

# Proyecto\_Estancia

November 24, 2018

## 1 Modelo de propensión a la contratación de créditos al consumo

Raúl Abraham Nieto Pacheco 51556

## 2 Proyecto: Selección de Clientes para campañas de préstamos bancarios.

### 3 Contenido

3. Introducción
4. Descripción del problema
5. Descripción de los datos
  - Análisis Exploratorio
6. Tratamiento de Datos
  - Limpieza
7. Data Partition
  - Transformación
8. Selección de variables
  - Correlación
  - árboles de decisión
  - ANOVA-Fvalue
  - AUC-variables
  - PCA
9. Modelos
  - SVM
  - Regresión Logística
  - Naive Bayes

- Redes Neuronales
- Random Forest

## 10. Selección del Modelo

- Cross-validation
- Champion Model

## 11. Selección del umbral de Clasificación

## 12. Conclusiones

## 13. Anexos

## 14. Referencias

## 4 Introducción

Es bien sabido que el otorgamiento de créditos es uno de los negocios más rentables para BBVA Bancomer, es por ello que se asigna mucho presupuesto a la generación de campañas para la colocación de estos dirigidas a clientes y no clientes de tal modo que el objetivo, en el primer caso, además de acrecentar el negocio también es incrementar la fidelidad del cliente con el banco y en el segundo caso hablamos de atraer nuevos clientes con ofertas crediticias atractivas. Por supuesto, se cuenta con información de clientes actuales y en este caso se pueden crear campañas dirigidas donde esto significa una personalización en el modo de comunicarle la oferta y esto es posible ya que se cuenta con datos como saldos, créditos, productos, movimientos,etc. del cliente de tal modo que se pueden hacer perfilamientos de estos a través de inferencias estadísticas que nos ayuden a seleccionar a los clientes de tal modo que la respuesta sea máxima y el costo lo menor posible, es decir si seleccionamos a los “mejores” clientes podemos ahorrar en gastos de comunicación(email,sms, etc.) ya que estaríamos escogiendo a los más propensos a aceptar la oferta. Entonces, para poder hacer una selección óptima tenemos que encontrar las características más importantes de los clientes relacionadas con la aceptación de un préstamo crediticio, además tomar en cuenta que sea solvente para poder pagarla, de tal modo que podamos jerarquizar a cada uno de ellos con respecto a la posibilidad de que acepten la oferta y con ello lograr que la campaña sea exitosa en términos de tasa de aceptación y de costos.

## 5 Descripción del Problema

Se quiere hacer un modelo que seleccione a los clientes adecuados para una campaña , hoy día la selección de estos clientes está basada en criterios simples como por ejemplo aquellos, que de acuerdo con los datos que se tienen, que tienen menor liquidez esto conlleva a preguntarse si los clientes seleccionados son los correctos de acuerdo con una métrica como la efectividad de la campaña, entonces es obvio que no se disponen de modelos de cálculo de propensiones para predecir quiénes son los clientes que les interesa contratar los préstamos, entonces como **objetivo: se busca construir un clasificador que pueda mejorar la efectividad de la campaña(1.17%) haciendo una selección inteligente.**

Se realizará un modelo que sea capaz de predecir la propensión que tiene cada cliente de aceptar la campaña. la idea es que este modelo pueda proporcionar una lista de clientes que sean

los más adecuados a aceptar los préstamos que el banco les ofrece utilizando una base de datos conformada principalmente por datos de saldos del cliente en el tiempo, así como saldos en los distintos productos de activo y pasivo con los que cuentan hoy dentro del banco, además de variables nominales como el sexo, estado de la república al que pertenece etc .se tienen poco menos de 100 variables disponibles independientes y una variable dependiente que es la respuesta si los clientes que tenemos en la base han contratado algún préstamo analizando la información del último año, y la idea es que con la construcción de este, podamos calificar a los clientes propensos para aceptar futuras campañas. Dado el alto número de variables es natural pensar en analizar la reducción de variables para poder hacer el clasificador utilizando el menor número de variables sin perder información, entonces dado que tenemos variables de tipos mixtos vale la pena considerar el uso de pca para reducción de variables numéricas y técnicas de mca para datos categóricos y finalmente utilizar varios algoritmos de clasificación ejemplo el SVM para clasificar los clientes que son los adecuados para responder a la campaña. Finalmente, para crear una conversación comercial tenemos que etiquetar a los clientes de forma que podamos gestionarlos en las campañas de forma distinta, no es suficiente con clasificar quienes son los clientes a acampañar sino además sabemos que dentro de estos existen distintos niveles de propensión entonces es importante saber quién es el cte más y menos propenso a aceptar la oferta para poder comunicar las ofertas de forma priorizada ya que por temas de negocio no se comunican todos los ctes. al mismo tiempo sino por bloques, entonces debemos etiquetar a los ctes. de acuerdo con su propensión de tal modo que se etiqueten como "muy propensos" al bloque de ctes. que se va a comunicar primero, "propensos medios" al segundo bloque etc. . . .por tanto es necesario traducir el score de los modelos a una etiqueta. . .

## 6 Descripción de los datos

Se cuenta con un dataset de 176,330 registros(ctes.) con 92 campos o variables donde existe una variable de identificación de clientes (id2) y una variable respuesta a modelar que es **contrata** la cual es una variable binaria donde **1** significa que el cte. acepta contratar un préstamo y **0** en otro caso. La tasa de aceptación general es de **1.17%**

resumen de tipos de variables **float64(67)**, **int64(13)**, **object(12)**

Out [1] : <matplotlib.axes.\_subplots.AxesSubplot at 0x7fe54f621240>

Variable	Descripción	Tipo de dato
id2	ID	int64
IM_MAX_SDO_MEDIO_12M	saldo máximo en los últimos 12 meses	float64
IM_PROM_HIPOTECA_12M	importe promedio de pagos de hipoteca en los últimos 12 meses	float64
IM_PROM_PREST_NOM_12M	importe promedio mensual de préstamos de nómina en los últimos 12 meses	float64
IM_PROM_AUTO_12M	importe promedio de pagos de crédito auto en los últimos 12 meses	float64
IM_PROM_PPIS_12M	importe promedio mensual de ppis en los últimos 12 meses	float64
IM_PROM_OTR0_12M	importe promedio mensual de otros productos en los últimos 12 meses	float64
IM_PROM_CARTERA_12M	promedio del saldo de créditos en los últimos 12 meses	float64
IM_PROM_SDO_CORTE_12M	promedio mensual de saldos al corte de los últimos 12 meses	float64
IM_PROM_SDO_MEDIO_12M	promedio mensual de saldos en los últimos 12 meses	float64
IM_PROM_VISTA_12M	saldo promedio de captación de los últimos 12 meses	float64
IM_PROM_PLAZO_12M	importe promedio mensual en plazos en los últimos 12 meses	float64
IM_PROM_FONDO_RF_12M	importe promedio del saldo en fondos de renta fija de los últimos 12 meses	float64
IM_PROM_FONDO_RV_12M	importe promedio del saldo en fondos de renta variable de los últimos 12 meses	float64
IM_PROM_FONDOS_AES_12M	importe promedio de saldos de fondos avanzados, esenciales y selectos	float64
IM_PROM_MERCADOS_12M	importe promedio de saldos en productos de mercados cap, div, in	float64
IM_PROM_ACTIVO_12M	importe promedio mensual de pagos a crédito en los últimos 12 meses	float64
IM_PROM_VISTA_6M	saldo promedio de captación de los últimos 6 meses	float64
IM_PROM_PREST_CONSUMO_6M	importe promedio mensual de préstamos al consumo en los últimos 6 meses	float64
IM_PROM_SDO_CORTE_3M	promedio mensual de saldos al corte de los últimos 3 meses	float64
IM_SUM_SDO_CORTE_3M	suma de saldos a la fecha de corte de los últimos 3 meses	float64
TO_MISS_SDO_CORTE_3M	número de veces que no se ha reportado el saldo al corte en los últimos 3 meses	float64
IM_PROM_SDO_MEDIO_3M	promedio mensual de saldos en los últimos 3 meses	float64
IM_PROM_VISTA_3M	saldo promedio de captación de los últimos 3 meses	float64
IM_PROM_NOMINA_3M	importe promedio de depósitos e nómina en los últimos 3 meses	float64
IM_PROM_PREST_CONSUMO_3M	importe promedio mensual de préstamos al consumo en los últimos 3 meses	float64
IM_SUM_FONDOS_AES_3M	suma de saldo promedio de fondos AES en los últimos 3 meses	float64
TO_MISS_FONDOS_AES_3M	número de veces que no se ha reportado el saldo de fondos AES en los últimos 3 meses	float64
IM_SUM_SDO_MEDIO_3M	suma de saldos medios captación mensuales de los últimos 3 meses	float64
TO_MISS_SDO_MEDIO_3M	número de veces que no se ha reportado el saldo medio en los últimos 3 meses	float64
IM_SUM_ACTIVO_TOTAL_3M	suma de saldos de todos los créditos en los últimos 3 meses	float64
TO_MISS_ACTIVO_TOTAL_3M	número de veces que el cte ha estado activo en los últimos 3 meses	float64
IM_PROM_SDO_CORTE_1M	promedio mensual de saldos al corte del último mes	float64
IM_PROM_SDO_MEDIO_1M	promedio mensual de saldos en el último mes	float64
IM_PROM_ACTIVO_1M	importe promedio mensual de pagos a créditos en el último mes	float64
IM_PROM_PREST_CONSUMO_1M	importe promedio mensual de préstamos al consumo en el último mes	float64
IM_SUM_SDO_CORTE_1M	suma de saldos a la fecha de corte del último mes	float64
TO_MISS_SDO_CORTE_1M	se ha reportado el saldo al corte en el último mes	float64
IM_SUM_SDO_MEDIO_1M	suma de saldos medios captación mensuales de los último mes	float64
TO_MISS_TO_SDO_MEDIO_1M	se ha reportado el saldo medio en el último mes	float64
IM_SUM_IM_SDO_CORTE_2MA	saldo al corte 2 meses antes	float64
TO_MISS_TO_SDO_CORTE_2MA	saldo al corte fue reportado hace 2 meses	float64
IM_SUM_IM_SDO_MEDIO_2MA	saldo medio 2 meses antes	float64
TO_MISS_TO_SDO_MEDIO_2MA	saldo medio fue reportado hace 2 meses	float64
IM_COC_IM_FONDOS_AES_CAP_0_2	0	float64
IM_COC_IM_SDO_CORTE_ACTIVO_0_2	0	float64
IM_DELTA_IM_SDO_CORTE_0_2	delta de saldo al corte actual vs. Hace 2 meses	float64
IM_DELTA_IM_SDO_MEDIO_0_2	delta de saldo medio actual vs. Hace 2 meses	float64
IM_PROM_IM_CARGOS_12M	importe promedio de cargos de los últimos 12 meses	float64
IM_PROM_IM_CARGOS_6M	importe promedio de cargos de los últimos 6 meses	float64
IM_PROM_IM_CARGOS_3M	importe promedio de cargos de los últimos 3 meses	float64
IM_PROM_IM_CARGOS_1M	importe promedio de cargos de los último mes	float64
TO_PROM_TO_CARGOS_6M	promedio del numero de cargos en los últimos 6 meses	float64
TO_PROM_TO_CARGOS_3M	promedio del numero de cargos en los últimos 3 meses	float64
IM_PROM_IM_OPERERS_TARJ_TDC_1M	importe promedio mensual de pagos con tdc en el último mes	float64
IM_PROM_IM_OPERERS_TARJ_TDC_3M	importe promedio mensual de pagos con tdc en los últimos 3 meses	float64
TO_PROM_TO_OPERERS_TARJ_TDC_6M	importe promedio mensual de transacciones promedio mensuales hechas con TDC en los últimos 3 meses	float64
IM_PROM_IM_OPERERS_TARJ_TDC_6M	importe promedio mensual de pagos con tdc en los últimos 6 meses	float64
TO_FONDOS_AES	número de fondos/avanzados/esenciales y selectos	int64
TO_MERCADOS	número de productos de mercado de capitales/divisas/dinero	int64
TO_PLAZO	cuenta con algún producto de inversión	int64
TO_CTA_HIPOTECA	tiene algún crédito hipotecario vigente	int64
TO_FONDO_RF	tiene fondos de renta fija	int64
TO_FONDO_RV	tiene fondos de renta variable	int64
TO_PREST_CONSUMO	número de préstamos contratados en el último año	int64
TO_DELTA_TO_PREST_CONSUMO_0_2	delta del número de créditos al consumo en los último 3 meses	float64
NU_REGULARIDAD_TDC_ULT_6M	Frecuencia mensual de uso de la tdc en los últimos 6 meses	int64
IM_PROM CUOTA_TOT_ULT_1M	importe promedio de cuota de tdc pagada en los últimos 6 meses	float64
IM_PROM CUOTA_TOT_ULT_3M	importe promedio de cuota de tdc pagada en los últimos 3 meses	float64
NU_VINC_BANCOPER	índice de fidelidad	int64
NU_VINC_COGNODATA	Estimación de la fidelidad del cliente	int64
TO_IR_AJUSTADO	estimación de ingresos anuales	float64
TP_NIVEL_IR_AJUSTADO	nivel de ingresos anuales estimados	object
IM_PROM_IM_NIBT_TOTAL_12M	rentabilidad promedio mensual del cte. en el último año	float64
TO_RANGO_EDAD	Rango de edad del cliente	int64
IM_PROM_GASTOS_1M	importe promedio de gastos en el último mes	float64
IM_PROM_GASTOS_3M	importe promedio de gastos en los últimos 3 meses	float64
TP_SEGMENTO_FINAL2	Segmento de ciclo de vida	object
TP_SUBSEGMENTO	clasificación del cliente	object
TO_NECESSIDAD_FINAN_CAP_1M	Ratio de captación vs. Créditos en el último mes	float64
TO_NECESSIDAD_FINAN_CAP_3M	Ratio de captación vs. Créditos en los últimos 3 meses	float64
TO_DELTA_NECC_CAP_0_2	delta de necesidad financiera con respecto a 3 meses anteriores	float64
TP_PERSONA	Persona Física o moral	object
TP_DOMICILIO	Tipo de domicilio	object
CD_ESTADO	Estado del Cliente	object
CD_EDO_CIVIL	estado civil	object
CD_SEGMENTACION	segmento operativo	object
CD_SEXO	sexo	object
CD_OCUPACION	Ocupación	object
CD_RESIDENCIA	Pais de Residencia	object
TP_VIVIENDA_SEPO	tipo de vivienda	object
contrata	variable de respuesta: 1 contrata y 0. e.o.c	int64

## Figura 1.-Descripción de Variables

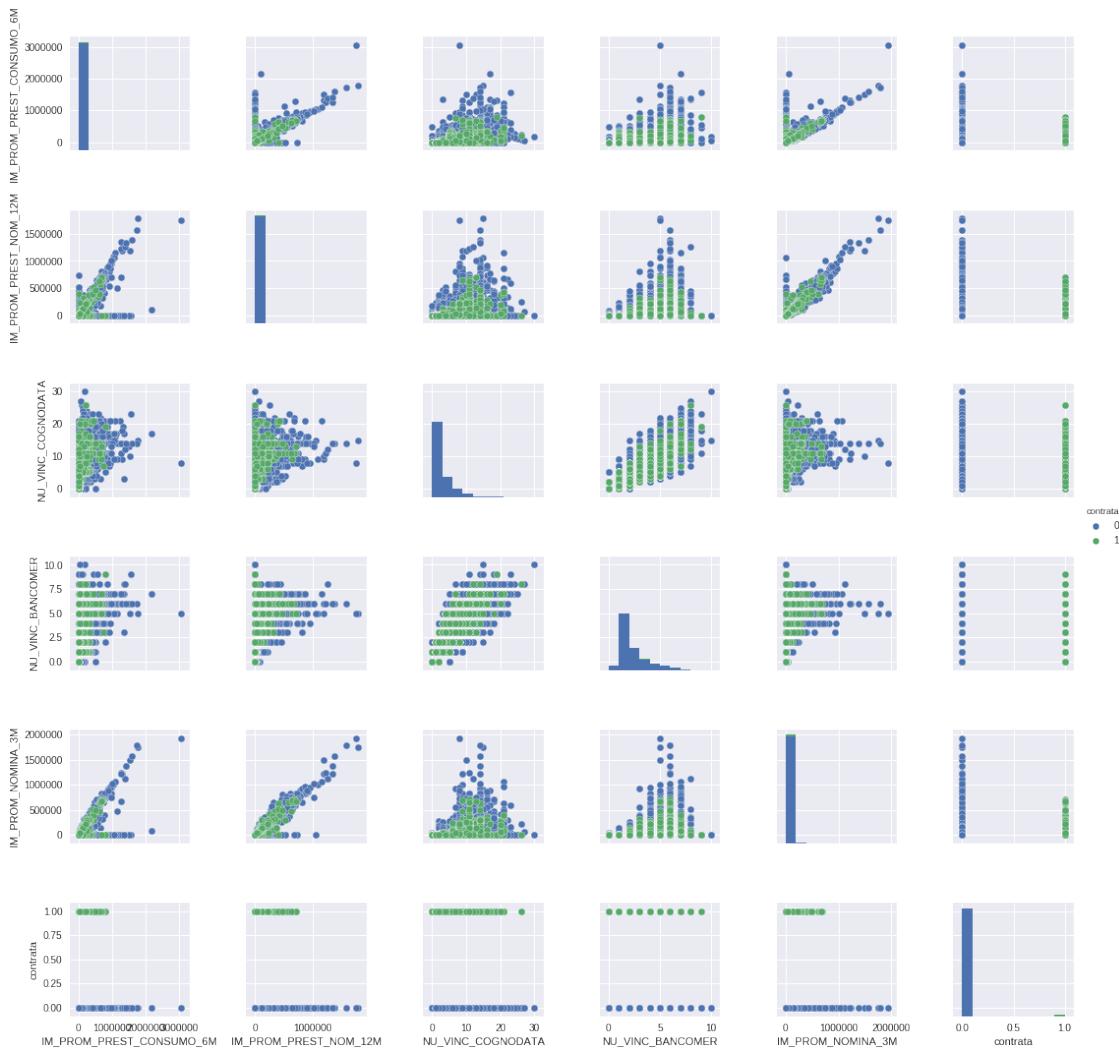
La explicación de las variables se resume en importes de saldos de productos, promedios mensuales de gastos, número promedio de transacciones todo esto visto en diferentes momentos del tiempo en un periodo anual por ejemplo muchas variables como el importe promedio de cargos se analizan a 12, 6 y 3 meses entonces tenemos 3 variables que por obvias razones están correlacionadas (casi siempre). todo lo anterior se repite para la mayoría de las variables numéricas, en el caso de las categóricas basta revisar cada descripción de estas.

## 7 Análisis Exploratorio(EDA)

Se carga la información y a continuación vamos a analizar los datos:

### Variables numéricas:

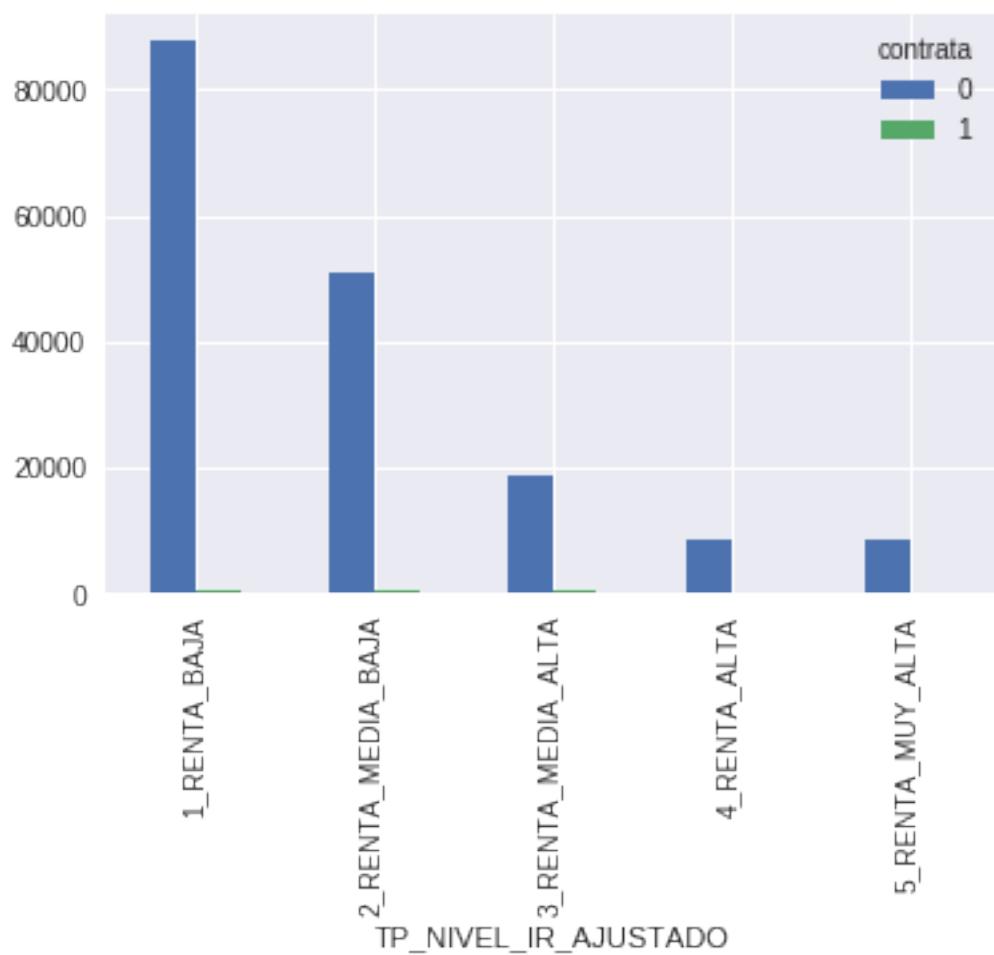
Out [4]: <seaborn.axisgrid.PairGrid at 0x7f8e88584a58>

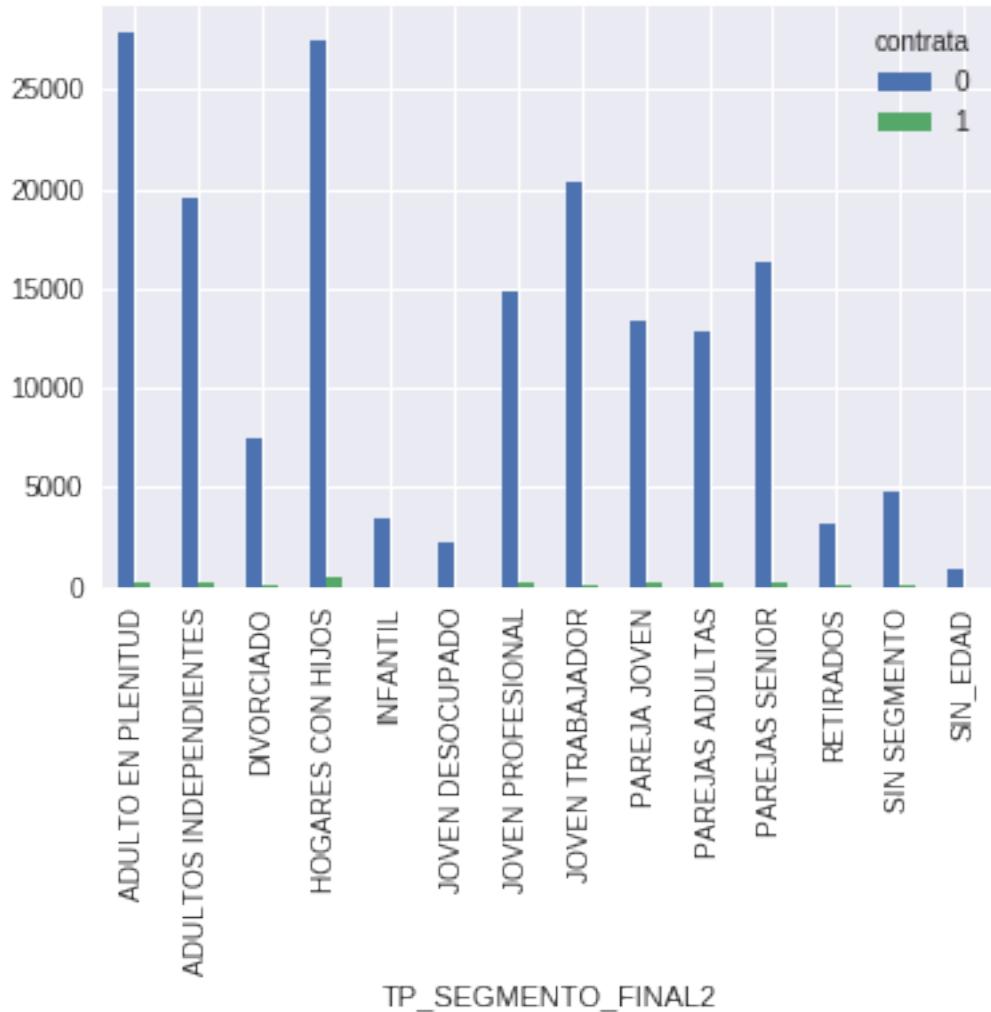


**Figura 2.- Pairplot de algunas variables numéricas.**-El resto de las variables se pueden ver en el Anexo1.

De acuerdo con las distribuciones de las variables numéricas podemos ver que la mayoría de estas se carga hacia el cero esto se da porque muchas de las variables son saldos e importes de productos con los que quizás no cuenta el cliente así que para todos aquellos que no tengan dicho producto el dato será 0, por ejemplo veamos IM\_PROM\_NOMINA\_3M donde la mayoría de los valores es cero, pero lo que esto significa es que existen muchos clientes que no tienen nómina del banco entonces todos estos no pueden tener un saldo por eso es 0. Además no se ve muy claro alguna variables que discrimine un poco las contrataciones. Otro dato importante es que las variables están muy correlacionadas ya que muchas son la misma variable vista en distintos momentos.

**Variables nominales:**





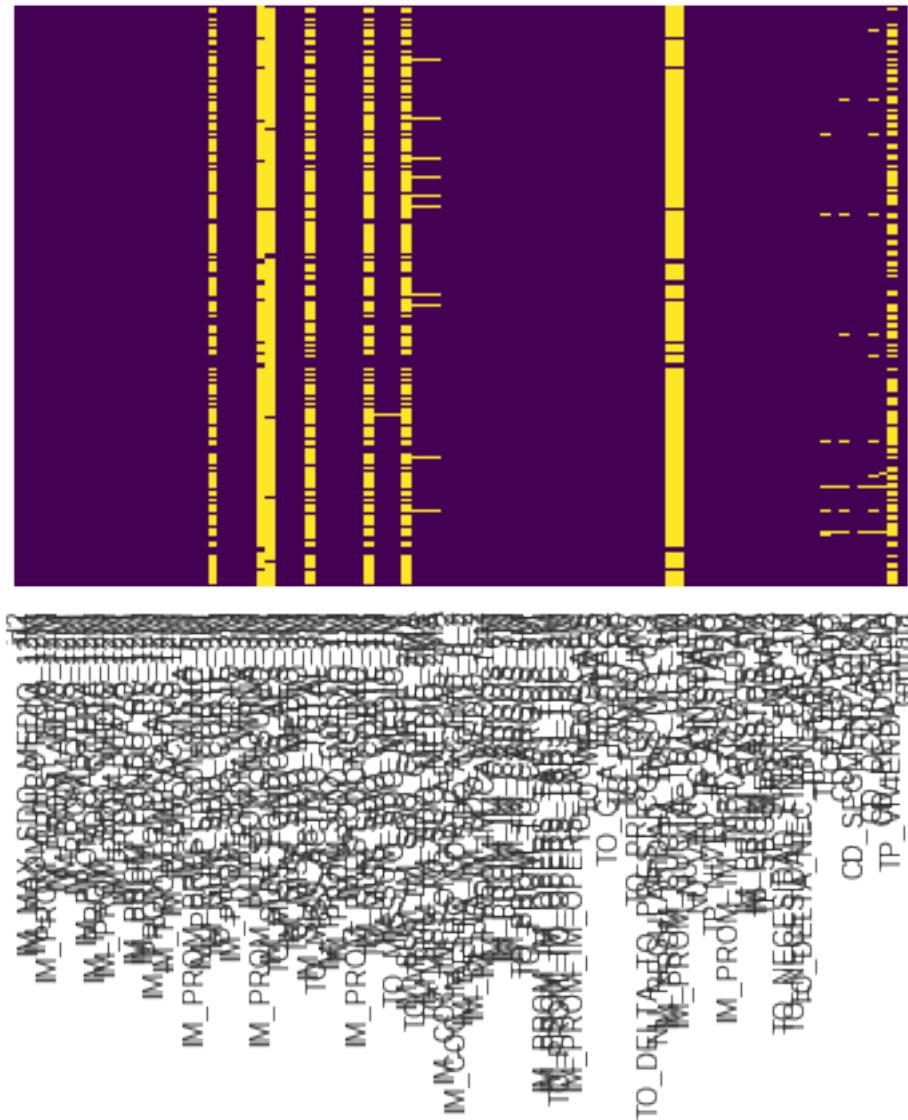
**Figura 3.-Gráfica de Frecuencias de variables categóricas.** El resto de las variables se pueden observar en el Anexo2.

En general podemos ver los siguientes hallazgos, la variable TP\_NIVEL\_IR\_AJUSTADO en realidad no es nominal es una variable ordinal ya que el 5 representa intervalo más alto de ingresos estimados anuales y el 1 el menor(**Figura 3**). La variable TP\_PERSONA sólo tiene una categoría por tanto no sirve para el modelado,casi igual tp\_domicilio, CD\_RESIDENCIA y TP\_VIVIENDA\_SEPO, por otro lado no se discriminan las contrataciones prácticamente en ninguna variable.

## 8 Tratamiento de Datos

Analizando la complezidad de los datos vamos a hacer un mapa de missings para revisar que tan completa está nuestra información...

Out [7]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7f8e322b1668>



**Figura 4.-Mapa de Datos con todas las variables.** En amarillo se muestran los valores ausentes por variable.

Como podemos observar existen varias variables que contienen una importante cantidad de valores ausentes(**Figura 4**), por tanto vamos a valorar a que variables deben imputarse y/o eliminarse bajo otros criterios, en este caso los datos ausentes, para el caso de las variables numéricas, significa que los ctes. no tienen, en la gran mayoría de los casos, el producto al que se hace referencia o que no tienen el dato reportado, pero en este caso esos ejemplos serán reemplazados por 0 ya que no afectan el significado del problema.

Vamos a cambiar el tipo de datos de algunas variables para hacer que tengan sentido de forma natural, primero cambiaremos el tipo de variable de TP\_NIVEL\_IR\_AJUSTADO de categórica a ordinal. Además, las variables NU\_VINC\_COGNODATA y NU\_VINC\_BANCOMER las haremos continuas en lugar de enteras o integers.

## 9 Limpieza

### Descripción variables numéricas

Out [9]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7f8e3223deb8>

metrífica	IM_MAX_SDO_MEDIO_12M	IM_PROM_HIPOTECA_12M	IM_PROM_PREST_NOM_12M	IM_PROM_AUTO_12M	IM_PROM_PPIS_12M
count	176330.0	176330.0	176330.0	176330.0	176330.0
mean	49184.96836040341	10982.817913688781	4577.007695177035	2258.6649101153967	1821.9801429823344
std	603185.4908882834	142481.6394128102	31609.50539692797	20224.95595450207	23646.77959568688
min	-13079.06	0.0	0.0	0.0	0.0
25%	391.8025	0.0	0.0	0.0	0.0
50%	1873.9099999999999	0.0	0.0	0.0	0.0
75%	12769.342500000002	0.0	0.0	0.0	0.0
max	367026076.03	34594177.371	1781076.5833	1129296.245	1748643.84

Figura 5.-Estadisticos de algunas variables numéricas

Se muestra el sumarizado de las variables numéricas dado que no se muestran todas por el número que hay vamos a buscar aquellas que no aporten valor a la variable de respuesta entonces si encontramos variables donde su máximo sea igual a su mínimo entonces el valor de la variable es único y se deben eliminar.

Out [10]: Index(['TO\_MISS\_SDO\_MEDIO\_3M', 'TO\_MISS\_TO\_SDO\_MEDIO\_1M', 'TO\_MISS\_TO\_SDO\_MEDIO\_2MA'],  
dtypes='object')

Estas 3 variables tienen un sólo valor por tanto vamos a eliminarlas del dataset.

Luego para aquellas variables que tienen en su gran mayoría ceros en sus datos es decir por ejemplo que su cuartil 1 y cuartil 3 son iguales a 0 vamos a crear variables pseudobinarias ya que lo que en realidad el significado de tener un dato mayor que cero significa que el cliente posee el producto en algunos y en otros que el dato ha sido informado por tanto hace más sentido utilizarlos así.

### Descripción variables categóricas

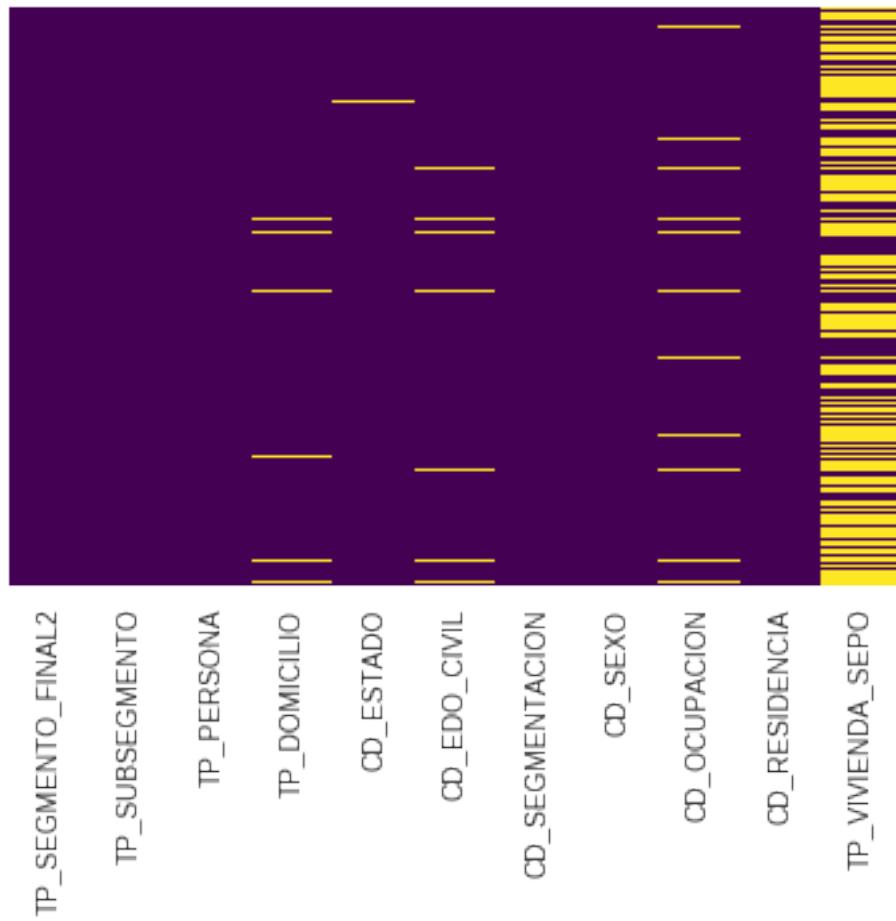
Out [14]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7f8e260d4320>

TP_SEGMENTO_FINAL2	TP_SUBSEGMENTO	TP_PERSONA	TP_DOMICILIO	CD_ESTADO	CD_EDO_CIVIL	CD_SEGMENTACION	CD_SEXO	CD_OCUPLICACION	CD_RESIDENCIA	TP_VIVIENDA_SEPO	metrica
176330	176330	176330	170545	172866	170208	176327	175458	168222	174903	71989	count
14	38	1	13	31	7	37	2	103	4	13	unique
ADULTO EN PLENTITUD ADULTO NO JUBILADO		F	H	DF	S	B1	M	SER	MEX	CASA	top
2800	21200	176330	169392	29628	83297	90860	94224	49728	374696	70829	freq

Figura 6.-Estadísticos de las variables categóricas. Los estadísticos se muestran en la variable métrica.

Al revisar las variables categóricas podemos ver que por ejemplo la variable tipo de persona (TP\_PERSONA) tiene 1 sólo valor(Figura 6) por tanto esta variable debe ser descartada ya que no aportará valor al modelo, la variable CD\_RESIDENCIA muestra que casi todos los ctes. viven México lo cual es obvio dado que es un banco mexicano por tanto esta variable tampoco se utilizará mismo caso de la variable TP\_VIVIENDA\_SEPO que además cuenta con más del 60% de los valores ausentes(Figura 7). TP\_DOMICILIO también se elimina ya que prácticamente todos los ctes. reportan que el domicilio que tienen dado de alta es su hogar.

Out[15]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7f8e25ff9ef0>



**Figura 7.-Mapa de Datos con variables categóricas.** En amarillo se muestran los valores ausentes por variable.

Se eliminan las variables comentadas:

TP\_PERSONA,CD\_RESIDENCIA,TP\_VIVIENDA\_SEPO,TP\_DOMICILIO,TO\_MISS\_SDO\_MEDIO\_3M, TO\_MISS\_TO\_SDO\_MEDIO\_1M y TO\_MISS\_TO\_SDO\_MEDIO\_2MA.

## 10 Data Partition

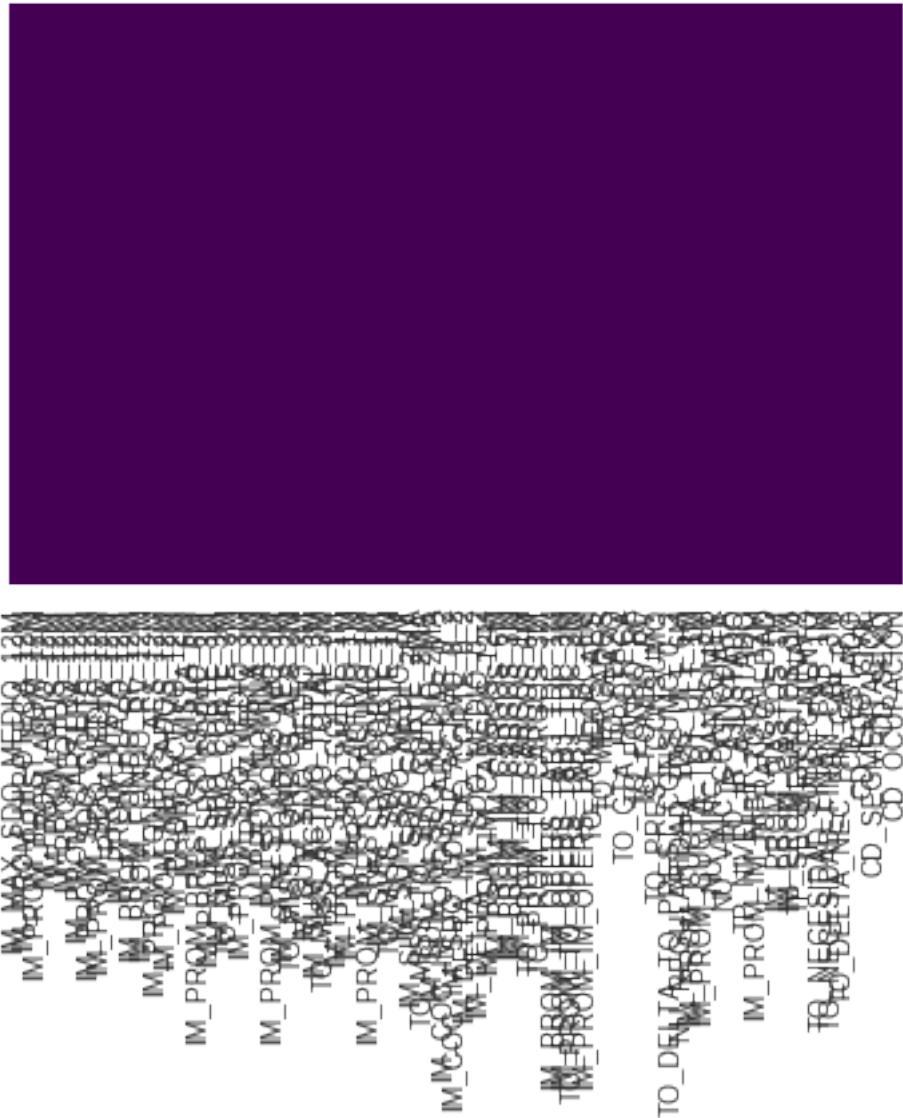
Vamos hacer nuestra partición de datos donde ocuparemos el 80% de la base para entrenar nuestros modelos y el otro 20% como dataset de validación, este dataframe de validación sólo se usará para ver **RESULTADOS** finales. A partir de este punto todo el tratamiento de datos que se le haga al entrenamiento también se hará con dataset de validación.

La densidad del set de test es: 1.17%

Los valores para las variables categóricas se reemplazan por la categorías más frecuentes en cada una de ellas y

Para valores numéricos vamos a imputar los missings con ceros y

Out [27]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7f8e387c4ac8>



Finalmente tenemos todos los valores ausentes imputados y la variables irrelevantes omitidas.

## 11 Transformación

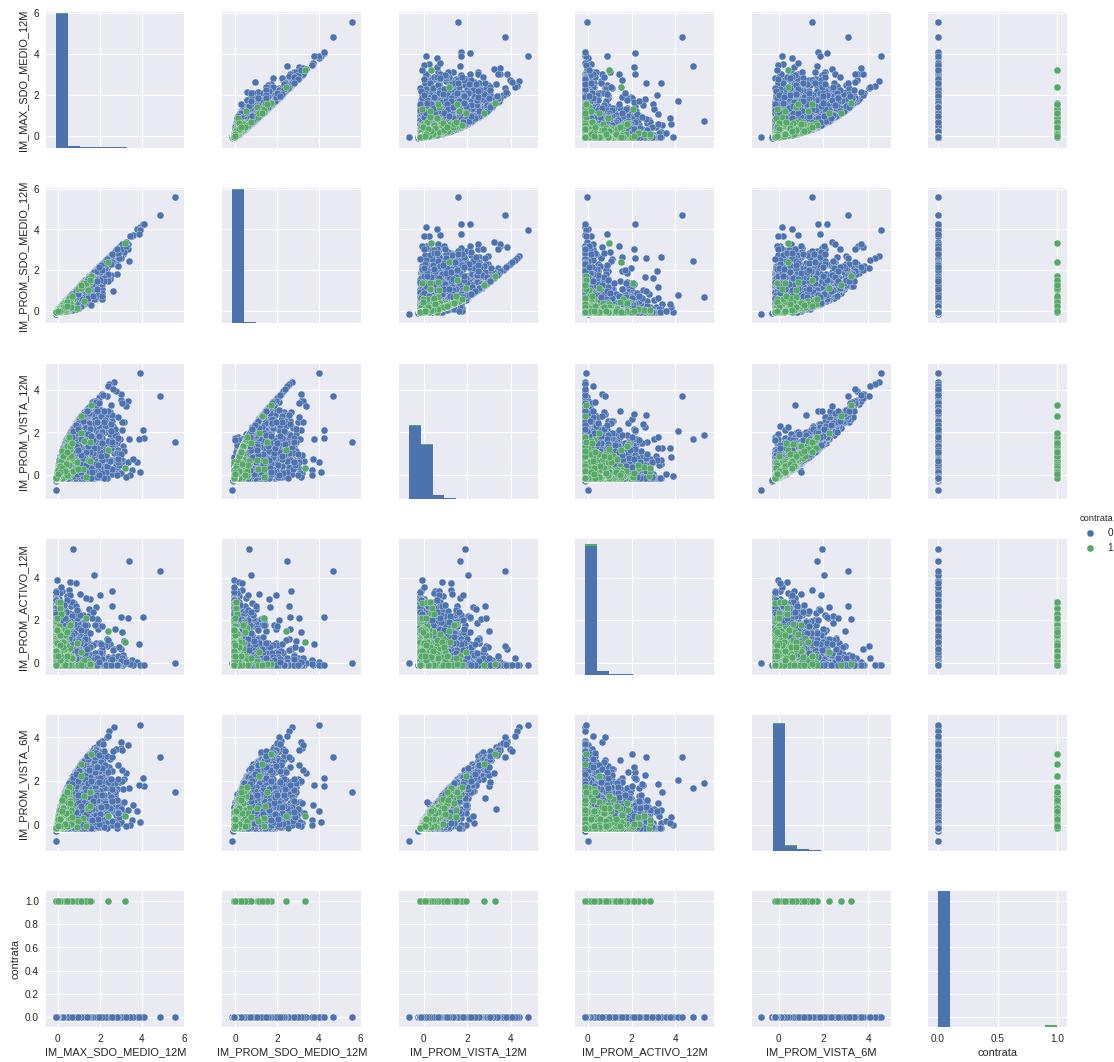
Vamos a explorar las variables con respecto al target de contratación para analizar datos como la distribución de las variables y su relación con la variable 'contrata' (Variable respuesta) así como

identificar qué variables requieren transformaciones o quizás cuales de ellas están muy correlacionadas no sin antes normalizarlas en el caso continuo para hacerlas comparables además de evitar problemas de escala y transformar las variables categóricas en númericas de alguna manera, para que estas últimas se puedan utilizar como inputs en los modelos.

**Estandarización y normalización de variables numéricas:** Además de estandarizar las variables restando su media y dividiendo por su desviación estándar, vamos a normalizarlas aplicando una versión del logaritmo natural  $\text{sign}(x) * \ln(|x| + 1)$ , esto para respetar el signo ya que tenemos variables con datos negativos, a cada una de ellas de tal modo que las variables numéricas se distribuyan y correlacionan como en la Figura 8.

**En este paso estandarizamos los datos de test usando la media y desviación del entrenamiento y aplicamos la versión del logaritmo en ambas particiones.**

Out [32] : <seaborn.axisgrid.PairGrid at 0x7f8e323c00f0>



**Figura 8.- pairplot de variables estandarizadas y normalizadas usando logaritmo natural.** El resto de variables se pueden observar en el Anexo3.

Con las variables numéricas normalizadas y los datos limpios es más clara la correlación que existe entre varias de ellas.

### **Reagrupación de categorías.**

Al revisar las categorías vemos que la mayoría de los clientes se concentran en pocas categorías por tanto vamos a reagrupar para tener menor número de categorías donde aplique. Reagrupación de variables categóricas:

CD\_OCUPACION se agupan: OTR= $x : x \in [SER, EPL, EDU, VTA, PEN]$

CD\_SEGMENTACION se agrupan: OT= $x : x \notin [B1, R1]$

CD\_EDO\_CIVIL se agrupan: O= $x : x \notin [S, C]$

CD\_ESTADO se agrupan: OT= $x : x \notin [DF, EM, JA, VE, PU, GU]$

TP\_SUBSEGMENTO se grupan: OTROSEG= $x : x \notin [ADULTO NO JUBILADO, ADULTO INDEPENDIENTE, JOVEN TRABAJADOR, PAREJAS SENIOR, PAREJA JOVEN, JOVEN PROFESIONAL]$

TP\_SEGMENTO\_FINAL2 se agrupan: OTROSEG= $x : x \notin [ADULTO EN PLENITUD, HOGARES CON HIJOS, JOVEN TRABAJADOR, ADULTOS INDEPENDIENTES]$

**El resto de categorías de cada variable permanece igual.**

**En este paso también reagrupamos las variables de test usando la forma que usamos en train.**

```
Out[35]: array(['OTROSEG', 'ADULTO EN PLENITUD', 'ADULTOS INDEPENDIENTES',
   'JOVEN TRABAJADOR', 'HOGARES CON HIJOS'], dtype=object)
```

Después de reagrupar las variables categóricas vamos recodificarlas para convertirlas en variables numéricas, entonces para esto vamos a crear variables binarias para cada variable por cada categoría de tal modo que entonces ahora contamos en total con 110 variables y 108 que pueden ser inputs de nuestros modelos(ya que la variable target 'contrata' y el 'id2' no pueden ser inputs).

### **Datos de entrenamiento:**

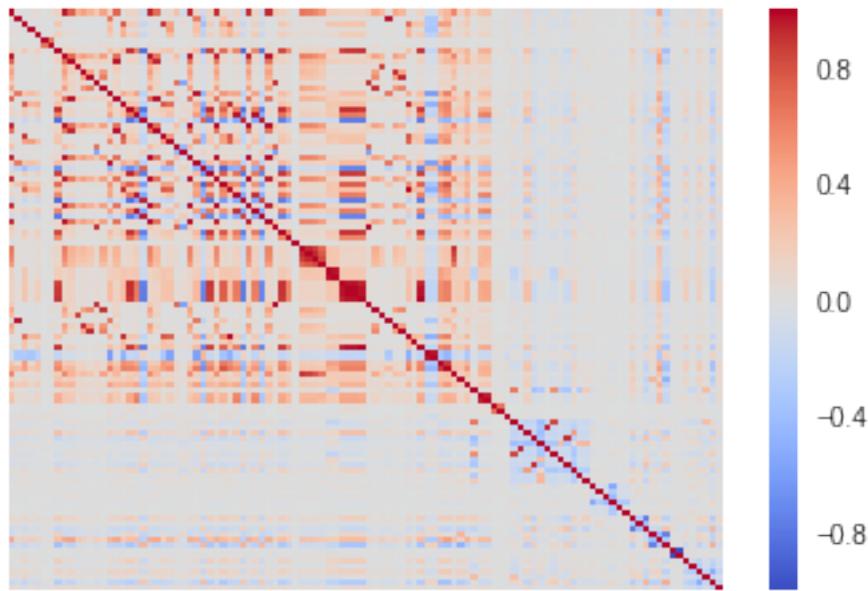
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 141064 entries, 144762 to 43567
Columns: 108 entries, IM_MAX_SDO_MEDIO_12M to CD_OCUPACION_VTA
dtypes: float64(40), int64(68)
memory usage: 117.3 MB
```

### **Datos de prueba:**

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 35266 entries, 51853 to 175971
Columns: 108 entries, IM_MAX_SDO_MEDIO_12M to CD_OCUPACION_VTA
dtypes: float64(40), int64(68)
memory usage: 29.3 MB
```

Dado que la tenemos muchas variables que son la misma sólo que vistas en diferentes momentos en el tiempo entonces es natural pensar en que tengamos variables correlacionadas(**Figura 9**) por tanto mirando la matriz de correlaciones podemos observar que tenemos muchas variables correlacionadas y cuando esto sucede significa redundancia de información por tanto es importante tratar de reducir dimensionalidad y para esto vamos a hacerlo utilizando la técnica de componentes principales como una de las posibles formas de seleccionar variables.

```
Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8e259f74e0>
```



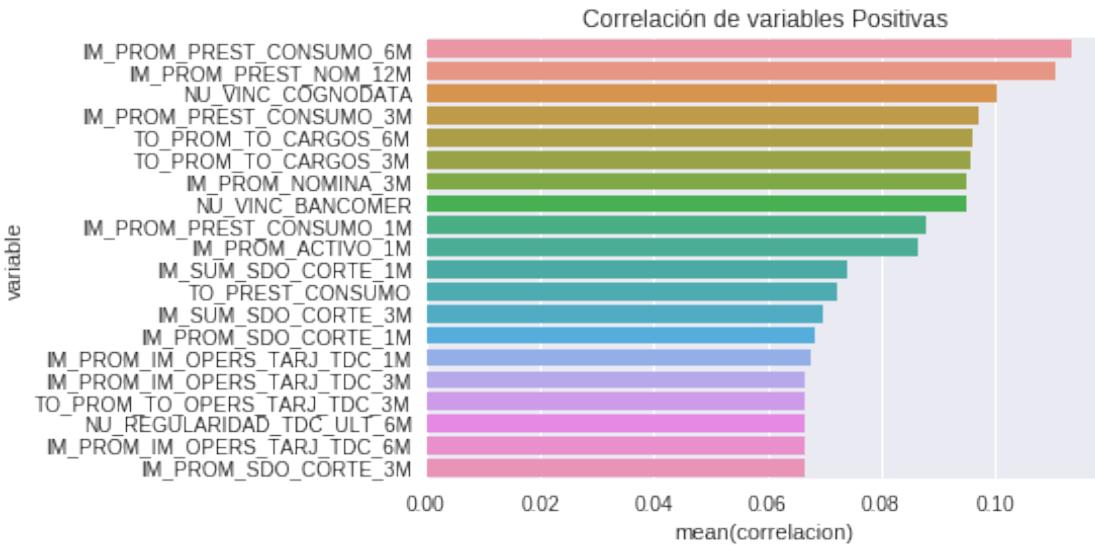
**Figura 9.-Matriz de correlaciones entre variables.**

## 12 Selección de variables

Para la selección de variables vamos a probar 5 formas distintas de seleccionar las variables más importantes y luego en la parte del modelado veremos cual selección es la que mejores resultados nos da utilizando los mismos modelos:

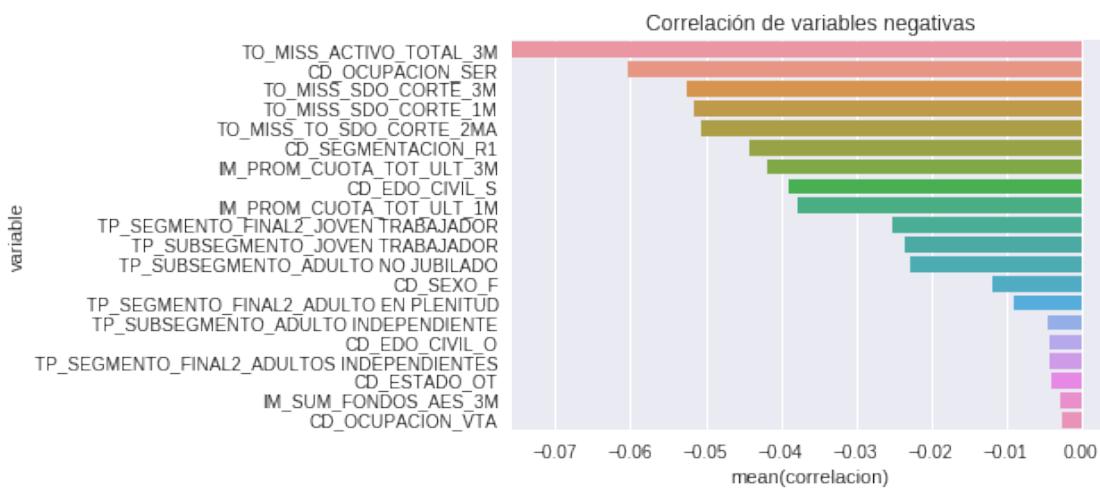
## 13 Correlación

Vamos a analizar la correlación de cada variable con respecto a nuestra variable target y así definir cuales son las variables que más relevancia tienen para modelar nuestro evento de tal modo que descartemos aquellas que no nos dan información.



CPU times: user 188 ms, sys: 252 ms, total: 440 ms  
Wall time: 152 ms

**Figura 10.- Lista de variables más correlacionadas con el target positivamente**



**Figura 11.-Lista de variables más correlacionadas con el target negativamente** El mapa completo de las variables se puede ver en el Anexo4.

Podemos ver que dentro de las variables más correlacionadas con las contrataciones de préstamos tenemos NU\_VINC\_COGNODATA que es una variable que significa a grandes rasgos el número de productos del banco con los que cuenta el cliente y la relación es positiva(**Figura 10**) “a mayor productos” mayor propensión a contratar. La variable TO\_MISS\_ACTIVO\_TOTAL\_3M es

la más correlacionada negativamente(**Figura 11**), es decir a mayor número de meses que el cliente no ha estado activo menor probabilidad tendrá de contratar el crédito.

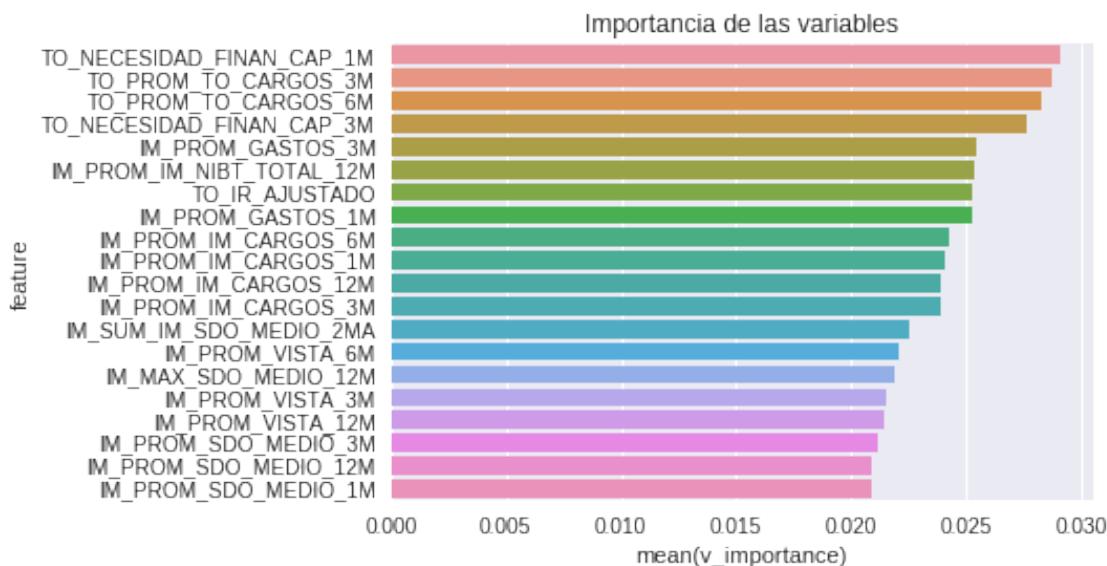
**Vamos a seleccionar aquellas variables que tengan una correlación mínima de 0.05 en valor absoluto**

Out [43] : 32

de tal modo que obtenemos 32 variables con mayor correlación con la contratación de préstamos.

Las variables se pueden observar en el Anexo5.

## 14 árboles de decisión(Random Forest)



**Figura 12.-Nivel de importancia de las variables a partir de árboles**

A diferencia de la correlación cuando usamos un estilo de random forest con 50 árboles las variables más importantes con respecto a nuestro evento vemos que ninguna variable sobresale demasiado por tanto(**Figura 12**) la información está muy distribuida entre varias variables pero también sabemos que existe mucha redundancia de información.

Out [45] : 36

utilizando un random forest para seleccionar las variables tenemos 36 seleccionadas bajo el corte de importancia mayor a 0.01.

La lista de Variables se puede observar en el Anexo6

## 15 ANOVA-Fvalue

Utilizaremos el método de f\_classif basado en la prueba estadística F para la selección de variables que sean relevantes, vamos a seleccionar aquellas cuyo p-value sea menor a 0.05 equivalente a un F-statistics mayor.

```
/home/abraham/anaconda3/lib/python3.6/site-packages/ipykernel/__main__.py:27: FutureWarning: IPython kernel is deprecated
```

Out [46]: 65

En este caso seleccionamos 65 variables.

La lista completa de variables se puede ver en el Anexo7

## 16 AUC-variables

Utilizaremos el método del área bajo la curva AUC o índice ROC para seleccionar aquellas variables que más discriminan la aceptación de las ofertas de préstamos esta técnica está altamente correlacionada con usar el índice de Gini, para este caso vamos a seleccionar aquellas variables cuyo  $AUC > 0.6$

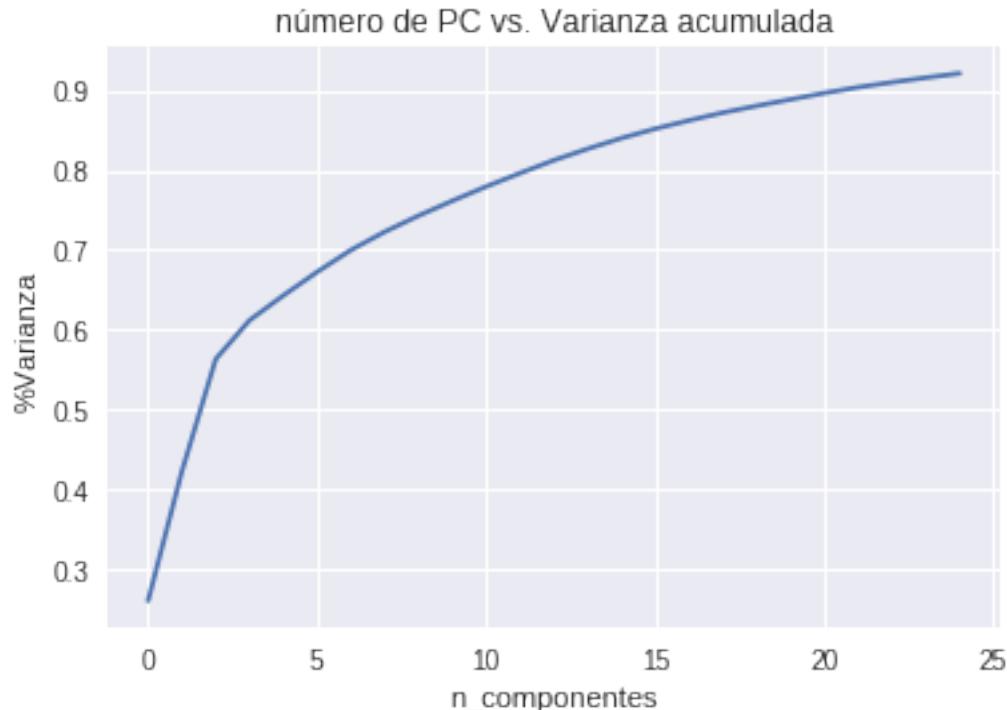
Out [49]: 43

Para este caso seleccionamos 43 variables.

La lista completa de variables la podemos ver en el Anexo8.

## 17 PCA

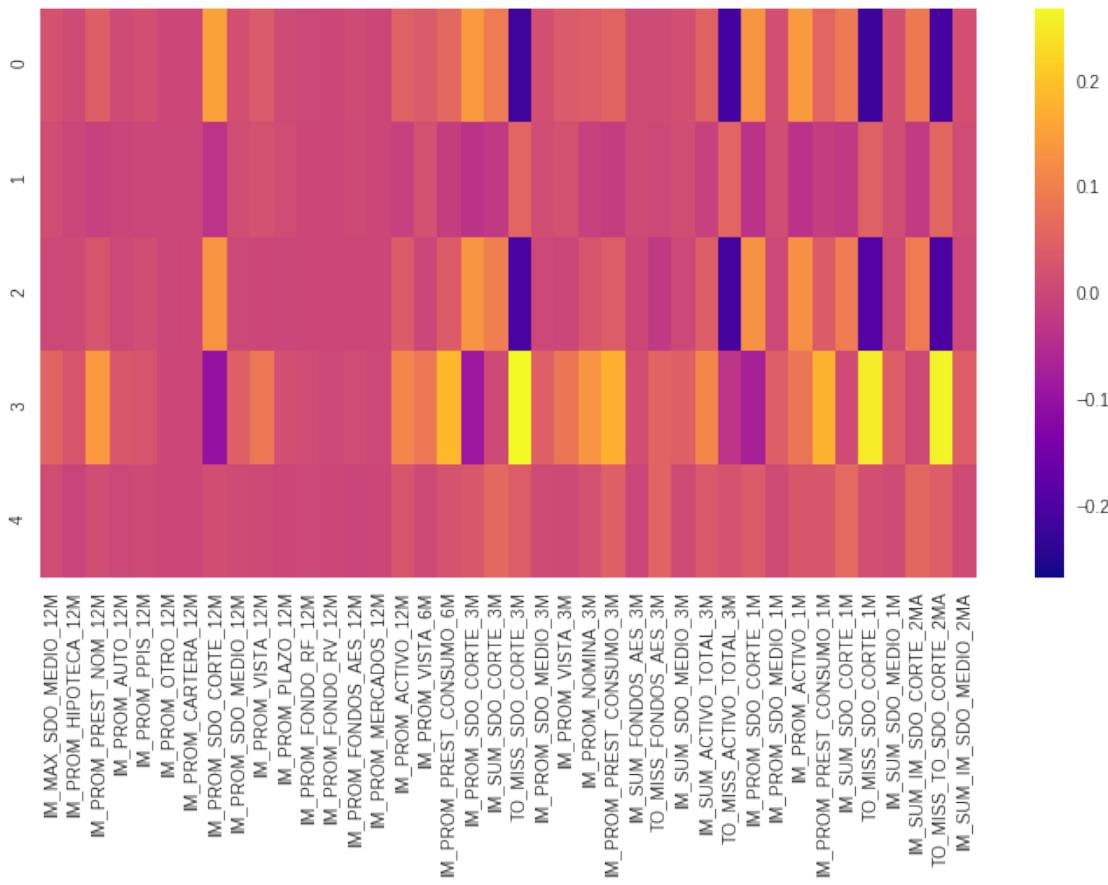
Utilizaremos el método de componentes principales para eliminar la redundancia de información y después vamos a verificar, utilizando las variables originales y las PCA, cuáles son las más importantes:



**Figura 13.- % de varianza explicada acumulada por el número de componentes escogidas**

Podemos observar que con 25 componentes podemos explicar más del 90% (Figura 13) de la varianza por tanto este es número de componentes que vamos a utilizar, entonces se agregarán al dataset y se seleccionarán las variables más importantes para modelar...

Out [51]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7f8e25546dd8>



**Figura 14.-Gráfico de influencia de cada variable en las primeras 5 componentes principales.**

El resto de componentes se pueden ver completas en el Anexo9.

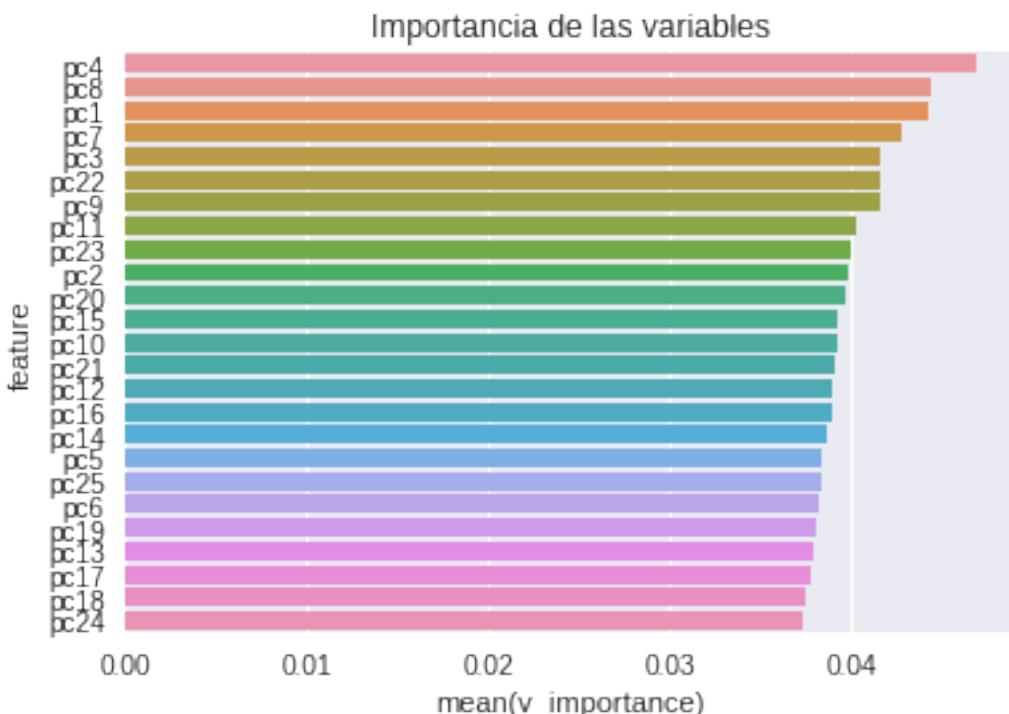
En la Figura 14 podemos observar las variables que más influyen en las primeras 5 componentes principales, por ejemplo para la componente 1 (PCA1.-se marca como 0 en la Figura 14) las variables más influyentes son 'IM\_SUM\_IM\_SDO\_MEDIO\_2MA', 'TO\_MISS\_SDO\_CORTE\_1M', 'TO\_MISS\_ACTIVITO\_TOTAL\_3M', 'TO\_MISS\_SDO\_CORTE\_3M', de forma negativa Y entre otras 'IM\_PROM\_SDO\_CORTE\_12M' de forma positiva entonces esta componente puede interpretarse como indicador de falta de información de saldos del cliente debido a su nula actividad en un periodo, entonces un valor mayor implicaría que el cte. reporta actividad casi todos los meses además de contar con el dato de 'IM\_PROM\_SDO\_CORTE\_12M'.

Otro ejemplo es la componente 4 (PCA4.-aparece como 3 en la Figura 14) vemos que las variables más influyentes son 'IM\_PREST\_CONSUMO\_6M', 'IM\_PREST\_CONSUMO\_1M',

TO\_MISS\_SDO\_CORTE\_1M, TO\_MISS\_TO\_SDO\_CORTE\_2MA de forma positiva e IM\_PROM\_SDO\_CORTE\_12M, IM\_PROM\_SDO\_CORTE\_1M, IM\_PROM\_SDO\_CORTE\_3M de forma negativa lo cual se puede interpretar como aquellos clientes que dado sus altos saldos de deuda de créditos dejan de tener actividad y por ello sus saldos de liquidez se ve afectado por tanto la componente habla de una necesidad financiera.

debido a que tenemos muchas componentes la interpretabilidad de cada una es bastante extensa para este trabajo mostraremos estas primeras 5.

**Aplicamos pca.transform a los datos de prueba como los ajustamos en el train para calcular las componentes como en el entrenamiento.**

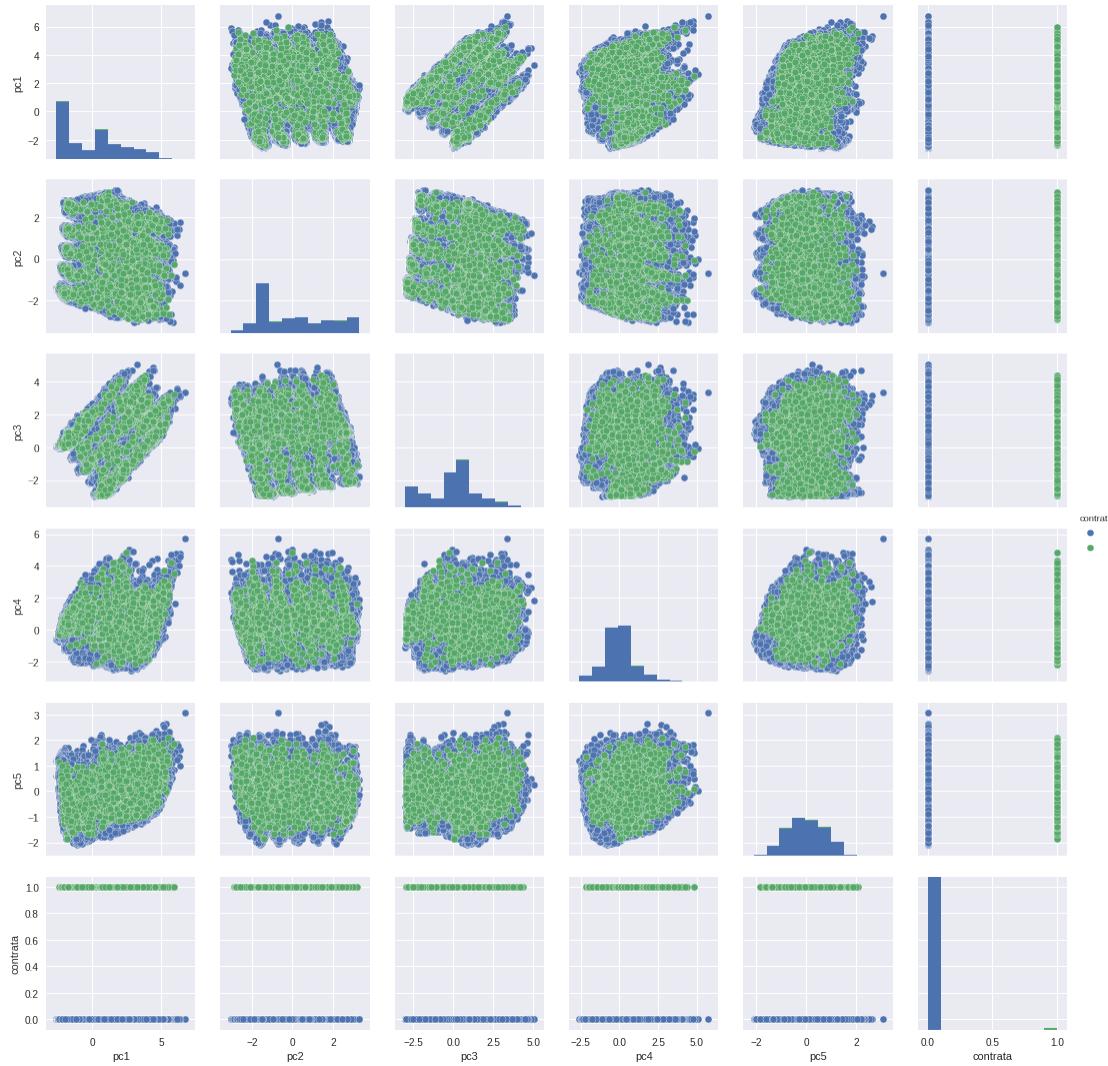


**Figura 15.- Importancia de las componentes principales usando el método de random forest de la Figura 12**

De acuerdo con las distintas selecciones de variables prácticamente con todas encontramos modelos que alcanzan curvas ROC AUC de más de 80 (Revisar Apéndice A.), pero en estricto sentido nos vamos a quedar con la selección de las primeras 25 componentes principales ya que de todas las selecciones es la que menor número de variables ocupa además de eliminar la redundancia de información entre ellas cosa que con otra forma de seleccionar los datos no sucede necesariamente además de que nuestro dataset tiene variables muy correlacionadas.

**Distribución de las primeras 5 componentes principales separadas por la variable target**

Out [53]: <seaborn.axisgrid.PairGrid at 0x7f8e256113c8>



**Figura 16.-Pairplot de las primeras 5 componentes principales.**

De acuerdo con el resultado vemos que el valor de la importancia de las variables mejora utilizando las 25 componentes principales además sólo con estas vamos a modelar.

Por otro lado, podemos ver que la distribución de las componentes es aprox normal(**Figura 16**) por lo que con esto buscamos que se aprovechen mejor los algoritmos.

## 18 Modelos

Para la parte del modelado vamos a probar con 5 modelos distintos utilizaremos Naive Bayes, red neuronal, un svm ,una regresión logística y un random forest con esto vamos a comparar las bajo las distintas métricas y escoger el mejor modelo de acuerdo con la solución al problema planteado.

**Todos los resultados y métricas obtenidas en esta sección están evaluados con la partición de TEST obtenida del set de entrenamiento definida al principio del problema(80%),es decir del set de entrenamiento vamos a hacer una partición de train y test tomando el 20% de los**

**datos para test o validación, así como los cross-validation, recordar que los datos de test finales(elegidos al principio del reporte) los vamos a tocar hasta la evaluación final.**

## 19 SVM

Para las maquinas de soporte vectorial vamos a probar con el kernel lineal, pero primero vamos a hacer un grid search para elegir los mejores parámetros del modelo. El **gridsearch** trata de probar diferentes combinaciones de parámetros hasta encontrar los ‘mejores’ basados en alguna métrica, este proceso funciona haciendo validación cruzada para obtener los resultados y compararlos. **Cómo lo hicimos en clase con la regularización** encontrando la mejor  $\lambda$ .

La métrica que vamos a ocupar en este y todos los algoritmos será AUC ya que de acuerdo con el contexto del problema nos dirá que parámetros nos dan mejor precisión y recall sin todavía pensar en el punto de corte.

Nota: se ajustan parámetros con C’s pequeñas [0.1, 0.5, 1.0, 1.5] y  $\gamma$ =[0.1, 0.01, 0.001, 0.0001] dado que usar C’s mayores implica tiempo de procesamiento muy alto por el tamaño de la base.

Entonces los Mejores Parámetros para svm son:

{‘C’: 1.5, ‘gamma’: 0.1, ‘kernel’: ‘linear’}

El output completo se puede ver en el Apéndice B punto 1.

Lo que significan estos parámetros es que en promedio de  $k = 3$  iteraciones(**esto es un 3-fold cross validation**) tienen el mejor AUC.

el mejor AUC promedio del cross validation: 0.626472402186

```
Out[61]: SVC(C=1.5, cache_size=200, class_weight=None, coef0=0.0,
              decision_function_shape='ovr', degree=3, gamma=0.1, kernel='linear',
              max_iter=-1, probability=False, random_state=None, shrinking=True,
              tol=0.001, verbose=False)
```

Los parámetros que maximizan AUC son: **Algoritmo SVM, complejidad: kernel lineal gamma=.1, C=1.5**

	precision	recall	f1-score	support
0	0.99	1.00	0.99	27881
1	0.00	0.00	0.00	332
avg / total	0.98	0.99	0.98	28213
[[27881 0]				[ 332 0]]

```
/home/abraham/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.  
'precision', 'predicted', average, warn_for)
```

Podemos ver que el svm con los parámetros utilizados no arroja discriminación entre las clases esto es debido a que la densidad de nuestro problema es muy baja y también que el punto de corte por **default** es de 0.5 entonces este modelo no scorea ninguna observación arriba de ese punto, por tanto las métricas de precision, recall y f1-score no nos ayudan a ver el desempeño verdadero de este modelo, más adelante veremos su curva ROC.

## 20 Regresión Logística

En este algoritmo también usaremos gridsearch para encontrar los mejores parámetros al igual que hicimos en SVM basado también en AUC como métrica de referencia.

vamos a probar los parámetros `penalty=['l2','l1']` recordar que estos hacen referencia la tipo de regularización de los coeficientes  $l1$  elimina variables llevando coeficientes a 0 para evitar overfitting y también usaremos el parámetro  $C = [0.001, 0.01, 0.1, 1, 10]$  es nuestra  $\lambda$

Entonces los Mejores Parámetros para la regresión Logística son:

`{'C': 0.01, 'penalty': 'l2'}`

El output completo se puede ver en el Apéndice B punto 2.

el mejor AUC promedio del cross validation: 0.812388810696

```
Out[64]: LogisticRegression(C=0.01, class_weight=None, dual=False, fit_intercept=True,
                             intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                             penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                             verbose=0, warm_start=False)
```

Los parámetros que maximizan AUC son: **Algoritmo Regresión Logística, complejidad: C=0.01 penalty=l2** l2 es la regularización Ridge.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	27881
1	0.00	0.00	0.00	332
avg / total	0.98	0.99	0.98	28213
[ [27881 0]				
[ 332 0]]				

```
/home/abraham/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.py:455: UserWarning: F-beta score requires precision and recall to have been computed. Need to call report_score first.
  'precision', 'predicted', average, warn_for)
```

La regresión logística tampoco califica ningún caso arriba de 0.5, veremos su curva ROC.

## 21 Naive Bayes

**Algoritmo Naive Bayes, complejidad: Gaussiano** en este caso el algoritmo sólo usa el parámetro prior que no es otra cosa que la densidad de 1's(1.17%) que se leen de los datos por tanto este parámetro se queda así.

	precision	recall	f1-score	support
0	0.99	0.97	0.98	27881
1	0.04	0.12	0.06	332
avg / total	0.98	0.96	0.97	28213
	[ [26978 903] [ 292 40] ]			

El algoritmo Naive Bayes, de acuerdo con la matriz de confusión, clasifica mas de 900 casos(clasificados arriba de 0.5) como 1's o contrataciones de los cuales la tasa de respuesta(efectividad) o precisión es de 4% lo cual mejora más de 3 veces la efectividad o densidad original (1.17%) este dato suena muy bien pero, si vemos el recall apenas capturamos el 12% de las contrataciones o clientes que aceptan las ofertas recordar que este 12% son 40 clientes. mismo caso veremos la curva ROC para definir el desempeño real del algoritmo.

## 22 Redes Neuronales

Para las redes neuronales crearemos 3 modelos con con número de neuronas y funciones de activación distintas.

En esta primera Red vamos a dejar fijo el número de capas ocultas=1 y vamos a probar con las funciones de activación 'relu','tanh','logistic', neuronas=[3,10,50,100] ....

Entonces los Mejores Parámetros para la mlp1 son:

```
{'activation': 'relu', 'hidden_layer_sizes': (50,), 'max_iter': 40000, 'random_state': 0, 'verbose': 0}
```

El output completo se puede ver en el Apéndice B punto 3.

```
el mejor AUC promedio del cross validation: 0.836584934903
```

```
Out[68]: MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9  
beta_2=0.999, early_stopping=False, epsilon=1e-08,  
hidden_layer_sizes=(50,), learning_rate='constant',  
learning_rate_init=0.001, max_iter=40000, momentum=0.9,  
nesterovs_momentum=True, power_t=0.5, random_state=0, shuffle=True,  
solver='adam', tol=0.0001, validation_fraction=0.1, verbose=0,  
warm_start=False)
```

**Algoritmo Red Neuronal, complejidad: función de activación relu,neuronas=50, capas ocultas=1**

	precision	recall	f1-score	support
0	0.99	1.00	0.99	27881
1	1.00	0.03	0.06	332
avg / total	0.99	0.99	0.98	28213
[ [27881 0]				
[ 321 11]]				

El modelo mlp1(Multilayer perceptron) con 50 neuronas y función de activación relu califica 11 casos arriba de 0.5 por tanto los clasifica como 1's, con esto tenemos una precisión de 100% lo cual implica que mejoramos la densidad original más de 95 veces pero, nuestro recall es de apenas 3% o sea 11 clientes, estos datos, al igual que en los casos anteriores no son métricas contundentes para medir el desempeño del modelo ya que están basados en el punto de corte 0.5, veremos la curva ROC adelante.

## 23 mlp2

Para la segunda Red Neuronal vamos probar modificando el número de capas y neuronas ocultas, usando la función de activación logística hidden\_layers=[(12,2),(15,3),(20,4),(25,2),(50,3),(100,)] vamos a escoger el mejor par de hidden layers y capas ocultas que de la mejor AUC... para obtener los parámetros de la red.

Entonces los Mejores Parámetros para la mlp2 son:

```
{'activation': 'logistic', 'hidden_layer_sizes': (20, 4), 'max_iter': 40000, 'random_state': 0, 'verbose': 0}
```

El output completo se puede ver en el Apéndice B punto 4.

el mejor AUC promedio del cross validation: 0.836937675666

```
Out[71]: MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08,
hidden_layer_sizes=(20, 4), learning_rate='constant',
learning_rate_init=0.001, max_iter=40000, momentum=0.9,
nesterovs_momentum=True, power_t=0.5, random_state=0, shuffle=True,
solver='adam', tol=0.0001, validation_fraction=0.1, verbose=0,
warm_start=False)
```

**Algoritmo Red Neuronal(mlp2), complejidad: función de activación logística, neuronas=(20,4)** en este caso tenemos una red profunda en este punto tal vez esta función ya no es tan interpretable pero ojo nuestro problema de negocio implica buscar la mejor efectividad o precision de contrataciones.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	27881

```

1          0.00      0.00      0.00      332
avg / total      0.98      0.99      0.98      28213
[[27881      0]
 [ 332      0]]

```

```
/home/abraham/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.
'precision', 'predicted', average, warn_for)
```

El modelo mlp2(Multilayer perceptron) con (20,4) neuronas y función de activación logística no clasifica casos como 1's, igual veremos su desempeño en la curva ROC y si resultara ganador revisaríamos el umbral adecuado.

## 24 mlp3

Para la 3a Red Neuronal vamos probar modificando el número de capas y neuronas ocultas, usando la función de activación tanh(Tangente Hiperbólica) hidden\_layers=[(12,4),(15,3),(20,4),(25,3),(50,3),(50,2)] vamos a escoger el mejor par de hidden layers y capas ocultas y también el número máximo de iteraciones=[400,4000,40000,100000] que de la mejor AUC... para obtener los paraméetros de la red.

Entonces los Mejores Parámetros para la mlp3 son: {'activation': 'tanh', 'hidden\_layer\_sizes': (15, 3), 'max\_iter': 400, 'random\_state': 0, 'verbose': 0}. El output completo se puede ver en el Apéndice B punto 5.

```
el mejor AUC promedio del cross validation: 0.835567762301
```

```
Out[74]: MLPClassifier(activation='tanh', alpha=0.0001, batch_size='auto', beta_1=0
beta_2=0.999, early_stopping=False, epsilon=1e-08,
hidden_layer_sizes=(15, 3), learning_rate='constant',
learning_rate_init=0.001, max_iter=400, momentum=0.9,
nesterovs_momentum=True, power_t=0.5, random_state=0, shuffle=True,
solver='adam', tol=0.0001, validation_fraction=0.1, verbose=0,
warm_start=False)
```

**Algoritmo Red Neuronal(mlp3), complejidad: función de activación tangente hip,neuronas=15, capas ocultas=3 y 400 iteraciones**

	precision	recall	f1-score	support
0	0.99	1.00	0.99	27881
1	0.91	0.03	0.06	332
avg / total	0.99	0.99	0.98	28213

```
[ [27880      1]  
[  322     10]]
```

El modelo mlp3(Multilayer perceptron) con 15 neuronas, 3 capas ocultas y función de activación tangente hiperbólica y califica 11 casos arriba de 0.5 por tanto los clasifica como 1's, on esto tenemos una precisión de 91% lo cual implica que mejoramos la densidad original más de 89 veces pero, nuestro recall es de apenas 3% o sea 10 clientes, estos datos, al igual que en los casos anteriores no son métricas contundentes para medir el desempeño del modelo ya que están basados en el punto de corte 0.5, veremos la curva ROC adelante.

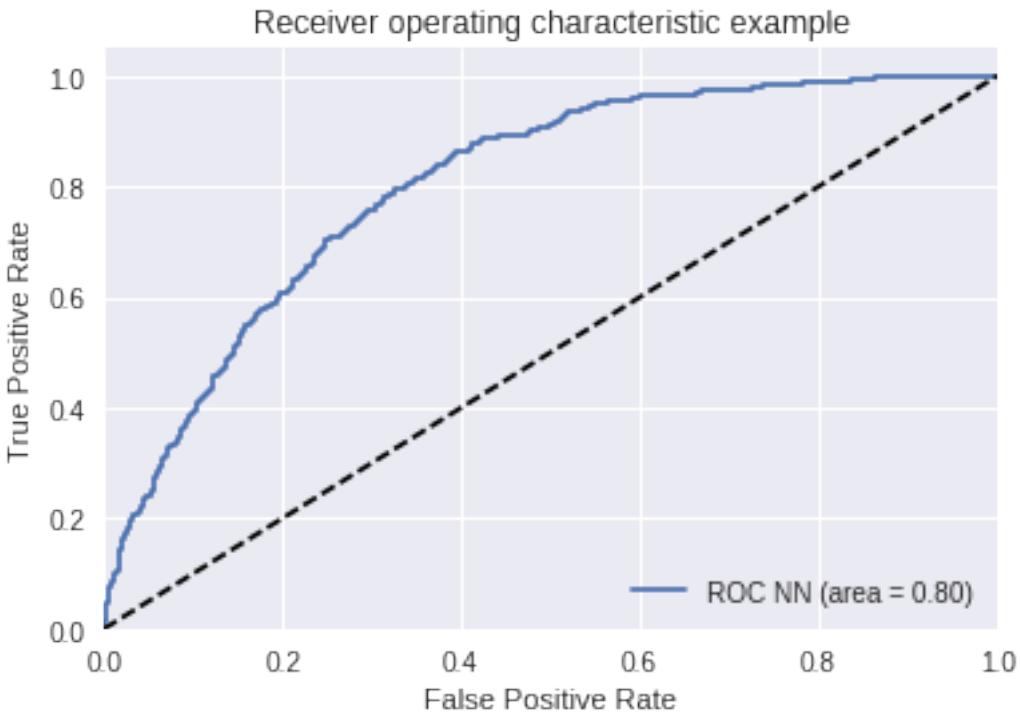
```
0.988215
```

```
Out[12]: 0.0
```

	precision	recall	f1-score	support
0	0.99	1.00	0.99	27881
1	0.00	0.00	0.00	332
avg / total	0.98	0.99	0.98	28213

```
[ [27881      0]  
[  332     0]]
```

```
/home/abraham/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.  
'precision', 'predicted', average, warn_for)
```



## 25 Random Forest

Para el Random Forest vamos a probar con números distintos de árboles entonces vamos a escoger `n_estimators=[10,15,35,50,100,200]` (número de árboles) para ver con cuantos maximizamos la AUC promedio, junto con el número de particiones dentro de cada rama=[2,4,8].

Entonces los Mejores Parámetros para el Random Forest son:

```
{'criterion': 'gini', 'min_samples_split': 8, 'n_estimators': 200}
```

El output completo se puede ver en el Apéndice B punto 6.

el mejor AUC promedio del cross validation: 0.789080113128

```
Out[77]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                 max_depth=None, max_features='auto', max_leaf_nodes=None,
                                 min_impurity_decrease=0.0, min_impurity_split=None,
                                 min_samples_leaf=1, min_samples_split=8,
                                 min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=1,
                                 oob_score=False, random_state=None, verbose=0,
                                 warm_start=False)
```

**Algoritmo Random Forest, complejidad:** 1000 árboles y 8 particiones por nodo escogimos el criterio de partición el estadístico gini dado que este método crea grupos homogéneos(Ganancia de información, queremos saber que tanto se diferencian estos clientes del general)

en este caso nuestros 1's y 0's que son los contratos y al final queremos clientes homogéneos para las campañas.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	27881
1	1.00	0.02	0.04	332
avg / total	0.99	0.99	0.98	28213
	[ [ 27881      0 ] [    326      6 ] ]			

El Random Forest clasifica 6 clientes como 1's donde tenemos una precisión de 100% pero un recall o captura de eventos de apenas el 2% veremos las curvas ROC.

## 26 Selección del Modelo

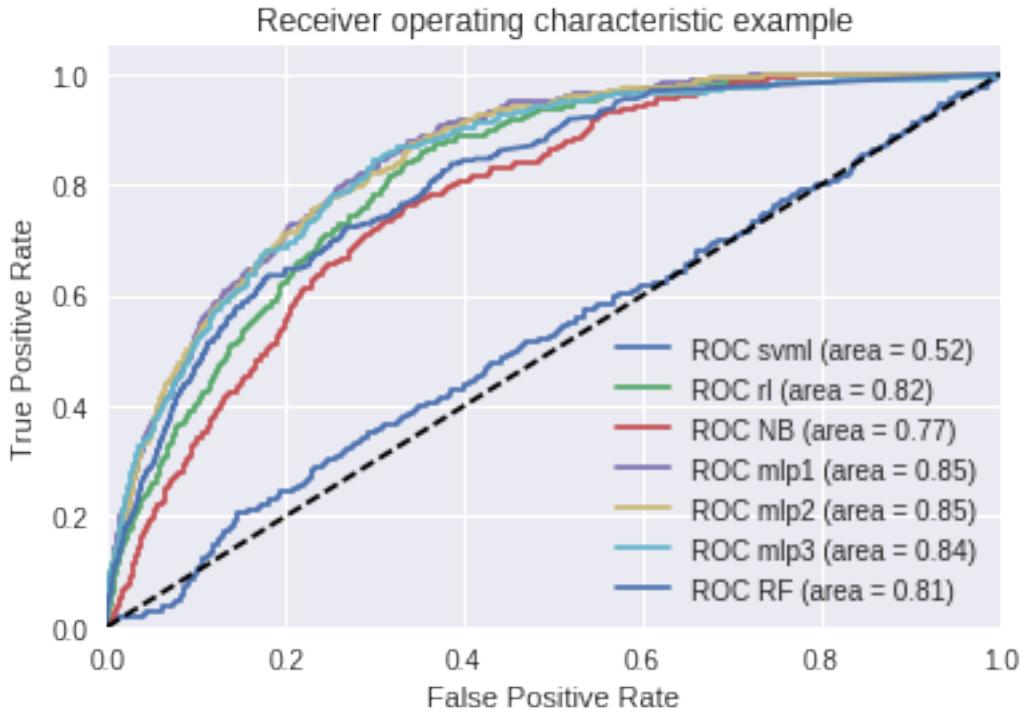
De acuerdo con las predicciones y el performance de cada modelo vamos a comparar sus distintas métricas, en particular para seleccionar el modelo ganador vamos a utilizar la **métrica AUC o índice ROC ya que este dato no depende de ningún punto de corte como serían en accuracy, F1-score** y los que se derivan de la matriz de confusión, entonces aquel modelo que capture más eventos con una menor tasa de falsos positivos será el más adecuado para resolver nuestro problema de negocio.

**Nota: en este punto vamos a comparar las curvas ROC de los modelos con la partición de TEST QUE SE HIZO DEL 80% de train, el set final de validación que definimos al principio del problema se utilizará después de los siguientes pasos:**

1.-comparación de modelos con test que viene del set de train que viene definido al principio del problema(80%).

2.-haremos un cross validation con 30 iteraciones de los modelos que se definieron y decidiremos cual es el mejor de acuerdo con el promedio de sus curvas ROC.

posterior a estos pasos probaremos el mejor modelo en la partición test definida al principio del problema y ahí verificaremos si el modelo ganador en realidad es el que mejor ajusta.

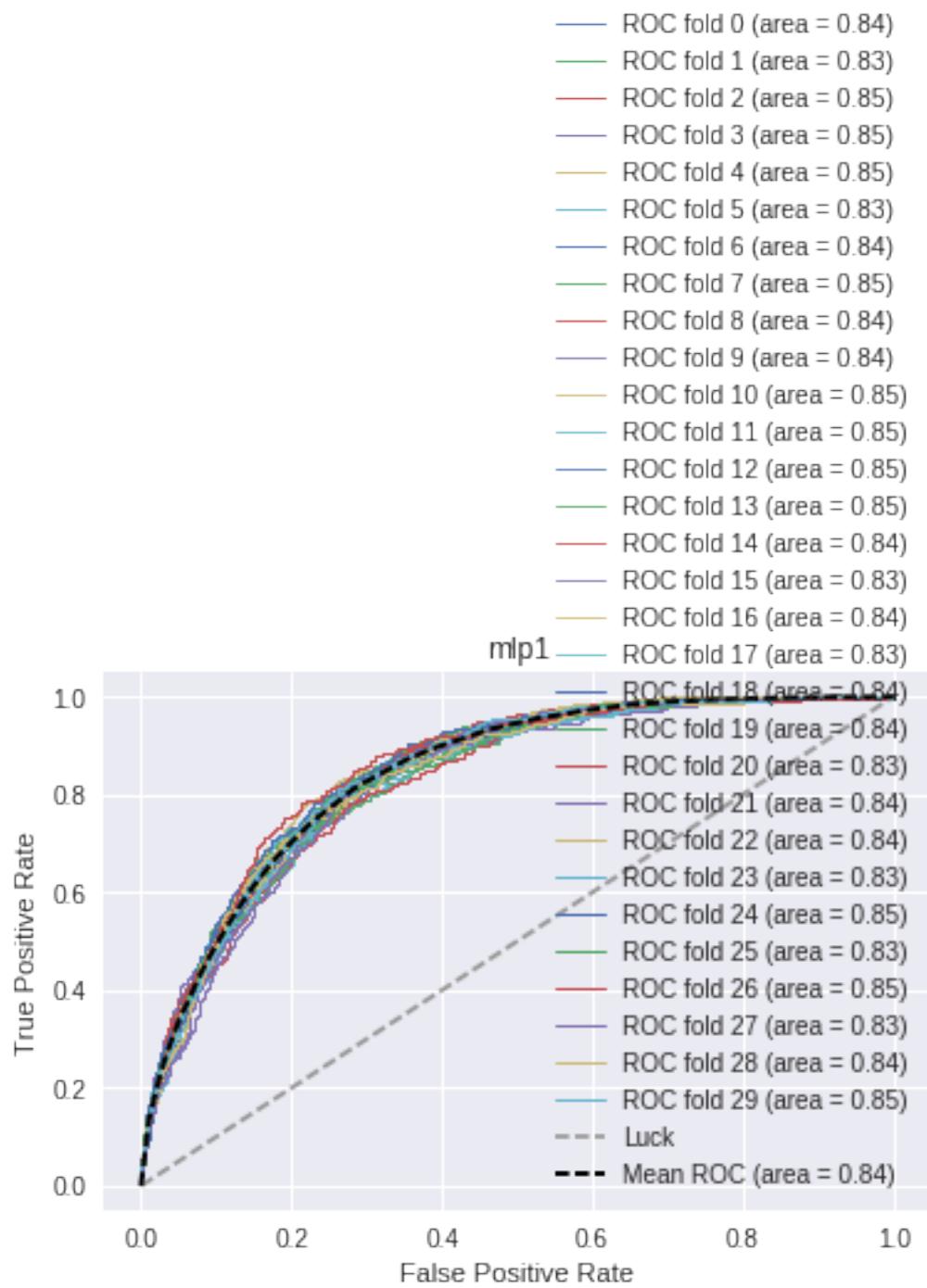


**Figura 17.- Comparativo de curvas ROC y su área bajo la curva(AUC).**

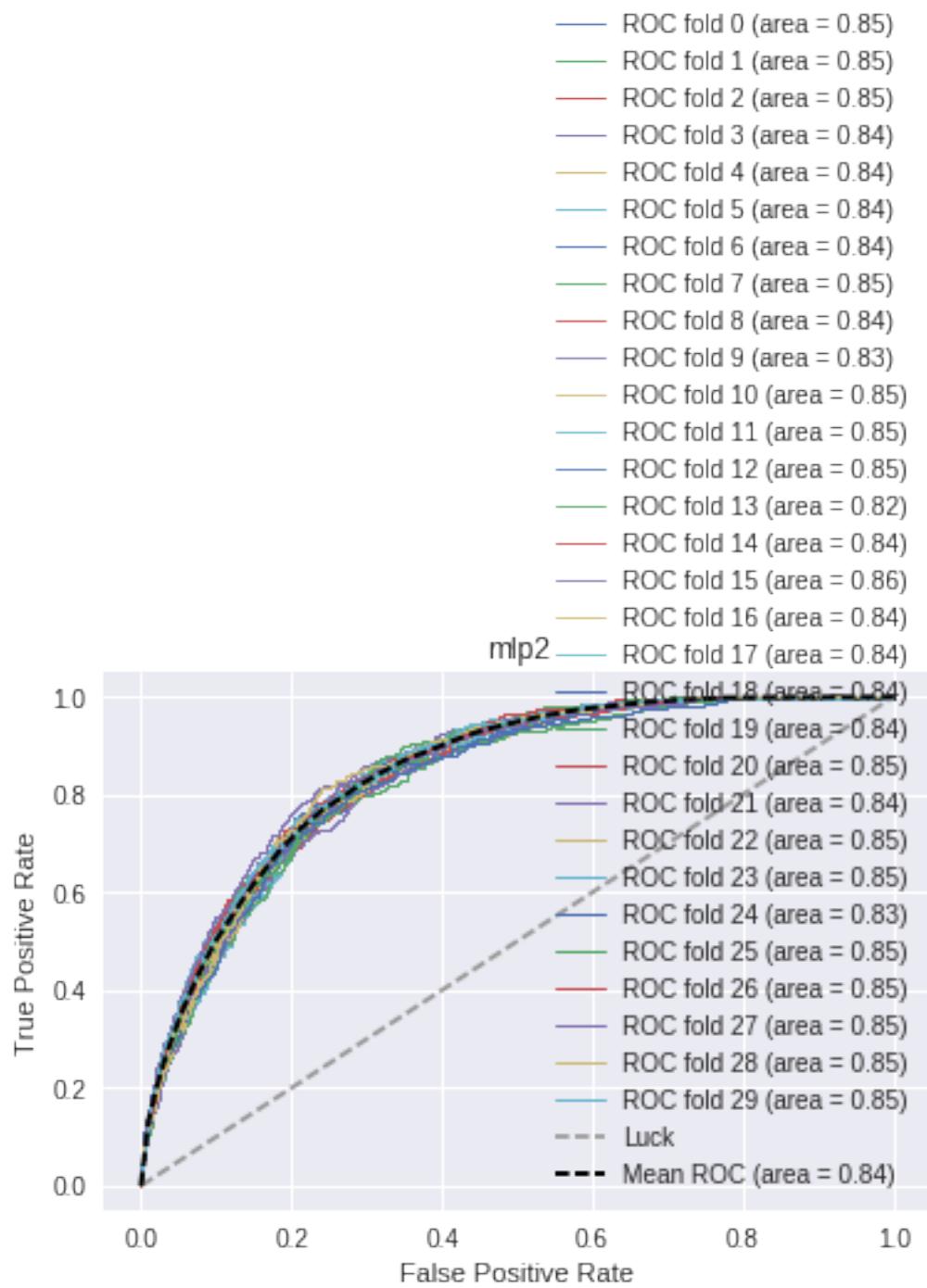
De acuerdo con la gráfica comparativa de cuvas ROC el mejor modelo es el mlp1 y mlp2 (**Figura 17**) ya que el índice ROC o AUC=0.85 de las redes neuronales es mayor que los del resto.

## 27 Cross-validation

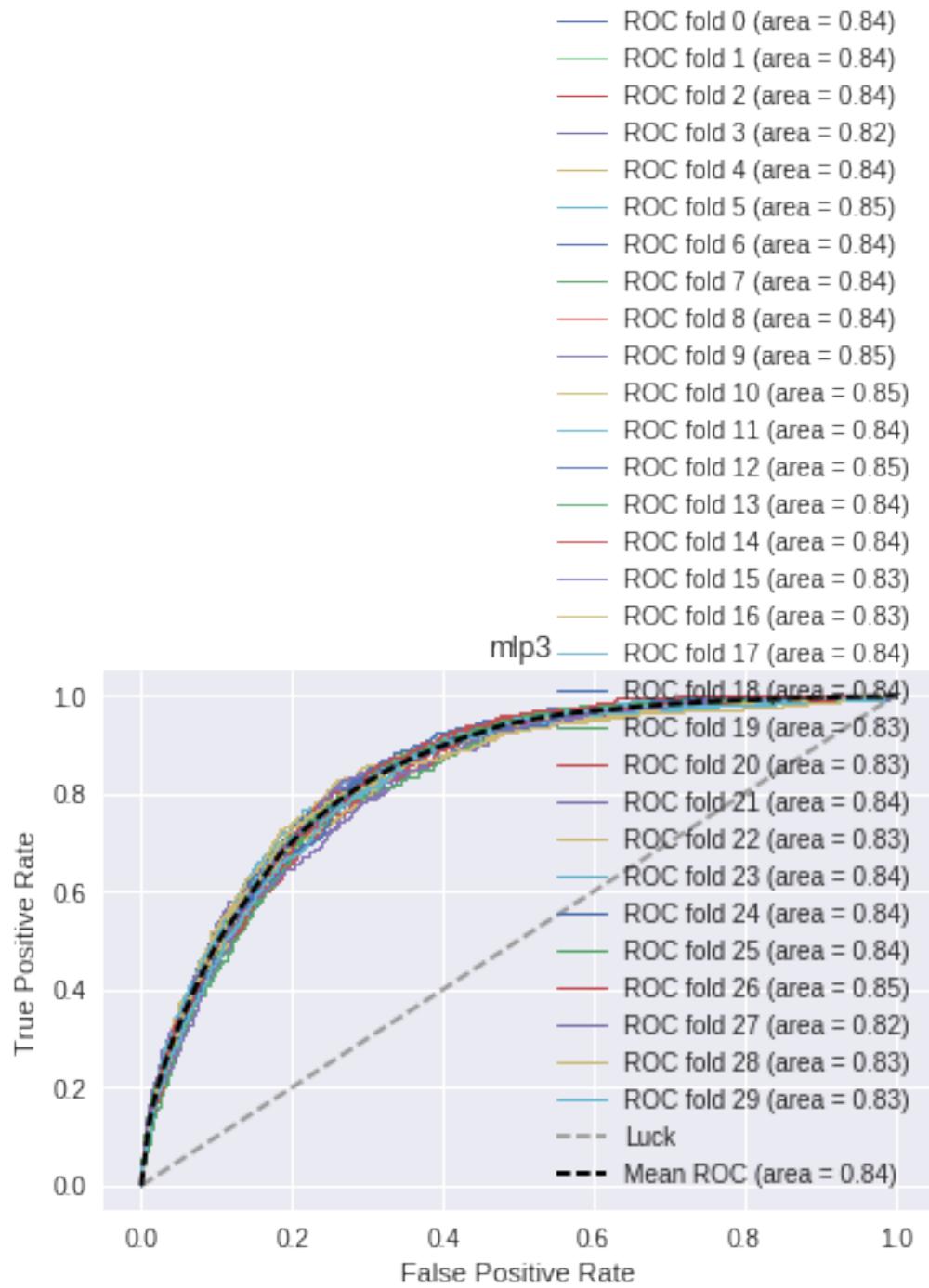
Con los resultados anteriores hemos seleccionado nuestro modelo ganador, pero hay que recordar que el performance del modelo o en otras palabras el índice ROC están probados en **una de muchas particiones posibles de datos de train y test**, entonces puede suceder que el mejor modelo en esta muestra no sea el mejor en alguna otra, para revisar la verosimilitud de resultado vamos hacer un proceso de cross validation repetido 30 veces al menos, seleccionando distintos datos de entrenamiento y test, con las mismas proporciones de partición i.e. `test_size` de 20%, de tal modo que este proceso se llama shuffle split. Con estas 30 iteraciones de partición podemos analizar la distribución del índice roc de cada uno de los modelos entrenados en cada iteración de tal forma que podamos ver cual de ellos es el más estable o confiable para ser seleccionado y finalmente, nos quedaremos con aquel modelo que mejor performance presente a lo largo de las iteraciones y ese será nuestro modelo final. `Out [88]` :



Out [89] :



Out [90] :



**Figura 18.- Desempeño de curvas ROC por cada modelo en cada una de las 30 iteraciones.**

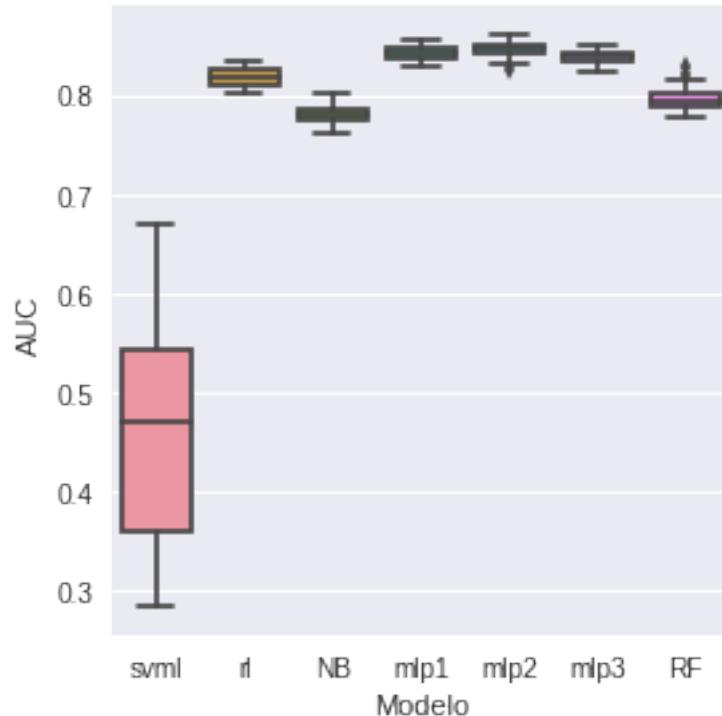
Podemos ver que para los modelos mlp1-mlp3 el AUC promedio=0.84.

El resto de los modelos pueden verse en el Anexo10.

De acuerdo con la Figura 18 haciendo las distintas iteraciones con excepción del modelo svml, el resto son muy estables en cuanto a la distribución del AUC, ahora bajo el AUC promedio los

modelos mlp1-mlp3 tienen el mismo dato, y para poder diferenciarlos vamos a ver un boxplot de su distribución:

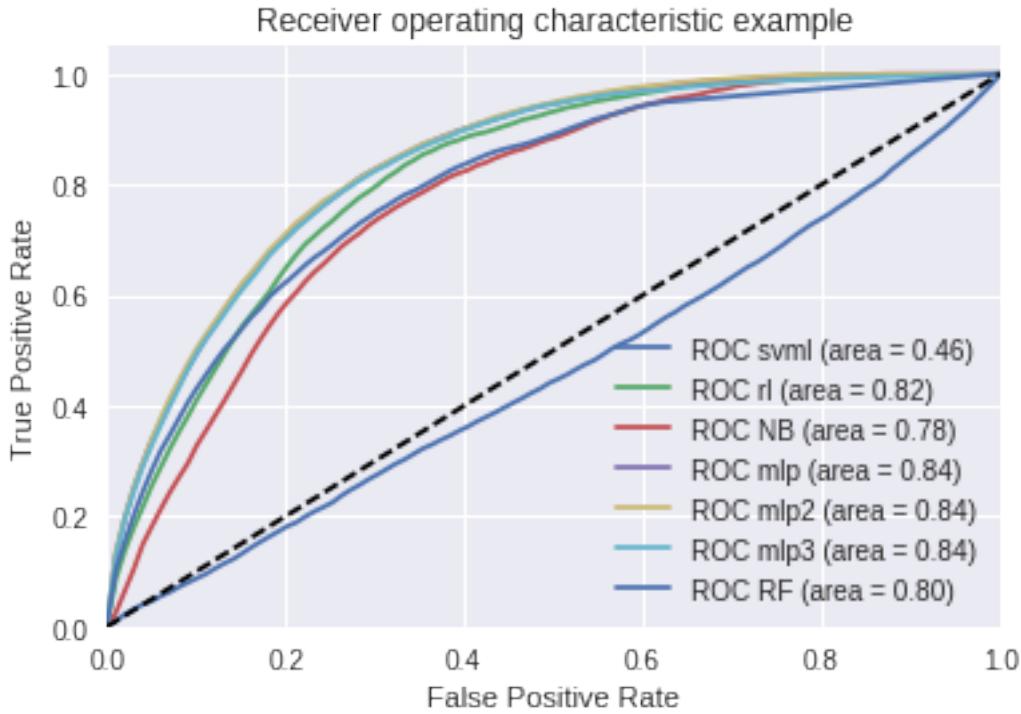
Out[100]: <seaborn.axisgrid.FacetGrid at 0x7f8ded556358>



**Figura 19.- Box-plot de la distribución de cada modelo con respecto a su AUC**

Modelo mlp1 es el más compacto en cuanto rango de AUC aunque no es muy distante de mlp2 y mlp3.

En general, con excepción de svml, todos los modelos son muy compactos en su distribución de AUC lo que habla de la estabilidad proporcionada en el proceso de validación cruzada.



**Figura 20.- Gráfico de curvas ROC promedio de cada modelo resultantes de Figura 18**

Con las gráficas de caja y brazos de las AUC en cada iteración vemos que de forma marginal el mejor modelo o el más estable es la red neuronal mlp1 (**complejidad: función de activación relu,neuronas=50, capas ocultas=1**)que cuenta con un índice **promedio** de 0.84 y con una mediana similar,además si vemos la **Figura 19** podemos observar que el modelo mlp1 es ligeramente más compacto en su distribución es decir que el rango del área bajo la curva o AUC es más cercano a cero que los demás y por otro lado la caja y brazos de este modelo está ligeramente más abajo que mlp2 aunque este es menos compacto, pero tiene AUC's mayores en las iteraciones con respecto al resto lo cual también se confirma con la **Figura 20** donde vemos las AUC's promedio de cada modelo, por tanto es el modelo con mejor desempeño aunque su diferencia o ganancia es marginal con respecto a las otros 2 modelos del mismo tipo mlp2 y mlp3.

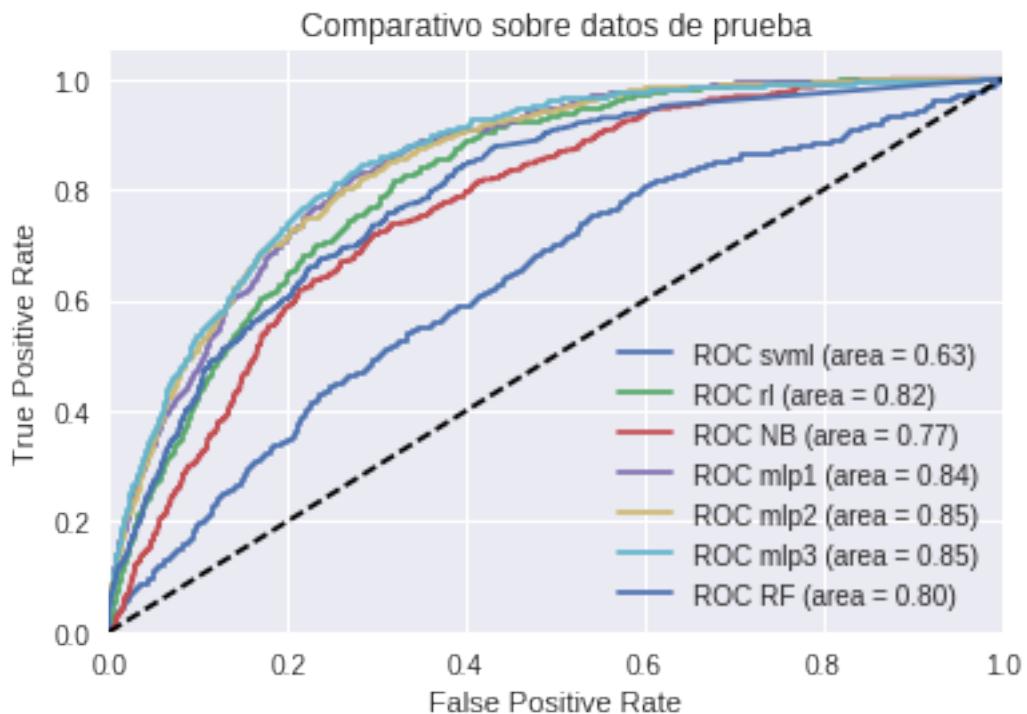
**Métrica de Éxito AUC** La razón por la que nuestra métrica de éxito es la curva ROC es debido a que métricas como Precisión, accuracy, recall, F1-Score, tasa de error de clasificación dependen del punto de corte de la probabilidad o score que cada uno de los modelos puede generar, por default los puntos de corte para la creación de todas estas métricas es 0.5, pero en este caso tenemos pocas observaciones calificadas arriba de este score lo que por default en algunos modelos tuvimos precisiones y recalls en 0 debido a la baja densidad de 1's que tenemos en nuestro dataset **1.17%** por tanto el área bajo la curva roc nos habla de que con nuestro modelo mlp1 podemos capturar más eventos o contrataciones con un menor % de falsos positivos lo cual se adecúa perfecto a nuestro caso de negocio ya que lo que buscamos es mejorar la tasa de respuesta con un menor universo o en otras palabras reducción de costos para después maximizar la utilidad de colocar préstamos.

Entonces, lo único que falta por confirmar es si los modelos se comportan igual en nuestros datos de prueba(20%) definidos al principio del problema y definir con esto el champion model.

## 28 Champion Model

Una vez revisado el preformance de cada modelo a través del cross validation y la **Figura 19** vamos a utilizar nuestros datos originales de TEST para comparar los distintos modelos y decidir el que vamos a elegir y las razones.

Primero, nuestro Dataset de prueba tiene una densidad de 1.17%, esta será nuestra métrica a mejorar..



**Figura 20.1.- Gráfico de curvas ROC en nuestros datos de TEST original**

Con la **Figura 20.1** podemos ver de acuerdo al desempeño de cada modelo en nuestros datos de Test originales que el **Champion Model** es mlp2 o mlp3, aunque vamos a elegir al segundo por menor complejidad, es decir recordemos que mlp2 tiene 20 neuronas y 4 capas ocultas y mlp3 15,3 respectivamente, también podríamos elegir mlp1 que tuvo una roc ligeramente menor pero dentro de lo esperado. En nuestro problema de negocio el objetivo es encontrar el mejor algoritmo no el más interpretable(Polémico y debatible!) concientes de que una red neuronal no es interpretable elegimos esta como nuestro mejor algoritmo mlp3

## 29 Selección del umbral de Clasificación

Densidad original 1.17 %

Estadístico ks

Una forma de seleccionar el umbral es basado en el estadístico KS(Kolmogorov-Smirnov) que es una métrica para medir la predictibilidad de un modelo con respuestas binarias, como nuestro caso, es una métrica muy utilizada en temas de riesgos y marketing.

$$KS = \max_{1 < i < n} |F_n(x_i) - F_0(x_i)|$$

donde:

- $x_i$  es el i-ésimo valor observado en la muestra (cuyos valores se han ordenado previamente de menor a mayor).

- $F_n(x_i)$  es un estimador de la probabilidad de observar valores menores o iguales que  $x_i$ .

- $F_0(x_i)$  es la probabilidad de observar valores menores o iguales que  $x_i$  cuando  $H_0$  es cierta.

De acuerdo con la definición anterior podemos pensar a  $F_n(x_i)$  y  $F_0(x_i)$  como los valores de recall o tasa de verdaderos positivos y tasa de falsos positivos respectivamente entonces el estadístico KS es la mayor o máxima diferencia absoluta entre estas 2 distribuciones, es decir es donde la diferencia entre la proporción acumulada de falsos positivos y verdaderos positivos es máxima, entonces para seleccionar el mejor umbral hay que encontrar el bucket del score o probabilidad donde se encuentra esta diferencia y la **Figura 21** nos puede mostrar esto.

Entonces, nuestro KS se da en el punto (*FalsoPositivo, VeraderoPos*) = (0.27, 0.83).

### Construcción de la tabla de distribución de score para encontrar el mejor umbral

`Out[121]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8dedf043c8>`

min_scr	max_scr	contrata	no_contrata	ks	max_ks	Precisión	Recall	cum_FP	%Universo	LIFT
0.0627	0.8041	123	1288	26.16		8.72%	29.85%	3.70%	4.00%	7.46
0.0422	0.0627	67	1344	38.57		6.73%	46.12%	7.55%	8.00%	5.76
0.0313	0.0422	41	1369	44.59		5.46%	56.07%	11.48%	12.00%	4.67
0.024	0.0313	34	1377	48.89		4.70%	64.32%	15.43%	16.00%	4.02
0.0181	0.024	35	1375	53.44		4.25%	72.82%	19.38%	20.00%	3.64
0.0136	0.0181	24	1387	55.29		3.83%	78.64%	23.35%	24.00%	3.28
0.0104	0.0136	17	1394	55.41	<----	3.45%	82.77%	27.35%	28.00%	2.96
0.008	0.0103	13	1397	54.56		3.14%	85.92%	31.36%	32.00%	2.69
0.0063	0.008	10	1401	52.97		2.87%	88.35%	35.38%	36.00%	2.45
0.005	0.0063	12	1398	51.87		2.67%	91.26%	39.39%	40.00%	2.28
0.0041	0.005	7	1404	49.54		2.47%	92.96%	43.42%	44.00%	2.11
0.0034	0.0041	7	1404	47.21		2.30%	94.66%	47.45%	48.00%	1.97
0.0029	0.0034	8	1402	45.13		2.17%	96.60%	51.47%	52.00%	1.86
0.0025	0.0029	0	1411	41.08		2.02%	96.60%	55.52%	56.00%	1.73
0.0022	0.0025	4	1406	38.02		1.90%	97.57%	59.55%	60.00%	1.63
0.002	0.0022	2	1409	34.46		1.79%	98.06%	63.60%	64.00%	1.53
0.0018	0.002	1	1410	30.66		1.69%	98.30%	67.64%	68.00%	1.45
0.0016	0.0018	1	1409	26.86		1.60%	98.54%	71.68%	72.00%	1.37
0.0015	0.0016	1	1410	23.06		1.52%	98.79%	75.73%	76.00%	1.3
0.0014	0.0015	0	1410	19.01		1.44%	98.79%	79.78%	80.00%	1.24
0.0013	0.0014	2	1409	15.45		1.38%	99.27%	83.82%	84.00%	1.18
0.0013	0.0013	1	1409	11.65		1.32%	99.51%	87.86%	88.00%	1.13
0.0012	0.0013	0	1411	7.61		1.26%	99.51%	91.91%	92.00%	1.08
0.0011	0.0012	0	1411	3.56		1.21%	99.51%	95.96%	96.00%	1.04
0.0009	0.0012	2	1409	0.0		1.17%	100.00%	100.00%	100.00%	1.0

**Figura 21.- Tabla de rangos de umbrales**

La tabla esta particionada en 25 buckets por intervalos de score cada bucket representa el 4% de los datos ordenados por score de forma descendente, entonces con esta tabla podemos ver la implicación de seleccionar distintos umbrales en términos de precisión, recall etc.

-Punto de corte K-S: 0.0104

-Mejora del modelo: 2.96 veces

-Precisión: 3.45%

-Recall: 82.77%

-% Universo seleccionado: 28.00%

-AUC: 0.844

Podemos observar que el punto de corte basado en el estadístico KS es 0.0104 aproximadamente con el cual tenemos los siguientes resultados:

	precision	recall	f1-score	support
0	1.00	0.73	0.84	34854
1	0.03	0.83	0.07	412
avg / total	0.99	0.73	0.83	35266
[[25344 9510]				
[ 71 341]]				

La matriz de confusión nos muestra que clasificamos a 9,851 como clientes tomadores de la oferta o en otras palabras son los clientes seleccionados para acampañar, de estos tenemos 341 clientes efectivos o que tomaron la oferta lo cual se traduce en una precisión cerca del 3.5%(3.45%), es decir, mi tasa de respuesta a la campaña. aunque la tasa suena baja recordar que con respecto a la densidad original estamos mejorando la efectividad casi 3 veces, además tenemos un recall de 83% es decir que con estos casi 10mil ctes. capturamos el 83% de los contratos.

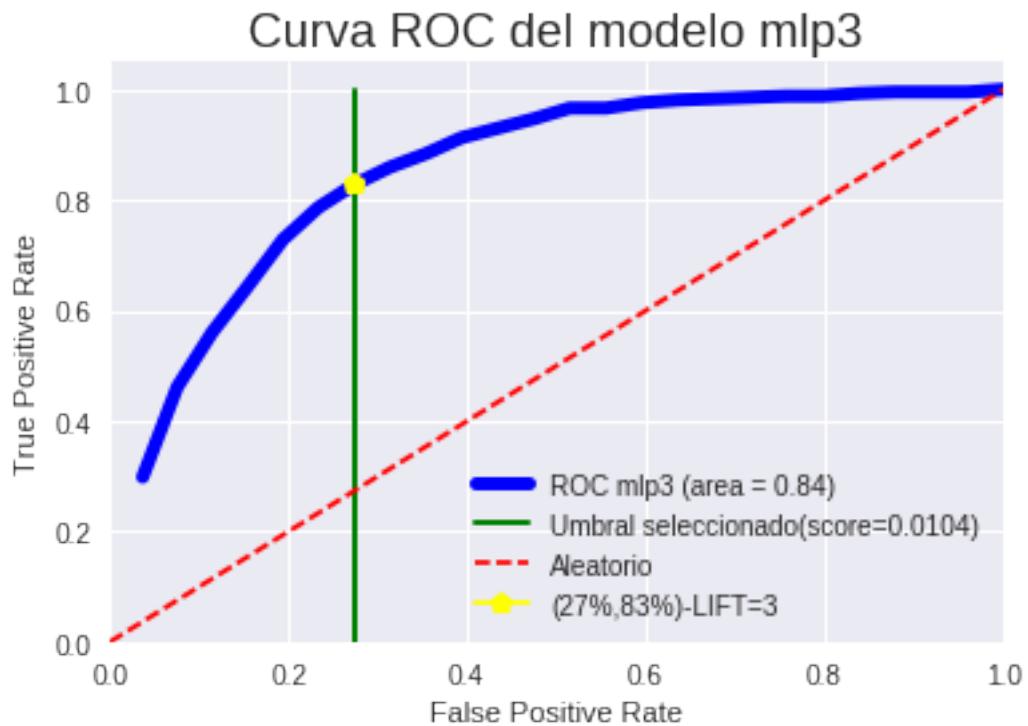
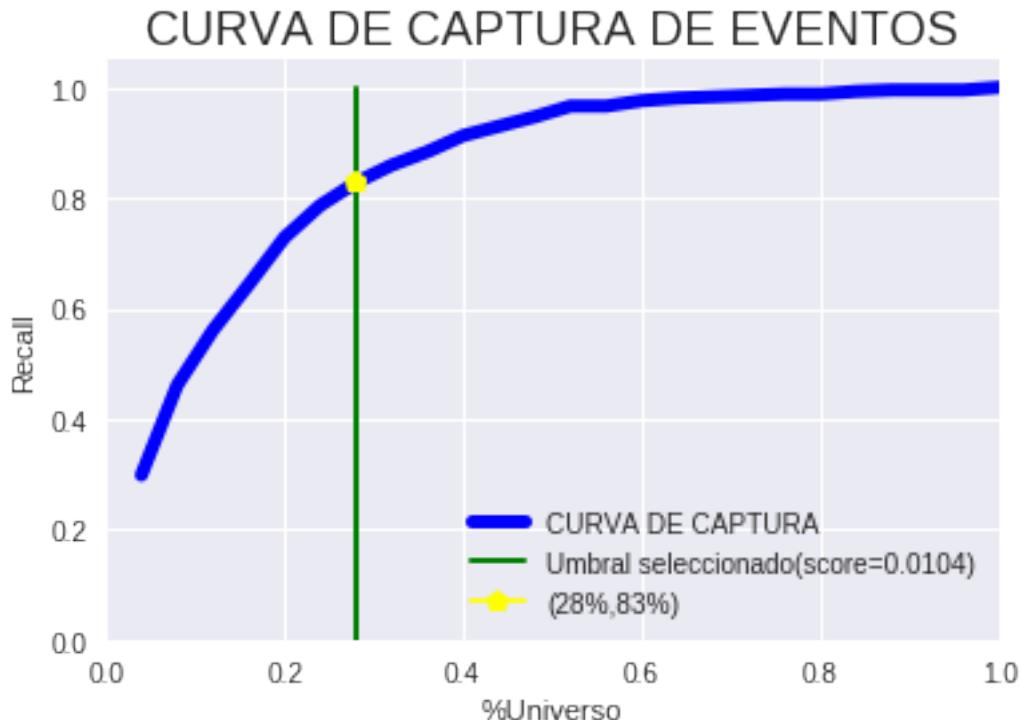


Figura22.- Curva ROC + umbral de selección

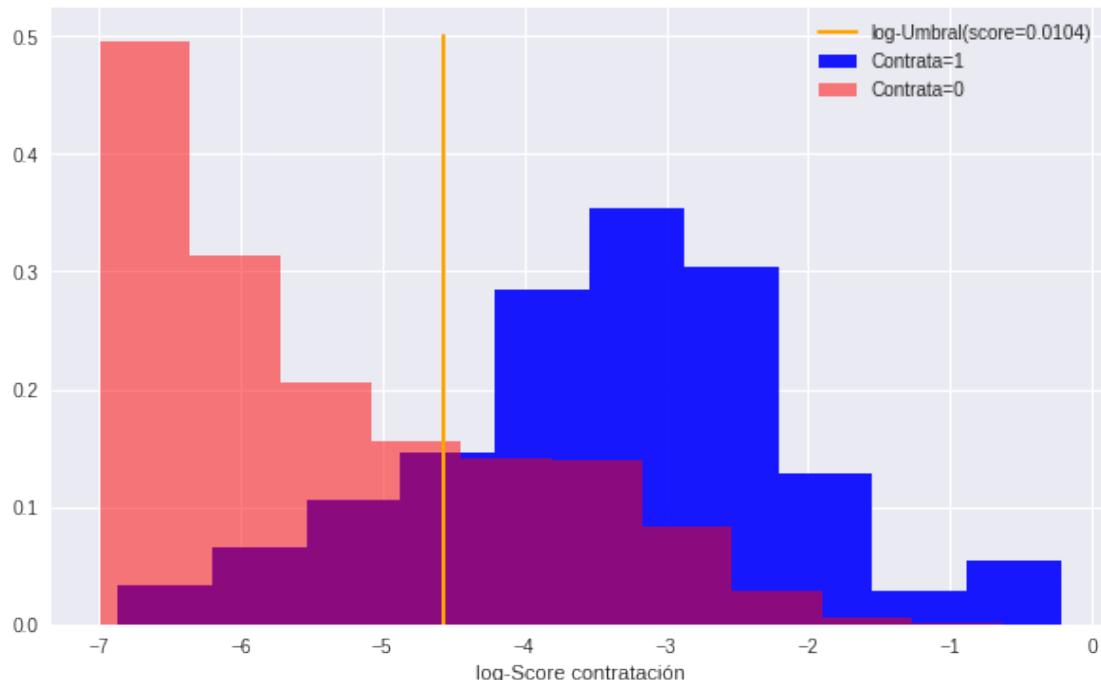
De acuerdo con la curva ROC al usar nuestro punto de corte 0.0104 alcanzamos a capturar casi el 83% de las contrataciones pero sólo con el 27% de los falsos positivos, es decir, eliminamos más del 72% de falsos positivos y nuestra tasa de clasificación errónea disminuye. Además nuestra área bajo la curva AUC=0.84 lo cual es bastante buena.



**Figura 23.-Curva de %universo vs. %eventos capturados o recall + umbral de selección**

Con esta curva podemos ver que con el 28% del universo de clientes capturamos el 83% de las contrataciones o Clientes que aceptan la oferta reduciendo de forma importante el tamaño de selección de la campaña.

Out [137]: <matplotlib.text.Text at 0x7f8decb1a3c8>



**Figura 24.-Gráfica del log-score + umbral de selección.**

Muestra las separación de las distribuciones del log-score de los clientes que contratan y no contratan, se puede ver que en los scores más altos la proporción de falsos positivos o universo de clientes que no contratan los préstamos, es menor, la separación ideal sería que en la gráfica las distribuciones no se intersectaran. En términos generales el modelo mlp3 separa bastante bien las poblaciones.

## 30 Conclusiones

De acuerdo con nuestros resultados podemos concluir lo siguiente con respecto al modelo:

1.-Mejoramos la efectividad de la campaña 3 veces es decir de una campaña de **1.17% a una de 3.45%**.

2.-Se redujo el número de clientes a acampañar a tan sólo un 28% del universo, es decir descartamos a un 72% de los clientes que originalmente se les mandaban ofertas de préstamos lo cual reduce costos de comunicación.

3-De acuerdo con la matriz de confusión podemos observar que la efectividad o tasa de respuesta de los clientes descartados de la campaña o clasificados como no contratadores es de **71/25415=0.28%** lo cual es peor que seleccionar al azar (1.17%).

4.-Con el 28% del universo que se selecciona se captura casi el 83% de las contrataciones o clientes que contratan, es decir sólo sacrificamos el 17% de contrataciones.

5.-Más que un clasificador tenemos un modelo con un score que puede ordenar a los clientes de mayor a menor propensión a contratar un préstamo, esto nos ayuda a poder seleccionar los primeros  $n$  clientes más propensos a tomar la oferta si es necesario escoger así.

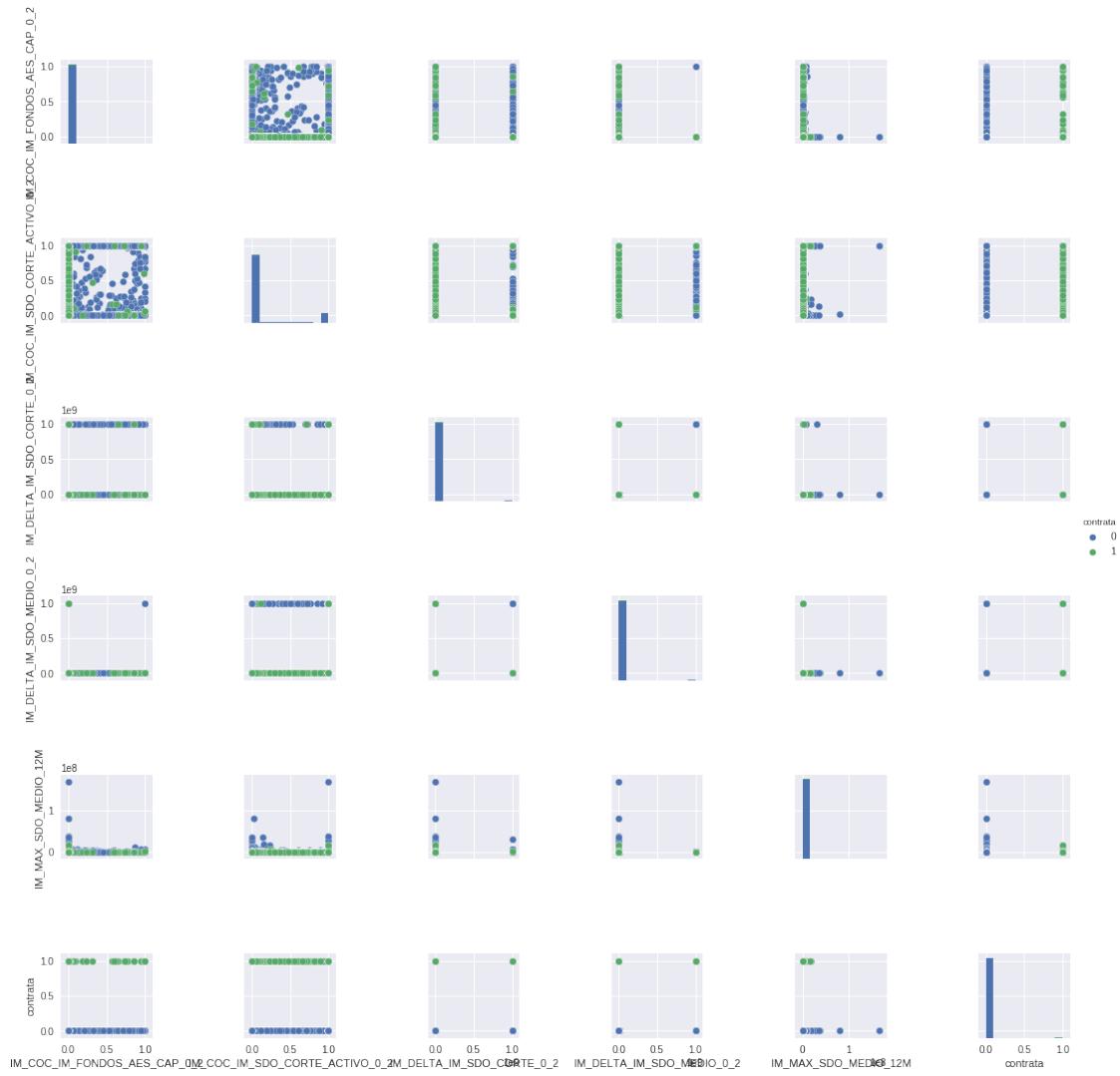
**Siguientes pasos**

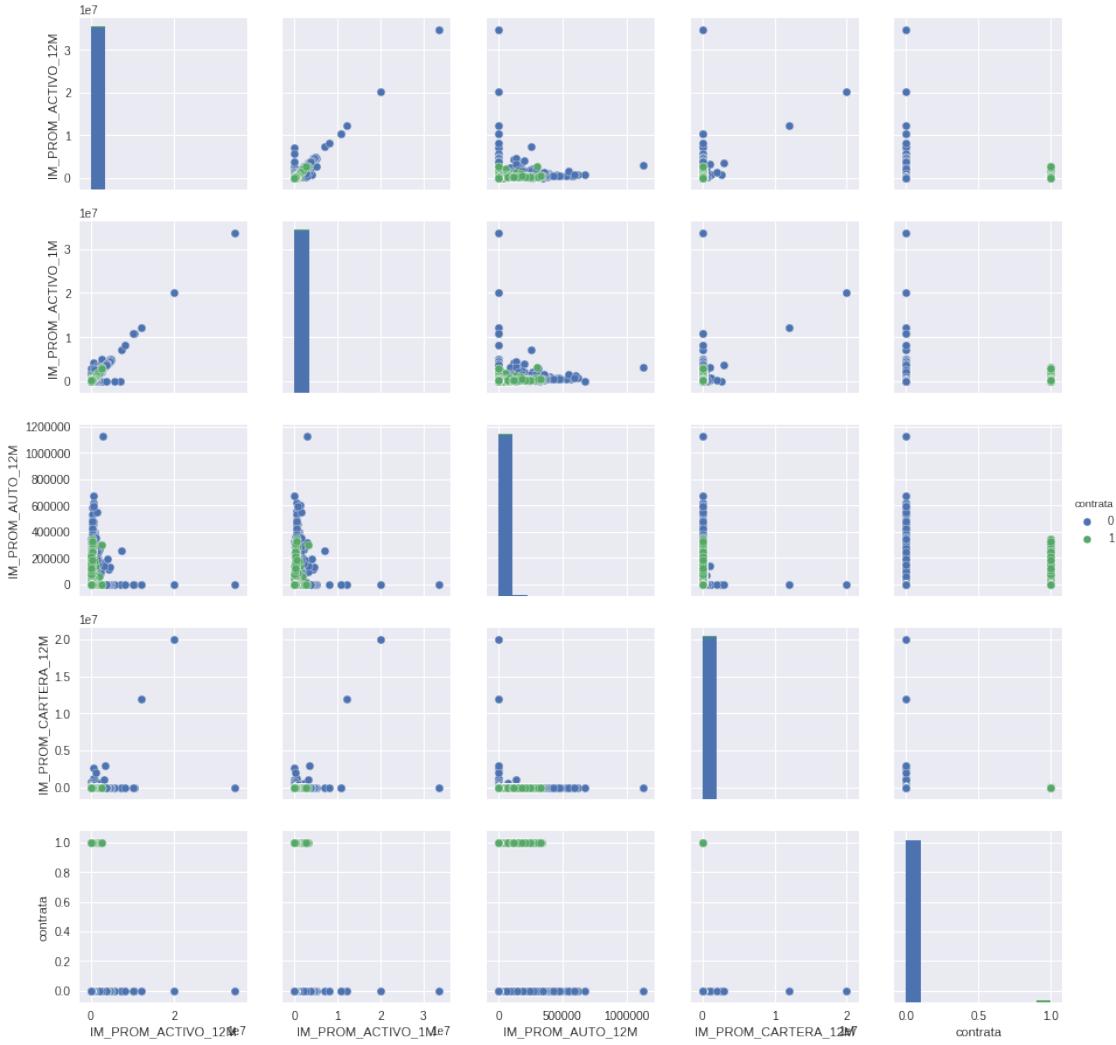
Uno de los problemas, planteados originalmente, a resolver es creación de distintas etiquetas de propensión a la toma de ofertas esta es necesaria para hacer grupos de prioridad de comunicación ya que como explicamos al principio no se comunican al mismo tiempo sino por bloques entonces al final para crear  $k$  grupos de propensión (Muy propenso,...,medio propenso,...,bajo propenso) tenemos que agrupar de acuerdo con nuestro score entonces hay que crear varios umbrales sobre el mismo score es decir un umbral para "Muy propensos", medios, bajos etc. una forma de hacerlo es podemos buscar su estadístico KS después de cada corte por ejemplo comenzando con lo que ya tenemos, de los que clasificamos como 1's que son casi 10mil podemos buscar su KS y hacer un corte y análogamente con los que clasificamos como 0's y así hasta obtener la segmentación requerida. Otro punto relevante es seguir buscando mejores parámetros para algunos algoritmos, con maquinas más potentes que puedan paralelizar ya que en este proyecto los tiempos de respuesta eran muy altos y por ello no se pudo probar con más. Finalmente, una de las tareas pendientes es asignar un costo a los falsos positivos y falsos negativos para seleccionar un modelo basado en la utilidad que nos puede dar una campaña en vez de sólo utilizar la efectividad o precisión.

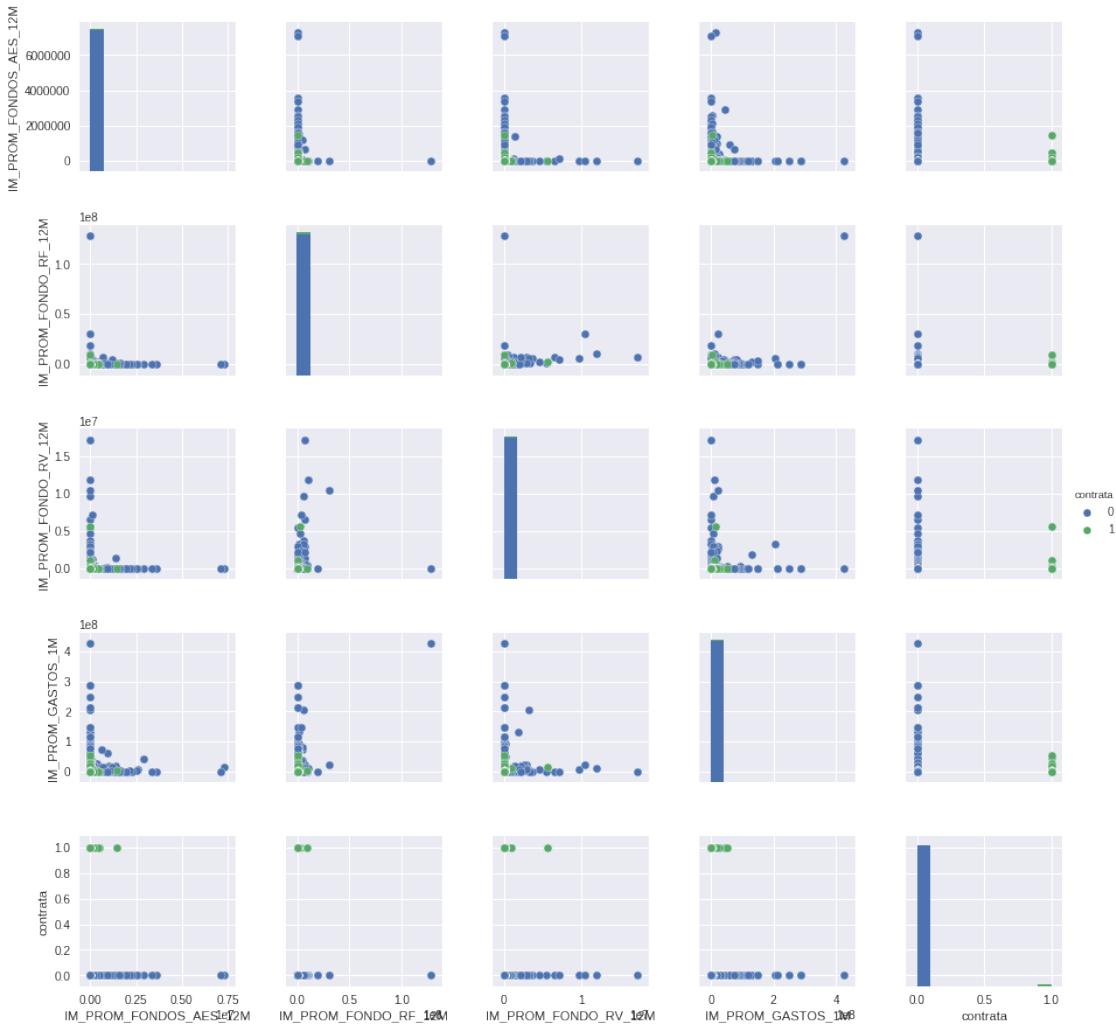
## **31 Anexos**

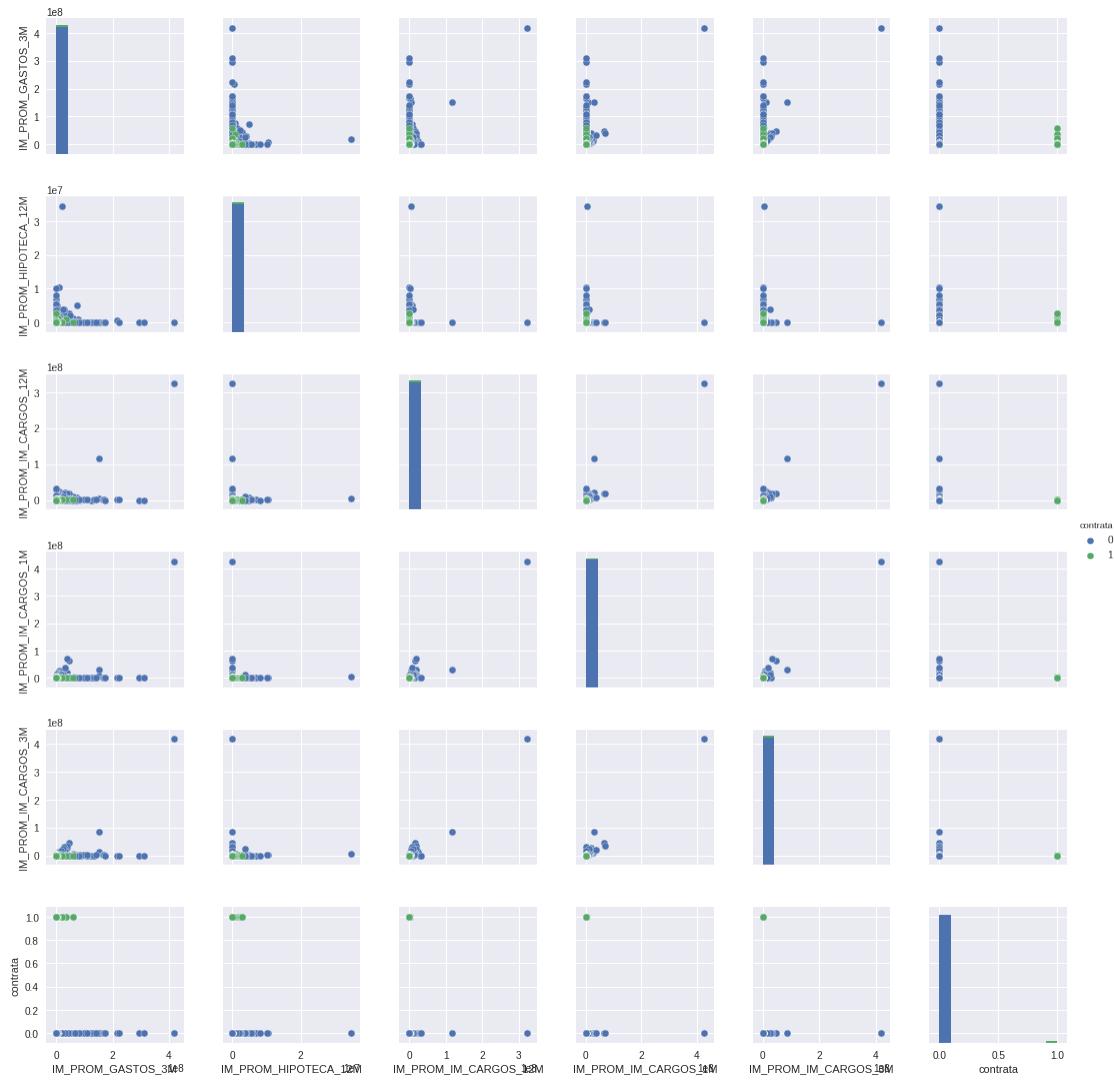
### **32 Anexo1**

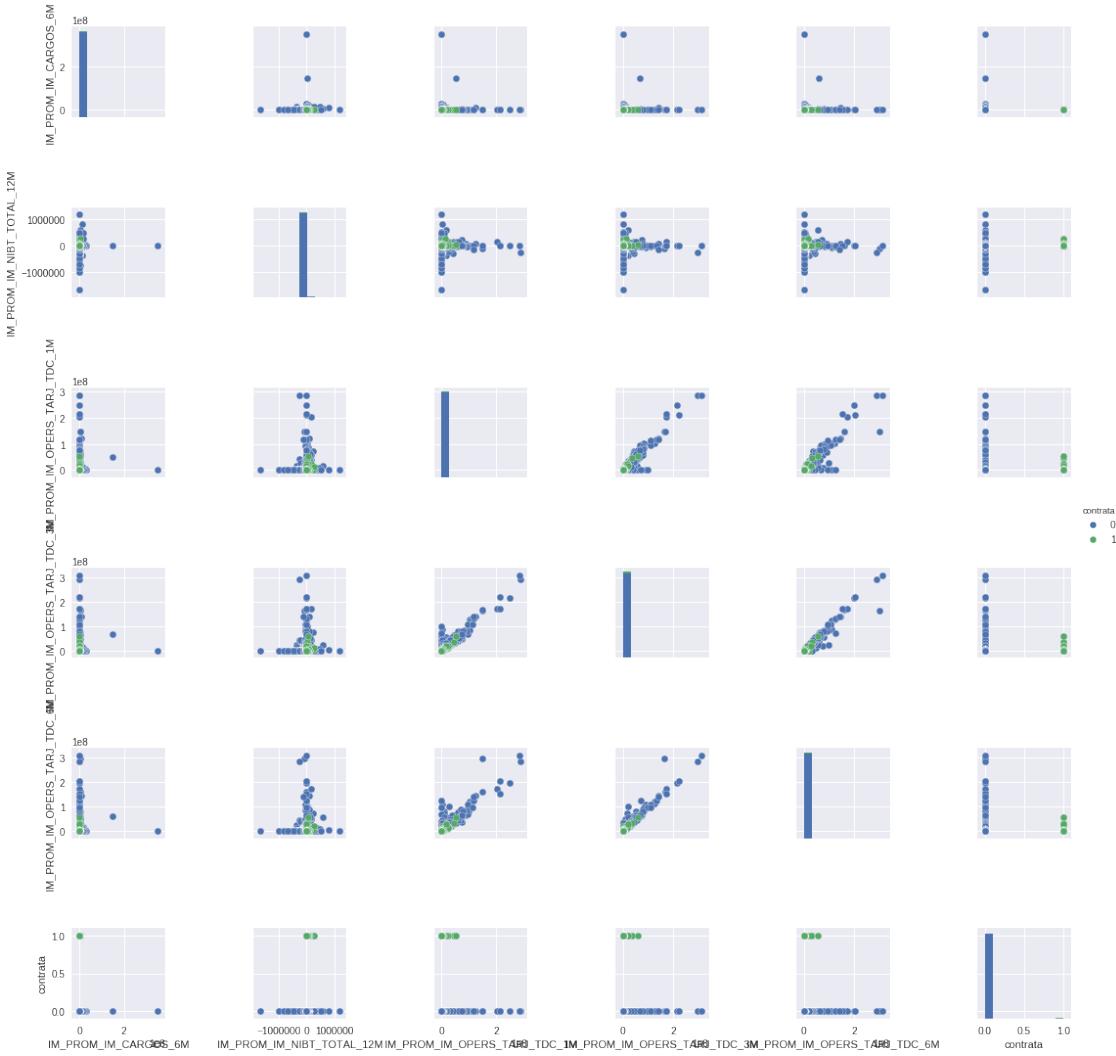
#### **Variables numéricas originales**

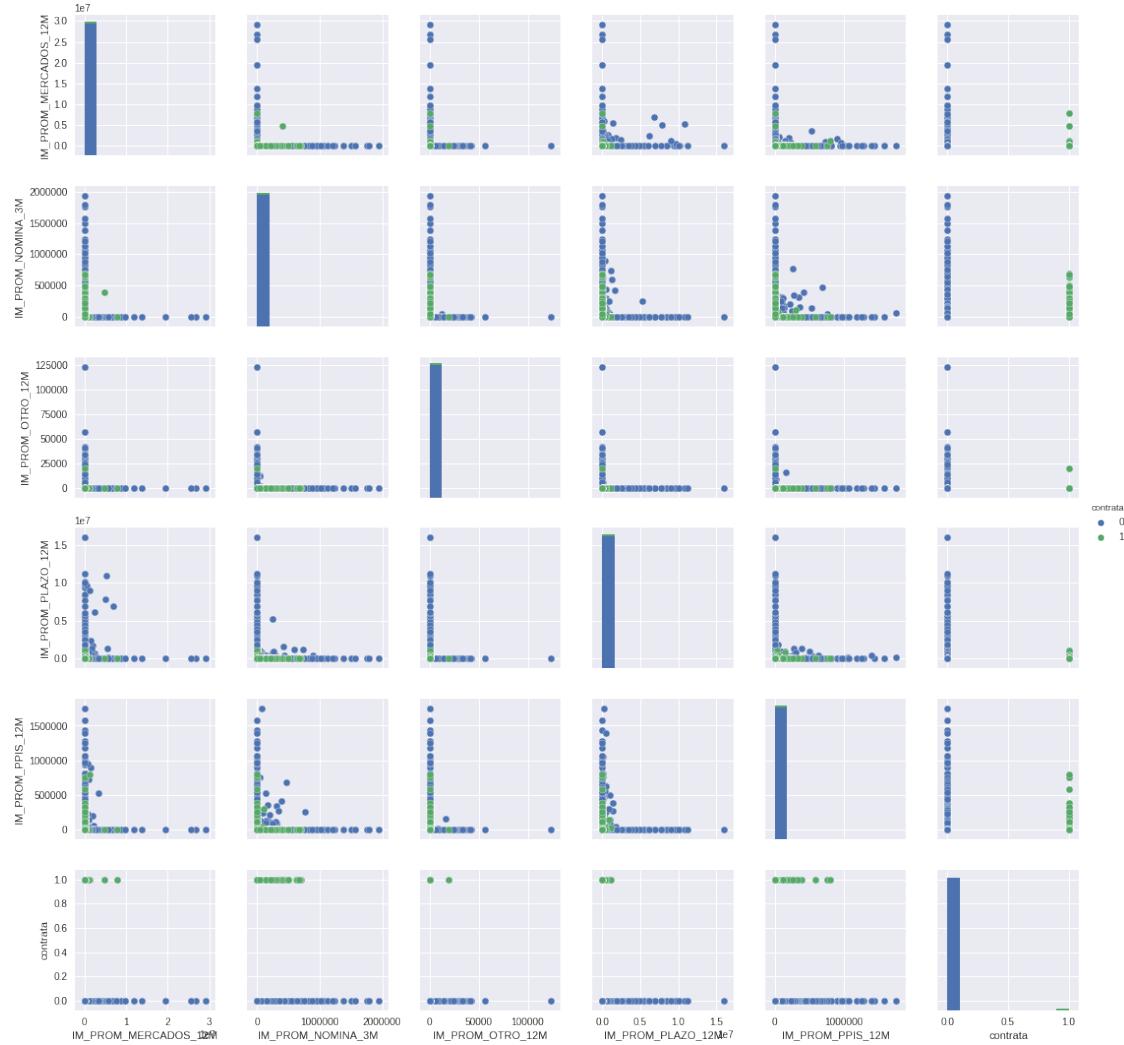


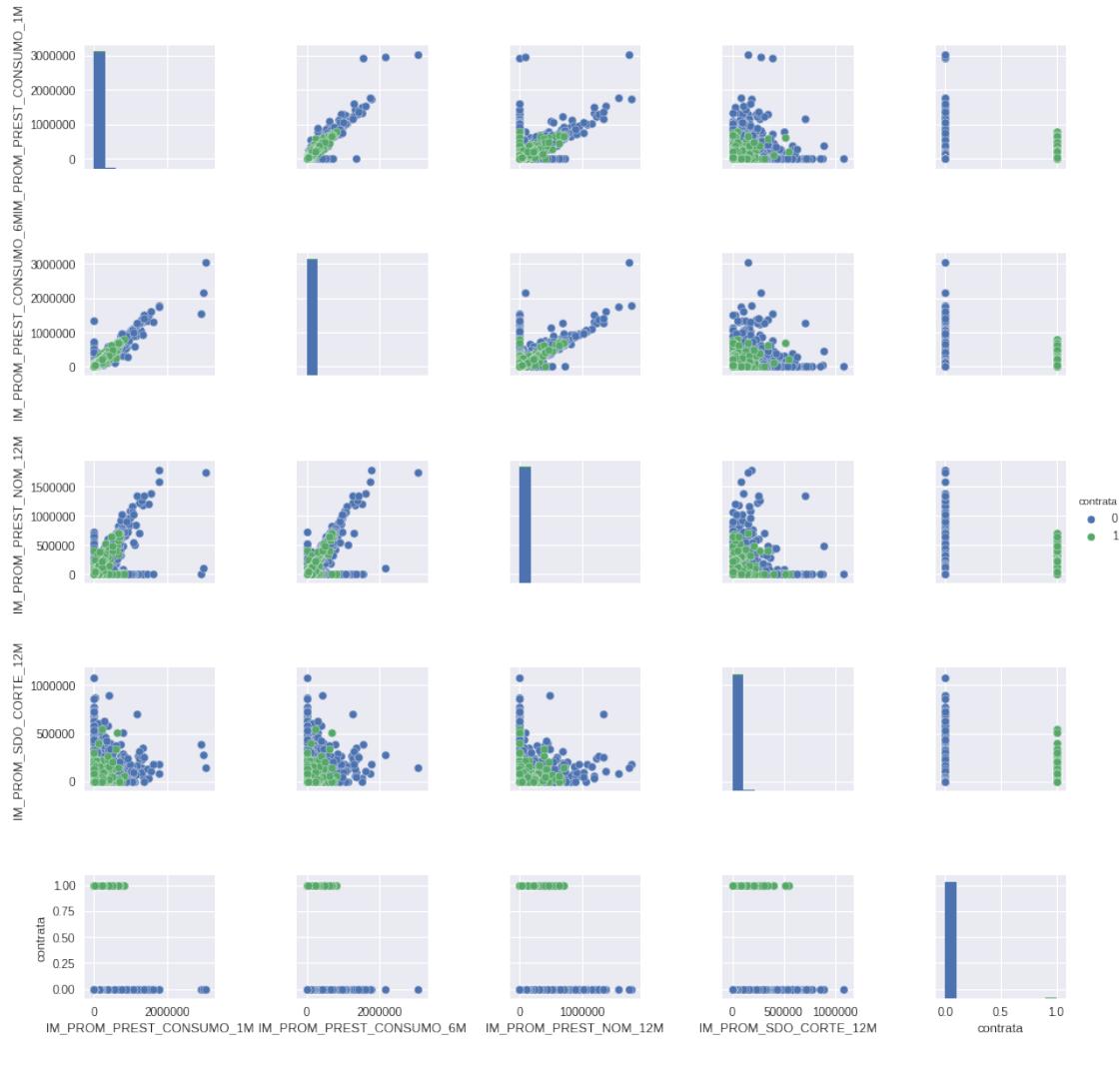


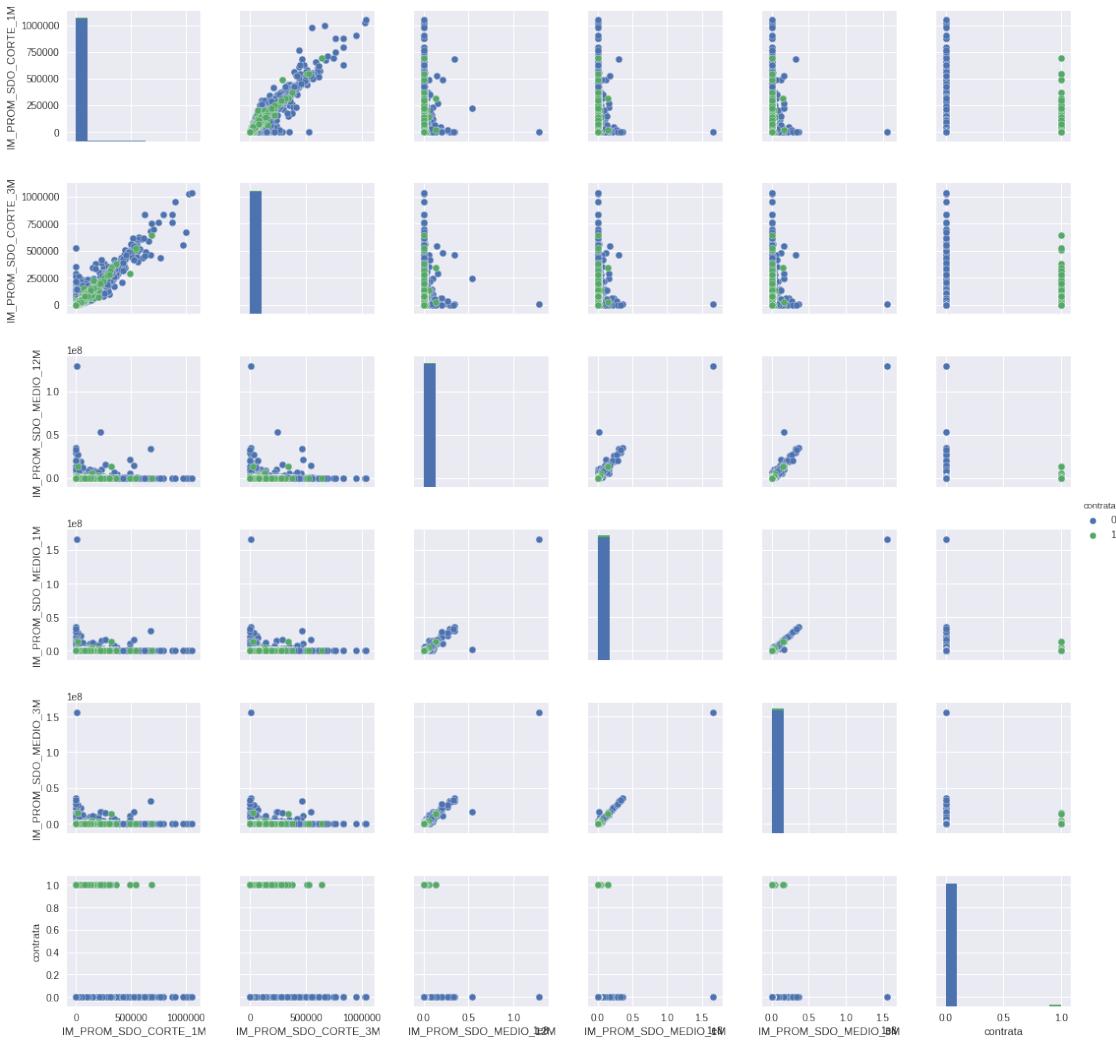


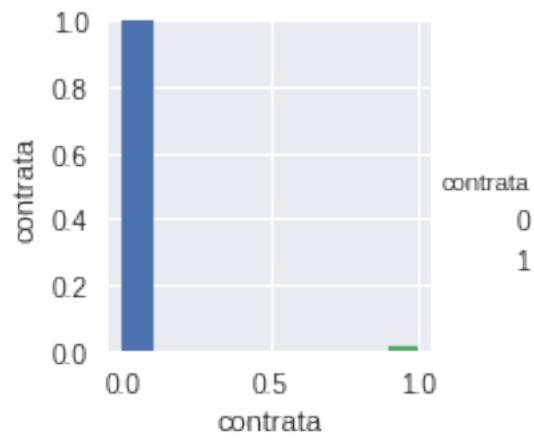
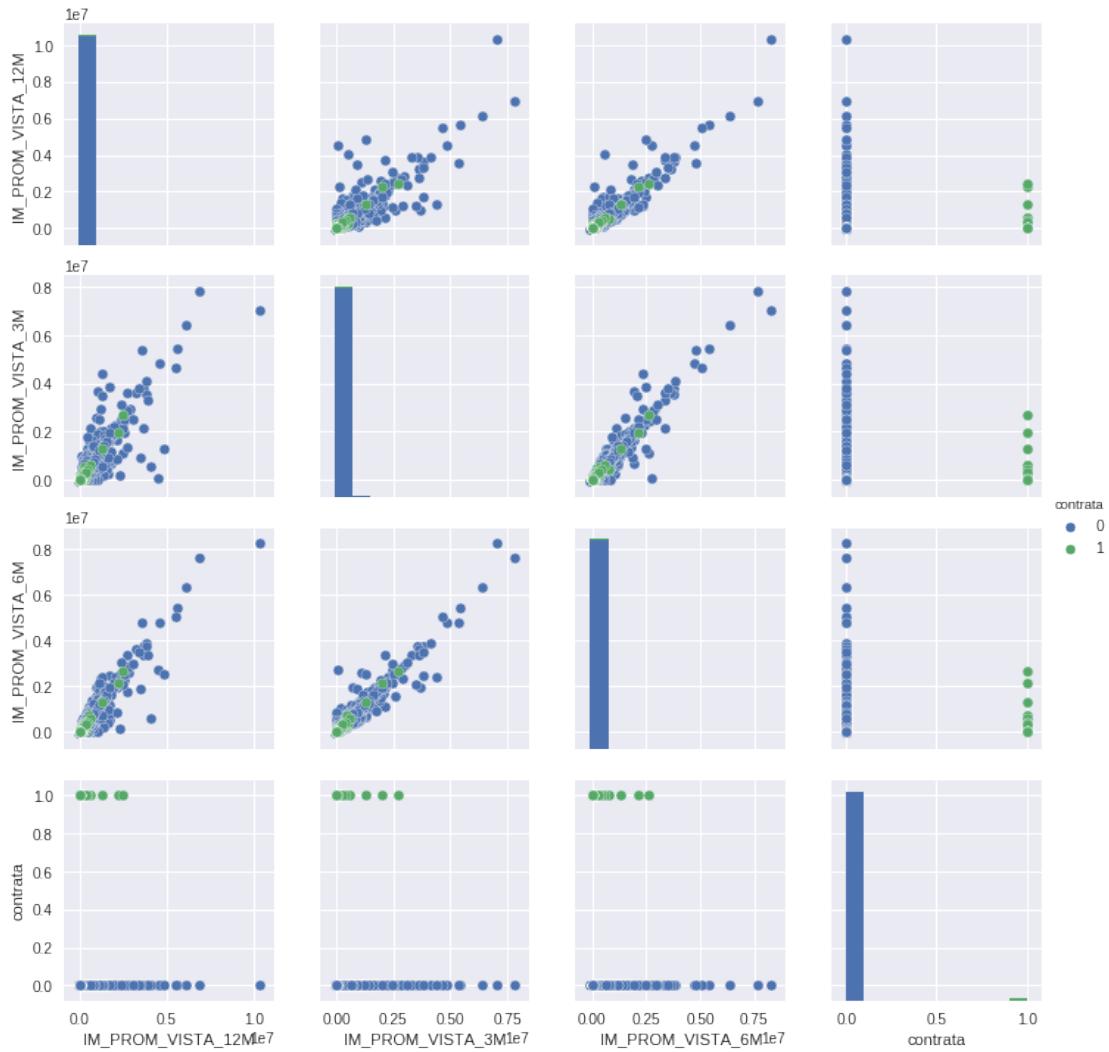






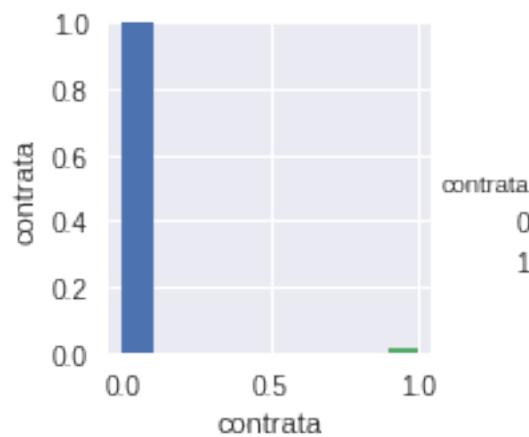
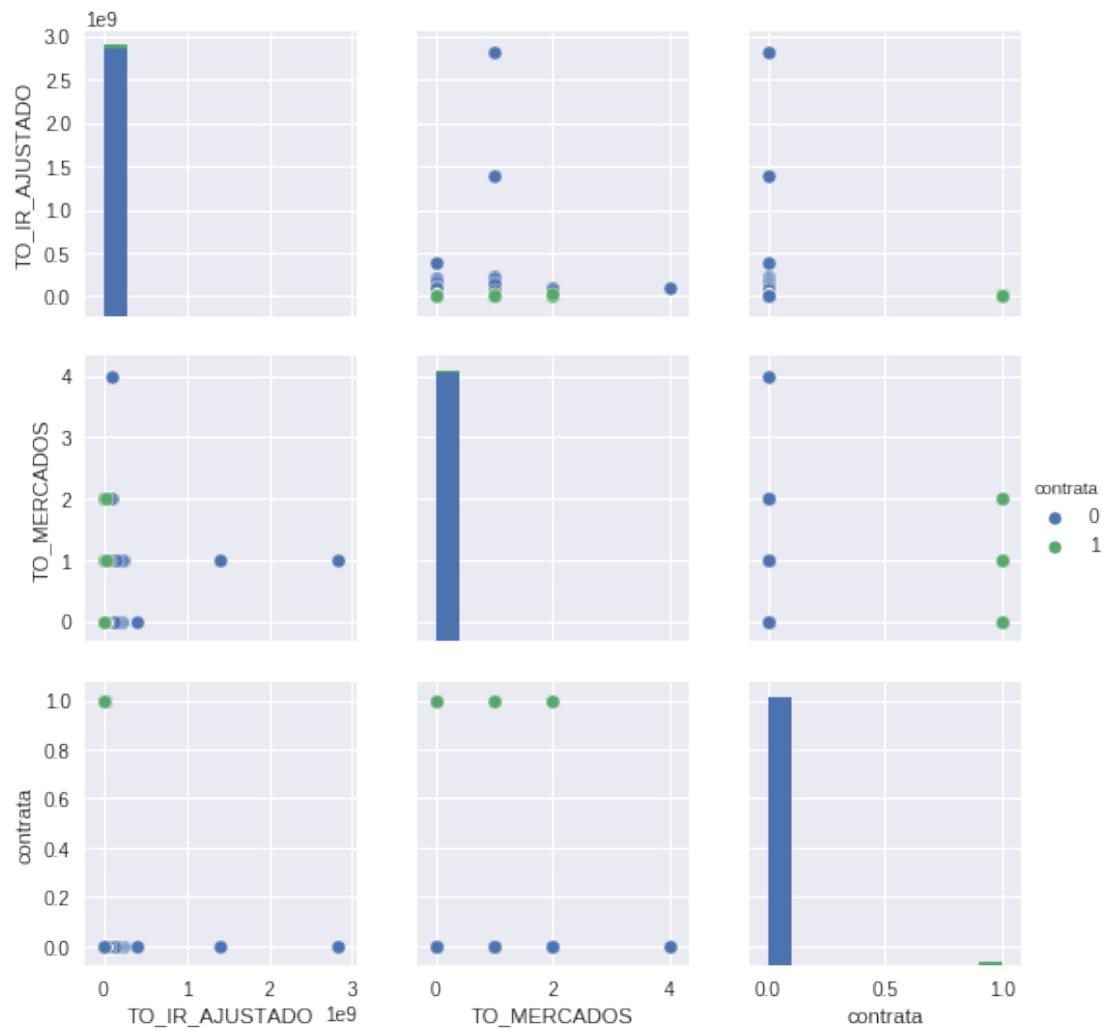


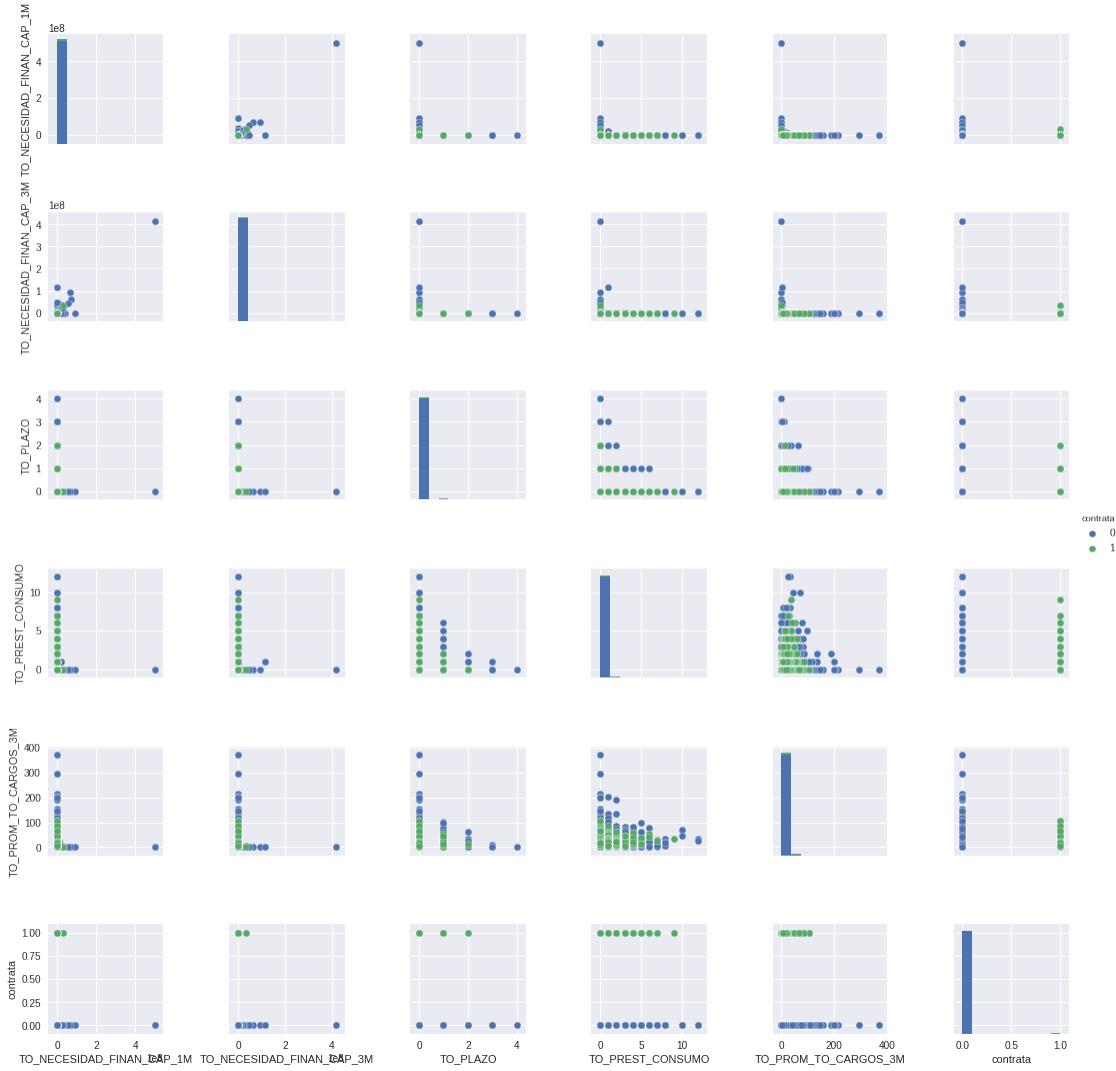


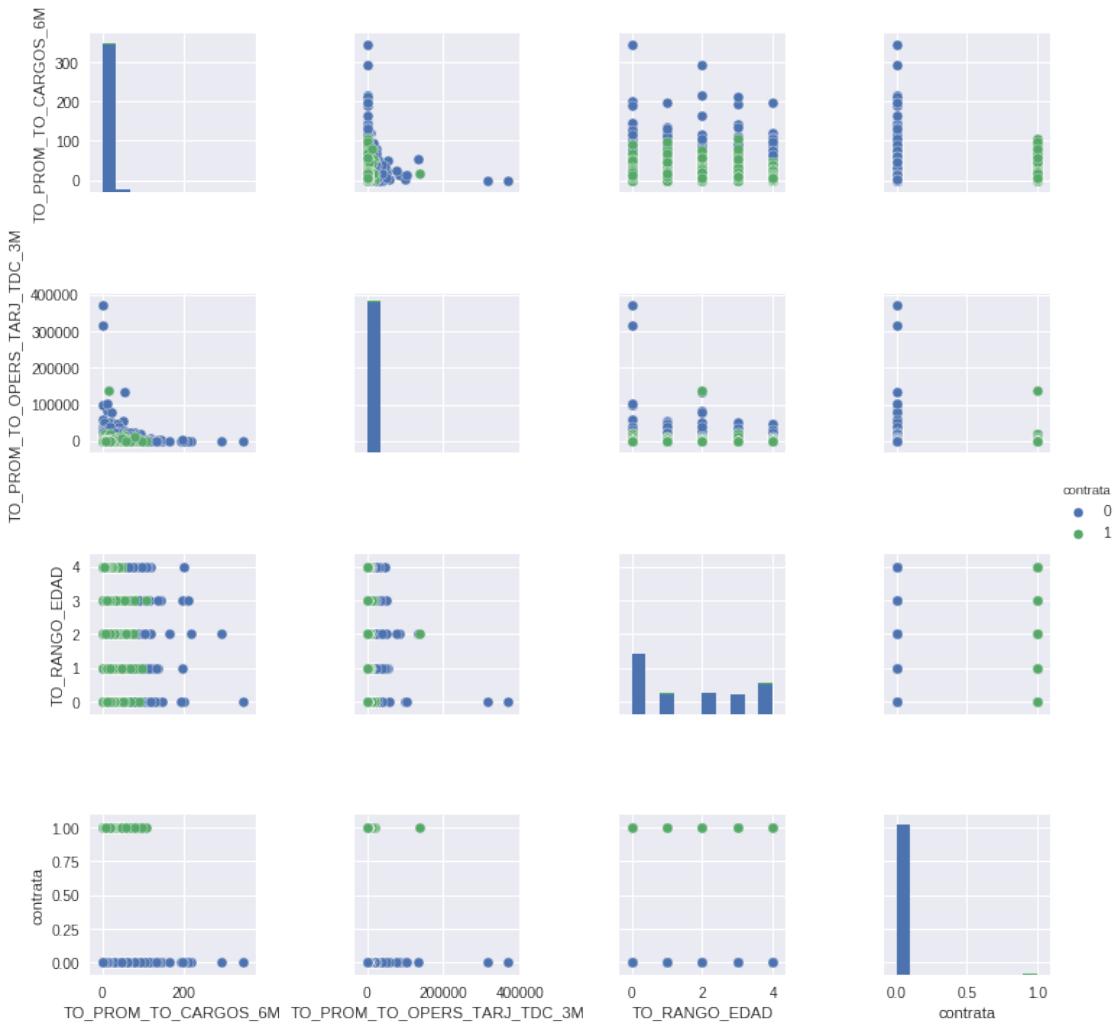






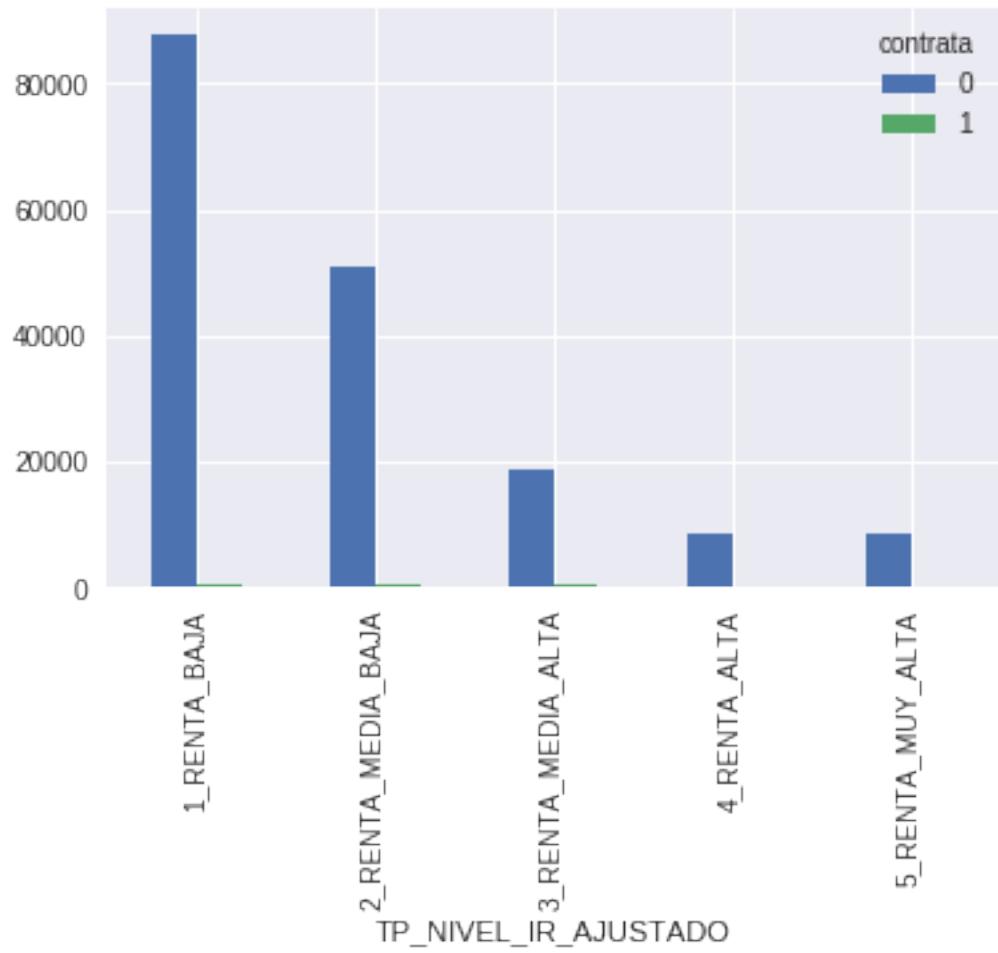


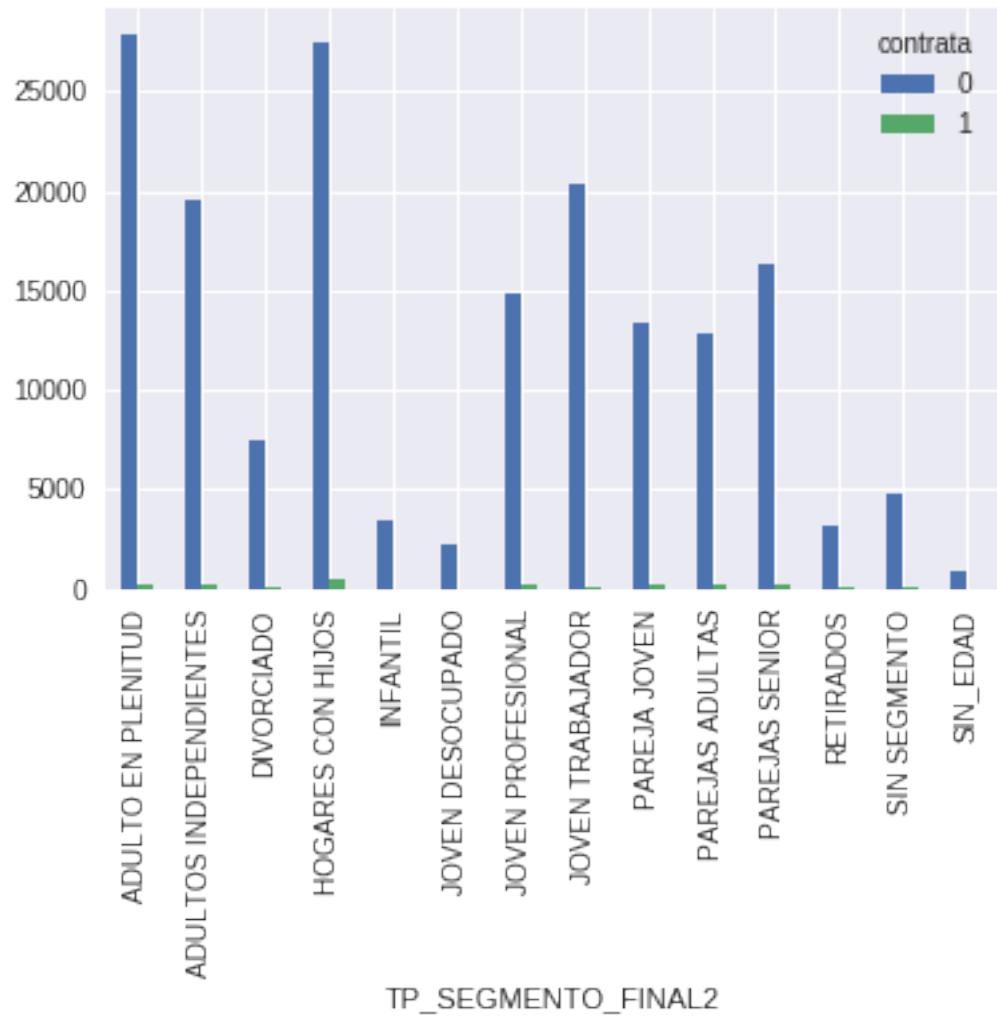


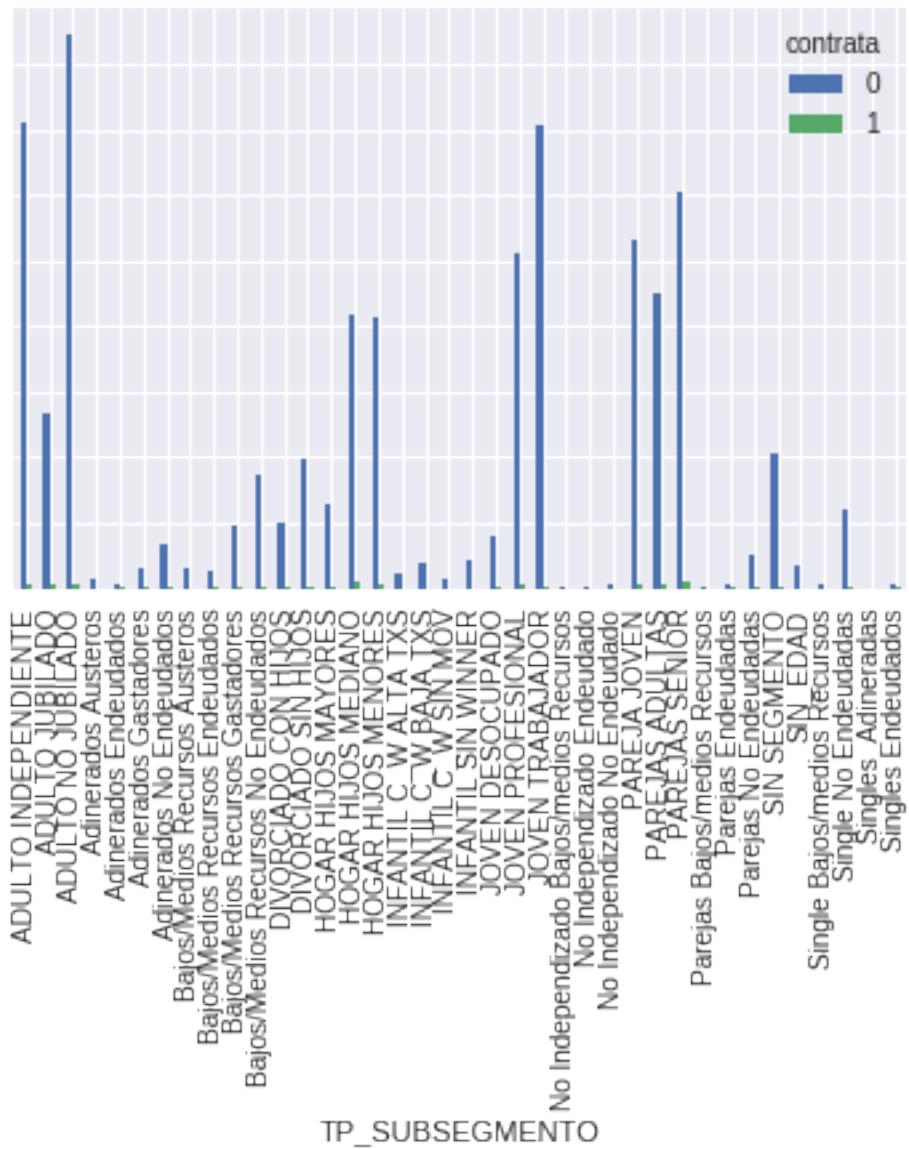


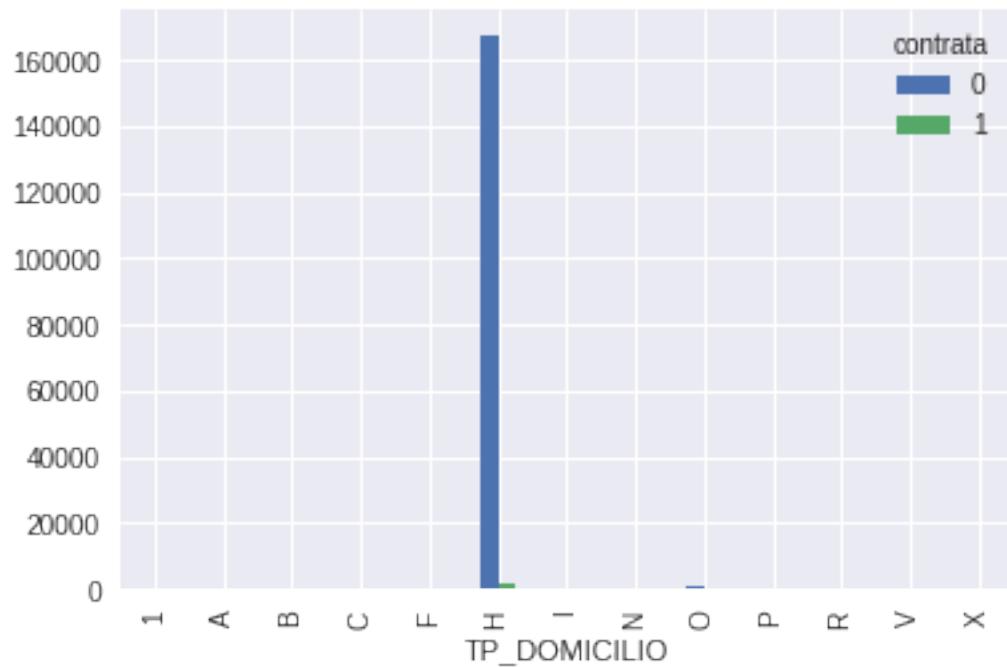
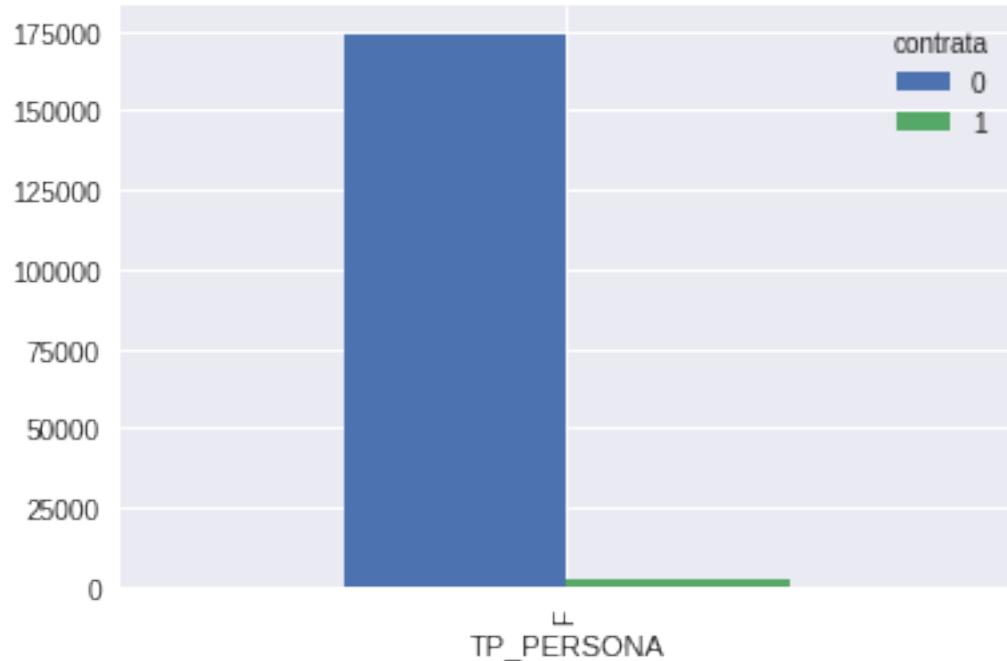
### 33 Anexo2

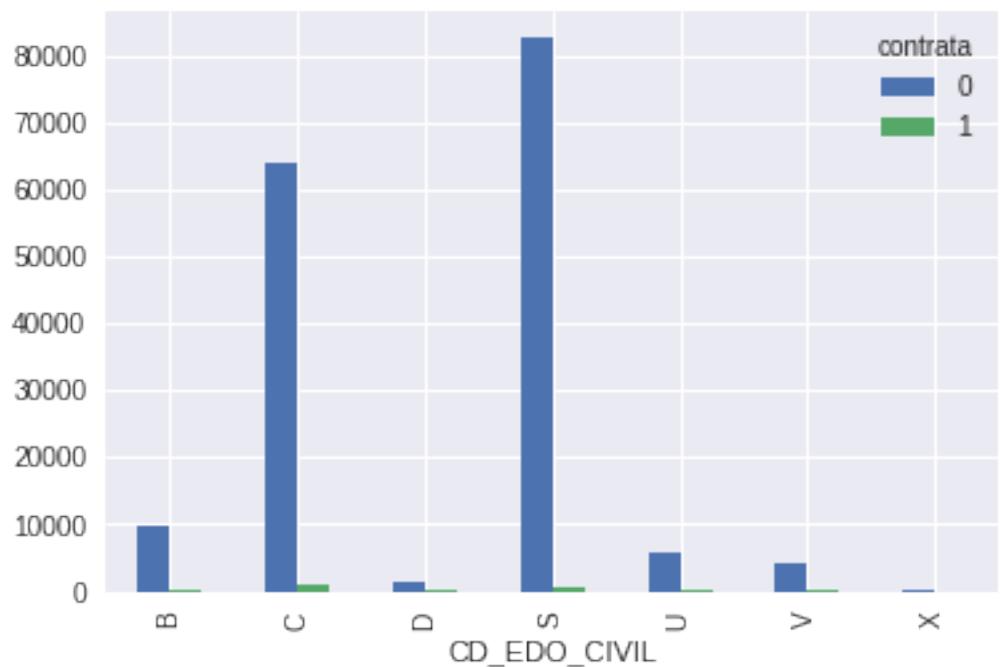
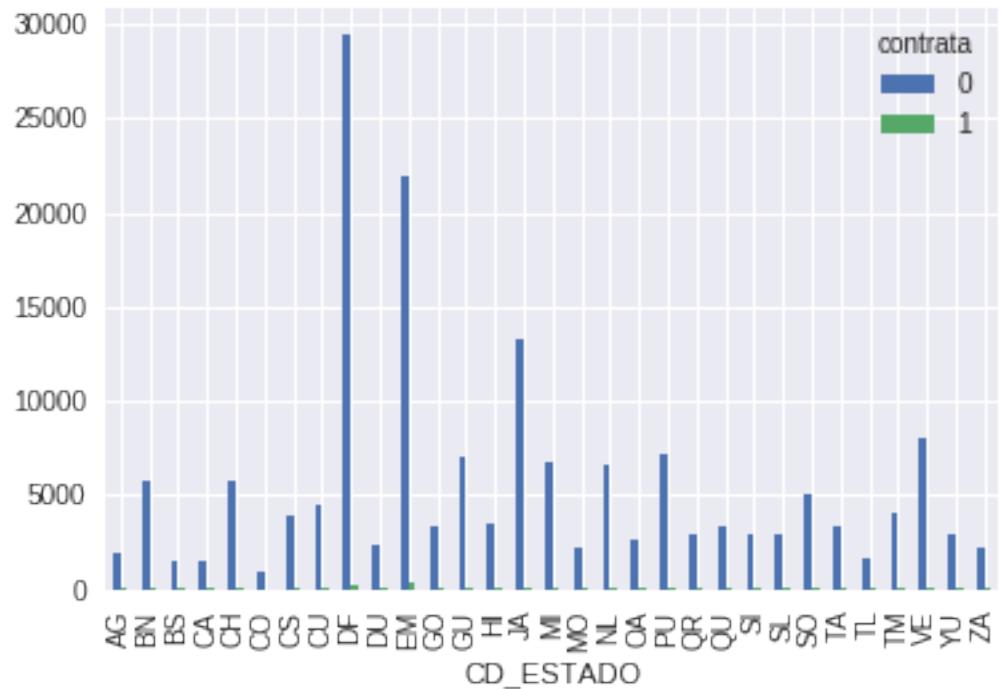
Variables categóricas originales

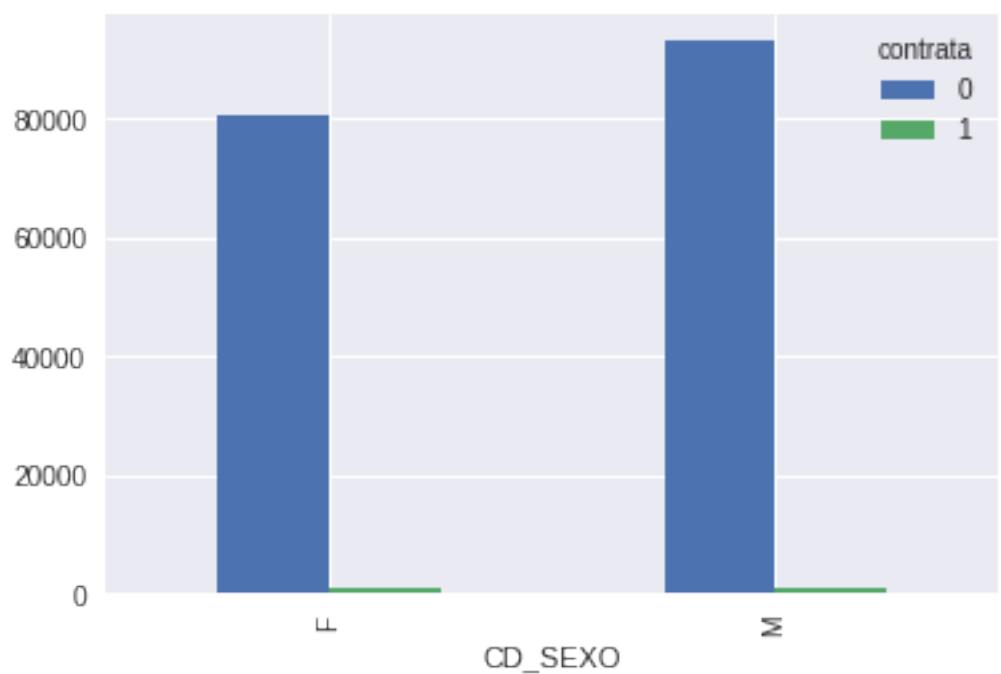
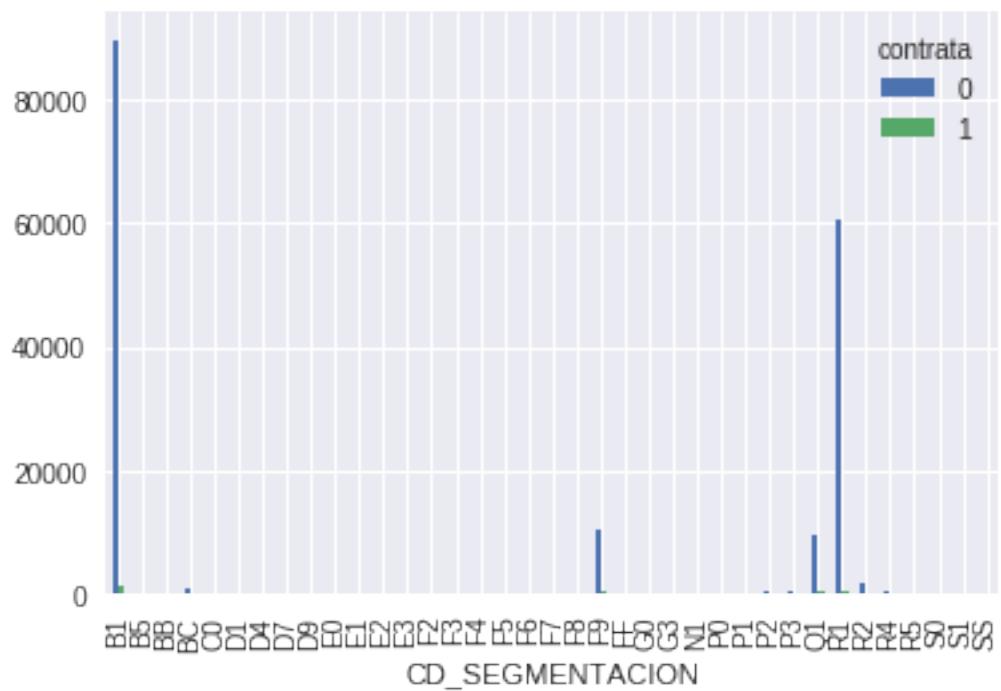


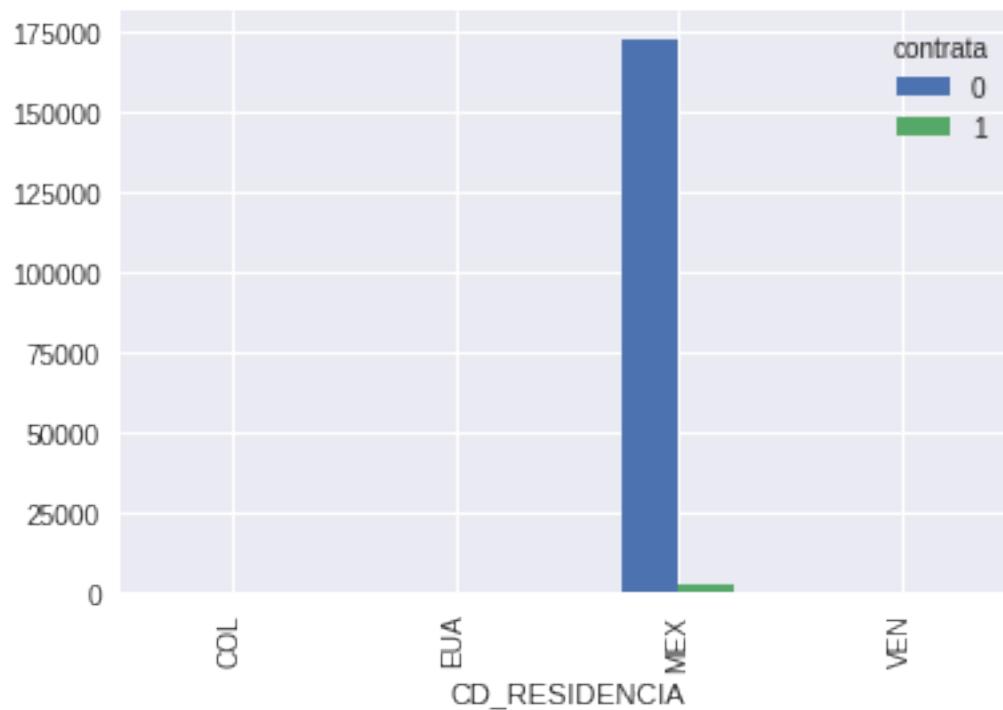
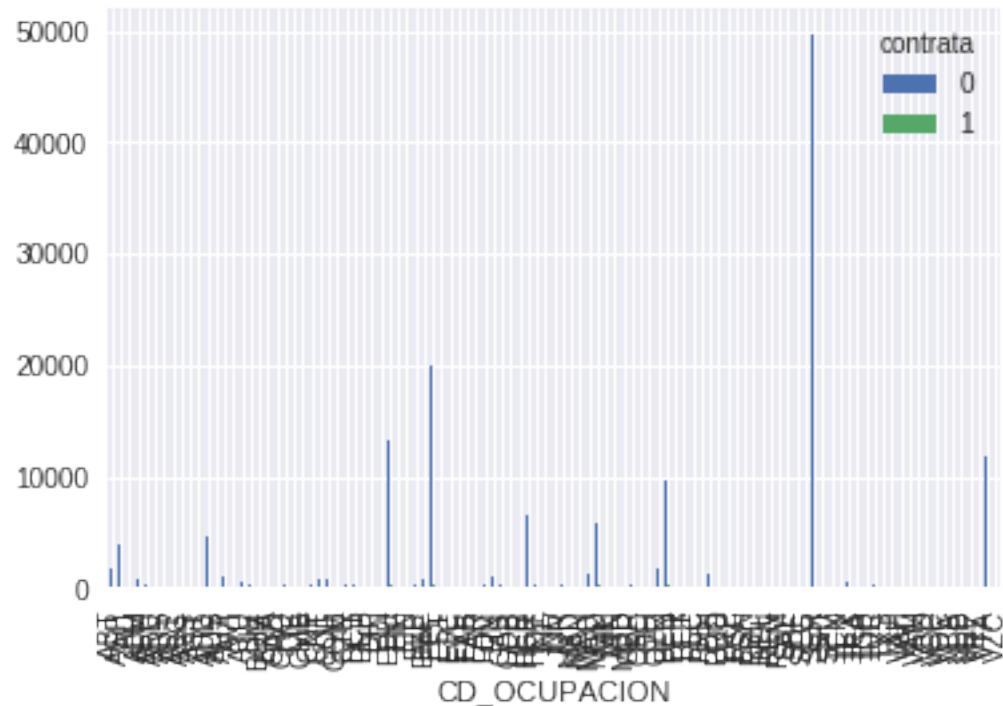


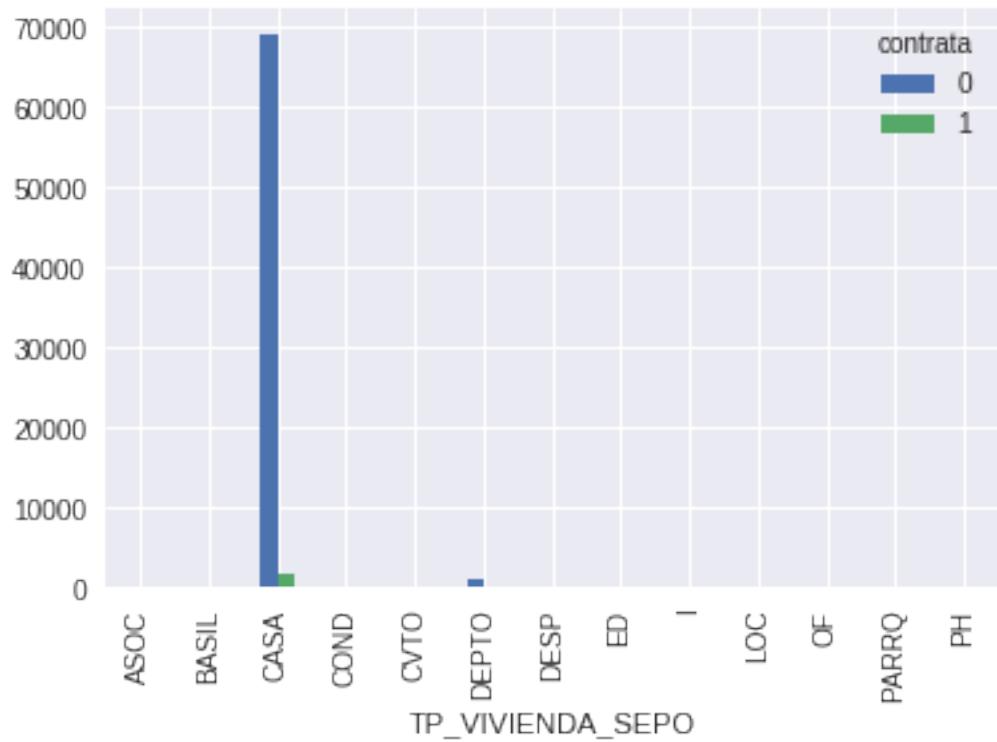






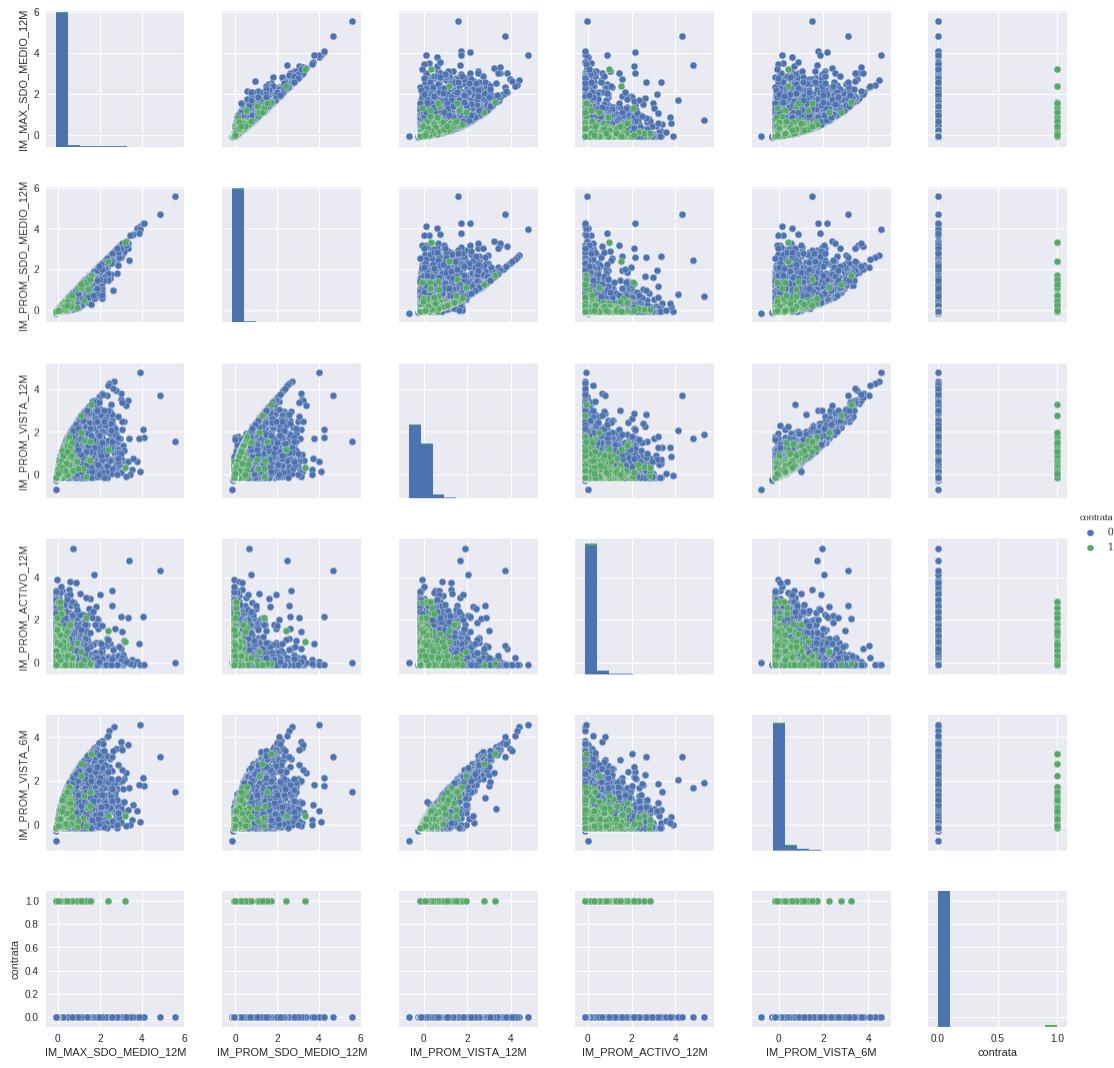


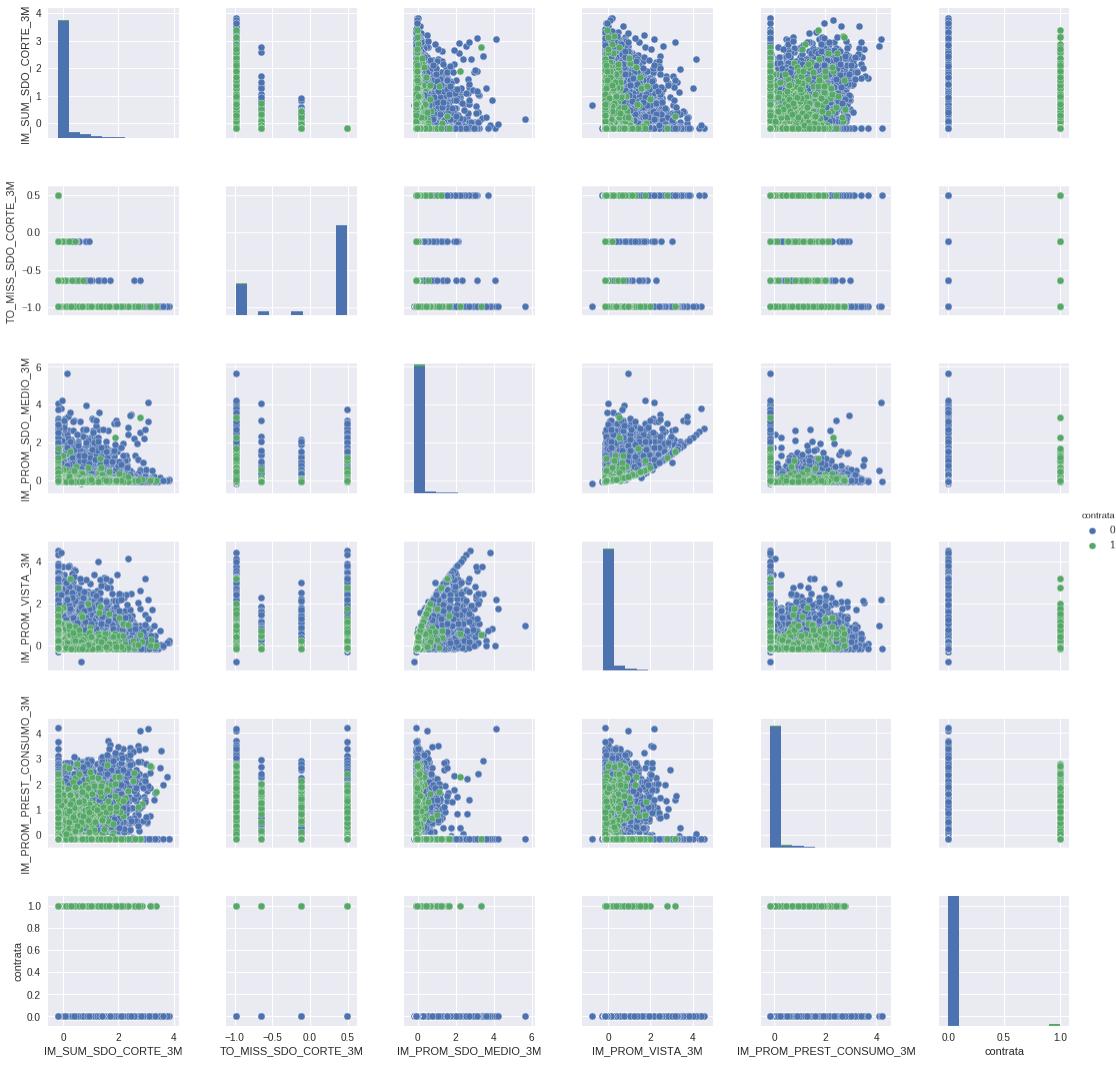




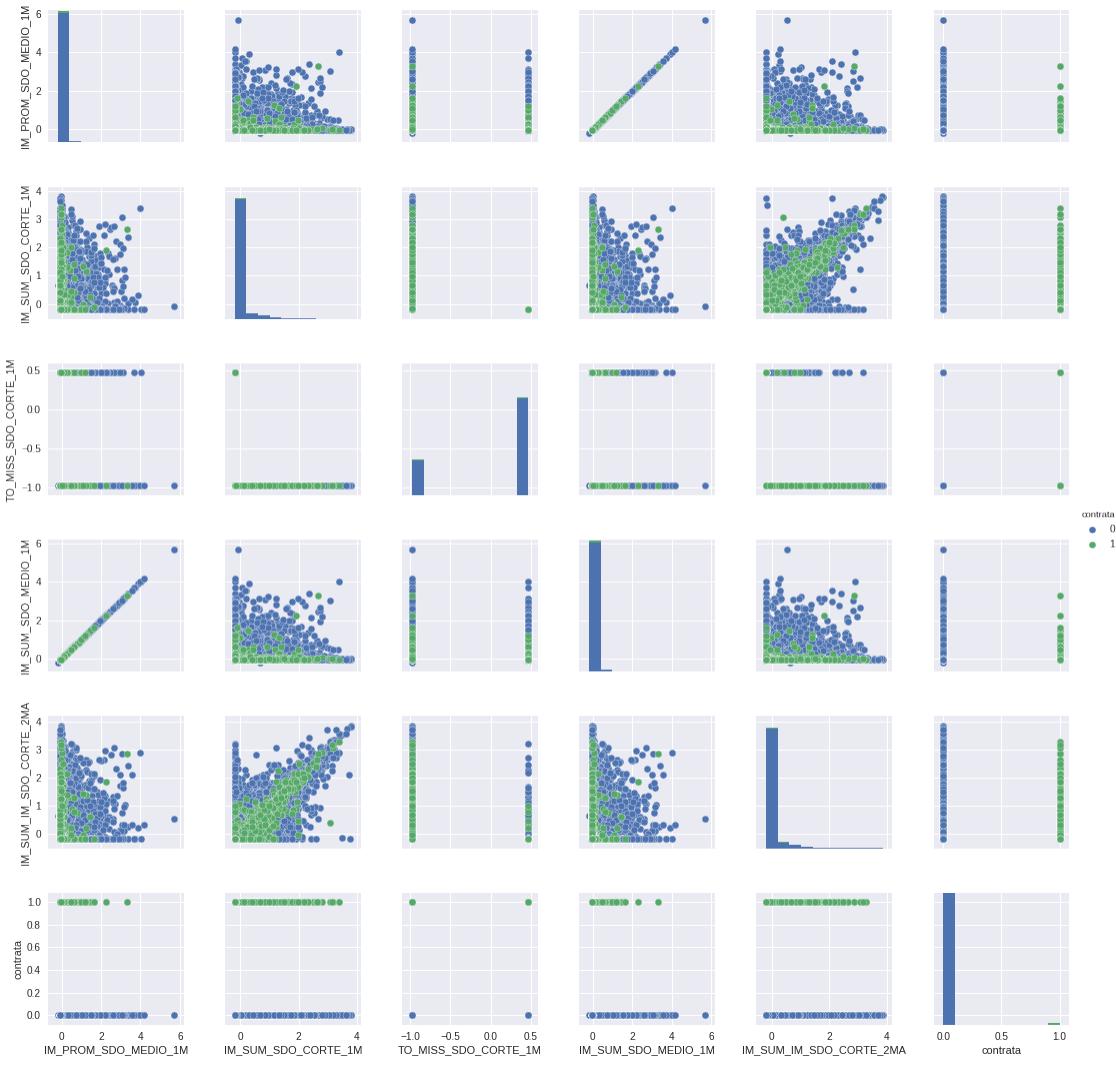
### 34 Anexo3

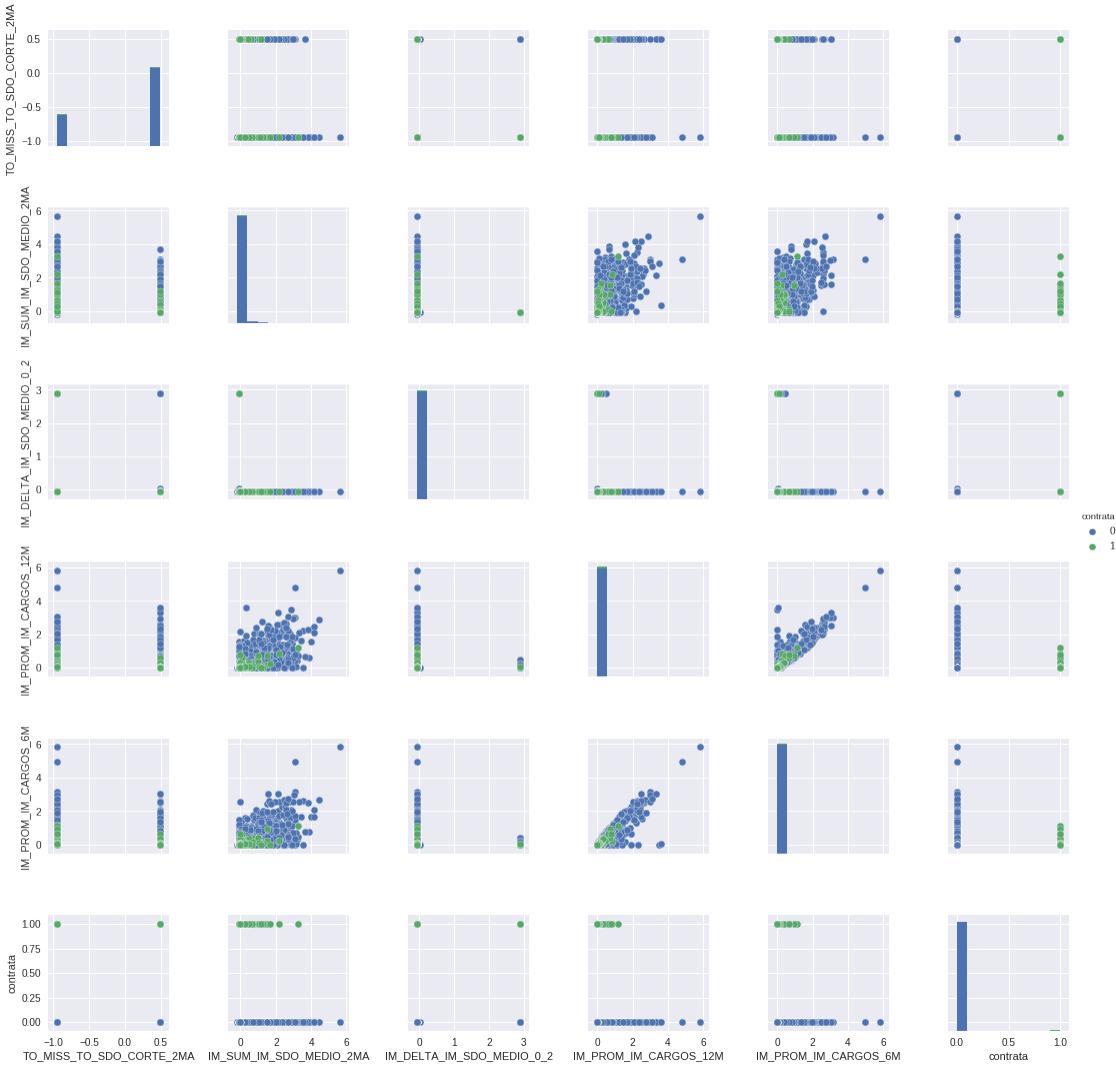
Variables numéricas transformadas

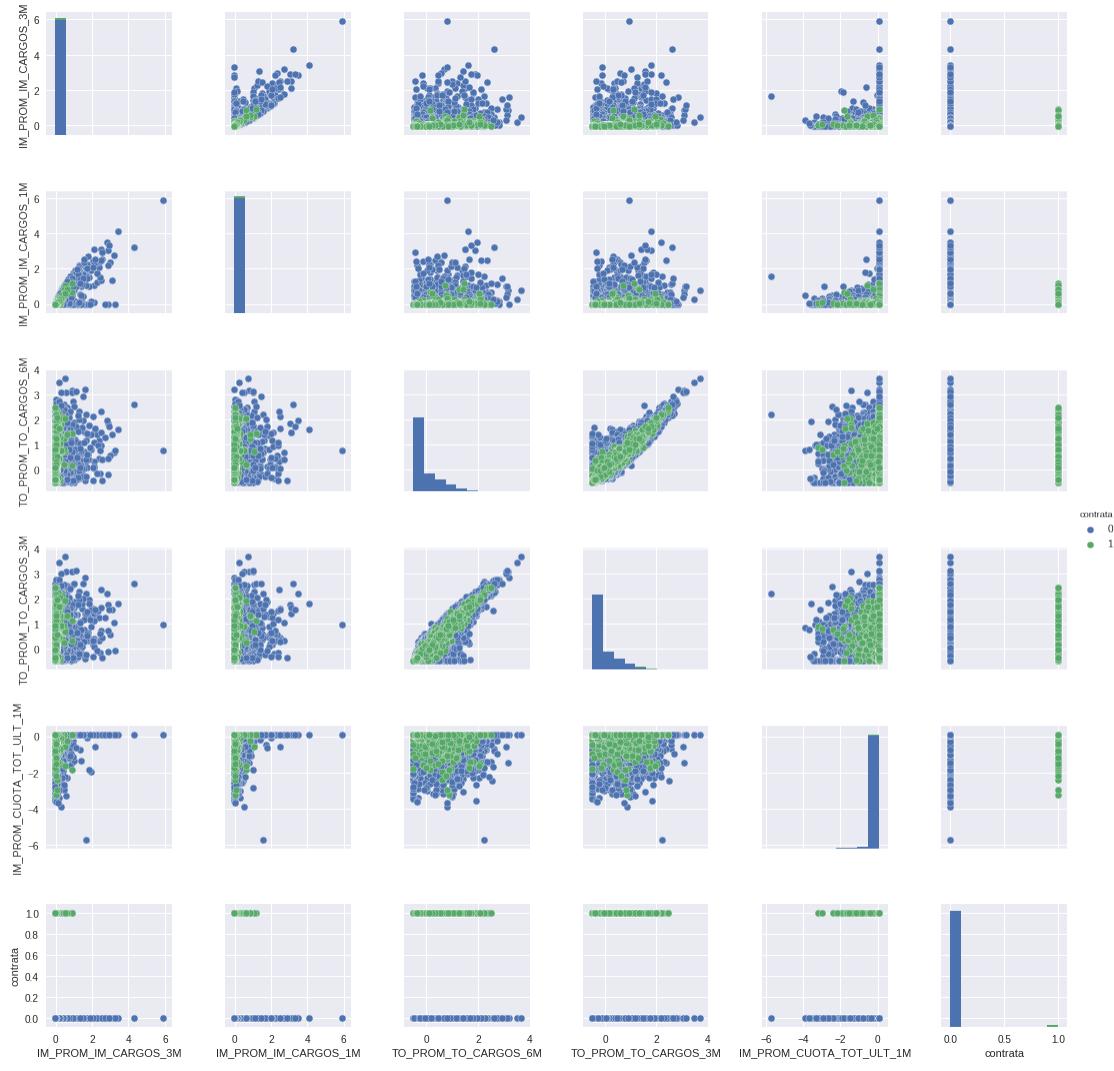


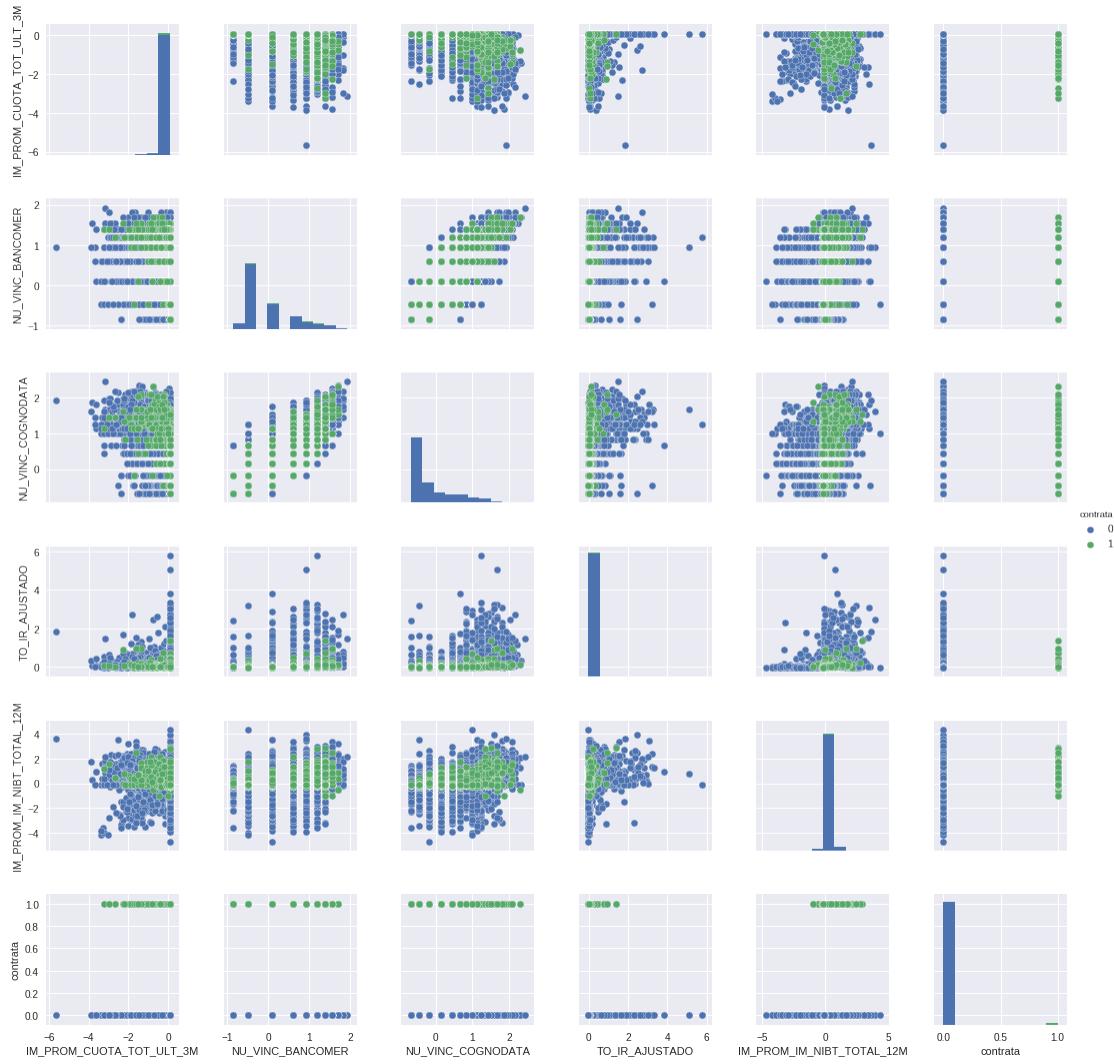






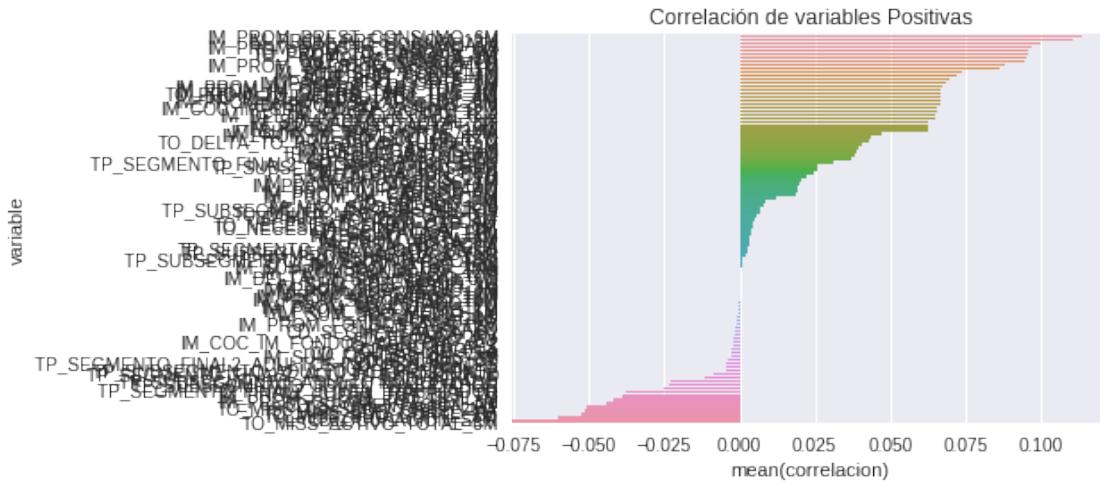






## 35 Anexo4

### Selección de variables por correlación



```
CPU times: user 2.14 s, sys: 404 ms, total: 2.54 s
Wall time: 2.12 s
```

## 36 Anexo5

### Selección de variables correlación con el target

Número de Variables: 32

Out[117] :

		variable	correlacion
17		IM_PROM_PREST_CONSUMO_6M	0.113385
2		IM_PROM_PREST_NOM_12M	0.110579
66		NU_VINC_COGNODATA	0.100227
24		IM_PROM_PREST_CONSUMO_3M	0.097032
48		TO_PROM_TO_CARGOS_6M	0.096008
49		TO_PROM_TO_CARGOS_3M	0.095624
23		IM_PROM_NOMINA_3M	0.095091
65		NU_VINC_BANCOMER	0.094894
33		IM_PROM_PREST_CONSUMO_1M	0.087862
32		IM_PROM_ACTIVO_1M	0.086590
34		IM_SUM_SDO_CORTE_1M	0.074010
60		TO_PREST_CONSUMO	0.072303
19		IM_SUM_SDO_CORTE_3M	0.069643
30		IM_PROM_SDO_CORTE_1M	0.068497
50		IM_PROM_IM_OPERS_TARJ_TDC_1M	0.067509
51		IM_PROM_IM_OPERS_TARJ_TDC_3M	0.066624
52		TO_PROM_TO_OPERS_TARJ_TDC_3M	0.066624
62		NU_REGULARIDAD_TDC_ULT_6M	0.066578

53	IM_PROM_IM_OPERS_TARJ_TDC_6M	0.066578
18	IM_PROM_SDO_CORTE_3M	0.066542
41	IM_CO_CIM_SDO_CORTE_ACTIVO_0_2	0.065833
15	IM_PROM_ACTIVO_12M	0.065814
42	IM_DELTA_IM_SDO_CORTE_0_2	0.064993
28	IM_SUM_ACTIVO_TOTAL_3M	0.064769
98	CD_SEGMENTACION_OT	0.062741
7	IM_PROM_SDO_CORTE_12M	0.062542
37	IM_SUM_IM_SDO_CORTE_2MA	0.062384
38	TO_MISS_TO_SDO_CORTE_2MA	-0.050689
35	TO_MISS_SDO_CORTE_1M	-0.051625
20	TO_MISS_SDO_CORTE_3M	-0.052575
106	CD_OCUPACION_SER	-0.060371
29	TO_MISS_ACTIVO_TOTAL_3M	-0.075900

## 37 Anexo6

### Selección de variables por importancia de variables Random Forest

Número de Variables: 36

Out [146] :

	feature	v_importance
73	TO_NECESIDAD_FINAN_CAP_1M	0.028514
48	TO_PROM_TO_CARGOS_6M	0.028227
49	TO_PROM_TO_CARGOS_3M	0.027313
74	TO_NECESIDAD_FINAN_CAP_3M	0.026983
69	IM_PROM_IM_NIBT_TOTAL_12M	0.025722
71	IM_PROM_GASTOS_1M	0.024893
45	IM_PROM_IM_CARGOS_6M	0.024427
67	TO_IR_AJUSTADO	0.024380
72	IM_PROM_GASTOS_3M	0.024120
47	IM_PROM_IM_CARGOS_1M	0.023777
44	IM_PROM_IM_CARGOS_12M	0.023657
46	IM_PROM_IM_CARGOS_3M	0.023180
39	IM_SUM_IM_SDO_MEDIO_2MA	0.022450
0	IM_MAX_SDO_MEDIO_12M	0.022443
16	IM_PROM_VISTA_6M	0.021906
9	IM_PROM_VISTA_12M	0.021570
22	IM_PROM_VISTA_3M	0.021410
27	IM_SUM_SDO_MEDIO_3M	0.021339
21	IM_PROM_SDO_MEDIO_3M	0.021282
36	IM_SUM_SDO_MEDIO_1M	0.021225
8	IM_PROM_SDO_MEDIO_12M	0.020639
31	IM_PROM_SDO_MEDIO_1M	0.020201
70	TO_RANGO_EDAD	0.018720
66	NU_VINC_COGNODATA	0.018047
65	NU_VINC_BANCOMER	0.016553

68	TP_NIVEL_IR_AJUSTADO	0.015685
15	IM_PROM_ACTIVO_12M	0.015298
28	IM_SUM_ACTIVO_TOTAL_3M	0.014075
19	IM_SUM_SDO_CORTE_3M	0.011742
24	IM_PROM_PREST_CONSUMO_3M	0.011444
75	TO_DELTA_NE_C_FIN_CAP_0_2	0.011259
34	IM_SUM_SDO_CORTE_1M	0.011229
60	TO_PREST_CONSUMO	0.011150
37	IM_SUM_IM_SDO_CORTE_2MA	0.010963
91	CD_ESTADO_OT	0.010873
104	CD_OCUPACION_OTR	0.010053

## 38 Anexo7

### Selección de variables por estadístico F

Número de Variables: 65

Out [152] :

	Attribute	F Score	P Value
17	IM_PROM_PREST_CONSUMO_6M	1837.134325	0.000000e+00
2	IM_PROM_PREST_NOM_12M	1746.211533	0.000000e+00
66	NU_VINC_COGNODATA	1431.418713	1.158220e-311
24	IM_PROM_PREST_CONSUMO_3M	1340.745299	3.764763e-292
48	TO_PROM_TO_CARGOS_6M	1312.337169	4.916031e-286
49	TO_PROM_TO_CARGOS_3M	1301.775504	9.241268e-284
23	IM_PROM_NOMINA_3M	1287.173769	1.288396e-280
65	NU_VINC_BANCOMER	1281.776133	1.872654e-279
33	IM_PROM_PREST_CONSUMO_1M	1097.429181	1.003492e-239
32	IM_PROM_ACTIVO_1M	1065.658664	7.148593e-233
29	TO_MISS_ACTIVO_TOTAL_3M	817.348257	2.976870e-179
34	IM_SUM_SDO_CORTE_1M	776.924877	1.633750e-170
60	TO_PREST_CONSUMO	741.318027	8.200389e-163
19	IM_SUM_SDO_CORTE_3M	687.509429	3.594019e-151
30	IM_PROM_SDO_CORTE_1M	664.956736	2.733112e-146
50	IM_PROM_IM_OPERS_TARJ_TDC_1M	645.837714	3.761724e-142
51	IM_PROM_IM_OPERS_TARJ_TDC_3M	628.932089	1.720283e-138
52	TO_PROM_TO_OPERS_TARJ_TDC_3M	628.932089	1.720283e-138
53	IM_PROM_IM_OPERS_TARJ_TDC_6M	628.056134	2.662366e-138
62	NU_REGULARIDAD_TDC_ULT_6M	628.056134	2.662366e-138
18	IM_PROM_SDO_CORTE_3M	627.372304	3.744005e-138
41	IM_CO_C_IM_SDO_CORTE_ACTIVO_0_2	614.021173	2.913616e-135
15	IM_PROM_ACTIVO_12M	613.661598	3.485789e-135
42	IM_DELTA_IM_SDO_CORTE_0_2	598.381530	7.105031e-132
28	IM_SUM_ACTIVO_TOTAL_3M	594.243662	5.595073e-131
98	CD_SEGMENTACION_OT	557.468790	5.184045e-123
7	IM_PROM_SDO_CORTE_12M	553.926226	3.035862e-122
37	IM_SUM_IM_SDO_CORTE_2MA	551.116840	1.233242e-121

106	CD_OCUPACION_SER	515.999314	5.037065e-114
20	TO_MISS_SDO_CORTE_3M	390.987838	6.620198e-87
35	TO_MISS_SDO_CORTE_1M	376.959912	7.355277e-84
38	TO_MISS_TO_SDO_CORTE_2MA	363.378067	6.547039e-81
69	IM_PROM_IM_NIBT_TOTAL_12M	314.960359	2.164682e-70
99	CD_SEGMENTACION_R1	275.146409	9.815817e-62
94	CD_EDO_CIVIL_C	265.333550	1.338138e-59
61	TO_DELTA_TO_PREST_CONSUMO_0_2	261.603806	8.668424e-59
64	IM_PROM CUOTA_TOT_ULT_3M	247.305303	1.120261e-55
3	IM_PROM_AUTO_12M	230.103166	6.221652e-52
72	IM_PROM_GASTOS_3M	219.835147	1.071120e-49
96	CD_EDO_CIVIL_S	214.562072	1.507745e-48
68	TP_NIVEL_IR_AJUSTADO	211.769959	6.116833e-48
71	IM_PROM_GASTOS_1M	206.229458	9.852488e-47
63	IM_PROM CUOTA_TOT_ULT_1M	201.075875	1.307536e-45
102	CD_OCUPACION_EDU	196.946839	1.038136e-44
78	TP_SEGMENTO_FINAL2_HOGARES CON HIJOS	135.957454	2.107373e-31
85	TP_SUBSEGMENTO_OTROSEGM	94.638284	2.322877e-22
105	CD_OCUPACION_PEN	94.072650	3.090612e-22
79	TP_SEGMENTO_FINAL2_JOVEN TRABAJADOR	89.426388	3.229057e-21
104	CD_OCUPACION_OTR	87.061002	1.066803e-20
84	TP_SUBSEGMENTO_JOVEN TRABAJADOR	77.684635	1.222107e-18
82	TP_SUBSEGMENTO_ADULTO NO JUBILADO	72.807484	1.443195e-17
4	IM_PROM_PPIS_12M	68.975385	1.005684e-16
47	IM_PROM_IM_CARGOS_1M	57.425731	3.531134e-14
44	IM_PROM_IM_CARGOS_12M	54.278217	1.749644e-13
67	TO_IR_AJUSTADO	52.922236	3.488099e-13
45	IM_PROM_IM_CARGOS_6M	52.585124	4.140982e-13
46	IM_PROM_IM_CARGOS_3M	49.261327	2.250403e-12
101	CD_SEXO_M	19.988043	7.798856e-06
100	CD_SEXO_F	19.988043	7.798856e-06
76	TP_SEGMENTO_FINAL2_ADULTO EN PLENITUD	11.052555	8.858760e-04
88	CD_ESTADO_EM	10.125958	1.462273e-03
0	IM_MAX_SDO_MEDIO_12M	9.266829	2.333843e-03
87	TP_SUBSEGMENTO_PAREJAS SENIOR	7.069350	7.842254e-03
75	TO_DELTA_NECK_FIN_CAP_0_2	6.331641	1.186136e-02
26	TO_MISS_FONDOS_AES_3M	4.325221	3.755333e-02

## 39 Anexo8

### Selección de variables por AUC-Variables

Número de Variables: 43

Out[116]:

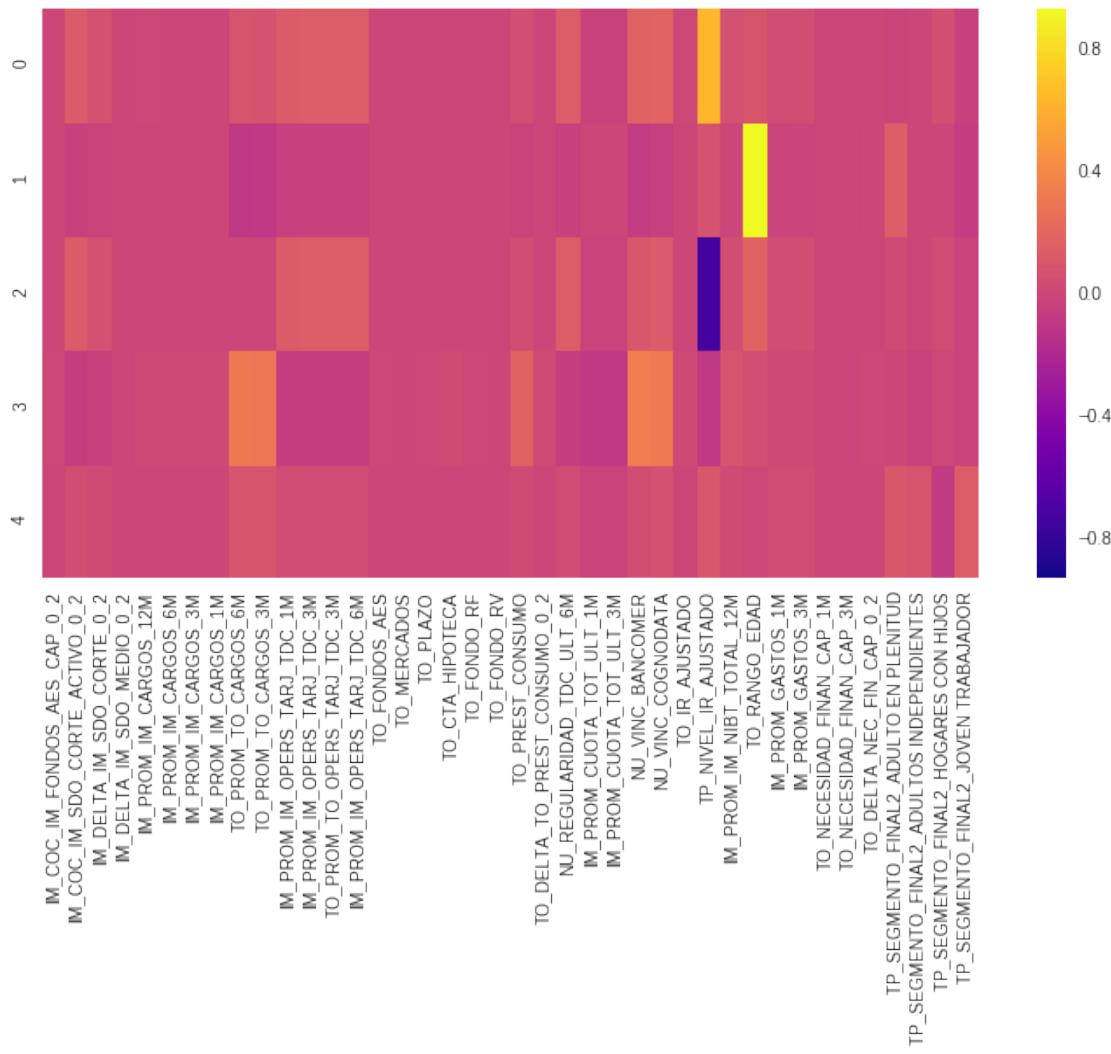
	variable	roc
45	IM_PROM_IM_CARGOS_6M	0.746829
46	IM_PROM_IM_CARGOS_3M	0.744452

44	IM_PROM_IM_CARGOS_12M	0.743873
47	IM_PROM_IM_CARGOS_1M	0.739342
48	TO_PROM_TO_CARGOS_6M	0.732587
72	IM_PROM_GASTOS_3M	0.732213
49	TO_PROM_TO_CARGOS_3M	0.731875
71	IM_PROM_GASTOS_1M	0.731477
66	NU_VINC_COGNODATA	0.720574
15	IM_PROM_ACTIVO_12M	0.716312
65	NU_VINC_BANCOMER	0.709723
73	TO_NECESSIDAD_FINAN_CAP_1M	0.705235
28	IM_SUM_ACTIVO_TOTAL_3M	0.699075
67	TO_IR_AJUSTADO	0.684564
74	TO_NECESSIDAD_FINAN_CAP_3M	0.683527
32	IM_PROM_ACTIVO_1M	0.670085
17	IM_PROM_PREST_CONSUMO_6M	0.665963
24	IM_PROM_PREST_CONSUMO_3M	0.647173
2	IM_PROM_PREST_NOM_12M	0.644600
0	IM_MAX_SDO_MEDIO_12M	0.632896
19	IM_SUM_SDO_CORTE_3M	0.632856
34	IM_SUM_SDO_CORTE_1M	0.631903
53	IM_PROM_IM_OPERS_TARJ_TDC_6M	0.625585
62	NU_REGULARIDAD_TDC_ULT_6M	0.625585
18	IM_PROM_SDO_CORTE_3M	0.624421
30	IM_PROM_SDO_CORTE_1M	0.623725
51	IM_PROM_IM_OPERS_TARJ_TDC_3M	0.623676
52	TO_PROM_TO_OPERS_TARJ_TDC_3M	0.623676
50	IM_PROM_IM_OPERS_TARJ_TDC_1M	0.622931
7	IM_PROM_SDO_CORTE_12M	0.622781
33	IM_PROM_PREST_CONSUMO_1M	0.620650
41	IM_COE_IM_SDO_CORTE_ACTIVO_0_2	0.616366
37	IM_SUM_IM_SDO_CORTE_2MA	0.616200
16	IM_PROM_VISTA_6M	0.615533
23	IM_PROM_NOMINA_3M	0.614473
9	IM_PROM_VISTA_12M	0.613844
22	IM_PROM_VISTA_3M	0.609063
8	IM_PROM_SDO_MEDIO_12M	0.608905
68	TP_NIVEL_IR_AJUSTADO	0.606427
27	IM_SUM_SDO_MEDIO_3M	0.606261
21	IM_PROM_SDO_MEDIO_3M	0.604394
39	IM_SUM_IM_SDO_MEDIO_2MA	0.601452
98	CD_SEGMENTACION_OT	0.600417

## 40 Anexo9

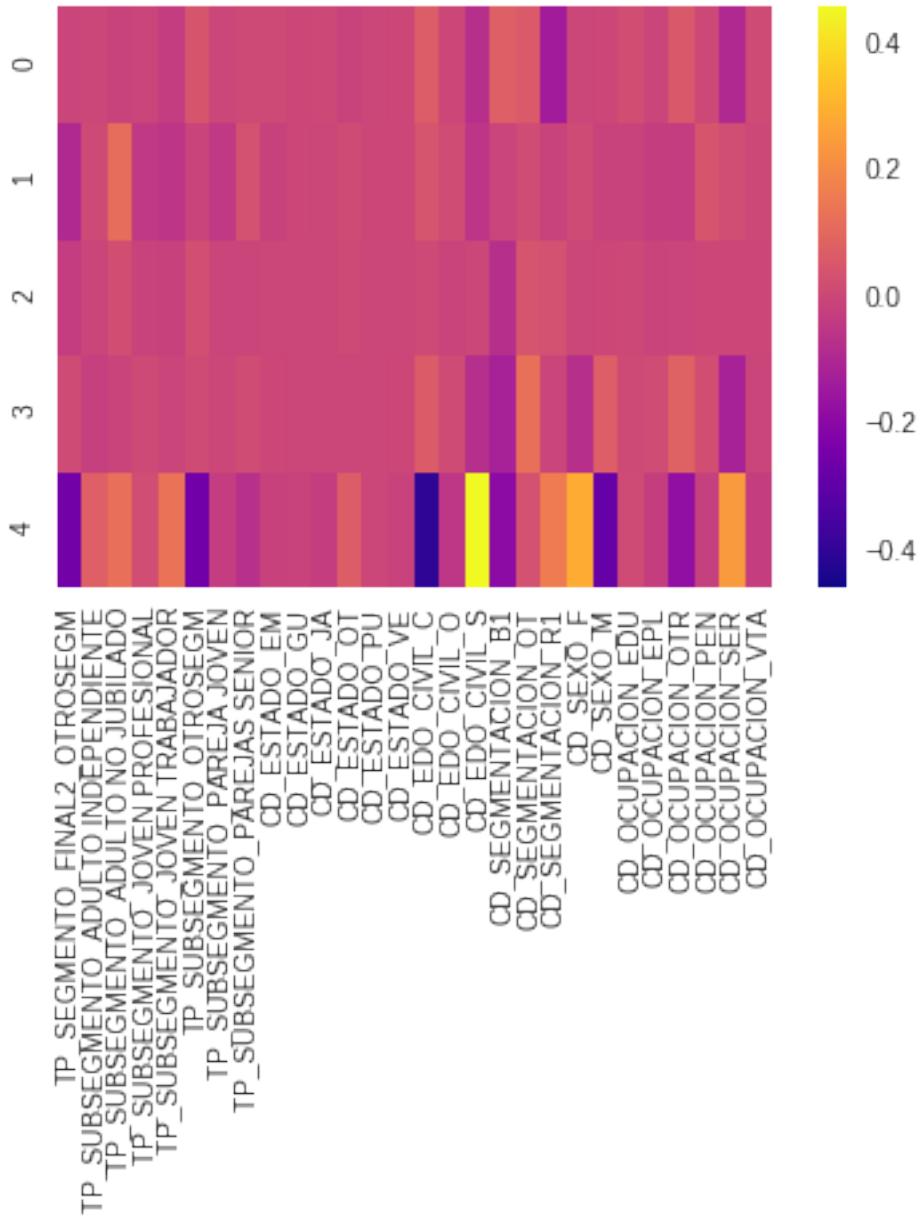
**Explicación de PCA con respecto a variables originales, en las primeras 5 componentes**  
 Gráfica de las variables 40-80

Out [120] : <matplotlib.axes.\_subplots.AxesSubplot at 0x7f1f18f0fc88>



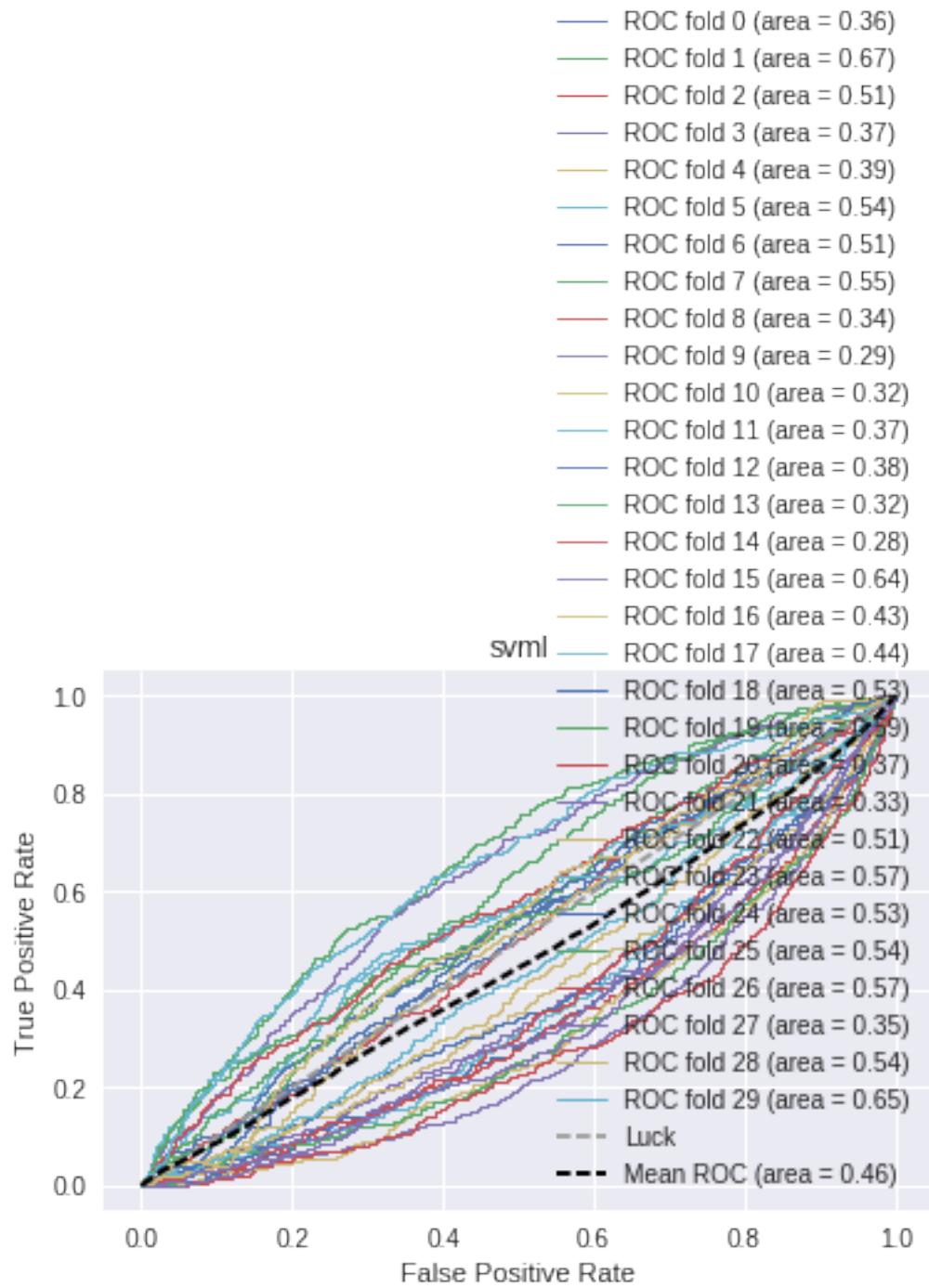
Variables de 80 a 120

Out [121]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7f1f3022e278>

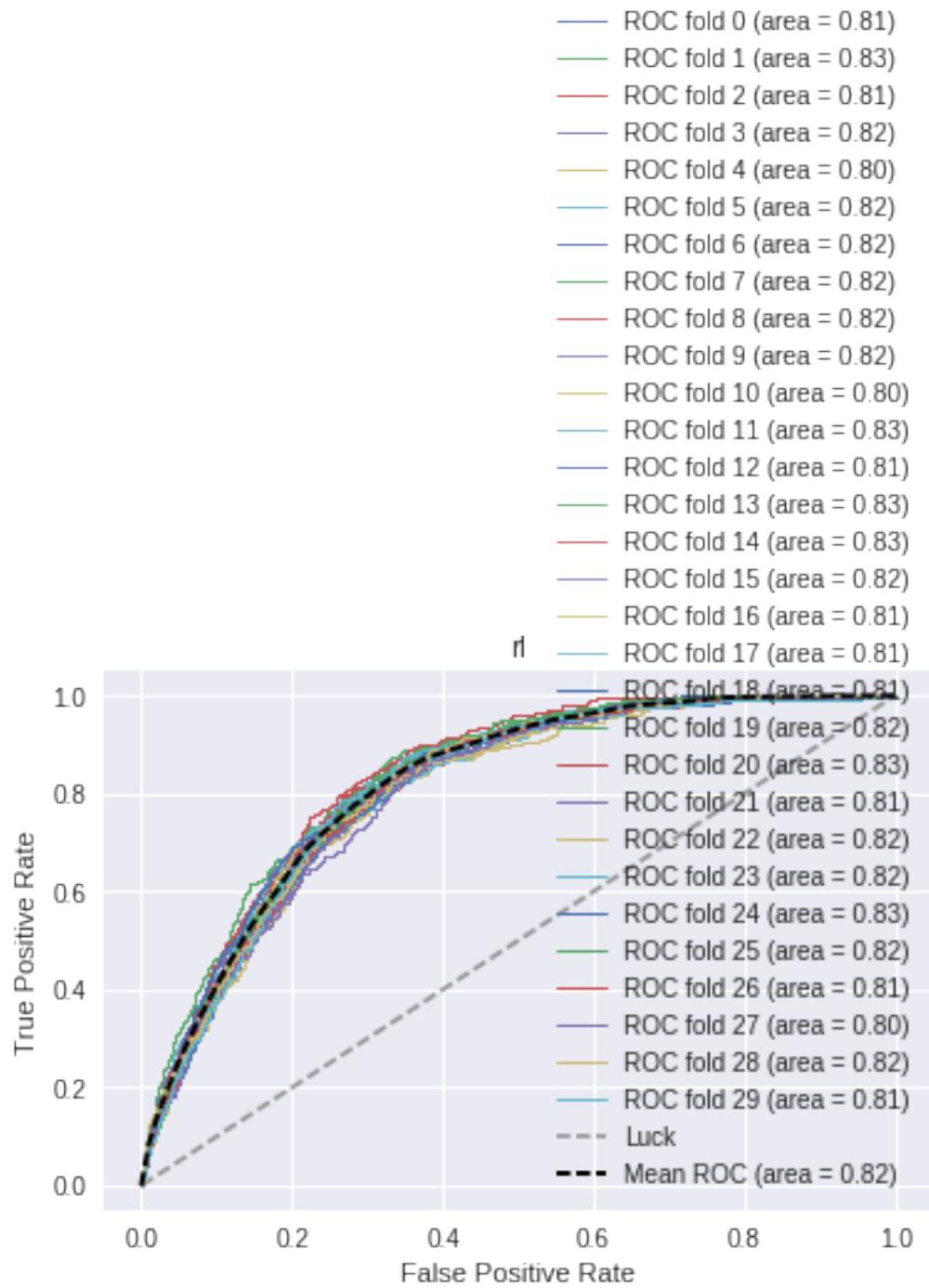


## 41 Anexo10

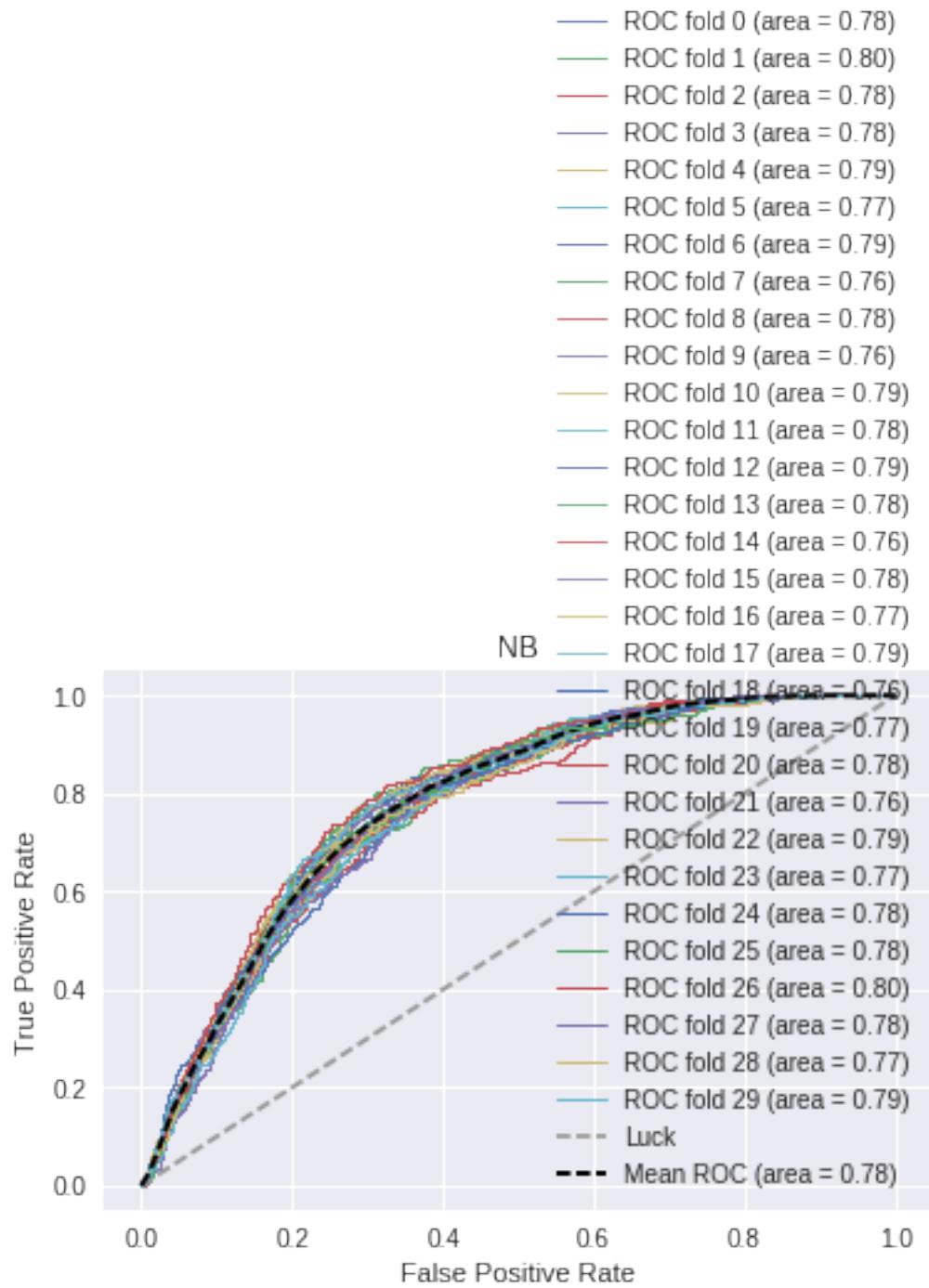
Desempeño de cada modelo haciendo cross validation con 30 iteraciones Out [91] :



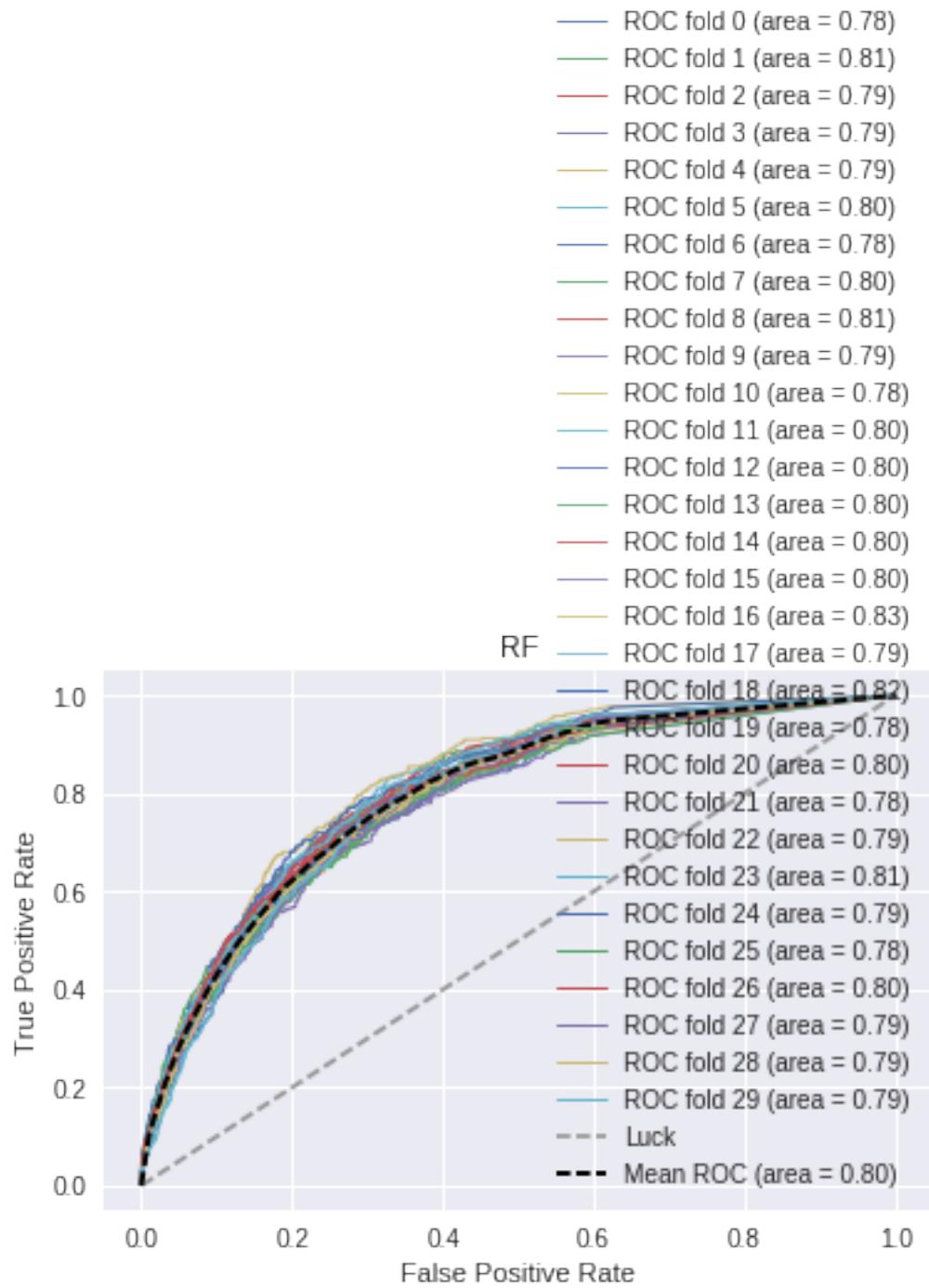
Curvas ROC por iteracion del modelo svml, maquina de soporte vectorial kernel lineal,AUC promedio=0.46 Out [92] :



Curvas ROC por iteracion del modelo rl, Regresión Logística,AUC promedio=0.82 Out [ 93 ] :



Curvas ROC por iteración del modelo NB, Naive Bayes,AUC promedio=0.78 [Out \[94\]](#) :



Curvas ROC por iteración del modelo RF, Random Forest,AUC promedio=0.80

## 42 Apéndice A

Diagrama de caja y brazos de cada modelo, construido con la selección de variables de correlación con el target con respecto a su AUC. Out [45] :

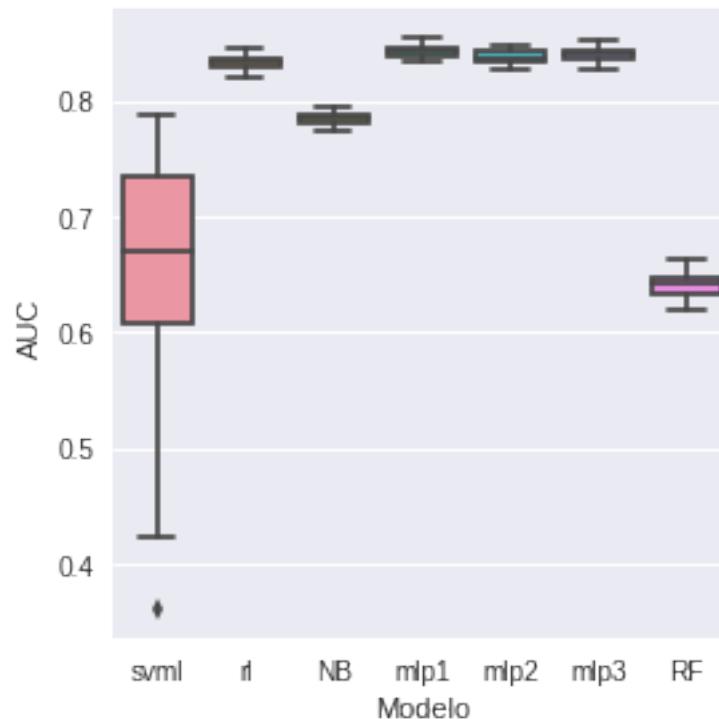


Diagrama de caja y brazos de cada modelo, construido con la selección de variables más importantes de acuerdo con el árbol de decisión con respecto a su AUC. Out [46] :

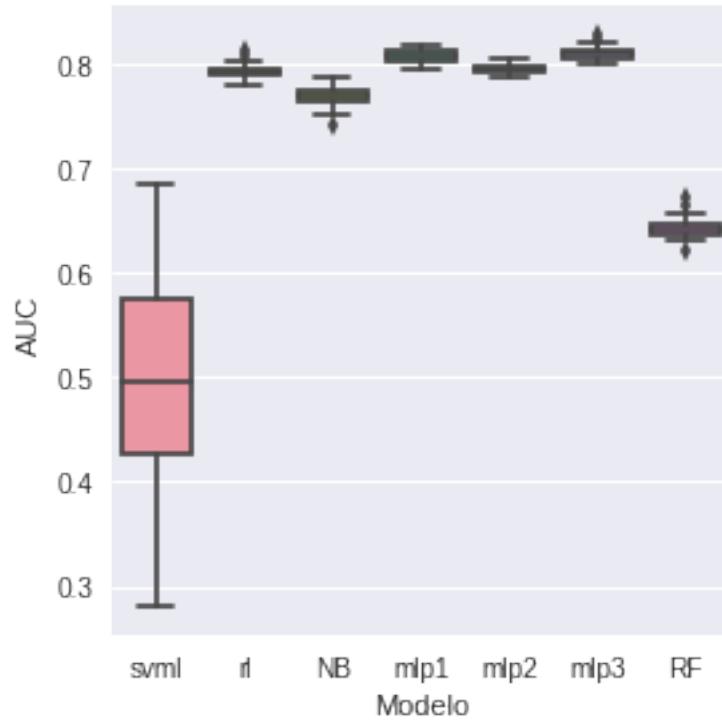


Diagrama de caja y brazos de cada modelo, construido con la selección de variables relevantes del ANOVA-Fvalue con respecto a su AUC. [Out \[48\]](#) :

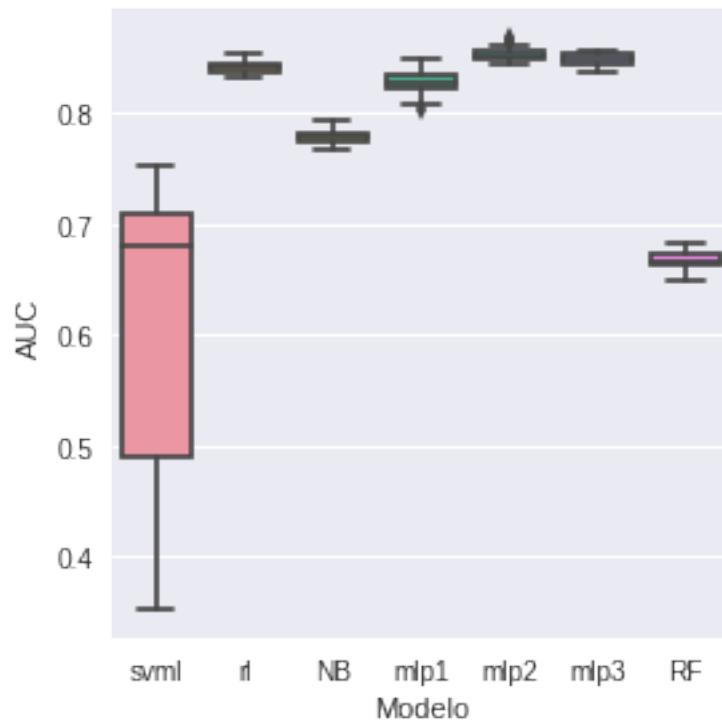
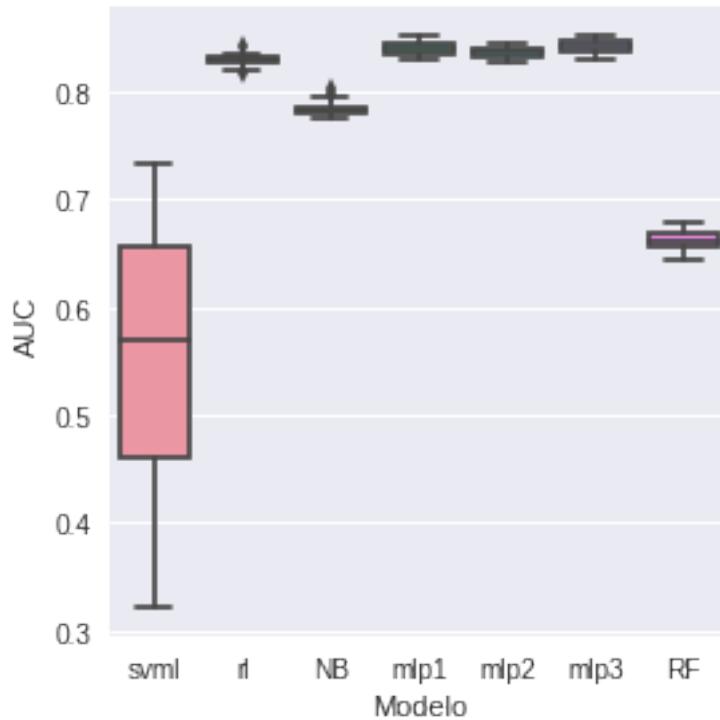


Diagrama de caja y brazos de cada modelo, construido con la selección de variables AUC o índice ROC con respecto a su AUC. Out [51] :



## 43 Apéndice B

### 1.-Output gridsearchcv de SVM

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits [CV] C=0.1, gamma=0.1, kernel=linear
..... [CV] C=0.1, gamma=0.1, kernel=linear, score=0.6701451343741653,
total= 5.1s [CV] C=0.1, gamma=0.1, kernel=linear ..... [Parallel(n_jobs=1)]:
Done 1 out of 1 | elapsed: 8.0s remaining: 0.0s [CV] C=0.1, gamma=0.1,
kernel=linear, score=0.5584316983605012, total= 5.2s [CV] C=0.1, gamma=0.1, kernel=linear
..... [Parallel(n_jobs=1)]:
Done 2 out of 2 | elapsed: 16.0s
remaining: 0.0s [CV] C=0.1, gamma=0.1, kernel=linear, score=0.5481613970284386,
total= 5.2s [CV] C=0.1, gamma=0.01, kernel=linear ..... [CV]
C=0.1, gamma=0.01, kernel=linear, score=0.6701451343741653, total= 5.2s [CV] C=0.1,
gamma=0.01, kernel=linear ..... [CV] C=0.1, gamma=0.01, kernel=linear
..... [CV] C=0.1, gamma=0.01, kernel=linear, score=0.5584316983605012, total= 5.1s [CV]
C=0.1, gamma=0.01, kernel=linear ..... [CV] C=0.1, gamma=0.01, kernel=linear, score=0.5481613970284386,
total= 5.1s [CV] C=0.1, gamma=0.001, kernel=linear ..... [CV]
C=0.1, gamma=0.001, kernel=linear, score=0.6701451343741653, total= 5.4s [CV] C=0.1,
```

gamma=0.001, kernel=linear ..... [CV] C=0.1, gamma=0.001, kernel=linear, score=0.5584316983605012, total= 5.2s [CV] C=0.1, gamma=0.001, kernel=linear ..... [CV] C=0.1, gamma=0.001, kernel=linear, score=0.5481613970284386, total= 5.8s [CV] C=0.1, gamma=0.0001, kernel=linear ..... [CV] C=0.1, gamma=0.0001, kernel=linear, score=0.6701451343741653, total= 5.7s [CV] C=0.1, gamma=0.0001, kernel=linear ..... [CV] C=0.1, gamma=0.0001, kernel=linear, score=0.5584316983605012, total= 5.6s [CV] C=0.1, gamma=0.0001, kernel=linear ..... [CV] C=0.1, gamma=0.0001, kernel=linear, score=0.5481613970284386, total= 5.5s [CV] C=0.5, gamma=0.1, kernel=linear ..... [CV] C=0.5, gamma=0.1, kernel=linear, score=0.3933815385838305, total= 7.9s [CV] C=0.5, gamma=0.1, kernel=linear ..... [CV] C=0.5, gamma=0.1, kernel=linear, score=0.6174610983841348, total= 7.4s [CV] C=0.5, gamma=0.1, kernel=linear ..... [CV] C=0.5, gamma=0.1, kernel=linear, score=0.6334799420852012, total= 7.0s [CV] C=0.5, gamma=0.01, kernel=linear ..... [CV] C=0.5, gamma=0.01, kernel=linear, score=0.3933815385838305, total= 7.7s [CV] C=0.5, gamma=0.01, kernel=linear ..... [CV] C=0.5, gamma=0.01, kernel=linear, score=0.6174610983841348, total= 7.1s [CV] C=0.5, gamma=0.01, kernel=linear ..... [CV] C=0.5, gamma=0.01, kernel=linear, score=0.6334799420852012, total= 7.0s [CV] C=0.5, gamma=0.001, kernel=linear ..... [CV] C=0.5, gamma=0.001, kernel=linear, score=0.3933815385838305, total= 7.0s [CV] C=0.5, gamma=0.001, kernel=linear ..... [CV] C=0.5, gamma=0.001, kernel=linear, score=0.6174610983841348, total= 7.2s [CV] C=0.5, gamma=0.001, kernel=linear ..... [CV] C=0.5, gamma=0.001, kernel=linear, score=0.6334799420852012, total= 9.1s [CV] C=0.5, gamma=0.0001, kernel=linear ..... [CV] C=0.5, gamma=0.0001, kernel=linear, score=0.3933815385838305, total= 7.6s [CV] C=0.5, gamma=0.0001, kernel=linear ..... [CV] C=0.5, gamma=0.0001, kernel=linear, score=0.6174610983841348, total= 7.3s [CV] C=0.5, gamma=0.0001, kernel=linear ..... [CV] C=0.5, gamma=0.0001, kernel=linear, score=0.6334799420852012, total= 7.1s [CV] C=1.0, gamma=0.1, kernel=linear ..... [CV] C=1.0, gamma=0.1, kernel=linear, score=0.5875109722624114, total= 11.5s [CV] C=1.0, gamma=0.1, kernel=linear ..... [CV] C=1.0, gamma=0.1, kernel=linear, score=0.42917475149807977, total= 11.2s [CV] C=1.0, gamma=0.1, kernel=linear ..... [CV] C=1.0, gamma=0.1, kernel=linear, score=0.6116885972542876, total= 11.1s [CV] C=1.0, gamma=0.01, kernel=linear ..... [CV] C=1.0, gamma=0.01, kernel=linear, score=0.5875109722624114, total= 12.2s [CV] C=1.0, gamma=0.01, kernel=linear ..... [CV] C=1.0, gamma=0.01, kernel=linear, score=0.42917475149807977, total= 10.2s [CV] C=1.0, gamma=0.01, kernel=linear ..... [CV] C=1.0, gamma=0.01, kernel=linear, score=0.6116885972542876, total= 11.0s [CV] C=1.0, gamma=0.001, kernel=linear ..... [CV] C=1.0, gamma=0.001, kernel=linear, score=0.5875109722624114, total= 13.7s [CV] C=1.0, gamma=0.001, kernel=linear ..... [CV] C=1.0, gamma=0.001, kernel=linear, score=0.42917475149807977, total= 11.3s [CV] C=1.0, gamma=0.001, kernel=linear ..... [CV] C=1.0, gamma=0.001, kernel=linear, score=0.6116885972542876, total= 11.2s [CV] C=1.0, gamma=0.0001, kernel=linear ..... [CV] C=1.0, gamma=0.0001, kernel=linear, score=0.5875109722624114, total= 12.9s [CV] C=1.0, gamma=0.0001, kernel=linear ..... [CV] C=1.0,

```

gamma=0.0001, kernel=linear, score=0.42917475149807977, total= 10.3s [CV] C=1.0,
gamma=0.0001, kernel=linear ..... [CV] C=1.0, gamma=0.0001, kernel=linear, score=0.6116885972542876, total= 10.6s [CV] C=1.5, gamma=0.1, kernel=linear ..... [CV] C=1.5, gamma=0.1, kernel=linear, score=0.640538625998894, total= 12.7s [CV] C=1.5, gamma=0.1, kernel=linear ..... [CV] C=1.5, gamma=0.1, kernel=linear, score=0.7336473063469079, total= 13.4s [CV] C=1.5, gamma=0.1, kernel=linear ..... [CV] C=1.5, gamma=0.1, kernel=linear, score=0.5052276771430319, total= 12.8s [CV] C=1.5, gamma=0.01, kernel=linear ..... [CV] C=1.5, gamma=0.01, kernel=linear, score=0.640538625998894, total= 13.2s [CV] C=1.5, gamma=0.01, kernel=linear ..... [CV] C=1.5, gamma=0.01, kernel=linear, score=0.7336473063469079, total= 14.1s [CV] C=1.5, gamma=0.01, kernel=linear ..... [CV] C=1.5, gamma=0.01, kernel=linear, score=0.5052276771430319, total= 13.0s [CV] C=1.5, gamma=0.001, kernel=linear ..... [CV] C=1.5, gamma=0.001, kernel=linear, score=0.640538625998894, total= 13.5s [CV] C=1.5, gamma=0.001, kernel=linear ..... [CV] C=1.5, gamma=0.001, kernel=linear, score=0.7336473063469079, total= 13.7s [CV] C=1.5, gamma=0.001, kernel=linear ..... [CV] C=1.5, gamma=0.001, kernel=linear, score=0.5052276771430319, total= 13.6s [CV] C=1.5, gamma=0.0001, kernel=linear ..... [CV] C=1.5, gamma=0.0001, kernel=linear, score=0.640538625998894, total= 14.1s [CV] C=1.5, gamma=0.0001, kernel=linear ..... [CV] C=1.5, gamma=0.0001, kernel=linear, score=0.7336473063469079, total= 14.0s [CV] C=1.5, gamma=0.0001, kernel=linear ..... [CV] C=1.5, gamma=0.0001, kernel=linear, score=0.5052276771430319, total= 14.9s [Parallel(n_jobs=1)]: Done 48 out of 48 | elapsed: 11.0min finished Out[60]: {'C': 1.5, 'gamma': 0.1, 'kernel': 'linear'}

```

## 2.-Output gridsearchcv de Regresión Logística

```

Fitting 3 folds for each of 12 candidates, totalling 36 fits [CV] C=0.01,
penalty=l1 ..... [CV] ..... C=0.01,
penalty=l1, score=0.7773153594224772, total= 0.9s [CV] C=0.01, penalty=l1 ..... [Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed:
0.9s remaining: 0.0s [CV] ..... C=0.01, penalty=l1, score=0.7828854021980216, total= 0.2s [CV]
C=0.01, penalty=l1 ..... [Parallel(n_jobs=1)]: Done 2 out of 2 | elapsed: 1.1s remaining: 0.0s [CV] ..... C=0.01, penalty=l1, score=0.7987257311142174,
total= 0.2s [CV] C=0.01, penalty=l2 ..... [CV] ..... C=0.01, penalty=l2, score=0.8049457700019922, total= 0.5s
[CV] C=0.01, penalty=l2 ..... [CV] ..... C=0.01, penalty=l2, score=0.805355596359066, total= 0.3s [CV]
C=0.01, penalty=l2 ..... [CV] ..... C=0.01, penalty=l2, score=0.8268656484393421, total= 0.3s
[CV] C=0.1, penalty=l1 ..... [CV] ..... C=0.1, penalty=l1, score=0.7991038733358635, total= 0.3s
[CV] C=0.1, penalty=l1 ..... [CV] ..... C=0.1, penalty=l1, score=0.798828363861681, total= 0.3s [CV]
C=0.1, penalty=l1 ..... [CV] ..... C=0.1, penalty=l1, score=0.8223480609454701, total= 0.3s
[CV] C=0.1, penalty=l2 ..... [CV] ..... C=0.1, penalty=l2, score=0.7988058470586582, total= 0.3s
[CV] C=0.1, penalty=l2 ..... [CV] ..... C=0.1, penalty=l2, score=0.7982814270660057, total= 0.4s

```

```

[CV] C=0.1, penalty=l2 ..... [CV]
..... C=0.1, penalty=l2, score=0.8230060787706118, total= 0.3s [CV]
C=1, penalty=l1 ..... [CV]
..... C=1, penalty=l1, score=0.7970536779314419, total= 0.4s [CV]
C=1, penalty=l1 ..... [CV]
..... C=1, penalty=l1, score=0.7958167198827406, total= 0.4s [CV]
C=1, penalty=l1 ..... [CV]
..... C=1, penalty=l1, score=0.8213044383593787, total= 0.4s [CV]
C=1, penalty=l2 ..... [CV]
..... C=1, penalty=l2, score=0.7968950012044648, total= 0.4s [CV]
C=1, penalty=l2 ..... [CV]
..... C=1, penalty=l2, score=0.7956465118402981, total= 0.4s [CV]
C=1, penalty=l2 ..... [CV]
..... C=1, penalty=l2, score=0.8212263458046415, total= 0.3s [CV]
C=10, penalty=l1 ..... [CV]
..... C=10, penalty=l1, score=0.7966828869543131, total= 0.4s [CV]
C=10, penalty=l1 ..... [CV]
..... C=10, penalty=l1, score=0.7953511526114576, total= 0.4s [CV]
C=10, penalty=l1 ..... [CV]
..... C=10, penalty=l1, score=0.821030082090092, total= 0.3s [CV]
C=10, penalty=l2 ..... [CV]
..... C=10, penalty=l2, score=0.7966706484285976, total= 0.3s [CV]
C=10, penalty=l2 ..... [CV]
..... C=10, penalty=l2, score=0.7953393722474785, total= 0.3s [CV]
C=10, penalty=l2 ..... [CV]
..... C=10, penalty=l2, score=0.8210201838891416, total= 0.3s [CV]
C=100, penalty=l1 ..... [CV]
..... C=100, penalty=l1, score=0.796646474310971, total= 0.4s [CV]
C=100, penalty=l1 ..... [CV]
..... C=100, penalty=l1, score=0.7953072495024011, total= 0.4s [CV]
[CV] C=100, penalty=l1 ..... [CV]
..... C=100, penalty=l1, score=0.8209989908085791, total= 0.4s [CV]
C=100, penalty=l2 ..... [CV]
..... C=100, penalty=l2, score=0.7966451414022296, total= 0.4s [CV] C=100,
penalty=l2 ..... [CV] ..... C=100,
penalty=l2, score=0.7953071280553498, total= 0.4s [CV] C=100, penalty=l2
..... [CV] ..... C=100, penalty=l2, score=0.8209978977557134, total= 0.3s [CV] C=100000.0, penalty=l1
..... [CV] .. C=100000.0, penalty=l1, score=0.796642536171508,
total= 0.4s [CV] C=100000.0, penalty=l1 ..... [CV] .
C=100000.0, penalty=l1, score=0.7953023308968221, total= 0.4s [CV] C=100000.0, penalty=l1
..... [CV] . C=100000.0, penalty=l1, score=0.8209954687493453,
total= 0.4s [CV] C=100000.0, penalty=l2 ..... [CV] .
C=100000.0, penalty=l2, score=0.7966416273700935, total= 0.3s [CV] C=100000.0, penalty=l2
..... [CV] . C=100000.0, penalty=l2, score=0.7953042133261177,
total= 0.3s [CV] C=100000.0, penalty=l2 ..... [CV] . C=100000.0,
penalty=l2, score=0.8209967439776886, total= 0.3s [Parallel(n_jobs=1)]: Done 36 out of 36 !
elapsed: 13.8s finished Out[63]: {'C': 0.01, 'penalty': 'l2'}

```

### 3.-Output gridsearchcv de mlp1

Fitting 3 folds for each of 12 candidates, totalling 36 fits [CV] activation=relu, hidden\_layer\_sizes=(3,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=relu, hidden\_layer\_sizes=(3,), max\_iter=40000, random\_state=0, verbose=0, score=0.7903407254028656, total= 2.9s [CV] activation=relu, hidden\_layer\_sizes=(3,), max\_iter=40000, random\_state=0, verbose=0 [Parallel(n\_jobs=1)]: Done 1 out of 1 | elapsed: 2.9s remaining: 0.0s [CV] activation=relu, hidden\_layer\_sizes=(3,), max\_iter=40000, random\_state=0, verbose=0, score=0.8162531617221725, total= 3.9s [CV] activation=relu, hidden\_layer\_sizes=(3,), max\_iter=40000, random\_state=0, verbose=0 [Parallel(n\_jobs=1)]: Done 2 out of 2 | elapsed: 6.8s remaining: 0.0s [CV] activation=relu, hidden\_layer\_sizes=(3,), max\_iter=40000, random\_state=0, verbose=0, score=0.8353283673512638, total= 3.3s [CV] activation=relu, hidden\_layer\_sizes=(10,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=relu, hidden\_layer\_sizes=(10,), max\_iter=40000, random\_state=0, verbose=0, score=0.8314291520228465, total= 3.7s [CV] activation=relu, hidden\_layer\_sizes=(10,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=relu, hidden\_layer\_sizes=(10,), max\_iter=40000, random\_state=0, verbose=0, score=0.8284546433519094, total= 3.8s [CV] activation=relu, hidden\_layer\_sizes=(10,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=relu, hidden\_layer\_sizes=(10,), max\_iter=40000, random\_state=0, verbose=0, score=0.8469690160198436, total= 4.7s [CV] activation=relu, hidden\_layer\_sizes=(50,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=relu, hidden\_layer\_sizes=(50,), max\_iter=40000, random\_state=0, verbose=0, score=0.8355366920694838, total= 9.9s [CV] activation=relu, hidden\_layer\_sizes=(50,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=relu, hidden\_layer\_sizes=(50,), max\_iter=40000, random\_state=0, verbose=0, score=0.8344461729058671, total= 10.1s [CV] activation=relu, hidden\_layer\_sizes=(50,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=relu, hidden\_layer\_sizes=(50,), max\_iter=40000, random\_state=0, verbose=0, score=0.8397720523263839, total= 15.5s [CV] activation=relu, hidden\_layer\_sizes=(100,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=relu, hidden\_layer\_sizes=(100,), max\_iter=40000, random\_state=0, verbose=0, score=0.8063907036643357, total= 56.2s [CV] activation=relu, hidden\_layer\_sizes=(100,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=relu, hidden\_layer\_sizes=(100,), max\_iter=40000, random\_state=0, verbose=0, score=0.8161411268173184, total= 49.2s [CV] activation=relu, hidden\_layer\_sizes=(100,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=relu, hidden\_layer\_sizes=(100,), max\_iter=40000, random\_state=0, verbose=0, score=0.8162342519167441, total= 1.1min [CV] activation=tanh, hidden\_layer\_sizes=(3,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(3,), max\_iter=40000, random\_state=0, verbose=0, score=0.7953601567888435, total= 2.9s [CV] activation=tanh, hidden\_layer\_sizes=(3,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(3,), max\_iter=40000, random\_state=0, verbose=0, score=0.807407565738378, total= 3.3s [CV] activation=tanh, hidden\_layer\_sizes=(3,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(3,), max\_iter=40000, random\_state=0, verbose=0, score=0.8203578545776962, total= 2.9s [CV] activation=tanh, hidden\_layer\_sizes=(10,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(10,), max\_iter=40000, random\_state=0, verbose=0, score=0.817630698972497, total= 5.0s [CV] activation=tanh, hidden\_layer\_sizes=(10,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(10,), max\_iter=40000, random\_state=0, verbose=0, score=0.8286460439048093, total= 7.7s [CV] activation=tanh, hidden\_layer\_sizes=(10,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(10,), max\_iter=40000, random\_state=0, verbose=0, score=0.8383108835456011, total= 4.9s [CV] ac-

```

tivation=tanh, hidden_layer_sizes=(50,), max_iter=40000, random_state=0, verbose=0 [CV]
activation=tanh, hidden_layer_sizes=(50,), max_iter=40000, random_state=0, verbose=0,
score=0.8195534198318403, total= 35.8s [CV] activation=tanh, hidden_layer_sizes=(50,),
max_iter=40000, random_state=0, verbose=0 [CV] activation=tanh, hidden_layer_sizes=(50,),
max_iter=40000, random_state=0, verbose=0, score=0.8162768438971825, total= 44.6s [CV]
activation=tanh, hidden_layer_sizes=(50,), max_iter=40000, random_state=0, verbose=0 [CV]
activation=tanh, hidden_layer_sizes=(50,), max_iter=40000, random_state=0, verbose=0,
score=0.8283670172755184, total= 1.0min [CV] activation=tanh, hidden_layer_sizes=(100,),
max_iter=40000, random_state=0, verbose=0 [CV] activation=tanh, hidden_layer_sizes=(100,),
max_iter=40000, random_state=0, verbose=0, score=0.7664598476994238, total= 2.8min [CV]
activation=tanh, hidden_layer_sizes=(100,), max_iter=40000, random_state=0, verbose=0 [CV]
activation=tanh, hidden_layer_sizes=(100,), max_iter=40000, random_state=0, verbose=0,
score=0.747921828419363, total= 3.2min [CV] activation=tanh, hidden_layer_sizes=(100,),
max_iter=40000, random_state=0, verbose=0 [CV] activation=tanh, hidden_layer_sizes=(100,),
max_iter=40000, random_state=0, verbose=0, score=0.768802740939366, total= 3.6min [CV]
activation=logistic, hidden_layer_sizes=(3,), max_iter=40000, random_state=0, verbose=0
[CV] activation=logistic, hidden_layer_sizes=(3,), max_iter=40000, random_state=0, verbose=0,
score=0.8095034696826272, total= 2.4s [CV] activation=logistic, hidden_layer_sizes=(3,),
max_iter=40000, random_state=0, verbose=0 [CV] activation=logistic, hidden_layer_sizes=(3,),
max_iter=40000, random_state=0, verbose=0, score=0.8078484792582404, total= 2.4s [CV]
activation=logistic, hidden_layer_sizes=(3,), max_iter=40000, random_state=0, verbose=0 [CV]
activation=logistic, hidden_layer_sizes=(3,), max_iter=40000, random_state=0, verbose=0,
score=0.8326410361558205, total= 2.9s [CV] activation=logistic, hidden_layer_sizes=(10,),
max_iter=40000, random_state=0, verbose=0 [CV] activation=logistic, hidden_layer_sizes=(10,),
max_iter=40000, random_state=0, verbose=0, score=0.8067191444955473, total= 2.2s [CV]
activation=logistic, hidden_layer_sizes=(10,), max_iter=40000, random_state=0, verbose=0 [CV]
activation=logistic, hidden_layer_sizes=(10,), max_iter=40000, random_state=0, verbose=0,
score=0.8059617993157916, total= 2.5s [CV] activation=logistic, hidden_layer_sizes=(10,),
max_iter=40000, random_state=0, verbose=0 [CV] activation=logistic, hidden_layer_sizes=(10,),
max_iter=40000, random_state=0, verbose=0, score=0.8281961974026757, total= 1.9s [CV]
activation=logistic, hidden_layer_sizes=(50,), max_iter=40000, random_state=0, verbose=0 [CV]
activation=logistic, hidden_layer_sizes=(50,), max_iter=40000, random_state=0, verbose=0,
score=0.8050081137790289, total= 4.5s [CV] activation=logistic, hidden_layer_sizes=(50,),
max_iter=40000, random_state=0, verbose=0 [CV] activation=logistic, hidden_layer_sizes=(50,),
max_iter=40000, random_state=0, verbose=0, score=0.8068875294645727, total= 7.3s [CV]
activation=logistic, hidden_layer_sizes=(50,), max_iter=40000, random_state=0, verbose=0 [CV]
activation=logistic, hidden_layer_sizes=(50,), max_iter=40000, random_state=0, verbose=0,
score=0.8291321542814972, total= 6.4s [CV] activation=logistic, hidden_layer_sizes=(100,),
max_iter=40000, random_state=0, verbose=0 [CV] activation=logistic, hidden_layer_sizes=(100,),
max_iter=40000, random_state=0, verbose=0, score=0.8039834098907646, total= 7.3s [CV]
activation=logistic, hidden_layer_sizes=(100,), max_iter=40000, random_state=0, verbose=0 [CV]
activation=logistic, hidden_layer_sizes=(100,), max_iter=40000, random_state=0, verbose=0,
score=0.8067351134151506, total= 12.2s [CV] activation=logistic, hidden_layer_sizes=(100,),
max_iter=40000, random_state=0, verbose=0 [CV] activation=logistic, hidden_layer_sizes=(100,),
max_iter=40000, random_state=0, verbose=0, score=0.8283858420748718, total= 7.1s [Parallel(n_jobs=1)]: Done 36 out of 36 | elapsed: 17.2min finished Out[67]: {'activation': 'relu',
'hidden_layer_sizes': (50,), 'max_iter': 40000, 'random_state': 0, 'verbose': 0}

```

#### 4.-Output gridsearchcv de mlp2

Fitting 3 folds for each of 6 candidates, totalling 18 fits [CV] activation=logistic, hidden\_layer\_sizes=(12, 2), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=logistic, hidden\_layer\_sizes=(12, 2), max\_iter=40000, random\_state=0, verbose=0, score=0.8233728092433101, total= 6.5s [CV] activation=logistic, hidden\_layer\_sizes=(12, 2), max\_iter=40000, random\_state=0, verbose=0 [Parallel(n\_jobs=1)]: Done 1 out of 1 | elapsed: 6.6s remaining: 0.0s [CV] activation=logistic, hidden\_layer\_sizes=(12, 2), max\_iter=40000, random\_state=0, verbose=0, score=0.8174029009571363, total= 5.7s [CV] activation=logistic, hidden\_layer\_sizes=(12, 2), max\_iter=40000, random\_state=0, verbose=0 [Parallel(n\_jobs=1)]: Done 2 out of 2 | elapsed: 12.4s remaining: 0.0s [CV] activation=logistic, hidden\_layer\_sizes=(12, 2), max\_iter=40000, random\_state=0, verbose=0, score=0.8371837031404441, total= 5.5s [CV] activation=logistic, hidden\_layer\_sizes=(15, 3), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=logistic, hidden\_layer\_sizes=(15, 3), max\_iter=40000, random\_state=0, verbose=0, score=0.8263164776117378, total= 8.0s [CV] activation=logistic, hidden\_layer\_sizes=(15, 3), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=logistic, hidden\_layer\_sizes=(15, 3), max\_iter=40000, random\_state=0, verbose=0, score=0.8220242041544366, total= 7.7s [CV] activation=logistic, hidden\_layer\_sizes=(15, 3), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=logistic, hidden\_layer\_sizes=(15, 3), max\_iter=40000, random\_state=0, verbose=0, score=0.8448305795384511, total= 8.5s [CV] activation=logistic, hidden\_layer\_sizes=(20, 4), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=logistic, hidden\_layer\_sizes=(20, 4), max\_iter=40000, random\_state=0, verbose=0, score=0.8323558265318636, total= 8.9s [CV] activation=logistic, hidden\_layer\_sizes=(20, 4), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=logistic, hidden\_layer\_sizes=(20, 4), max\_iter=40000, random\_state=0, verbose=0, score=0.8471429936009648, total= 9.8s [CV] activation=logistic, hidden\_layer\_sizes=(25, 2), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=logistic, hidden\_layer\_sizes=(25, 2), max\_iter=40000, random\_state=0, verbose=0, score=0.8273096157975375, total= 8.4s [CV] activation=logistic, hidden\_layer\_sizes=(25, 2), max\_iter=40000, random\_state=0, verbose=0, score=0.8251079876818685, total= 8.3s [CV] activation=logistic, hidden\_layer\_sizes=(25, 2), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=logistic, hidden\_layer\_sizes=(25, 2), max\_iter=40000, random\_state=0, verbose=0, score=0.8490070130879114, total= 9.4s [CV] activation=logistic, hidden\_layer\_sizes=(50, 3), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=logistic, hidden\_layer\_sizes=(50, 3), max\_iter=40000, random\_state=0, verbose=0, score=0.8331650838981232, total= 14.1s [CV] activation=logistic, hidden\_layer\_sizes=(50, 3), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=logistic, hidden\_layer\_sizes=(50, 3), max\_iter=40000, random\_state=0, verbose=0, score=0.8305428039525186, total= 12.5s [CV] activation=logistic, hidden\_layer\_sizes=(50, 3), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=logistic, hidden\_layer\_sizes=(50, 3), max\_iter=40000, random\_state=0, verbose=0, score=0.8382175489759035, total= 10.9s [CV] activation=logistic, hidden\_layer\_sizes=(100,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=logistic, hidden\_layer\_sizes=(100,), max\_iter=40000, random\_state=0, verbose=0, score=0.8039834098907646, total= 7.0s [CV] activation=logistic, hidden\_layer\_sizes=(100,), max\_iter=40000, random\_state=0, verbose=0, score=0.8067351134151506, total= 10.7s [CV] activation=logistic, hidden\_layer\_sizes=(100,), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=logistic, hidden\_layer\_sizes=(100,), max\_iter=40000, random\_state=0, verbose=0, score=0.8283858420748718, total= 7.1s [Parallel(n\_jobs=1)]: Done 18 out of 18 | elapsed:

2.7min finished Out[70]: {'activation': 'logistic', 'hidden\_layer\_sizes': (20, 4), 'max\_iter': 40000, 'random\_state': 0, 'verbose': 0}

### 5.-Output gridsearchcv de mlp3

Fitting 3 folds for each of 24 candidates, totalling 72 fits [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=400, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=400, random\_state=0, verbose=0, score=0.8257272713613167, total= 6.6s [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=400, random\_state=0, verbose=0 [Parallel(n\_jobs=1)]: Done 1 out of 1 | elapsed: 6.7s remaining: 0.0s [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=400, random\_state=0, verbose=0, score=0.8277660385708547, total= 9.4s [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=400, random\_state=0, verbose=0 [Parallel(n\_jobs=1)]: Done 2 out of 2 | elapsed: 16.3s remaining: 0.0s [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=400, random\_state=0, verbose=0, score=0.851151157734269, total= 7.0s [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=4000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=4000, random\_state=0, verbose=0, score=0.8257272713613167, total= 6.8s [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=4000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=4000, random\_state=0, verbose=0, score=0.8277660385708547, total= 9.4s [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=4000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=4000, random\_state=0, verbose=0, score=0.851151157734269, total= 7.4s [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=40000, random\_state=0, verbose=0, score=0.8257272713613167, total= 6.4s [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=40000, random\_state=0, verbose=0, score=0.8277660385708547, total= 9.1s [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=40000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=40000, random\_state=0, verbose=0, score=0.851151157734269, total= 7.0s [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=100000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=100000, random\_state=0, verbose=0, score=0.8257272713613167, total= 6.5s [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=100000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=100000, random\_state=0, verbose=0, score=0.8277660385708547, total= 9.3s [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=100000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(12, 4), max\_iter=100000, random\_state=0, verbose=0, score=0.851151157734269, total= 7.4s [CV] activation=tanh, hidden\_layer\_sizes=(15, 3), max\_iter=400, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(15, 3), max\_iter=400, random\_state=0, verbose=0, score=0.83539073856231, total= 8.9s [CV] activation=tanh, hidden\_layer\_sizes=(15, 3), max\_iter=400, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(15, 3), max\_iter=400, random\_state=0, verbose=0, score=0.8329336106050481, total= 9.5s [CV] activation=tanh, hidden\_layer\_sizes=(15, 3), max\_iter=400, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(15, 3), max\_iter=400, random\_state=0, verbose=0, score=0.8383790171742288, total= 9.3s [CV] activation=tanh, hidden\_layer\_sizes=(15, 3), max\_iter=4000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(15, 3), max\_iter=4000, random\_state=0, verbose=0, score=0.83539073856231, total= 8.8s [CV] activation=tanh, hidden\_layer\_sizes=(15, 3), max\_iter=4000, random\_state=0, verbose=0 [CV] activation=tanh, hidden\_layer\_sizes=(15, 3), max\_iter=4000, random\_state=0, verbose=0, score=0.8329336106050481, total= 10.0s [CV] activation=tanh, hidden\_layer\_sizes=(15, 3),





```

3), max_iter=40000, random_state=0, verbose=0, score=0.799165440152654, total= 23.7s [CV]
activation=tanh, hidden_layer_sizes=(50, 3), max_iter=40000, random_state=0, verbose=0
[CV] activation=tanh, hidden_layer_sizes=(50, 3), max_iter=40000, random_state=0, ver-
bose=0, score=0.812145991298449, total= 27.3s [CV] activation=tanh, hidden_layer_sizes=(50,
3), max_iter=100000, random_state=0, verbose=0 [CV] activation=tanh, hidden_layer_sizes=(50,
3), max_iter=100000, random_state=0, verbose=0, score=0.7836384967309807, total= 31.5s [CV]
activation=tanh, hidden_layer_sizes=(50, 3), max_iter=100000, random_state=0, verbose=0
[CV] activation=tanh, hidden_layer_sizes=(50, 3), max_iter=100000, random_state=0, ver-
bose=0, score=0.799165440152654, total= 23.2s [CV] activation=tanh, hidden_layer_sizes=(50, 3),
max_iter=100000, random_state=0, verbose=0 [CV] activation=tanh, hidden_layer_sizes=(50,
3), max_iter=100000, random_state=0, verbose=0, score=0.812145991298449, total= 27.5s
[CV] activation=tanh, hidden_layer_sizes=(50, 2), max_iter=400, random_state=0, verbose=0
[CV] activation=tanh, hidden_layer_sizes=(50, 2), max_iter=400, random_state=0, verbose=0,
score=0.8252688719278272, total= 13.5s [CV] activation=tanh, hidden_layer_sizes=(50, 2),
max_iter=400, random_state=0, verbose=0 [CV] activation=tanh, hidden_layer_sizes=(50, 2),
max_iter=400, random_state=0, verbose=0, score=0.821187859035436, total= 12.5s [CV] ac-
tivation=tanh, hidden_layer_sizes=(50, 2), max_iter=400, random_state=0, verbose=0 [CV]
activation=tanh, hidden_layer_sizes=(50, 2), max_iter=400, random_state=0, verbose=0,
score=0.8372776449617336, total= 19.4s [CV] activation=tanh, hidden_layer_sizes=(50, 2),
max_iter=4000, random_state=0, verbose=0 [CV] activation=tanh, hidden_layer_sizes=(50,
2), max_iter=4000, random_state=0, verbose=0, score=0.8252688719278272, total= 13.3s [CV]
activation=tanh, hidden_layer_sizes=(50, 2), max_iter=4000, random_state=0, verbose=0 [CV]
activation=tanh, hidden_layer_sizes=(50, 2), max_iter=4000, random_state=0, verbose=0,
score=0.821187859035436, total= 12.4s [CV] activation=tanh, hidden_layer_sizes=(50, 2),
max_iter=4000, random_state=0, verbose=0 [CV] activation=tanh, hidden_layer_sizes=(50,
2), max_iter=4000, random_state=0, verbose=0, score=0.8372776449617336, total= 21.0s [CV]
activation=tanh, hidden_layer_sizes=(50, 2), max_iter=40000, random_state=0, verbose=0 [CV]
activation=tanh, hidden_layer_sizes=(50, 2), max_iter=40000, random_state=0, verbose=0,
score=0.8252688719278272, total= 13.7s [CV] activation=tanh, hidden_layer_sizes=(50, 2),
max_iter=40000, random_state=0, verbose=0 [CV] activation=tanh, hidden_layer_sizes=(50,
2), max_iter=40000, random_state=0, verbose=0, score=0.821187859035436, total= 12.2s [CV]
activation=tanh, hidden_layer_sizes=(50, 2), max_iter=40000, random_state=0, verbose=0 [CV]
activation=tanh, hidden_layer_sizes=(50, 2), max_iter=40000, random_state=0, verbose=0,
score=0.8372776449617336, total= 19.3s [CV] activation=tanh, hidden_layer_sizes=(50, 2),
max_iter=100000, random_state=0, verbose=0 [CV] activation=tanh, hidden_layer_sizes=(50,
2), max_iter=100000, random_state=0, verbose=0, score=0.8252688719278272, total= 13.7s [CV]
activation=tanh, hidden_layer_sizes=(50, 2), max_iter=100000, random_state=0, verbose=0
[CV] activation=tanh, hidden_layer_sizes=(50, 2), max_iter=100000, random_state=0, ver-
bose=0, score=0.821187859035436, total= 13.5s [CV] activation=tanh, hidden_layer_sizes=(50,
2), max_iter=100000, random_state=0, verbose=0 [CV] activation=tanh, hidden_layer_sizes=(50,
2), max_iter=100000, random_state=0, verbose=0, score=0.8372776449617336, total= 20.2s [Par-
allel(n_jobs=1)]: Done 72 out of 72 | elapsed: 16.1min finished Out[73]: {'activation': 'tanh',
'hidden_layer_sizes': (15, 3), 'max_iter': 400, 'random_state': 0, 'verbose': 0}

```

## 6.-Output gridsearchcv de Random Forest

Fitting 3 folds for each of 18 candidates, totalling 54 fits [CV] criterion=gini, min\_samples\_split=2, n\_estimators=10 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=10, score=0.6186216681522541, total= 5.7s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=10 ..... [Parallel(n\_jobs=1)]: Done 1 out of 1 |

elapsed: 5.8s remaining: 0.0s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=10, score=0.6070639252342804, total= 6.1s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=10 ..... [Parallel(n\_jobs=1)]: Done 2 out of 2 | elapsed: 12.0s remaining: 0.0s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=10, score=0.6109342996892269, total= 6.1s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=15 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=15, score=0.6258862934328312, total= 8.7s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=15 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=15, score=0.6326463215327687, total= 8.6s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=15 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=15, score=0.6434755461909264, total= 9.1s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=35 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=35, score=0.6765849617842946, total= 21.1s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=35 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=35, score=0.7112796438589509, total= 21.0s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=35 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=35, score=0.7160187626167904, total= 20.6s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=50 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=50, score=0.7052839410670152, total= 30.9s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=50 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=50, score=0.7223490567996929, total= 31.7s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=50 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=50, score=0.7400997799380955, total= 31.1s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=100 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=100, score=0.7586813860990705, total= 59.5s [CV] criterion=gini, min\_samples\_split=2, n\_estimators=100 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=100, score=0.758673262617954, total= 1.0min [CV] criterion=gini, min\_samples\_split=2, n\_estimators=100 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=100, score=0.7758580024738215, total= 1.0min [CV] criterion=gini, min\_samples\_split=2, n\_estimators=200 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=200, score=0.773027634348719, total= 2.0min [CV] criterion=gini, min\_samples\_split=2, n\_estimators=200 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=200, score=0.7783320182641791, total= 2.0min [CV] criterion=gini, min\_samples\_split=2, n\_estimators=200 ..... [CV] criterion=gini, min\_samples\_split=2, n\_estimators=200, score=0.7983146217864018, total= 2.1min [CV] criterion=gini, min\_samples\_split=4, n\_estimators=10 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=10, score=0.6002616681619479, total= 5.9s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=10 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=10, score=0.6162802687040301, total= 5.7s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=10 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=10, score=0.6275954919827913, total= 5.9s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=15 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=15, score=0.6395610283172828, total= 8.3s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=15 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=15, score=0.6396622569647151, total= 9.1s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=15 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=15, score=0.675927526708595, total= 9.0s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=35 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=35, score=0.710252812800965, total=

20.4s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=35 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=35, score=0.697109505527116, total=20.4s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=35 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=35, score=0.7347236601433879, total=21.5s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=50 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=50, score=0.7241083688735274, total=30.2s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=50 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=50, score=0.7282931612801056, total=30.7s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=50 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=50, score=0.7572667824452551, total=31.1s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=100 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=100, score=0.757079169205252, total=59.6s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=100 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=100, score=0.7586751146854868, total=58.6s [CV] criterion=gini, min\_samples\_split=4, n\_estimators=100 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=100, score=0.7739481658542552, total=1.0min [CV] criterion=gini, min\_samples\_split=4, n\_estimators=200 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=200, score=0.7746129475362273, total=2.0min [CV] criterion=gini, min\_samples\_split=4, n\_estimators=200 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=200, score=0.7676479871790778, total=2.0min [CV] criterion=gini, min\_samples\_split=4, n\_estimators=200 ..... [CV] criterion=gini, min\_samples\_split=4, n\_estimators=200, score=0.8047882577459949, total=2.1min [CV] criterion=gini, min\_samples\_split=8, n\_estimators=10 ..... [CV] criterion=gini, min\_samples\_split=8, n\_estimators=10, score=0.6578733162333259, total=6.3s [CV] criterion=gini, min\_samples\_split=8, n\_estimators=10 ..... [CV] criterion=gini, min\_samples\_split=8, n\_estimators=10, score=0.6540219437819171, total=6.0s [CV] criterion=gini, min\_samples\_split=8, n\_estimators=10 ..... [CV] criterion=gini, min\_samples\_split=8, n\_estimators=10, score=0.658230970450591, total=6.4s [CV] criterion=gini, min\_samples\_split=8, n\_estimators=15 ..... [CV] criterion=gini, min\_samples\_split=8, n\_estimators=15, score=0.6635686535246711, total=8.5s [CV] criterion=gini, min\_samples\_split=8, n\_estimators=15 ..... [CV] criterion=gini, min\_samples\_split=8, n\_estimators=15, score=0.6860455273419213, total=8.3s [CV] criterion=gini, min\_samples\_split=8, n\_estimators=15 ..... [CV] criterion=gini, min\_samples\_split=8, n\_estimators=15, score=0.7088797914503712, total=9.0s [CV] criterion=gini, min\_samples\_split=8, n\_estimators=35 ..... [CV] criterion=gini, min\_samples\_split=8, n\_estimators=35, score=0.7311404924588871, total=20.7s [CV] criterion=gini, min\_samples\_split=8, n\_estimators=35 ..... [CV] criterion=gini, min\_samples\_split=8, n\_estimators=35, score=0.7430755445594697, total=20.4s [CV] criterion=gini, min\_samples\_split=8, n\_estimators=35 ..... [CV] criterion=gini, min\_samples\_split=8, n\_estimators=35, score=0.7579948771041192, total=20.8s [CV] criterion=gini, min\_samples\_split=8, n\_estimators=50 ..... [CV] criterion=gini, min\_samples\_split=8, n\_estimators=50, score=0.7356242157043791, total=33.1s [CV] criterion=gini, min\_samples\_split=8, n\_estimators=50 ..... [CV] criterion=gini, min\_samples\_split=8, n\_estimators=50, score=0.7412662567504825, total=29.8s [CV] criterion=gini, min\_samples\_split=8, n\_estimators=50 ..... [CV] criterion=gini, min\_samples\_split=8, n\_estimators=50, score=0.7713715062614622, total=32.0s [CV] criterion=gini, min\_samples\_split=8, n\_estimators=100 ..... [CV] criterion=gini, min\_samples\_split=8, n\_estimators=100, score=0.7686362756203236, total=

```
58.2s [CV] criterion=gini, min_samples_split=8, n_estimators=100 ..... [CV] cri-  
terion=gini, min_samples_split=8, n_estimators=100, score=0.7689022619634758, total=  
1.0min [CV] criterion=gini, min_samples_split=8, n_estimators=100 ..... [CV] cri-  
terion=gini, min_samples_split=8, n_estimators=100, score=0.7916882013262497, total=  
1.0min [CV] criterion=gini, min_samples_split=8, n_estimators=200 ..... [CV] cri-  
terion=gini, min_samples_split=8, n_estimators=200, score=0.7764457879356734, total=  
2.0min [CV] criterion=gini, min_samples_split=8, n_estimators=200 ..... [CV] cri-  
terion=gini, min_samples_split=8, n_estimators=200, score=0.7850384762475678, total= 2.0min  
[CV] criterion=gini, min_samples_split=8, n_estimators=200 ..... [CV] criterion=gini,  
min_samples_split=8, n_estimators=200, score=0.805756854397889, total= 2.1min [Paral-  
lel(n_jobs=1)]: Done 54 out of 54 | elapsed: 38.0min finished Out[76]: {'criterion': 'gini',  
'min_samples_split': 8, 'n_estimators': 200}
```

## 44 Referencias

[https://statcompute.wordpress.com/2012/11/18/calculating-k-s-statistic-with-  
python/amp/%20Recibidos%20x/](https://statcompute.wordpress.com/2012/11/18/calculating-k-s-statistic-with-python/)

[https://www2.ulpgc.es/hege/almacen/download/5/5015/Complemento\\_3\\_  
Prueba\\_de\\_Bondad\\_de\\_Ajuste\\_de\\_Kolmogorov\\_Smirnov.pdf](https://www2.ulpgc.es/hege/almacen/download/5/5015/Complemento_3_<br/>Prueba_de_Bondad_de_Ajuste_de_Kolmogorov_Smirnov.pdf)