Jonathan Daniel García Herrera
Diana Paola Lagunes Blanco
Arturo Ricardo Villafuerte Vega

Introducción

El sector bancario, en su línea de negocios de retail banking, es quizá una de las industrias más antiguas en el sector financiero y quizá una de las más reguladas por las autoridades del mismo sector. Durante muchos años, décadas tal vez, el sector ha estado controlado por casi los mismos actores, enfocados en atender a casi los mismos sectores poblaciones.

La banca minorista o banca de consumo es la línea de negocio encargada de atender con servicios financieros al público que no son empresas ni instituciones. Los servicios de crédito principales ofrecidos son créditos personales, créditos de nómina, tarjetas de crédito, créditos hipotecarios y créditos para el financiamiento de un auto. De acuerdo al tipo de producto se pueden definir dos segmentaciones importantes: secured lending y unsecured lending.

Los créditos garantizados o secured lending son aquellos en los que se tiene una garantía de pago de por medio, ya sea un auto o una casa habitación, es decir, al adquirir un crédito de este tipo, la propiedad (casa) queda hipotecada a favor del banco. Y en el caso de un auto, la factura de este quedará endosada a favor el banco. Por el contrario, en los créditos unsecured no hay garantía de por medio más que la promesa de pago del solicitante.

Es claro que cada uno de estos productos atrae a diferentes perfiles de clientes, y al realizar la solicitud para adquirir algún producto, el banco o institución deberá evaluar si el solicitante puede o no ser sujeto de crédito. De acuerdo con la naturaleza de cada préstamo las instituciones financieras definen los límites de riesgo de incumplimiento de pago que están dispuestos a asumir. Entiéndase que "los límites de riesgo" se traducen (de manera muy general) en diversos indicadores de mora, por ejemplo, puede ser de interés medir y controlar el porcentaje de cuentas y saldo que incurren en 30 días o más de mora durante los primeros 3 meses de vida de un crédito; a este indicador se le conoce como 30+@MOB3 y se calcula como

$$\frac{\sum_i Bal_{i_{MOB_0}}}{\sum_k Bal^{PV2}_{k_{MOB_3}}}$$

Donde:

- $BAL_{i_{MOB_0}}$ es el balance / saldo del i-ésimo crédito originado en el mes 0 y
- $Bal^{PV2}_{k_{MOB_3}}$ es el balance del k-ésimo crédito con 30 o más días de mora al cierre del mes 3

Tomando este indicador como ejemplo, una pregunta natural sería: ¿qué variables describen el comportamiento de este indicador?, de tal forma que durante la evaluación de un cliente se pueda inferir en qué niveles de morosidad se encontrará 3 meses después de su formalización.

Para controlar los niveles de riesgo que la institución tolerará, tradicionalmente (en particular la banca) se desarrollan modelos matemáticos con el fin de predecir si un cliente incurrirá en impago. Estos modelos se han desarrollado por mucho tiempo con variables que empiezan a ser muy tradicionales, además de que los propios modelos ya no parecen ser tan robustos frente a las técnicas de *machine learning* que las nuevas tecnologías y lenguajes de programación ponen al alcance. Esto, combinado con que los bancos han mantenido una baja tolerancia al riesgo, ha permitido el buen posicionamiento de los llamados neobancos y fintechs, que han visto una gran oportunidad de negocio en los segmentos desatendidos por la banca tradicional y que han aprovechado el "boom" del desarrollo tecnológico en el área de analítica de datos.

Regresando al negocio de banca minorista de las instituciones más tradicionales, es claro que se vuelve necesario introducir el uso de nuevas y diferentes técnicas de modelado. Los bancos lo tienen absolutamente todo para disrumpir nuevamente en su industria: datos en grandes volúmenes, tecnología, presupuesto y experiencia. Sin embargo también hay grandes limitantes: procesos burocráticos y la alta regulación del sector.

El riesgo de crédito se vuelve entonces el riesgo que asume una institución financiera al realizar un préstamo. La gestión del riesgo de crédito es el desarrollo de estrategias de admisión de crédito basadas en análisis de datos históricos sobre el comportamiento de los clientes respecto a su deuda, de tal forma que sea posible segmentar a los buenos clientes (pagadores) de los malos (que pueden incurrir en impago). Estas estrategias son comúnmente parametrizadas en motores de dcisión propios de las instituciones.

Una vez formalizado un crédito, este se vuelve parte de la cartera y se debe comenzar

su gertión con la finalidad anticiparce a un posible deterioro a lo largo del plazo de financimaiento. Esto debido a que, pese a que el cliente pasó por un proceso de evaluación, las estrategias de admisión no son perfectas y pueden admitir clientes que eventualmente no podrán hacer frente a su compromiso de pago, ya sea por un posible sobre endeudamiento o cualquier otro factor atribuible o no al cliente.

Los reportes de seguimiento y gestión de cartera se enfocan típicamente en clasificar a los créditos de acuerdo a su estatus de mora en el momento de observación: créditos al corriente son aquéllos que tiene 0 días de atraso en su pago, créditos en un pago vencido (1 pv) son aquéllos con 1 a 30 días de atraso, créditos con 2 pv son aquéllos con 31 a 60 días de atraso, créditos con 3 pv son aquéllos con 61 a 70 días de atraso, créditos con 4 pv son aquellos con 71 a 80 días de atraso, créditos con 5 pv son aquéllos con 81 a 90 días de atraso, créditos con 6 pv son aquellos con 91 a 120, si el crédito incurre el más de 120 días de atraso se dice que es un crédito castigado o default

El impacto económico que genera un crédito en mora es muy alto, pues la institución se ve obligada, por regulación, a generar reservas ante el deterioro de la cartera, además de que se incurren en gastos de recuperación y cobranza. De aquí la relevancia en tener mejores modelos tanto de admisión como de comportamiento de crédito.

Desarrollo

En este trabajo se aplican diferentes técnicas de exploración de datos con el fin de ajustar dos modelos de clasificación para predecir el incumplimiento de pago de clientes tarjeta-habientes de AMEX. Los datos se obtuvieron del set de datos *amex-default-prediction* de la competencia de kaggle.

El desarrollo de este trabajo se resume, de forma general, de la siguiente manera:

- Eliminación de variables con 30% o más de valores vacíos
- Imputación de valores ausentes en variables con menos de 30% de valores vacíos
- Selección de variables para ajustar los modelos
- Ajuste del modelo

Para el caso de imputación de valores se eligió para todas las variables imputar la media. Por otro lado, para la selección de variables se emplearon dos métodos: Información mutua, con el que se seleccionaron todas las variables con score >=0.01, y posterior menete para reducir la multicolinealidad se realizó un cluster jerárquico de variables.

Finalmete se ajustaron, a una muestra de entrenamiento del 30% del total de los datos, una regresión logística y un arbol de decisión de clasificación. El desempeño de los modelos refleja que el mejor modelo fue la regresión logística, con un roc score de 0.81 versus un roc de 0.76 obtenido en el entrenamniento del arbol.

Librerías

In [3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from varclushi import VarClusHi
from sklearn.impute import SimpleImputer
from sklearn.metrics import roc_auc_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import mutual_info_classif
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
C:\Users\DIANAL~1\AppData\Local\Temp/ipykernel_2732/2684137303.py in <module
>
      2 import pandas as pd
      3 import seaborn as sns
----> 4 from varclushi import VarClusHi
      5 from sklearn.impute import SimpleImputer
      6 from sklearn.metrics import roc_auc_score

ModuleNotFoundError: No module named 'varclushi'
```

Carga de documentos

In [2]:
```
ruta = "../Datos/amex-default-prediction/"
```

In [3]:
```
training = pd.read_csv(ruta + "train_data.csv")
target = pd.read_csv(ruta + "train_labels.csv")
```

In [4]:
```
training.shape
```

Out[4]:
```
(5531451, 190)
```

Porcentaje de missings por variable

En las siguientes líneas de código se realiza la selección de variables que cumplen con un cierto porcentaje de valores missing. Dado que no se cuenta con una descripción detallada sobre las variables, y que estas están escaladas a valores dentro del rango [0,1], no se tiene mayor detalle sobre los valores ausentes. Es decir, no podemos identificar que valores es normal que sean vacíos. Ante esta situación, y con el fin no afectar las distribuciones de cada variable, se ha decidido conservar aquéllas que tienen completitud de 30% o más. Las que no cumplan esta condición serán descartadas.

In [5]:
```
def missings(df = training, pct = 0.3):
    vacios = df.isnull().sum()/len(df)
    ls_missings = [variable for variable, valor in vacios.items() if valor>0
    ls_remover = [variable for variable, valor in vacios.items() if valor>=p
    return ls_missings, ls_remover
```

In [6]:
```
missings, remover = missings(training, 0.3)
```

In [7]:
```
variables = [v for v in list(training.columns) if v not in remover]
```

In [8]:
```
training = training[variables].copy()
```

Pegamos el target

En las siguientes celdas realizamos simplemente un cruce para pegar las variables predictoras a su target correspondiente.

In [9]:
```
df_training = pd.merge(left = training,
                       right = target,
                       how = 'left',
                       on = ['customer_ID']).reset_index(drop = True)
```

In [10]:
```
df_training.dtypes
```

Out[10]:
```
customer_ID     object
S_2             object
P_2            float64
D_39           float64
B_1            float64
                 ...
D_141          float64
D_143          float64
D_144          float64
D_145          float64
target           int64
Length: 160, dtype: object
```

Separar las variables según su tipo

El siguiente código tiene el fin de identificar y separar las vairbales numéricas de las variables de texto

In [11]:
```
ID = ['customer_ID']
tgt = ['target']
ls_discretas = [v for v in df_training.columns if df_training[v].dtype == 'c
ls_numericas = [v for v in df_training.columns if df_training[v].dtype != 'c
print(len(ls_discretas))
print(len(ls_numericas))
```

```
3
155
```

Imputacion de valores ausentes para variables numéricas

Recordando que anteriormente se descartaron las variables con más del 30% de valores missings, en las siguiente líneas imputaremos la mediana a los valores ausentes de las variables restantes, en los casos donde aplique.

In [12]:
```
imputar = [var for var in ls_numericas if var in missings]
len(imputar)
```

Out[12]:
```
90
```

In [13]:
```
im = SimpleImputer(strategy='median')
im.fit(df_training[imputar])
```

Out[13]:
```
SimpleImputer
SimpleImputer(strategy='median')
```

In [14]:
```
aux = df_training.copy()
aux[imputar]=im.transform(aux[imputar])
```

In [15]:
```
aux.shape
```

Out[15]:
```
(5531451, 160)
```

Training-Test Split

Con el fin de tener ua muestra reducida y poder realizar los entrenamientos de los modelos de clasificación, se realizará una partición del conjunto de datos, con 70% de observaciones para el conjunto de entrenaniento y 30% para el conjunto de validación.

In [16]:
```
#aux_sample = aux.sample(frac=0.2, random_state=135)
#aux_sample.shape
#aux_sample.to_csv('muestreo.csv', index = False)
```

In [17]:
```
aux_sample, test = train_test_split(aux, test_size = 0.70)
```

In [18]:
```
aux_sample.shape
```

Out[18]:
```
(1659435, 160)
```

Seleccion de variables

Una vez particionado los datos, queda seleccionar las variables que más pueden predecir la variable target (default o no default), por lo que tenemos que elegir, de las 159 variables del dataset, las que mejor ayuden a describir el fenómeno. Para esto haremos uso de dos técnicas: Información Mutua para identificar cualquier tipo de relación (lineal o no lineal) entre las variables predictorias y la variable objetivo.

También haremos clustering de variables para eliminar la multicolinealidad y seleccionar las variables que mejor describen la varianza de los datos

Mutual Information

In [19]:
```
# Label encoding for categoricals
for colname in aux_sample[ls_numericas].select_dtypes("object"):
    aux_sample[ls_numericas][colname], _ = aux_sample[ls_numericas][colname]

# All discrete features should now have integer dtypes (double-check this be
discrete_features = aux_sample[ls_numericas].dtypes == int
```

In [20]:
```
def make_mi_scores(X, y, discrete_features):
    mi_scores = mutual_info_classif(X, y, discrete_features=discrete_feature
    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(aux_sample[ls_numericas], aux_sample.target, disc
```

In [21]:
```
# Seleccionamos todas las variables con score>=0.01
features = [variable for variable, valor in mi_scores.items() if valor>=0.01
```

Clustering de variables

In [22]:
```
vc = VarClusHi(aux_sample[features])
```

In [23]:
```
vc = VarClusHi(aux_sample[features])
vc.varclus()
aux2 = vc.rsquare.copy()
aux2 = aux2.sort_values(['Cluster','RS_Ratio'],ascending=[1,1]).reset_index(
aux2['k'] = aux2.groupby('Cluster').cumcount()+1
varc = aux2.loc[aux2.k==1]['Variable'].to_list()
```

In [24]:
```
len(varc)
```

Out[24]:
```
26
```

In [25]:
```
aux_sample[tgt + varc].target.value_counts(True)
```

Out[25]:
```
0    0.75111
1    0.24889
Name: target, dtype: float64
```

Una vez implementados los algoritmos de información mutua y clustering jerárquico de variables, nos quedamos al final con 28 variables predictoras;

In [26]:
```
varc
```

Out[26]:
```
['B_33',
 'D_118',
 'R_4',
 'D_75',
 'S_3',
 'D_131',
 'D_51',
 'S_24',
 'B_13',
 'D_41',
 'D_43',
 'D_45',
 'S_16',
 'B_14',
 'D_48',
 'D_81',
 'R_7',
 'B_37',
 'S_15',
 'D_128',
 'R_10',
 'S_27',
 'D_120',
 'D_121',
 'B_26',
 'B_40']
```

Las variables seleccionadas a partir del análisis realizado fueron 28, de las cuales saqueremos algunas descripciones y veremos la correlación existente entre el target.

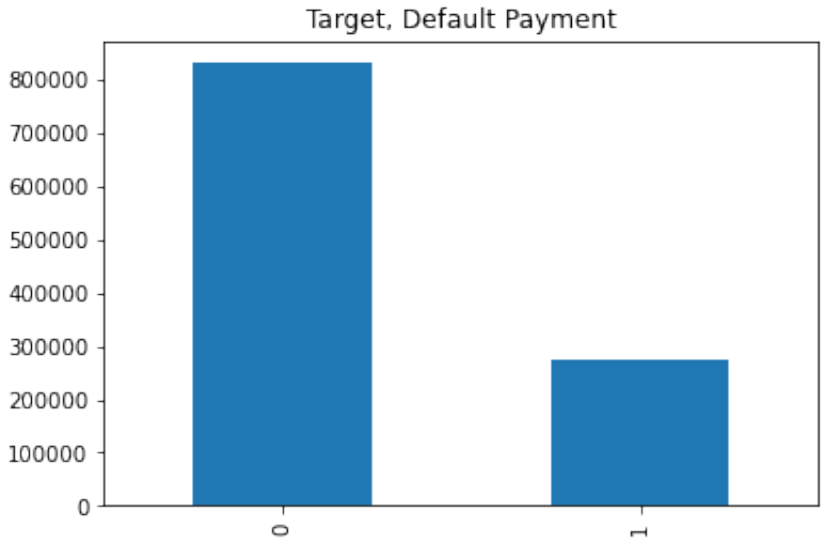
A continuación se presentan algunas gráficas que nos permiten ver:

1. La diferencia entre Default y No Default
2. Correlación existente entre las variables, donde observamos que target tiene tanto correlación positiva como negativa con las variables seleccionadas.

In [5]:
```
varc = varc[['B_33', 'D_118',
             'R_4', 'D_75', 'S_3', 'D_131', 'D_51', 'S_24', 'B_13', 'D_41',
             'B_14', 'D_48', 'D_81', 'R_7', 'B_37', 'S_15', 'D_128', 'R_10',
```

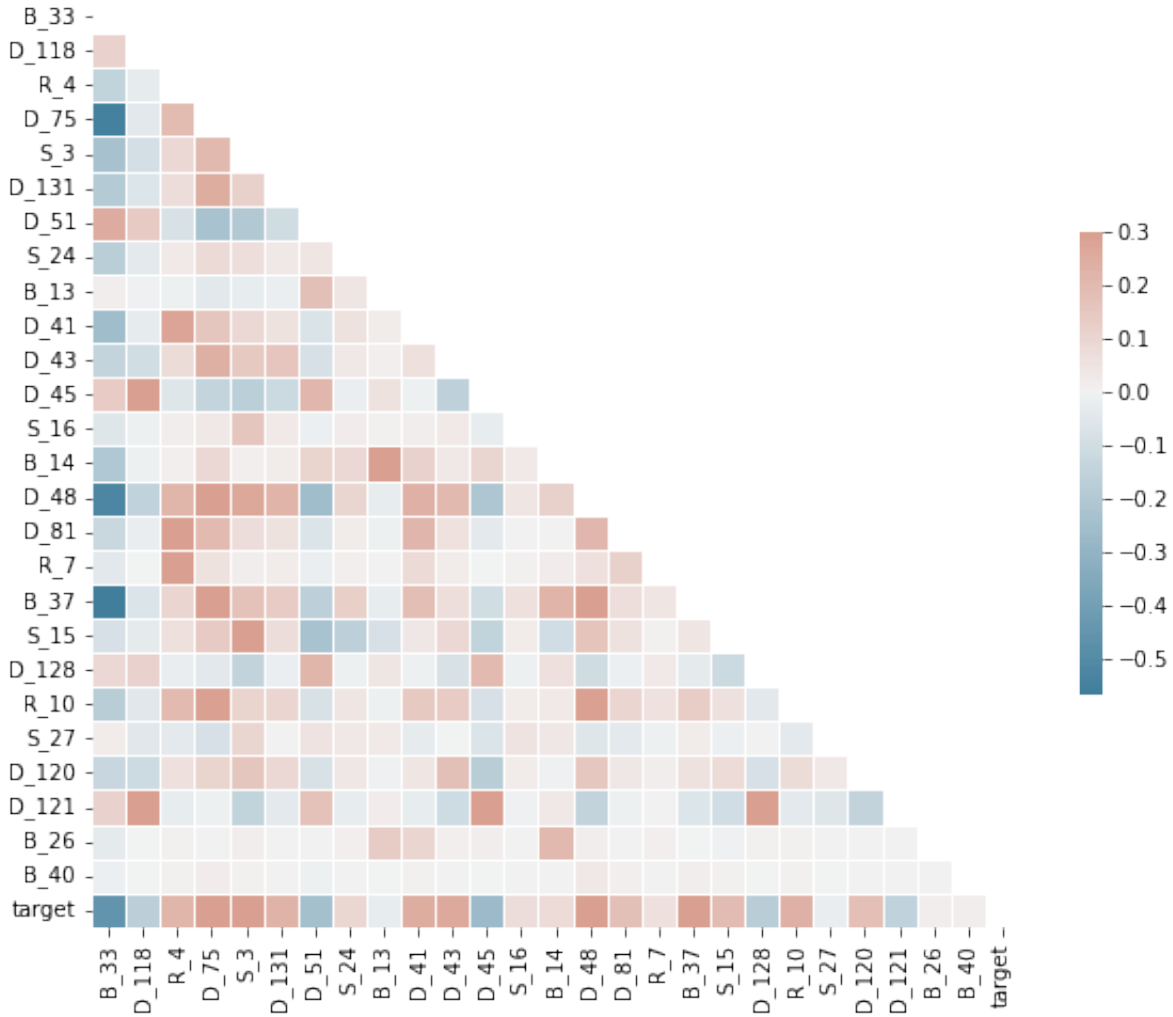
In [6]:
```
varc['target'].value_counts().plot(kind='bar',
                                   title='Target, Default Payment')
```

Out[6]:
```
<AxesSubplot:title={'center':'Target, Default Payment'}>
```



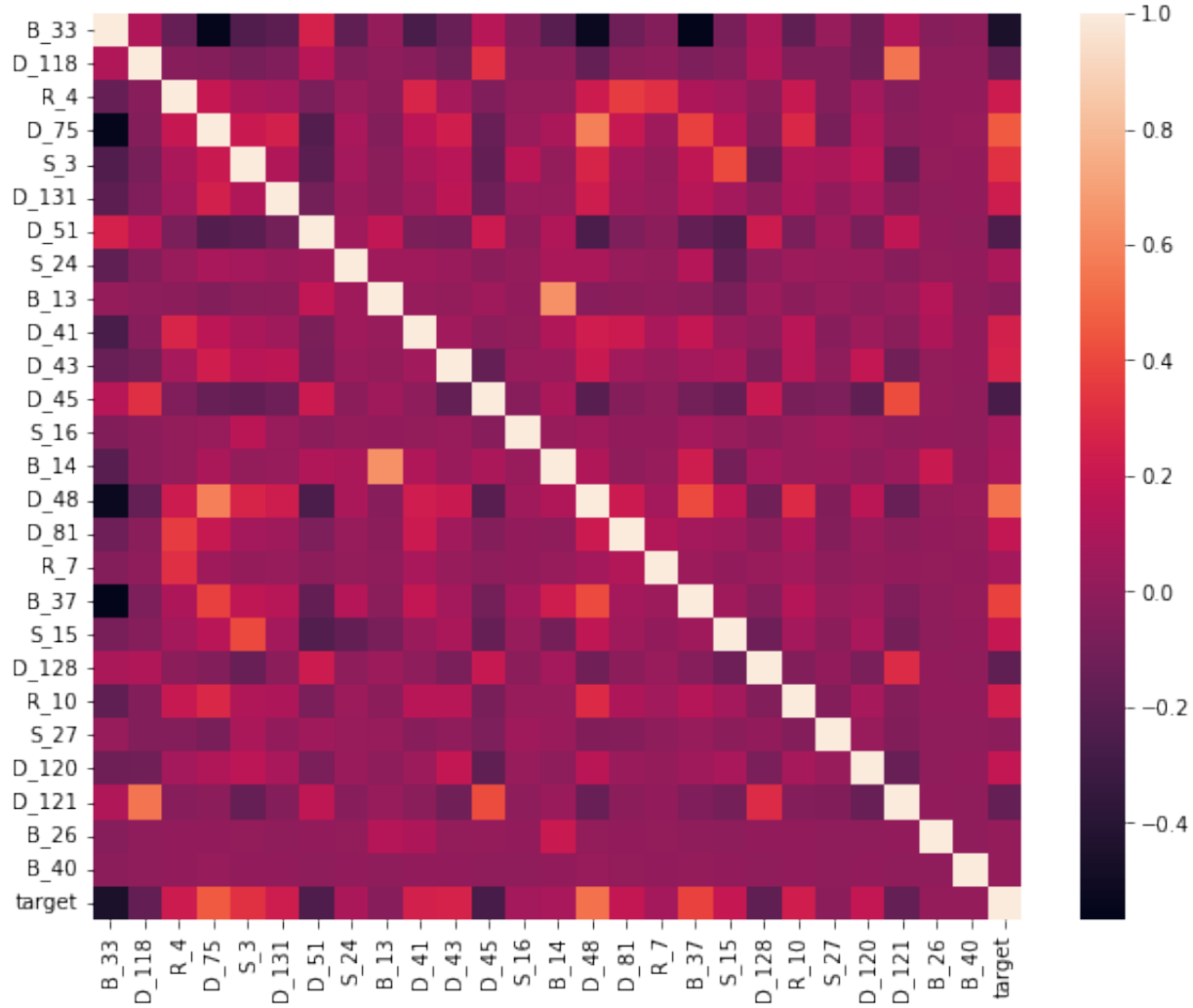
In [11]:
```
# Compute the correlation matrix
corr = varc.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(10, 8))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

Out[11]:
```
<AxesSubplot:>
```



In [10]:
```
plt.subplots(figsize=(10, 8))
sns.heatmap(varc.corr())
```

Out[10]:
```
<AxesSubplot:>
```



Implementación de modelos

Por simplicidad renombramos las variables de la siguiente forma:

In [28]:
```
# Variables predictoras:
X = aux_sample[features]
#Variable objetivo
y = aux_sample[tgt]
```

Regresión logística

Se ha elegido, en primera instancia, una regresión logística por su simplicidad. Pensando que los modelos de originación de crédito son auditables y fáciles de implementar se opta, en primer lugar, por abordar el problema usando un modelo sencillo.

In [29]:
```
log_r = LogisticRegression()
log_r.fit(X, y)
```

Out[29]:
```
▾ LogisticRegression
LogisticRegression()
```

Accuracy

In [30]:
```
log_r.score(X, y)
```

Out[30]:
```
0.8687137489567233
```

ROC

In [31]:
```
roc_auc_score(y_true=y, y_score=log_r.predict(X))
```

Out[31]:
```
0.81216772740409
```

Validación

In [32]:
```
X_val = test[features]
y_val = test[tgt]
```

In [33]:
```
roc_auc_score(y_true=y_val, y_score=log_r.predict(X_val))
```

Out[33]:
```
0.8116957310633377
```

Árbol de decisión

Con el fin de comparar el desempeño de la regresión logística, se entrena un árbol de decisión.

In [34]:
```
arb = DecisionTreeClassifier(criterion = 'entropy')
```

In [35]:
```
arb.fit(X, y)
```

Out[35]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

In [36]:
```
ls_res = cross_val_score(estimator = arb, X=X, y=y, cv=4, scoring="roc_auc")
```

In [37]:
```
np.mean(ls_res), np.std(ls_res)
```

Out[37]:
```
(0.77030954666152, 0.00029015111107701607)
```

Se observa que el desempeño de la regresión logística es mucho mejor que el desempeño del árbol.

Conclusiones

El desarrollo de modelos predictivos para estimar la probabilidad de default se puede reducir a un problema relativamente sencillo de clasificación binaria. Identificar aquéllos clientes que muy probablemente incurrirán en impago representa reducir pérdidas económicas para la insitución que otorga el crédito y optimización en la rentabilidad de la cartera.

En este ejercicio el problema de predicción de impago se abordó, en primer lugar y ante el desconocimiento del detalle de cada variable, excluyendo las características con maypt porcentaje de valores vacíos (más del 30%) e imputando la media en los casos donde los valores vacíos representaban 30% o menos del total de observaciones. Posteriormente se redujo la dimensionalidad del dataset, seleccionando las variables mejor relacionadas a la variable objetivo y seleccionando aquéllas que mejor describieran la varianza de los datos. Finalmete se entrenaron los modelos de clasificación. Para este caso se entrenó, por simplicidad, una regresión logística que generó muy buenos resultados en el desempeño, incluso frente al árbol de decisión.

In [ ]:
```
```