level_0	level_1	0
326	target	R_1	0.362677
317	target	R_3	0.293585
309	target	R_2	0.259418
1	target	R_27	-0.269047

In [24]:
```
# Imprimimos una gráfica de la matriz de correlación de todas las variables de Riesgo,
sns.heatmap(df[['R_1', 'R_10', 'R_11', 'R_12', 'R_13', 'R_14', 'R_15', 'R_16', 'R_17',
                'R_20', 'R_21', 'R_22', 'R_23', 'R_24', 'R_25', 'R_27', 'R_28', 'R_3',
                'R_8','target']].corr(method='pearson'),square=True)
```

Out[24]:
```
<AxesSubplot:>
```



Comentario

La variable de riesgo que correlaciona más con la variable target lo hace con un 36% en un sentido positivo.

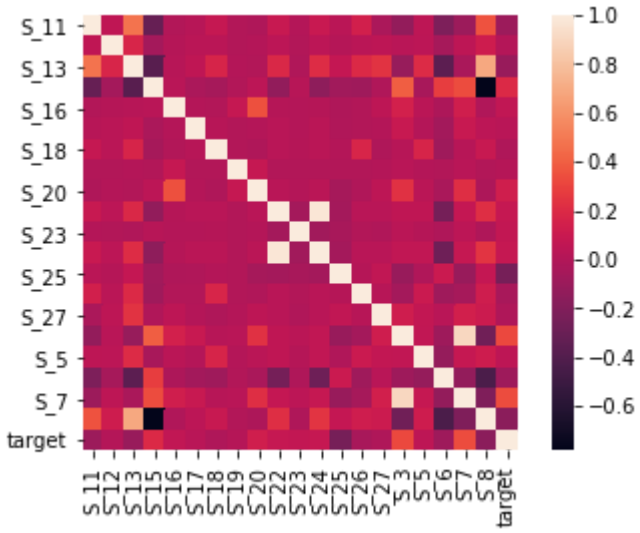
In [25]:
```
#Correlación variables de gasto
# El siguiente código obtiene las correlaciones por el método de pearson y construye u
corr_D=df[['S_11', 'S_12', 'S_13', 'S_15', 'S_16', 'S_17', 'S_18', 'S_19', 'S_20', 'S_
 'S_27', 'S_3', 'S_5', 'S_6', 'S_7', 'S_8','target']].corr(method='pearson')
upper_corr_D=corr_D.where(np.triu(np.ones(corr_D.shape),k=1).astype(bool))
unique_corr_D=upper_corr_D.unstack().dropna()
sorted_corr_D=unique_corr_D.sort_values()
#sorted_corr_B=corr_B.unstack().sort_values()
sorted_corr_D
a=pd.DataFrame(sorted_corr_D)
a.reset_index(inplace=True)
#(a[a["level_0"]=='target'].sort_values(by=0,ascending=False)).where(np.abs(a[0])>.3)
b=a[a["level_0"]=='target'].sort_values(by=0,ascending=False)
b[np.abs(b[0])>.2] # Por convención visualizamos aquellas con mas del 20% de correlaci
```

Out[25]:

	level_0	level_1	0
201	target	S_7	0.335293
200	target	S_3	0.323726
8	target	S_25	-0.243644

In [26]:
```
# Imprimimos una gráfica de la matriz de correlación de todas las variables de Gasto,
sns.heatmap(df[['S_11', 'S_12', 'S_13', 'S_15', 'S_16', 'S_17', 'S_18', 'S_19', 'S_20'
 'S_27', 'S_3', 'S_5', 'S_6', 'S_7', 'S_8','target']].corr(method='pearson'),square=Tr
```

Out[26]:
```
<AxesSubplot:>
```



Comentario

La variable de gasto que correlaciona más con la variable target lo hace con un 33% en un sentido positivo.

Observación final del análisis de correlación

La variable objetivo "target" tiene una mayor relación con variables de balance en el análisis bivariado,detrás variables de morosidad. Las variables de gasto son menos las que correlacionan con un valor mayor al 30%.

Clustering de variables

En esta sección del código vamos a aplicar un método que consiste en obtener la relación directa que existe entre grupos de las variables de entrada de nuestro análisis, esto con la finalidad de reducir la dimensionalidad de nuestro dataset y eliminar la redundancia entre dichas variables.

In [27]:
```
# Si no se cuenta con el paquete varclushi instalado descomentar y ejecutar esta celda
#!pip install varclushi
```

In [28]:
```
#Realizamos una copia del Data Set Original para eliminar la Target ya que en el siguie
df_sin_target=df.copy()
```

In [29]:
```
#Clusterin de variables
from varclushi import VarClusHi
#Para este analisis no necesitamos la variable objetivo
df_sin_target.drop('target', inplace=True, axis=1)
#Realizamos el clustering
var_clust_model=VarClusHi(df_sin_target,maxeigval2=0.7, maxclus=None)
var_clust_model.varclus()
```

Out[29]:
```
<varclushi.varclushi.VarClusHi at 0x1d3e6e0ddf0>
```

In [30]:
```
#Visualizamos los clusters formados
var_clust_model.rsquare
```

Out[30]:

	Cluster	Variable	RS_Own	RS_NC	RS_Ratio
0	0	B_2	0.853129	0.511265	0.300513
1	0	B_9	0.487800	0.301069	0.732834
2	0	B_16	0.820730	0.467375	0.336578
3	0	B_18	0.813792	0.520910	0.388669
4	0	B_20	0.792746	0.520247	0.432000
...	...	...	...	...	...
149	81	D_114	1.000000	0.253386	0.000000
150	82	R_20	0.841753	0.388705	0.258872
151	82	R_24	0.841753	0.281577	0.220271
152	83	D_122	1.000000	0.170414	0.000000
153	84	D_39	1.000000	0.185513	0.000000

154 rows × 5 columns

In [31]:
```
#Exportamos la tabla resultante en el proceso anterior
var_clus=var_clust_model.rsquare
var_clus.to_csv('var_clust.csv')
```

Observación

Tras el Análisis de Clustering de variables podemos tomar la variable con la distancia menor al centroide dentro de cada cluster, esto es con el valor del RS_Ratio y así elegiriamos una variable representante de cada grupo de acuerdo a su nivel de correlación. Por otro lado aquellos cluster cuyo número de variables es uno significa que son variables que no se correlaconan con ninguna otra.

El criterio tomado es que la iteración del algoritmo pare cuando se tenga el 70% de la Varianza explicada.

En este caso de 154 variables que entraron al análisis podemos explicar el 70% de la varianza del fenómeno solo con 85 variables.

In [32]:
```
#Eliminamos la variables correlacionada de nuestro Data set original
col_drop2 = ['B_2','B_16','B_18','B_20','B_38','B_9','D_118','D_115','D_89','R_5','S_8
             'D_55','P_2','D_143','D_141','D_145','D_103','D_107','D_102','R_21','D_79'
             'D_130','R_25','R_12','B_13','B_5','D_45','P_4','S_24','D_72','D_84','D_11
             'R_3','R_16','S_3','D_91','D_92','B_11','B_37','D_128','D_125','R_1','D_78
             'D_74','D_58','D_70','D_44','B_4','R_2','R_8','B_28','D_62','D_47','B_22',
```

```
            'B_3','R_6','P_3','B_14','B_23','B_15','D_68','B_21','S_13','R_20']
df.drop(col_drop2, inplace=True, axis=1)
```

In [33]:
```
# Visualizamos que las variables ya no esten en el dataframe
df
```

Out[33]:

	D_39	B_1	D_41	D_43	D_46	D_48	B_6	B_7	B_8	
0	0.000703	0.010017	0.008048	0.074066	0.450230	0.053509	0.203157	0.021792	0.002591	0.33
1	0.008442	0.048399	0.000891	0.050604	0.459737	0.292940	0.031634	0.128801	0.001507	0.00
2	0.006712	0.011058	0.007368	0.038598	0.501244	0.941572	0.058611	0.073880	1.008261	0.34
3	0.004250	0.221267	0.008111	0.052670	0.459737	0.292940	0.004841	0.090462	1.003994	0.00
4	0.242586	0.060069	0.411929	0.089859	0.459737	0.536306	0.086689	0.072637	0.008448	0.00
...	...	...	...	...	...	...	...	...	...	
442537	0.179749	0.036868	0.007442	0.073280	0.452435	0.059010	0.202117	0.031874	0.002307	1.00
442538	0.000013	0.013432	0.007888	0.063877	0.449594	0.026090	0.382536	0.049315	0.006618	0.33
442539	0.035479	0.873243	0.004677	0.122226	0.459737	0.039915	0.237358	0.599893	1.009359	0.00
442540	0.008020	0.007429	0.007628	0.089859	0.459737	0.292940	0.178784	0.001239	1.006196	0.00
442541	0.004288	0.105060	0.007221	0.089859	0.459737	0.292940	0.175042	0.048031	1.008910	0.00

442542 rows × 86 columns

Conclusión del Análisis exploratorio

Tras los resultados del análisis exploratorio concluimos que se cuenta con variables que tienen cierto nivel de relación con la variable objetivo, por ello podemos modelar el fenómeno. De igual manera hemos quitado la redundancia entre las variables de entrada reduciendo el dataset de 154 a 85 variables las cuales ya no tienen valores perdidos.

** Modelación Parte 1 **

En esta parte vamos correr algunos modelos que nos puedan decir la relación que existe entre nuestras variables de entrada y la variable objetivo. Se busca obtener un modelo que prediga el impago con las variables porpuestas resultantes de nuestro Análisis exploratorio.

** Modelo 1: Regresión Logística **

El primer modelo a probar es una regresión Logística, la cual es el algoritmo más común para temas de riesgo de crédito.

In [34]:
```
# Separación del conjunto en inputs y target
Y=df['target']
X=df.copy()
X.drop('target', inplace=True, axis=1)
```

In [35]:
```
# Realizamos un proceso de estandarización de las variables con el fin de que todas se
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = pd.DataFrame(scaler.fit_transform(X), index=X.index, columns=X.columns)
```

In [36]:
```
# Conjuntos de entrenamiento y test con un 80%-20%
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state = 1, shuffle=Tr
```

In [37]:
```
# Importamos librerías
from sklearn.metrics import classification_report,accuracy_score
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
```

In [38]:
```
# Creamos un Primer modelo utilizando matrices como en scikitlearn y Visualizamos el r
# ==============================================================================
# A la matriz de predictores se le tiene que añadir una columna de 1s para el intercep
X_train = sm.add_constant(X_train, prepend=True)
modelo = sm.Logit(endog=Y_train, exog=X_train,)
modelo = modelo.fit()
print(modelo.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.313508
         Iterations 9
                           Logit Regression Results
==============================================================================
Dep. Variable:                 target   No. Observations:               354033
Model:                          Logit   Df Residuals:                   353947
Method:                           MLE   Df Model:                           85
Date:                Thu, 01 Dec 2022   Pseudo R-squ.:                  0.4490
Time:                        10:54:10   Log-Likelihood:            -1.1099e+05
converged:                       True   LL-Null:                   -2.0143e+05
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -1.9025      0.008   -235.348      0.000      -1.918      -1.887
D_39           0.1292      0.006     20.492      0.000       0.117       0.142
B_1            0.2200      0.008     29.144      0.000       0.205       0.235
D_41           0.2510      0.007     37.004      0.000       0.238       0.264
D_43           0.2631      0.006     42.819      0.000       0.251       0.275
D_46           0.1300      0.006     21.822      0.000       0.118       0.142
D_48           0.5380      0.007     73.773      0.000       0.524       0.552
B_6           -0.1098      0.023     -4.811      0.000      -0.154      -0.065
B_7            0.1640      0.007     22.037      0.000       0.149       0.179
B_8            0.4215      0.007     63.638      0.000       0.408       0.434
D_51          -0.3972      0.010    -39.840      0.000      -0.417      -0.378
D_52          -0.1174      0.007    -17.226      0.000      -0.131      -0.104
B_10          -0.0685      0.026     -2.626      0.009      -0.120      -0.017
S_5            0.0465      0.006      8.203      0.000       0.035       0.058
S_6            0.1726      0.008     22.836      0.000       0.158       0.187
R_4            0.1401      0.007     20.376      0.000       0.127       0.154
S_7            0.2691      0.006     45.324      0.000       0.257       0.281
B_12           0.0090      0.006      1.568      0.117      -0.002       0.020
D_59           0.0185      0.006      3.127      0.002       0.007       0.030
D_60           0.1462      0.008     18.102      0.000       0.130       0.162
S_11          -0.0800      0.006    -12.935      0.000      -0.092      -0.068
D_65           0.0347      0.009      3.686      0.000       0.016       0.053
S_12           0.0376      0.007      5.780      0.000       0.025       0.050
D_69           0.0063      0.005      1.212      0.225      -0.004       0.017
S_15           0.1119      0.006     18.356      0.000       0.100       0.124
D_75           0.2909      0.007     39.832      0.000       0.277       0.305
B_24           0.0716      0.007     10.206      0.000       0.058       0.085
R_7            0.0145      0.005      2.846      0.004       0.005       0.025
B_25           0.0070      0.006      1.101      0.271      -0.005       0.019
B_26           0.0016      0.006      0.264      0.792      -0.010       0.013
S_16           0.0306      0.007      4.196      0.000       0.016       0.045
D_80           0.0231      0.006      4.100      0.000       0.012       0.034
R_10           0.0547      0.006      9.656      0.000       0.044       0.066
R_11           0.0848      0.005     17.298      0.000       0.075       0.094
B_27           0.0073      0.005      1.357      0.175      -0.003       0.018
D_81           0.0904      0.007     12.924      0.000       0.077       0.104
S_17           0.0137      0.005      2.872      0.004       0.004       0.023
R_13          -0.0079      0.008     -0.956      0.339      -0.024       0.008
D_83           0.0436      0.005      9.090      0.000       0.034       0.053
R_14           0.0138      0.005      2.896      0.004       0.004       0.023
R_15           0.0749      0.005     14.105      0.000       0.065       0.085
B_30           0.0730      0.006     12.899      0.000       0.062       0.084
S_18           0.0358      0.006      6.050      0.000       0.024       0.047
D_86          -0.0518      0.008     -6.648      0.000      -0.067      -0.037
R_17          -0.0134      0.006     -2.152      0.031      -0.026      -0.001
```

```
R_18          0.0031      0.005      0.590      0.555      -0.007       0.013
S_19          0.0120      0.005      2.208      0.027       0.001       0.023
R_19          0.0081      0.005      1.779      0.075      -0.001       0.017
B_32         -0.0130      0.005     -2.814      0.005      -0.022      -0.004
S_20          0.0681      0.006     12.345      0.000       0.057       0.079
B_33         -0.1924      0.008    -24.774      0.000      -0.208      -0.177
R_22          0.0182      0.005      4.033      0.000       0.009       0.027
R_23          0.0210      0.005      4.159      0.000       0.011       0.031
D_93          0.0260      0.008      3.375      0.001       0.011       0.041
D_94         -0.0429      0.012     -3.567      0.000      -0.066      -0.019
R_24          0.0693      0.006     12.104      0.000       0.058       0.081
D_96         -0.0131      0.008     -1.748      0.081      -0.028       0.002
S_22          0.2402      0.024     10.170      0.000       0.194       0.287
S_23          0.1143      0.014      8.393      0.000       0.088       0.141
S_25         -0.0501      0.005    -10.365      0.000      -0.060      -0.041
S_26         -0.0081      0.009     -0.934      0.350      -0.025       0.009
D_104        -0.0109      0.006     -1.897      0.058      -0.022       0.000
B_36          0.0489      0.004     11.055      0.000       0.040       0.058
R_27         -0.1770      0.005    -35.932      0.000      -0.187      -0.167
D_109        -0.0220      0.013     -1.758      0.079      -0.047       0.003
B_40          0.0040      0.004      0.980      0.327      -0.004       0.012
S_27         -0.0519      0.005    -11.380      0.000      -0.061      -0.043
D_113        -0.0244      0.006     -3.805      0.000      -0.037      -0.012
D_114         0.0027      0.007      0.409      0.683      -0.010       0.016
D_116         0.0207      0.005      4.380      0.000       0.011       0.030
D_117        -0.0333      0.006     -5.414      0.000      -0.045      -0.021
D_119        -0.1745      0.008    -20.674      0.000      -0.191      -0.158
D_120         0.0708      0.005     14.029      0.000       0.061       0.081
D_121         0.0751      0.009      8.368      0.000       0.057       0.093
D_122        -0.0564      0.007     -7.976      0.000      -0.070      -0.043
D_123         0.0252      0.006      4.522      0.000       0.014       0.036
D_124         0.0385      0.007      5.463      0.000       0.025       0.052
D_126         0.0255      0.006      4.308      0.000       0.014       0.037
D_127        -0.1950      0.013    -14.485      0.000      -0.221      -0.169
D_129        -0.2814      0.007    -41.830      0.000      -0.295      -0.268
B_41          0.0449      0.006      7.528      0.000       0.033       0.057
D_131         0.1402      0.005     29.621      0.000       0.131       0.149
R_28          0.0118      0.004      2.753      0.006       0.003       0.020
D_139        -0.0212      0.007     -3.264      0.001      -0.034      -0.008
D_140         0.0476      0.005      9.526      0.000       0.038       0.057
D_144        -0.0657      0.007     -9.987      0.000      -0.079      -0.053
==============================================================================
```

Observación

Las variables que no aportan al modelo debido a que la probabilidad de Z es mayor al 0.005 (p-value) con lo que no se rechaza H0 (H0 : el predictor xp no contribuye al modelo (βp=0), en presencia del resto de predictores) son:

B_10 B_12 D_69 B_25 B_26 B_27 R_13 R_17 R_18 S_19 R_19 B_32 D_96 S_26 D_104 D_109 B_40 D_114 R_28

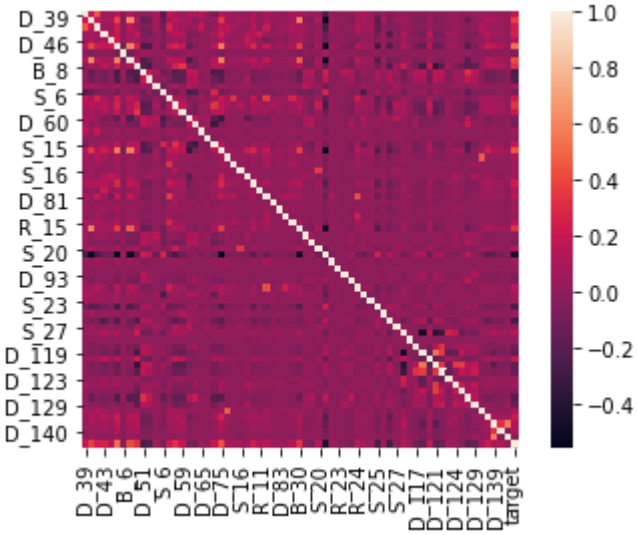
Por otro lado las Variables que si contribuyen al modelo son:

D_39 B_1 D_41 D_43 D_46 D_48 B_6 B_7 B_8 D_51 D_52 S_5 S_6 R_4 S_7 D_59 D_60 S_11 D_65 S_12 S_15 D_75 B_24 R_7 S_16 D_80 R_10 R_11 D_81 S_17 D_83 R_14 R_15 B_30 S_18 D_86 S_20

B_33 R_22 R_23 D_93 D_94 R_24 S_22 S_23 S_25 B_36 R_27 S_27 D_113 D_116 D_117 D_119
D_120 D_121 D_122 D_123 D_124 D_126 D_127 D_129 B_41 D_131 D_139 D_140 D_144

In [39]:
```
#Correlación variables de que aportan al modelo
corr_D=df[['D_39','B_1','D_41','D_43','D_46','D_48','B_6','B_7','B_8','D_51','D_52','S
           'S_11','D_65','S_12','S_15','D_75','B_24','R_7','S_16','D_80','R_10','R_11'
           'B_30','S_18','D_86','S_20','B_33','R_22','R_23','D_93','D_94','R_24','S_22
           'D_113','D_116','D_117','D_119','D_120','D_121','D_122','D_123','D_124','D_
           'D_139','D_140','D_144','target']].corr(method='pearson')
sns.heatmap(corr_D,square=True)
```

Out[39]: <AxesSubplot:>



In [40]:
```
#Variables que contribuyen al modelo y su correlación con la variable objetivo
upper_corr_D=corr_D.where(np.triu(np.ones(corr_D.shape),k=1).astype(bool))
unique_corr_D=upper_corr_D.unstack().dropna()
sorted_corr_D=unique_corr_D.sort_values()
#sorted_corr_B=corr_B.unstack().sort_values()
sorted_corr_D
a=pd.DataFrame(sorted_corr_D)
a.reset_index(inplace=True)
#(a[a["level_0"]=='target'].sort_values(by=0,ascending=False)).where(np.abs(a[0])>.3)
b=a[a["level_0"]=='target'].sort_values(by=0,ascending=False)
b[np.abs(b[0])>.2]
```

Out[40]:

	level_0	level_1	0
2203	target	D_48	0.527277
2200	target	D_75	0.446553
2198	target	B_7	0.419232
2195	target	B_1	0.378933
2184	target	B_30	0.337646
2182	target	S_7	0.335293
2177	target	B_8	0.322482
2168	target	D_43	0.259537
2161	target	D_41	0.240186
2157	target	R_10	0.230870
2153	target	D_131	0.226012
2147	target	R_4	0.218199
29	target	D_52	-0.226765
22	target	S_25	-0.243644
19	target	D_51	-0.249963
14	target	R_27	-0.269047
6	target	B_33	-0.440924

Comentario

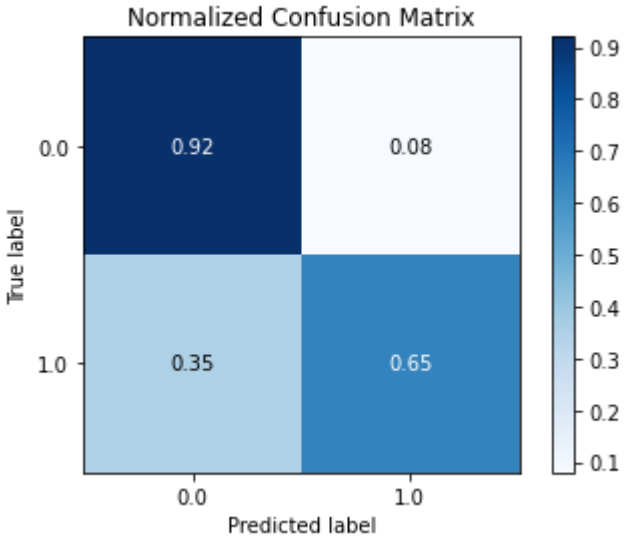
El porcentaje de correlación de las variables de entrada que más aportan al modelo con la variable objetivo van del 30% al 52%

In [41]:
```
# Visualizamos las métricas de desempeño de nuestro modelo con el conjunto de validaci
X_test = sm.add_constant(X_test, prepend=True)
predict_test_y_log = modelo.predict(exog = X_test)
clasificacion = np.where(predict_test_y_log<0.5, 0, 1)
print('Valor del Accuracy en el conjunto de Test es:',accuracy_score(Y_test, clasifica
print("\n")
print(classification_report(Y_test, clasificacion))
```

```
Valor del Accuracy en el conjunto de Test es: 0.8549413054039702


              precision    recall  f1-score   support

         0.0       0.89      0.92      0.90     65840
         1.0       0.75      0.65      0.70     22669

    accuracy                           0.85     88509
   macro avg       0.82      0.79      0.80     88509
weighted avg       0.85      0.85      0.85     88509
```

In [42]:
```
# Visualizamos la Matriz de Confusión
import scikitplot as skplt
skplt.metrics.plot_confusion_matrix(Y_test, clasificacion, normalize=True)
```

Out[42]:
```
<AxesSubplot:title={'center':'Normalized Confusion Matrix'}, xlabel='Predicted labe
l', ylabel='True label'>
```



Comentario

El resultado obtenido de la regresión logística nos indica que atrapa bien los casos que no incumplirán a futuro, sin embargo, se equivoca más en predecir los casos que incumplirán (captura apenas un 65% del total de casos), es decir, con este modelo tenemos más falsos negativos, que en la práctica, y por el caso de uso de clientes que caeran en impago se busca que el modelo tenga precisión (la mayor cantidad de clientes predichos sean verdaderos positivos) y que a su ves capture el patrón lo más posible (que prediga el mayor número posible de verdaderos positivos) para así minimizar las pérdidas monetarias.

** Modelo 2: Árbol de Decisión **

En esta sección probabremos el algoritmo de árbol de decisión, veremos si un algoritmo de este estilo puede darnos mejores resultados que clásica regresión lógistica.

In [44]:
```
# Importamos la Librería de árboles de decisión
from sklearn.tree import DecisionTreeClassifier
```

In [45]:
```
# Construimos un árbol de decisión simple únicamente fijando la semilla
DT = DecisionTreeClassifier(random_state=0)
# Entrenamos los datos
DT.fit(X_train, Y_train)
```

Out[45]:
```
DecisionTreeClassifier(random_state=0)
```

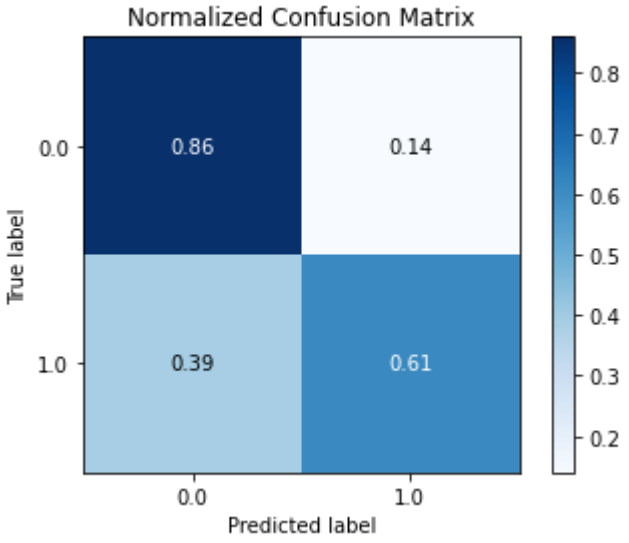
In [46]:
```
#Realizamos predicciones
predic_y_dc = DT.predict(X_test)
```

```
print('Accuracy on Validation set :',accuracy_score(Y_test, predic_y_dc))
print("\n")
# Visualizamos métricas de desempeño
print(classification_report(Y_test, predic_y_dc))
```

```
Accuracy on Validation set : 0.7966195528138381


              precision    recall  f1-score   support

         0.0       0.86      0.86      0.86     65840
         1.0       0.60      0.61      0.60     22669

    accuracy                           0.80     88509
   macro avg       0.73      0.73      0.73     88509
weighted avg       0.80      0.80      0.80     88509
```

In [47]:
```
# Visualizamos la Matriz de Confusión
skplt.metrics.plot_confusion_matrix(Y_test, predic_y_dc, normalize=True)
```

Out[47]:
```
<AxesSubplot:title={'center':'Normalized Confusion Matrix'}, xlabel='Predicted labe
l', ylabel='True label'>
```



Observación

El modelo de Árbol de Decisión presenta ligeramente un menor desempeño que la Regresión Logística, es decir atrapa solo el 61% de los verdaderos positivos, es decir de cada 10 clientes impagadores el modelo no detecta 4.

** Modelo 3: Random Forest **

Debido a que el árbol de decisión simple no arrojo muy buenos resultados probaremos con un modelo ensamblado de árboles de decisión simples.

In [48]:
```
# Importamos librería de modelos ensamblados
from sklearn import ensemble
```

```
# Creamos el bosque aleatorio simple (Random Forest).
modeloRF = ensemble.RandomForestClassifier(n_estimators = 100, max_features = 'sqrt')
```

In [49]:
```
# Entrenamos el modelo con los datos train
modeloRF.fit(X_train, Y_train)
```

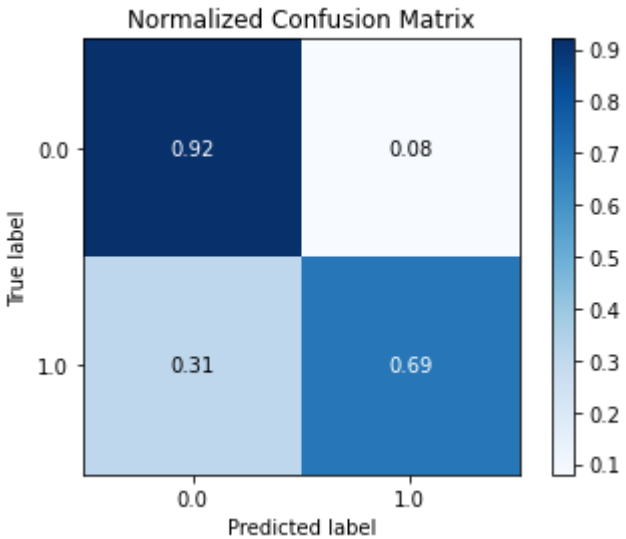
Out[49]:
```
RandomForestClassifier(max_features='sqrt')
```

In [50]:
```
#Realizamos predicciones
predic_y_rf = modeloRF.predict(X_test)
print('Accuracy on Validation set :',accuracy_score(Y_test, predic_y_rf))
print("\n")
# Visualizamos métricas de desempeño
print(classification_report(Y_test, predic_y_rf))
```

```
Accuracy on Validation set : 0.8587940209470223


              precision    recall  f1-score   support

         0.0       0.90      0.92      0.91     65840
         1.0       0.74      0.69      0.72     22669

    accuracy                           0.86     88509
   macro avg       0.82      0.80      0.81     88509
weighted avg       0.86      0.86      0.86     88509
```

In [51]:
```
# Visualizamos la Matriz de Confusión
skplt.metrics.plot_confusion_matrix(Y_test, predic_y_rf, normalize=True)
```

Out[51]:
```
<AxesSubplot:title={'center':'Normalized Confusion Matrix'}, xlabel='Predicted labe
l', ylabel='True label'>
```



Observación

El modelo de Random Forest logra una mejora de 4% en la precisión con respecto a la Regresión Logística, ademas de que solo 26 de cada 100 clientes propensos resultan ser falsos positivos.

Approach 2

A continuación se hará algunos cambios a la preparación de los datos y se validará si se obtiene un mejor desempeño en los modelos propuestos con esta nueva estrategia

In [52]:
```
# Cargamos la Base nuevamente
df2=pd.read_pickle('df_sample_amex.pkl')
df2.head(2)
```

Out[52]:

	customer_ID	S_2	P_2	D_39	B_1	
5230464	f1e881682c6eac031984d20d3620b30514dcdf3ec9ce0d...	2018-01-12	0.875387	0.000703	0.010017	0.81
35131	01a4df870c9a606a9463f6daa20fcfe6e9fd1d089a3369...	2017-12-08	0.825674	0.008442	0.048399	0.81

2 rows × 191 columns

In [53]:
```
#Eliminamos la Fecha de transacción ya que no nos es util, así como las variables con
col_drop2 = ['S_2','customer_ID','D_42','D_49','D_50','D_53','D_56','S_9','B_17','D_66
             'B_29','D_87','D_88','D_105','D_106','R_26','D_108','D_110','D_111','B_39'
             'D_136','D_137','D_138','D_142','D_63','D_64','B_31','D_43','D_46','D_48',
df2.drop(col_drop2, inplace=True, axis=1)
```

In [54]:
```
# Visualizamos la dimensión del dataframe
df2.shape
```

Out[54]:
```
(442542, 147)
```

Missing Values: Extensión de la imputación

Dado que ya eliminamos variables que tenían un porcentaje significativo de valores perdidos, el approach que seguiremos ahora será eliminar los registros con valores pérdidos a diferencia del primer approah en donde se decidió imputar valores con la mediana.

In [55]:
```
# Obtenemos el % total de valores perdidos que aún quedan en la base
df2.isnull().values.mean() * 100
```

Out[55]:
```
1.1322404327232924
```

In [56]:
```
# Copiamos el dataframe
df2_plus = df2.copy()
# Obtenemos los registros con valores perdidos
cols_with_missing = [col for col in df2.columns
                     if df2[col].isnull().any()]

# Hacemos una nueva columna la cual indica que sería imputada
for col in cols_with_missing:
    df2[col + '_was_missing'] = df2[col].isnull()
```

```
# Imputacion
my_imputer = SimpleImputer()
imputed_df2_plus = pd.DataFrame(my_imputer.fit_transform(df2_plus))
imputed_df2_plus.columns = df2_plus.columns
```

In [57]:
```
# Visualizamos nuevamente la dimensión del dataframe
imputed_df2_plus.shape
```

Out[57]:
```
(442542, 147)
```

In [58]:
```
# Verificamos que no existan valores perdidos
imputed_df2_plus.isnull().values.mean() * 100
```

Out[58]:
```
0.0
```

Aplicando ingeniería de variables

Para elegir un conjunto más pequeño de las variables más útiles y para desarrollar inicialmente la métrica que usaremos, la cual se llama "información mutua".

La información mutua se parece mucho a la correlación en el sentido de que mide una relación entre dos cantidades. La ventaja de la información mutua es que puede detectar cualquier tipo de relación, mientras que la correlación solo detecta relaciones lineales.

In [59]:
```
# Partimos el conjunto en Inputs y Target
X = imputed_df2_plus.copy()
y = imputed_df2_plus.pop("target")

#Factorizamos la variables de tipo objeto
for colname in X.select_dtypes("object"):
    X[colname], _ = X[colname].factorize()
# Alas caracteristicas discretas asignamos un tipo int ya que será un proceso de one h
discrete_features = X.dtypes == int
```

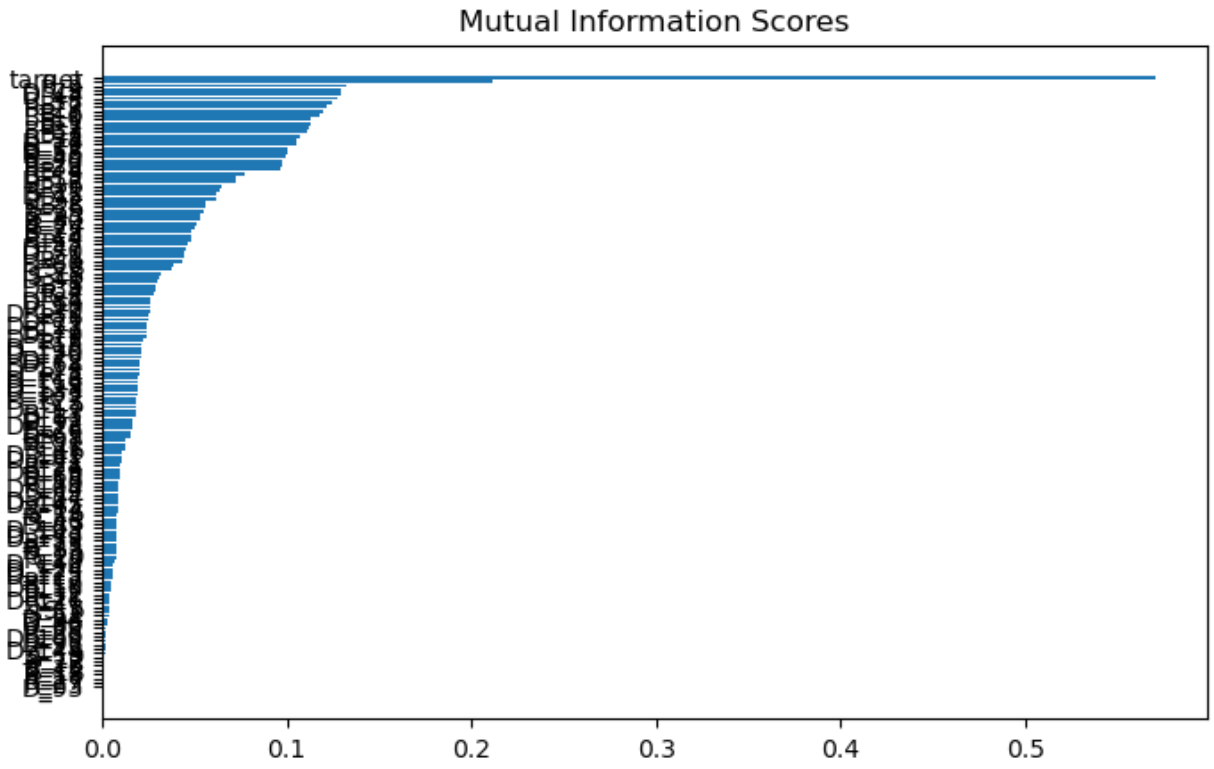
In [60]:
```
# Definimos una función que nos obtenga la relación mutua ente las variables
from sklearn.feature_selection import mutual_info_regression

def make_mi_scores(X, y, discrete_features):
    mi_scores = mutual_info_regression(X, y, discrete_features=discrete_features)
    mi_scores = pd.Series(mi_scores, name="MI Scores", index=X.columns)
    mi_scores = mi_scores.sort_values(ascending=False)
    return mi_scores

mi_scores = make_mi_scores(X, y, discrete_features)
mi_scores[::3]
```

Out[60]:
```
target    0.570220
D_75      0.129251
B_7       0.124740
B_10      0.117618
B_37      0.111743
D_74      0.105522
D_55      0.099991
B_33      0.097876
D_52      0.072550
S_22      0.063555
B_8       0.055661
S_23      0.052714
R_27      0.049646
P_3       0.047911
D_51      0.044635
S_8       0.038058
D_41      0.030614
P_4       0.028546
D_59      0.025749
R_6       0.025474
B_12      0.023988
R_5       0.021652
D_120     0.021085
D_114     0.020169
D_118     0.019482
D_133     0.019178
S_5       0.018657
D_81      0.018263
B_26      0.016120
D_91      0.012907
D_102     0.010754
R_24      0.009754
R_16      0.009101
S_11      0.008644
D_145     0.008374
D_54      0.008111
D_103     0.007947
R_13      0.007869
S_20      0.007270
D_124     0.006160
R_17      0.005920
B_32      0.004514
B_41      0.003689
D_94      0.003380
R_25      0.002154
D_96      0.001723
S_19      0.001508
B_15      0.000726
B_27      0.000000
Name: MI Scores, dtype: float64
```

Comentario

Graficando los cores obtenidos a través de Información mutua se observan las principales variables que aportan a la variable target, tomando como input el resultado se definen las variables a participar en los modelos posteriores.

In [83]:
```
# Definimos una función para graficar los scores anteriores
def plot_mi_scores(scores):
    scores = scores.sort_values(ascending=True)
    width = np.arange(len(scores))
    ticks = list(scores.index)
    plt.barh(width, scores)
    plt.yticks(width, ticks)
    plt.title("Mutual Information Scores")

# Gráficamos los scores anteriores por variables
plt.figure(dpi=100, figsize=(8, 5))
plot_mi_scores(mi_scores)
```



In [62]:
```
# Guardamos el dataset de score en formato .pkl para optimizar espacio
mi_scores.to_pickle("mi_scores.pkl")
```

In [63]:
```
# Guardamos el dataset de valores imputados en formato .pkl para optimizar espacio
imputed_df2_plus.to_pickle("imputed_df2_plus.pkl")
```

In [64]:
```
# Imprimimos el número de variables
len(mi_scores)
```

Out[64]:
```
147
```

In [65]:
```
# Guardamos el excel los scores para dar un vistazo posterior
mi_scores.to_excel("mi_scores.xlsx")
```

In [66]:
```
#Filtrando un conjunto de variables que más aportan al modelo a través de Información
X_2 =X[['P_2','B_9','D_44','D_75','B_18','B_7','B_23','B_6','B_10','B_1','B_2','B_37',
'B_40','B_33','B_20','B_4','B_19','D_52','B_16','R_1','S_22','D_45','B_22','B_8','S_25
'B_28','B_14','P_3','D_47','D_70','D_51','R_3','D_60','D_78','S_8','S_15','D_41','B_5'
'D_59','R_6','D_71','D_127','D_79','B_12','D_112','R_5','D_72','D_115','D_114','D_130'
```

```
'D_121','D_118','D_133','D_119','R_7','S_5','B_13','D_81','D_122','D_92','D_113','B_26
'D_102','D_141','R_21']]
```

In [67]:
```
# Nos quedamos con 92 variables que son las que más aportan de acuerdo a su índice de
print(X_2.shape)
print(y.shape)
```

```
(442542, 92)
(442542,)
```

** Modelación Parte 2**

En esta parte vamos correr nuevamente los modelos que hemos aplicado en el apartado de Modelación Parte 1, esta vez con un conjunto de datos que tiene un tratamiento diferente y ahora con 92 variables en vez de 85.

** Modelo 1: Regresión Logística **

El primer modelo a probar es una regresión Logística, la cual es el algoritmo más común para temas de riesgo de crédito.

In [68]:
```
# Declaramos librerías a utilizar
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,accuracy_score
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
```

In [69]:
```
#Estandarización de variables
scaler = StandardScaler()
X2 = pd.DataFrame(scaler.fit_transform(X_2), index=X_2.index, columns=X_2.columns)

#Probando un 80 20 en las bases de train y test para el modelo
X_train, X_test, Y_train, Y_test = train_test_split(X_2, y, random_state = 1, shuffle=

# Creamos un Primer modelo utilizando matrices como en scikitlearn y Visualizamos el r
# ==============================================================================
# A la matriz de predictores se le tiene que añadir una columna de 1s para el intercep
X_train = sm.add_constant(X_train, prepend=True)
modelo = sm.Logit(endog=Y_train, exog=X_train,)
modelo = modelo.fit()
print(modelo.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.296673
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:                 target   No. Observations:               354033
Model:                          Logit   Df Residuals:                   353940
Method:                           MLE   Df Model:                           92
Date:                Thu, 01 Dec 2022   Pseudo R-squ.:                 0.4786
Time:                        13:22:45   Log-Likelihood:            -1.0503e+05
converged:                       True   LL-Null:                   -2.0143e+05
Covariance Type:            nonrobust   LLR p-value:                    0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.4562      0.077      5.927      0.000       0.305       0.607
P_2           -3.8360      0.045    -85.013      0.000      -3.924      -3.748
B_9            0.1222      0.025      4.938      0.000       0.074       0.171
D_44           0.5825      0.042     13.977      0.000       0.501       0.664
D_75          -0.5336      0.274     -1.947      0.052      -1.071       0.004
B_18          -0.8710      0.054    -16.273      0.000      -0.976      -0.766
B_7            3.2278      0.310     10.401      0.000       2.620       3.836
B_23          -3.0073      0.324     -9.288      0.000      -3.642      -2.373
B_6           -0.0569      0.019     -3.049      0.002      -0.094      -0.020
B_10          -0.0089      0.005     -1.700      0.089      -0.019       0.001
B_1            4.1207      0.401     10.284      0.000       3.335       4.906
B_2           -0.1034      0.042     -2.438      0.015      -0.186      -0.020
B_37          -1.4001      0.188     -7.458      0.000      -1.768      -1.032
B_3            0.7907      0.041     19.308      0.000       0.710       0.871
B_38          -0.0263      0.005     -5.136      0.000      -0.036      -0.016
D_74           0.2913      0.261      1.117      0.264      -0.220       0.802
B_11          -1.9039      0.361     -5.277      0.000      -2.611      -1.197
D_58           0.2886      0.066      4.381      0.000       0.160       0.418
D_55           0.1311      0.035      3.794      0.000       0.063       0.199
B_40       -1.629e-05      0.000     -0.041      0.967      -0.001       0.001
B_33           0.1520      0.037      4.145      0.000       0.080       0.224
B_20          -0.7014      0.033    -21.276      0.000      -0.766      -0.637
B_4            1.5500      0.045     34.584      0.000       1.462       1.638
B_19          -0.0453      0.032     -1.400      0.162      -0.109       0.018
D_52          -0.4215      0.039    -10.905      0.000      -0.497      -0.346
B_16           0.4403      0.035     12.663      0.000       0.372       0.508
R_1            0.5721      0.046     12.569      0.000       0.483       0.661
S_22          -0.0502      0.026     -1.950      0.051      -0.101       0.000
D_45          -0.0359      0.050     -0.712      0.476      -0.135       0.063
B_22           0.3065      0.042      7.323      0.000       0.224       0.389
B_8            0.4108      0.020     20.714      0.000       0.372       0.450
S_25          -0.0042      0.023     -0.181      0.857      -0.050       0.042
B_25          -0.0406      0.037     -1.099      0.272      -0.113       0.032
B_30          -0.0965      0.021     -4.664      0.000      -0.137      -0.056
S_23           0.0903      0.019      4.787      0.000       0.053       0.127
S_24           0.0732      0.022      3.356      0.001       0.030       0.116
R_27          -0.4316      0.016    -26.599      0.000      -0.463      -0.400
B_28          -0.7405      0.065    -11.449      0.000      -0.867      -0.614
B_14           0.6238      0.054     11.581      0.000       0.518       0.729
P_3            0.3757      0.037     10.027      0.000       0.302       0.449
D_47          -0.9884      0.041    -24.273      0.000      -1.068      -0.909
D_70           0.3967      0.029     13.559      0.000       0.339       0.454
D_51          -1.4562      0.057    -25.435      0.000      -1.568      -1.344
R_3            1.2982      0.028     46.704      0.000       1.244       1.353
D_60           0.3373      0.026     12.924      0.000       0.286       0.388
```

```
D_78          0.0088      0.027      0.329      0.742      -0.044       0.061
S_8          -0.1894      0.036     -5.192      0.000      -0.261      -0.118
S_15          0.7392      0.043     17.216      0.000       0.655       0.823
D_41          0.4970      0.040     12.515      0.000       0.419       0.575
B_5          -0.3273      0.037     -8.862      0.000      -0.400      -0.255
D_39          0.4638      0.024     19.611      0.000       0.417       0.510
P_4           0.4778      0.019     25.362      0.000       0.441       0.515
R_2          -0.0212      0.044     -0.481      0.630      -0.108       0.065
D_84         -0.1652      0.032     -5.088      0.000      -0.229      -0.102
D_131         0.5871      0.040     14.659      0.000       0.509       0.666
R_10         -0.0783      0.025     -3.194      0.001      -0.126      -0.030
D_59          0.0186      0.032      0.579      0.563      -0.044       0.081
R_6           0.0087      0.010      0.865      0.387      -0.011       0.029
D_71         -0.0495      0.048     -1.032      0.302      -0.144       0.045
D_127        -0.4784      0.046    -10.389      0.000      -0.569      -0.388
D_79          0.2421      0.045      5.381      0.000       0.154       0.330
B_12         -0.1496      0.039     -3.859      0.000      -0.226      -0.074
D_112        -0.3804      0.019    -20.242      0.000      -0.417      -0.344
R_5          -0.0623      0.052     -1.197      0.231      -0.164       0.040
D_72         -0.0111      0.034     -0.329      0.742      -0.077       0.055
D_115         0.1046      0.048      2.183      0.029       0.011       0.198
D_114        -0.1181      0.015     -7.823      0.000      -0.148      -0.088
D_130         0.0836      0.018      4.646      0.000       0.048       0.119
D_120         0.0117      0.016      0.717      0.474      -0.020       0.044
D_68         -0.0214      0.006     -3.364      0.001      -0.034      -0.009
R_4           0.1504      0.066      2.268      0.023       0.020       0.280
D_117        -0.0248      0.003     -8.696      0.000      -0.030      -0.019
D_128        -0.2198      0.015    -14.930      0.000      -0.249      -0.191
D_121         1.0040      0.037     27.506      0.000       0.932       1.076
D_118        -0.5047      0.250     -2.022      0.043      -0.994      -0.016
D_133        -0.7550      0.039    -19.359      0.000      -0.831      -0.679
D_119         0.1039      0.249      0.418      0.676      -0.384       0.592
R_7           0.0034      0.003      1.079      0.280      -0.003       0.010
S_5           0.1385      0.018      7.753      0.000       0.104       0.174
B_13          0.0845      0.042      2.023      0.043       0.003       0.166
D_81         -0.0273      0.021     -1.274      0.203      -0.069       0.015
D_122        -0.1970      0.033     -5.968      0.000      -0.262      -0.132
D_92          0.2184      0.045      4.844      0.000       0.130       0.307
D_113        -0.0018      0.034     -0.053      0.957      -0.068       0.065
B_26          0.0066      0.003      2.311      0.021       0.001       0.012
D_65          0.0351      0.017      2.099      0.036       0.002       0.068
R_8           0.1641      0.042      3.882      0.000       0.081       0.247
D_91          0.3035      0.062      4.895      0.000       0.182       0.425
B_21          0.0370      0.011      3.313      0.001       0.015       0.059
S_16          0.0197      0.009      2.130      0.033       0.002       0.038
D_102        -0.0293      0.022     -1.351      0.177      -0.072       0.013
D_141        -0.1181      0.015     -7.862      0.000      -0.148      -0.089
R_21          0.4466      0.052      8.600      0.000       0.345       0.548
==============================================================================
```

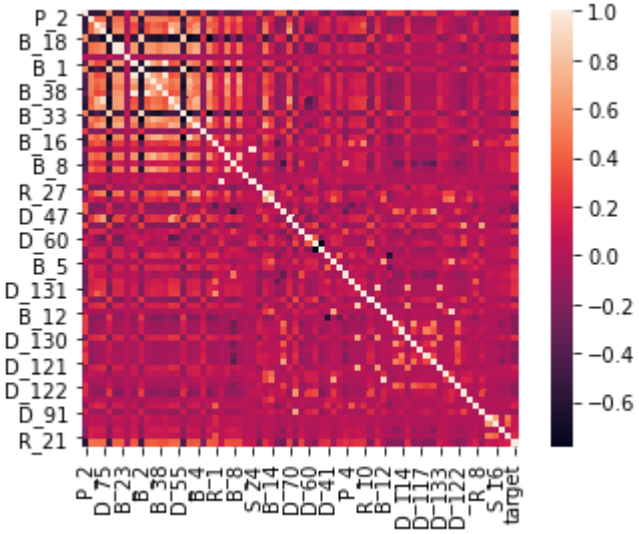
Comentario

A continuación filtramos las variables cuyo p-value resultó aceptable en la regresión y las observaremos gráficamente.

In [70]:
```
#Correlación variables de que aportan al modelo
corr_D=X[['P_2','B_9','D_44','D_75','B_18','B_7','B_23','B_6','B_1','B_2','B_37','B_3'
```

```
'B_20','B_4','D_52','B_16','R_1','S_22','B_22','B_8','B_30','S_23','S_24','R_27','B_28
'R_3','D_60','S_8','S_15','D_41','B_5','D_39','P_4','D_84','D_131','R_10','D_127','D_7
'D_130','D_68','D_117','D_128','D_121','D_133','S_5','B_13','D_122','D_92','B_26','R_8
'target']].corr(method='pearson')
sns.heatmap(corr_D,square=True)
```

Out[70]:
```
<AxesSubplot:>
```

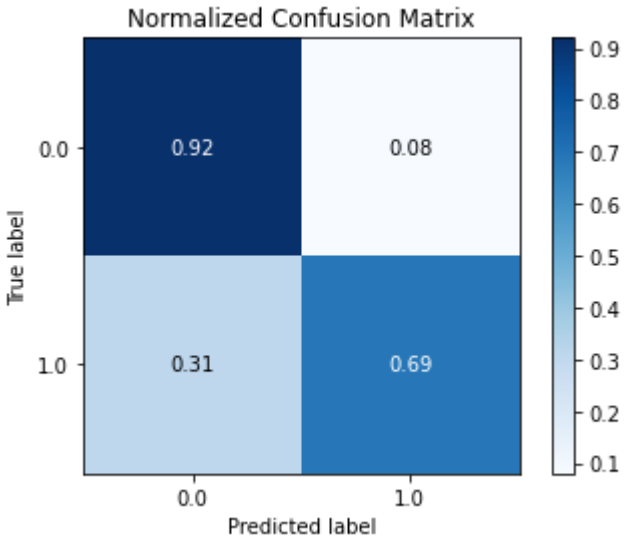


In [71]:
```
# Predecimos sobre el conjunto de test y visualizamos métricas de desempeño
X_test = sm.add_constant(X_test, prepend=True)
predict_test_y_log = modelo.predict(exog = X_test)
clasificacion = np.where(predict_test_y_log<0.5, 0, 1)
print('Valor del Accuracy en el conjunto de Test es:',accuracy_score(Y_test, clasifica
print("\n")
print(classification_report(Y_test, clasificacion))
```

```
Valor del Accuracy en el conjunto de Test es: 0.8614039250245737


              precision    recall  f1-score   support

         0.0       0.90      0.92      0.91     65840
         1.0       0.75      0.69      0.72     22669

    accuracy                           0.86     88509
   macro avg       0.82      0.80      0.81     88509
weighted avg       0.86      0.86      0.86     88509
```

In [72]:
```
# Visualizamos la Matriz de Confusión
#!pip install scikit-plot
skplt.metrics.plot_confusion_matrix(Y_test, clasificacion, normalize=True)
```

Out[72]:
```
<AxesSubplot:title={'center':'Normalized Confusion Matrix'}, xlabel='Predicted labe
l', ylabel='True label'>
```



Observación

Aplicando ingeniería de variables, en este caso Información Mutua el modelo de regresión mejora y atrapa mejor a los casos que caeran en impago en el futuro, en este caso igualando al Modelo Random Forest en captura del patrón que corresponde con la métodología anterior del datatset.

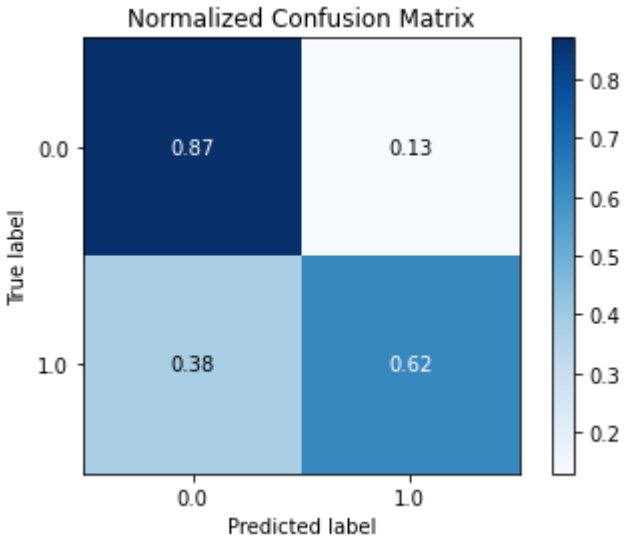
** Modelo 2: Árbol de decisión **

En esta sección probabremos el algoritmo de árbol de decisión, veremos si un algoritmo de este estilo puede darnos mejores resultados que clásica regresión lógistica.

In [73]:
```
# construimos nuevamente el modelo de arbol de decisión simple
DT = DecisionTreeClassifier(random_state=0)
# Entrenamos el Modelo
DT.fit(X_train, Y_train)
#Realizamos predicciones
predic_y_dc = DT.predict(X_test)
# Imprimimos métricas de desempeño
print('Accuracy on Validation set :',accuracy_score(Y_test, predic_y_dc))
print("\n")
print(classification_report(Y_test, predic_y_dc))
# Visualizamos la Matriz de Confusión
skplt.metrics.plot_confusion_matrix(Y_test, predic_y_dc, normalize=True)
```

```
Accuracy on Validation set : 0.8048673016303427


              precision    recall  f1-score   support

         0.0       0.87      0.87      0.87     65840
         1.0       0.62      0.62      0.62     22669

    accuracy                           0.80     88509
   macro avg       0.74      0.74      0.74     88509
weighted avg       0.80      0.80      0.80     88509
```

Out[73]:
```
<AxesSubplot:title={'center':'Normalized Confusion Matrix'}, xlabel='Predicted labe
l', ylabel='True label'>
```



Observación

También observaciomos una ligera mejora (de 1% en la precisión) en el árbol de decisión aplicando ingeniería de variables en el conjunto de datos.

** Modelo 3: Random Forest **

Nuevamente probaremos con un modelo ensamblado de árboles de decisión simples.

In [74]:
```
# Creamos el bosque aleatorio simple (Random Forest).
modeloRF = ensemble.RandomForestClassifier(n_estimators = 100, max_features = 'sqrt')
```

In [75]:
```
# Entrenamos el modelo con los datos train
modeloRF.fit(X_train, Y_train)
```

Out[75]:
```
RandomForestClassifier(max_features='sqrt')
```

In [76]:
```
#Realizamos predicciones
predic_y_rf_1 = modeloRF.predict(X_test)
print('Accuracy on Validation set :',accuracy_score(Y_test, predic_y_rf_1))
print("\n")
```

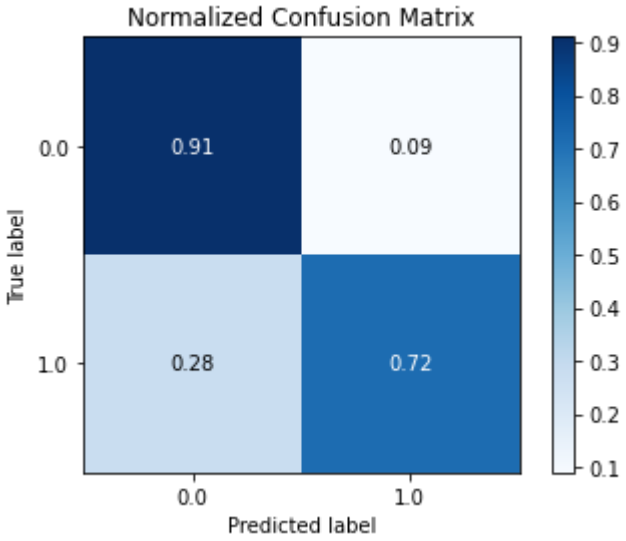
```
# Visualizamos métricas de desempeño
print(classification_report(Y_test, predic_y_rf_1))
```

```
Accuracy on Validation set : 0.8627936142087246
```

```
              precision    recall  f1-score   support

         0.0       0.90      0.91      0.91     65840
         1.0       0.74      0.72      0.73     22669

    accuracy                           0.86     88509
   macro avg       0.82      0.81      0.82     88509
weighted avg       0.86      0.86      0.86     88509
```

In [78]:
```
# Visualizamos la Matriz de Confusión
skplt.metrics.plot_confusion_matrix(Y_test, predic_y_rf_1, normalize=True)
```

Out[78]:
```
<AxesSubplot:title={'center':'Normalized Confusion Matrix'}, xlabel='Predicted label', ylabel='True label'>
```



Observación

El modelo de Random Forest logra una mejora de 3 % en la captura del patrón con respecto al Random Forest con en elfoque anterior, ademas de que solo 26 de cada 100 clientes propensos resultan ser falsos positivos.

Conclusiones Finales

Como se puede apreciar dentro del trabajo que se realizó se enfrentaron diversas problemáticas una de ellas es trabajar con grandes volúmenes de información, debido a que el trabajar con información en ordenadores con muy poca capacidad de memoria se tuvo que trabajar con una muestra del 20%.

Cabe destacar que mediante el modelo de información mutua en comparación con el modelo de correlación se observó que las variables más significativas para el modelo de correlación

fueron :

- B_10,B_12,D_69,B_25,B_26,B_27,R_13,R_17,R_18,S_19,R_19,B_32,D_96,S_26,D_104,D_109,B_40,D_1
y para el modelo de información mutua fueron las siguientes.
- 'P_2','B_9','D_44','D_75','B_18','B_7','B_23','B_6','B_1','B_2','B_37','B_3','B_38','B_11','D_58',
- 'D_55','B_33','B_20','B_4','D_52','B_16','R_1','S_22','B_22','B_8','B_30','S_23','S_24','R_27','B_28',
- 'B_14','P_3','D_47','D_70','D_51','R_3','D_60','S_8','S_15','D_41','B_5','D_39','P_4','D_84','D_131',
- 'R_10','D_127','D_79','B_12','D_112','D_115','D_114','D_130','D_68','D_117',
- 'D_128','D_121','D_133','S_5','B_13','D_122','D_92','B_26','R_8','D_91','B_21','S_16','D_141','R_21'

Esto se debe a que mediante el modelo de información mutua analiza no solo las relaciones lineales como es el caso del modelo de correlación esto nos da más seguridad al momento de seleccionar las variables feature para predecir valores futuros, como fue el caso de nuestro trabajo.

De los tres tipos de modelos que se proponen obtuvimos los mejores resultados en todos aplicando Ingeniería de variables. Nuestros Resultados son:

1. Regresión Logística: Captura el 69% del patrón (de cada 100 clientes que caerán en impago 69 son predichos por el modelo y 31 se le escapan) y tiene una precisión de 75% (de cada 100 clientes que el modelo dice que incumplirán, 75 resultan ser positivos reales).

2. Árbol de Decisión: Captura el 62% del patrón (de cada 100 clientes que caerán en impago 69 son predichos por el modelo y 31 se le escapan) y tiene una precisión de 62% (de cada 100 clientes que el modelo dice que incumplirán, 62 resultan ser positivos reales).

3. Random Forest: Captura el 74% del patrón (de cada 100 clientes que caerán en impago 69 son predichos por el modelo y 31 se le escapan) y tiene una precisión de 73% (de cada 100 clientes que el modelo dice que incumplirán, 73 resultan ser positivos reales).

De lo anterior podríamos concluir que el mejor modelo es el que utiliza el Random Forest, con una diferencia de 1% en la captura del patrón con respecto a la Regresión Logística. Sin embargo dado que la diferencia es muy poca nosotros optaríamos por seleccionar a la regresión logística como el modelo campeón, esto debido a que es un algoritmo que puede explicar mejor la influencia de las variables utilizadas en el caso de caer en impago, lo cual resulta ser de mayor utilidad para el planteamiento de estrategias que minimicen este riesgo.

Sugerencias para mejoras

Dentro de este trabajo hemos incluido modelos muy simples obteniendo resultados muy aceptables para su uso, sin embargo reconocemos que se puede llegar a mejorar bastante probando otro tipo de procesamiento a las variables, probar con distintos hiperparámetos en los algoritmos e incluso probar con otros de mayor complejidad; estamos seguros de que se obtendrían mejoras tanto en la precisión como en la captura del patrón que es el objetivo principal dentro de este caso de negocio.

Liga donde se puede consultar los datos y el trabajo realizado

- https://github.com/pac05an/Equipo_S_modelos_avanzados_infotec

Referencias

1. https://www.kaggle.com/competitions/amex-default-prediction/
2. Gunter Löffler & Peter N. Posch, Credit Risk Modeling Using Excel and VBA, 2011 John Wiley & Sons.
3. Comisión Nacional Bancaria y de Valores CNBV,https://www.cnbv.gob.mx/Normatividad/Disposiciones%20de%20car%C3%A1cter%20ge , junio de 2022
4. BBVA 2010,probabilidades de incumlimiento,https://accionistaseinversores.bbva.com/microsites/informes2009/es/Gestiondel
5. Banco de México, https://www.banxico.org.mx/publicaciones-y-prensa/reportes-sobre-el-sistema-financiero/recuadros/%7BE165B93E-88FF-4195-B476-0FF711A6BC78%7D.pdf

In [ ]: