Minería de datos
Marketing Bancario

Manuel Lagunas Arto

Junio 2017

1 Introducción

El siguiente trabajo trata de analizar un conjunto de datos directamente recogidos por una institución bancaria portuguesa dentro de sus campañas de marketing. Los datos recolectados provienen de llamadas telefónicas donde se buscaba conocer si el cliente se inscribiría o no en cierto deposito bancario. El objetivo final será incrementar la eficacia y efectividad de futuras campañas de marketing, identificando cuales son las variables con mayor influencia dentro de las decisiones de los clientes.

El documento se estructura en tres secciones principales: la Sección 2 donde se comenta la naturaleza de los datos, sus variables, la existencia de valores nulos y la separación en subconjuntos de entrenamiento, validación y test. La Sección 3 comenta los modelos de árboles de decisión ajustados así como posibles variantes como bagging o boosting y la importancia que cada uno da a las variables predictoras. Por último, la Sección 4 busca una solución al problema haciendo uso de máquinas de vector soporte con tres kernels: de función de base radial, lineal y polinomial.

2 Análisis de los datos

Como se ha mencionado, los datos provienen de un banco portugués, que dentro de sus campañas de marketing buscaba que los clientes se subscribieran a cierto deposito bancario. El conjunto obtenido contiene **41188 observaciones, 20 variables predictoras y una variable respuesta**. En las variables predictoras encontramos varios tipos: binarias, enteras, numéricas continuas y de tipo cualitativo.

```
     age                 job            marital
 Min.   :17.00   admin.     :10422   divorced: 4612
 1st Qu.:32.00   blue-collar: 9254   married :24928
 Median :38.00   technician : 6743   single  :11568
 Mean   :40.02   services   : 3969   unknown :   80
 3rd Qu.:47.00   management : 2924
 Max.   :98.00   retired    : 1720
                 (Other)    : 6156
               education        default         housing           loan
 university.degree  :12168   no     :32588   no     :18622   no     :33950
 high.school        : 9515   unknown: 8597   unknown:  990   unknown:  990
 basic.9y           : 6045   yes    :    3   yes    :21576   yes    : 6248
 professional.course: 5243
 basic.4y           : 4176
 basic.6y           : 2292
 (Other)            : 1749
     contact          month       day_of_week    duration
 cellular :26144   may    :13769   fri:7827    Min.   :   0.0
 telephone:15044   jul    : 7174   mon:8514    1st Qu.: 102.0
                   aug    : 6178   thu:8623    Median : 180.0
                   jun    : 5318   tue:8090    Mean   : 258.3
                   nov    : 4101   wed:8134    3rd Qu.: 319.0
                   apr    : 2632               Max.   :4918.0
                   (Other): 2016
   campaign          pdays          previous            poutcome
 Min.   : 1.000   Min.   :  0.0   Min.   :0.000   failure    : 4252
 1st Qu.: 1.000   1st Qu.:999.0   1st Qu.:0.000   nonexistent:35563
 Median : 2.000   Median :999.0   Median :0.000   success    : 1373
 Mean   : 2.568   Mean   :962.5   Mean   :0.173
 3rd Qu.: 3.000   3rd Qu.:999.0   3rd Qu.:0.000
 Max.   :56.000   Max.   :999.0   Max.   :7.000

  emp.var.rate      cons.price.idx  cons.conf.idx     euribor3m
 Min.   :-3.40000   Min.   :92.20   Min.   :-50.8   Min.   :0.634
 1st Qu.:-1.80000   1st Qu.:93.08   1st Qu.:-42.7   1st Qu.:1.344
 Median : 1.10000   Median :93.75   Median :-41.8   Median :4.857
 Mean   : 0.08189   Mean   :93.58   Mean   :-40.5   Mean   :3.621
 3rd Qu.: 1.40000   3rd Qu.:93.99   3rd Qu.:-36.4   3rd Qu.:4.961
 Max.   : 1.40000   Max.   :94.77   Max.   :-26.9   Max.   :5.045

  nr.employed     y
 Min.   :4964   no :36548
 1st Qu.:5099   yes: 4640
 Median :5191
 Mean   :5167
 3rd Qu.:5228
 Max.   :5228
```

Figure 1: Resumen de los datos.

2.1 Existencia de valores nulos

Mostrando un resumen de los datos podemos observar como algunas de las variables de tipo cualitativo o binario tienen valores *unknown*, lo que puede significar que el cliente no quisiera dar dicha información durante la llamada telefónica. La existencia de este tipo de valores nulos puede afectar en el entrenamiento final del modelo dando resultados no deseados. Por ello, su

tratamiento será crucial para obtener ajustar de mejor forma los modelos.

Las variables *education*, *housing* y *loan* tienen una cantidad considerable de valores nulos (1731, 989 y 989 respectivamente) por lo que se decide imputar los valores en sus observaciones en vez de eliminarlas. Para la imputación se hace uso del paquete *mice*.

En las variables con un menor número de valores nulos: *job* y *marital* no se hace ninguna imputación y dado que es un porcentaje realmente bajo del total, se decide eliminar las observaciones con nulos.

La variable binaria *default* contiene el mayor número de valores *"unknown"* con 8597, además su porcentaje de valores *"yes"* es muy bajo respecto al de *"no"*, con tan solo 3 (mencionar que son valores "yes de la variable *default* y no la variable respuesta), lo que dificulta la imputación de los valores nulos. Por ello se decide usar el valor "*default*" como una categoría más dentro del predictor.

Posteriormente se prueba a dejar todas variables sin realizar ningun tipo de imputación, a imputar todas las variables o a imputar todas menos *default* teniendo que el mejor resultado es el descrito anteriormente.

2.2 Desbalanceo de los datos

Además de la existencia de valores nulos, existe otro problema en el conjunto de datos. La variable respuesta tiene aproximadamente **88.74% de respuestas *no* y 11.26% de respuestas *yes***. El desbalanceo obligará a replantear las métricas a usar para evaluar los modelos. Ahora la precisión no nos dará tanta información (un 90% de precisión implica que el predictor de como resultado siempre *no*), se hará uso además de otras medidas como:

- La curva **ROC** y el **área por debajo** de esta (**AUC**). La curva ROC relaciona el ratio de positivos verdaderos (TPR), las observaciones positivas que son predichas como positivas, con el ratio de falsos positivos (FPR), que corresponde a las observaciones negativas predichas incorrectamente como positivas. Por último, el AUC nos informará de la probabilidad de que el modelo clasifique un ejemplo aleatorio positivo con mayor valor que un ejemplo aleatorio negativo $P\Big(\text{score}(x^+) > \text{score}(x^-)\Big)$. Esta será la métrica usada al entrenar los modelos.
- Estadístico **Kappa** el cual compara la precisión esperada (asumiendo un clasificador aleatorio) con la precisión observada con el modelo.

También se realizará un cambio durante el entrenamiento que tendrá en

cuenta dicho desbalanceo para muestrear los datos. Así pues, se usaran distintas metodologías como **sobre-muestreo** (mayor muestreo del grupo con menos datos), el **sub-muestreo** (menor muestreo del grupo con más datos), **muestreos mixtos** y la **aplicación de pesos** (relativos a las cantidades de datos de cada clase) a las observaciones del conjunto de datos.

2.3 Búsqueda de valores atípicos

Con el fin de localizar posibles valores anómalos dentro del conjunto de datos realizamos diferentes visualizaciones descriptivas de los datos que nos puedan dar una visión global del contenido de cada variable. Primero mostramos los histogramas de cada variable:

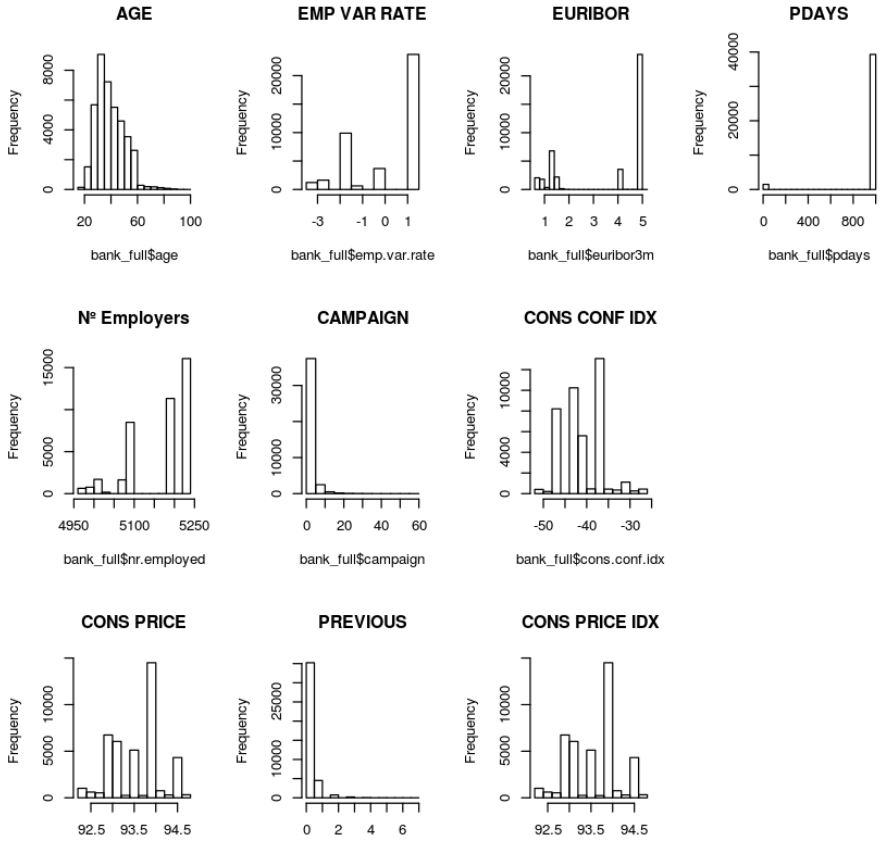

Figure 2: Histograma de las variables numéricas con el fin de buscar datos atípicos.

A priori, puede observarse que *pdays* (días pasados desde la última vez que se hablo con un cliente) tiene un conjunto de valores anormales cerca del 1000, pero leyendo la descripción de las variables se observa que 999 es el valor por defecto cuando no se ha contactado antes con dicho cliente. Respecto al resto de datos parecen encontrarse dentro de los rangos normales sin ninguna variación extraña dentro de sus histogramas. Para explorar las variables con mayor precisión visualizamos también los diagramas de cajas de cada variable:

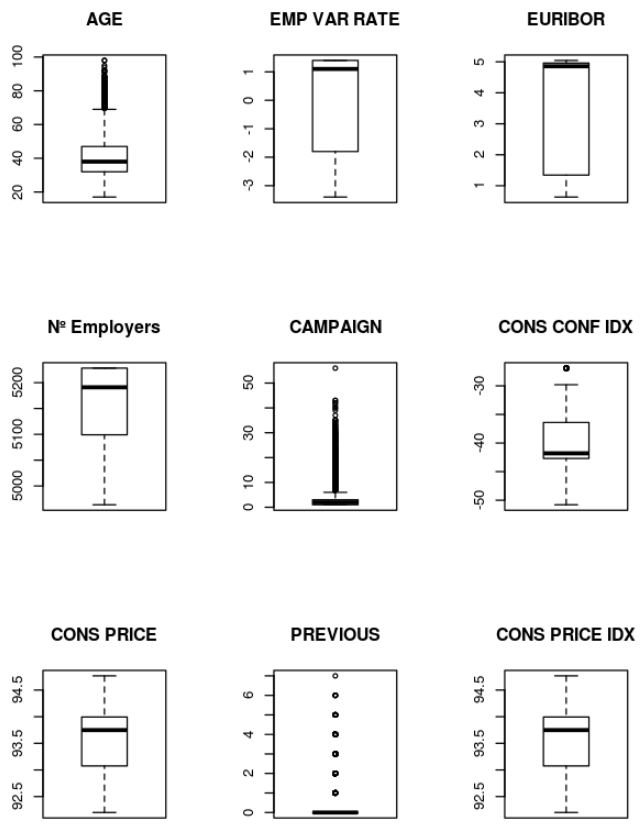

Figure 3: Boxplot de las variables numéricas con el fin de buscar datos atípicos.

De nuevo observamos que existe cierta normalidad en el rango de los datos, existiendo el valor de 999 en *pdays*. Además, aparecen datos atípicos en variables como *campaign* y *duration* donde hay 3 observaciones relativamente alejadas de la distribución de sus respectivas variables. Los datos extraños corresponden a un número bajo de observaciones, por lo que se en un principio se decide eliminarlas del conjunto, tras observar que los modelos no tenían mejor resultado se piensa que aunque de la sensación de que son datos anómalos quizás sigan estando dentro de la distribución original de las observaciones. Otras variables

como *cons.conf.idx* o *age* tienen también valores que parece que salen de la distribución original de los datos pero es un número alto de observaciones las que se encuentran en esta situación por lo que se decide mantenerlas.

2.4 Separación de los datos

Tras el preprocesado se separaran los datos en un conjunto de test y entrenamiento con un 80% y 20% de los datos respectivamente. Además 15% de los datos de entrenamiento serán usados para la búsqueda de parámetros con validación cruzada.

Además de esta separación (por defecto) también se crean nuevos conjuntos a partir de los de entrenamiento y test haciendo uso de sobre-muestreo, sub-muestreo y muestreo mixto (*SMOTE*). También se calculará el peso de cada observación de acuerdo a la clase a la que pertenece para los conjuntos de datos de entrenamiento y validación.

3 Uso de árboles de decisión

Una vez los datos han sido preprocesados se entrenarán diferentes árboles de decisión con sus variaciones como *bagging* o *boosting*. Para evitar la sobre-parametrización se hace uso de validación cruzada para la selección de los parámetros. Todos los procesos se entrenan usando 5 formas de muestreo: por defecto, con pesos en las observaciones, sobre-muestreo, sub-muestreo y muestreo mixto con la función *SMOTE*. Para la búsqueda de parámetros se hace uso del paquete *caret*.

3.1 Árboles de decisión tradicionales

El primer modelo a entrenar será un árbol de decisión tradicional. Para ello haremos uso de la librería *rpart*. Con la ayuda de *caret* seleccionamos los parámetros para cada modelo de muestreo. Finalmente, entrenamos los modelos con los datos de entrenamiento usando los parámetros de la validación cruzada.

Como se observa en las Figuras 5, 6, la métrica AUC se beneficia de considerar el desbalanceo de los datos. Comparando las curvas ROC de entrenamiento y validación vemos como no se tiene casi sobre-parametrización lo que lleva a pensar que los árboles de decisión no son capaces de modelar de mejor forma el problema.

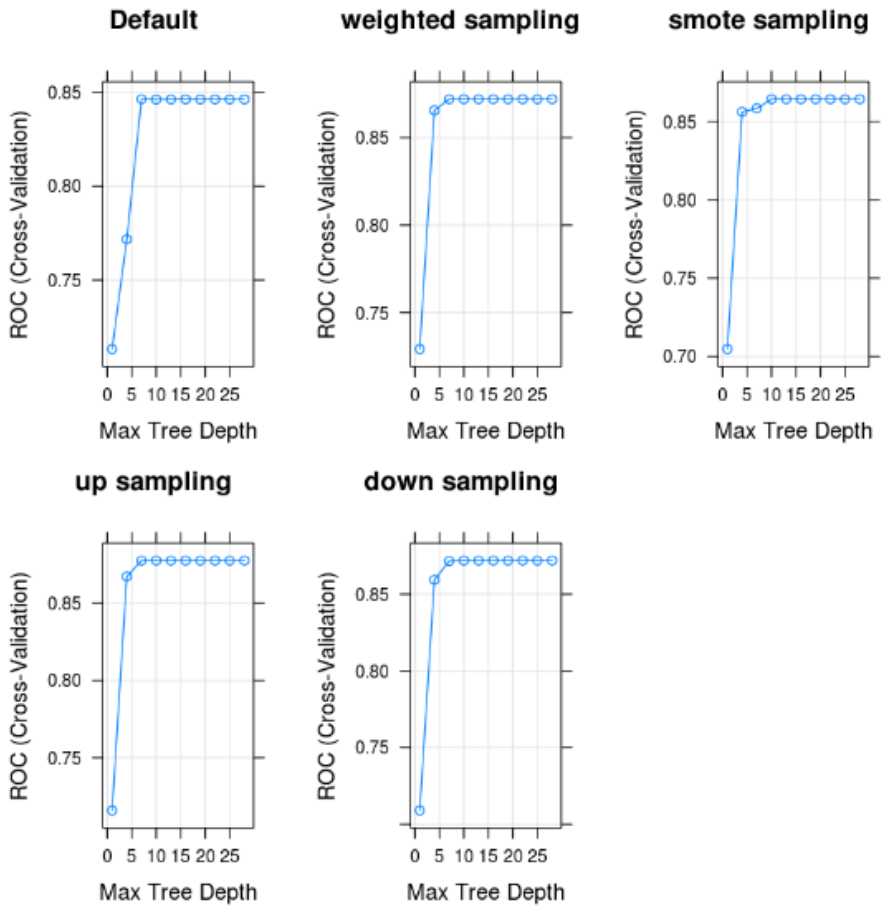

Figure 4: Resumen de la búsqueda de parámetros para los árboles de decisión haciendo uso de la librería caret.

Si observamos las tablas de resultados vemos como el muestreo teniendo en cuenta el desbalanceo de los datos mejora los resultados en test de AUC. Sin embargo, aunque con peor AUC, el modelo con mejor precisión y valor del estadístico kappa corresponde al modelo con el muestreo por defecto. Si observamos la importancia de las variables obtenidas por cada modelo en la Figura 7 podemos apreciar como hay 4 variables que coexisten entre las que el modelo ha considerado más importantes: *duration*, *euribor3m*, *nr.employed* y *emp.var.rate*. Lo que empieza a dar una idea de cuales son los factores que hacen que el cliente se decline por una decisión positiva o negativa.

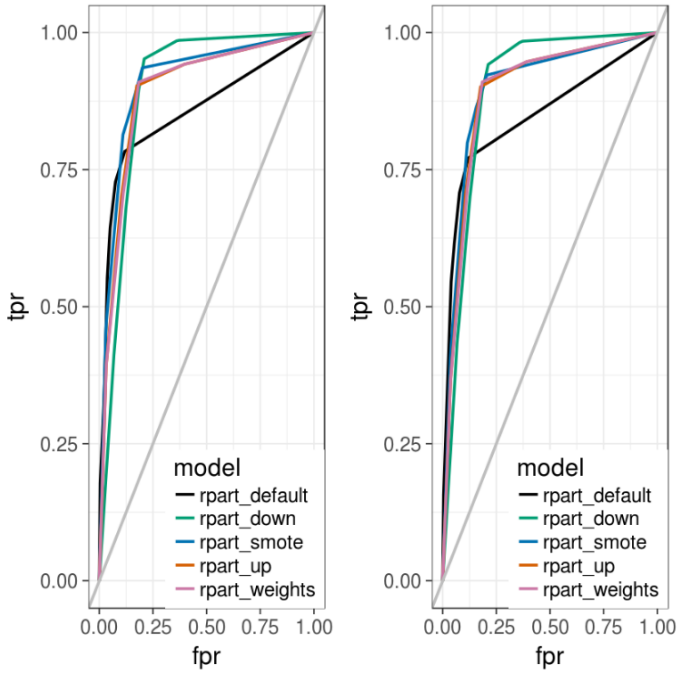

Figure 5: Curvas ROC para los muestreos con rpart. La gráfica de la izquierda corresponde al subconjunto de test, la gráfica de la derecha al de entrenamiento.

	rpart_default	rpart_smote	rpart_up	rpart_down	rpart_weights
AUC	0.8551846	0.9105469	0.8931139	0.8964853	0.8912487
prec.	0.9167484	0.8811918	0.8338646	0.8089750	0.8267533
kappa	0.5528598	0.5418883	0.4668280	0.4372275	0.4556248
AUC(train)	0.8475270	0.8972699	0.8914610	0.8967482	0.8902388
prec.(train)	0.9138549	0.8753946	0.8322454	0.8052772	0.8262082
kappa(train)	0.5401158	0.5232554	0.4634392	0.4283195	0.4549320

Figure 6: Resultados de varias métricas para los modelos de árboles de decisión en entrenamiento y test.

3.2 Árboles de inferencia condicional

El segundo modelo que se entrena se trata de árboles de inferencia condicional. Para ello se hace uso del paquete *party* y de *caret* para la búsqueda de parámetros. Este tipo de árboles evaluan la hipotesis nula de independencia entre cualquier variable de entrada y la variable respuesta. Si no puede ser rec-

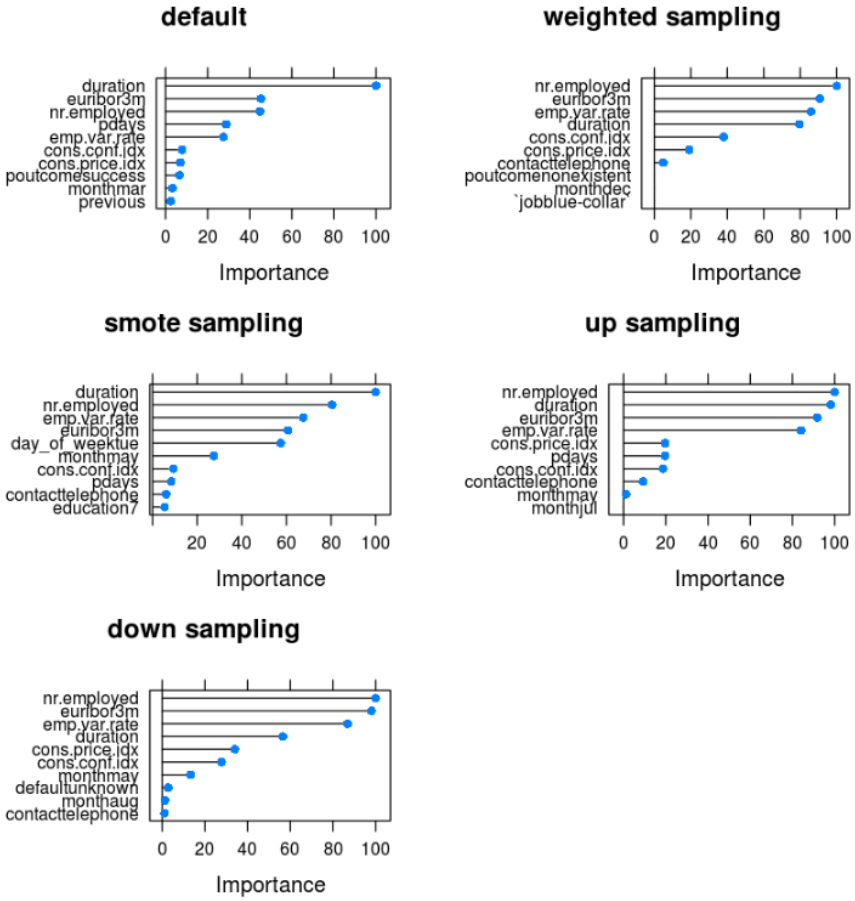

Figure 7: Resultados de varias métricas para los modelos de árboles de decisión en entrenamiento y test.

hazada se para, en caso contrario se selecciona la variable con mayor asociación con la respuesta (con el p-valor). Después realiza una separación binaria de la variable seleccionada y repite estos pasos recursivamente.

Con la librería *caret* buscaremos la máxima profundidad del árbol (*maxdepth*), así como el valor que debe sobrepasar el test para implementar la separación (*mincriterion*).

De nuevo, se entrenan los modelos con la totalidad de los datos de entrenamiento y los mejores parámetros obtenidos. Los árboles de inferencia condicional mejoran el resultado pues tratan de evitar el sesgo introducido por la selección de variables de los árboles tradicionales tal y como se observa en las Figuras 9, 10. En este caso **las métricas han mejorado de manera general**

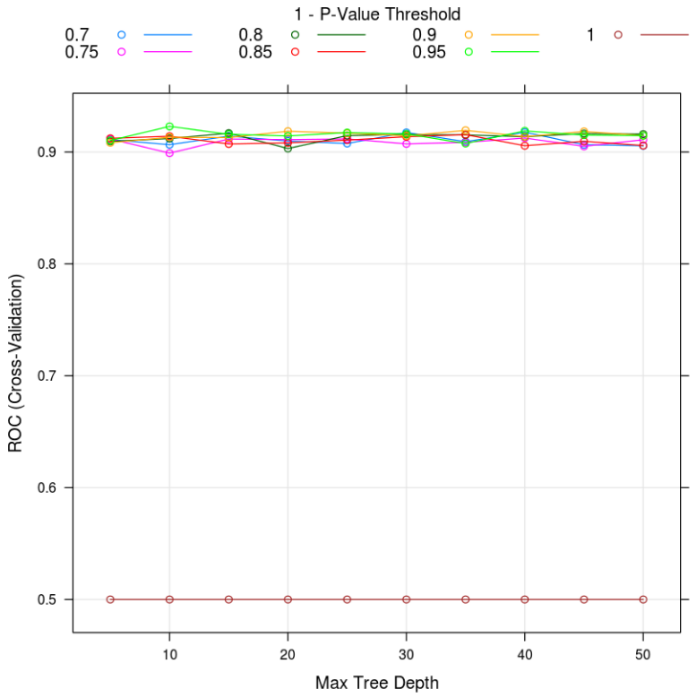

Figure 8: Ejemplo del resultado de la búsqueda de parámetros con árboles de inferencia condicional. Solo se muestra para el caso del muestreo por defecto dado que el tamaño de las visualizaciones hacía que fuera imposible distinguir la leyenda y dificultaba su comprensión aunque en el código se hizo para todos los tipos de muestreo.

respecto a los árboles de decisión. El AUC con un valor más alto es el que devuelve el árbol de inferencia condicional entrenado con el muestreo por defecto, que además tiene el mejor porcentaje de precisión del modelo, sin embargo su estadístico kappa no es el más alto, correspondiendo al del modelo entrenado con el muestreo *smote*.

La Figura 11 nos muestra como los árboles de inferencia condicional consideran con mayor importancia las mismas variables que las mencionadas en los árboles de decisión tradicionales. Todos consideran la duración como la variable predictora más importante.

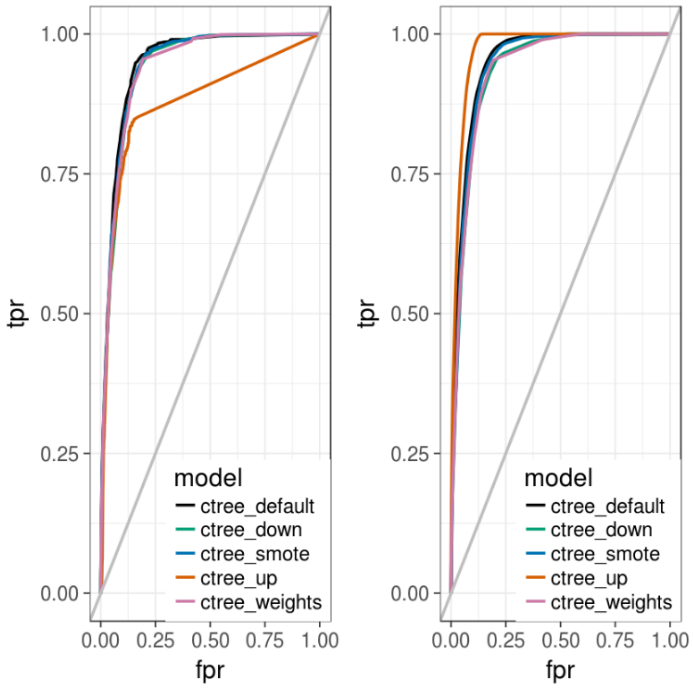

Figure 9: Visualización de las curvas ROC generadas por los diferentes muestreos para el método de árboles de inferencia condicional. La gráfica de la izquierda corresponde al subconjunto de test, la gráfica de la derecha al de entrenamiento.

	ctree_default	ctree_smote	ctree_up	ctree_down	ctree_weights
AUC	0.9462359	0.9420471	0.8782868	0.9390641	0.9388658
prec.	0.9140510	0.8994605	0.8654978	0.8352133	0.8364394
kappa	0.5439573	0.5781793	0.5018148	0.4801905	0.4806337
AUC(train)	0.9513800	0.9450236	0.9725928	0.9360612	0.9364939
prec.(train)	0.9194937	0.8990837	0.9030370	0.8339616	0.8374858
kappa(train)	0.5737838	0.5849763	0.6422110	0.4753032	0.4840013

Figure 10: Diferentes curvas ROC para los distintos muestreos con rpart.

3.3 treebag

El tercer modelo hace uso de la técnica de *bagging*. Se emplea la librería *treebag* y *caret* para el entrenamiento de los modelos. En este caso, los modelos

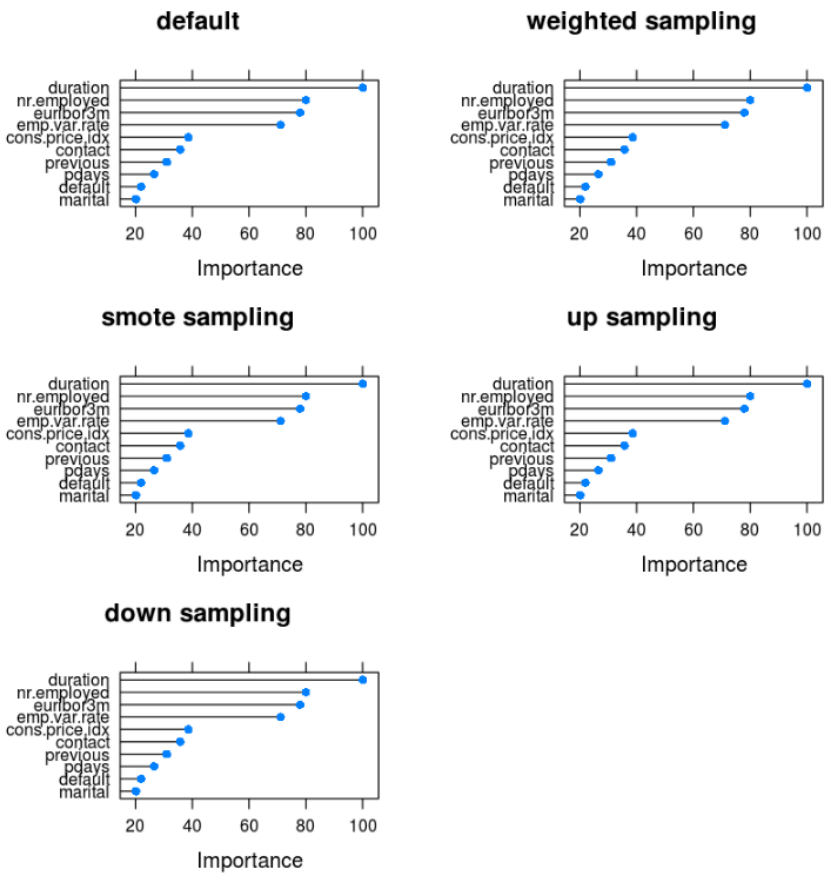

Figure 11: Gráfica comparativa con la importancia de las variables dadas por los modelos de árboles de inferencia condicional entrenados usando los distintos muestreos.

carecen de parámetros a seleccionar, por ello se entrenaran directamente con la totalidad de los datos de entrenamiento.

En este caso se hace *bagging* con los árboles de decisión tradicionales y obtiene resultados similares en cuanto a AUC con respecto a los árboles de inferencia condicional, tal y como vemos en las Figuras 12, 13. Todos modelos obtienen un AUC de aproximadamente 0.93 además, la precisión parece ser también mejor (en términos generales) que en los modelos anteriores siendo que todos superan el 90% a excepción del modelo con sub-muestreo. Lo mismo ocurre con el estadístico kappa, con valores por encima de 0.55 en todos modelos a excepción del entrenado con sub-muestreo, que obtiene un 0.51.

En el caso de la importancia de las variables predictoras volvemos a ob-

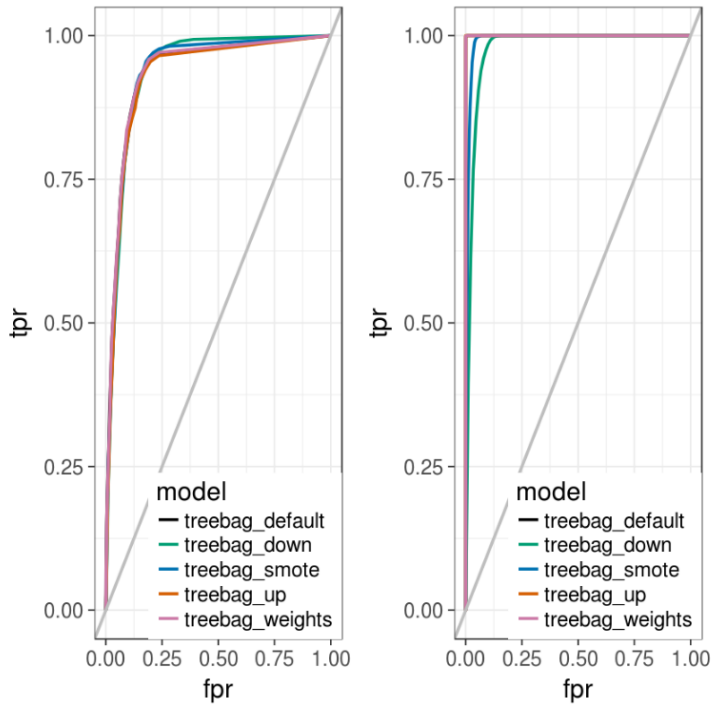

Figure 12: Visualización de las curvas ROC generados haciendo uso de bagging con árboles de decisión, la curva de la izquierda pertenece al test mientras que la derecha al entrenamiento. Se puede observar cierto sobre-ajuste de los modelos en la curva de la derecha donde algunas métricas aparecen perfectas.

	treebag_default	treebag_smote	treebag_up	treebag_down	treebag_weights
AUC	0.9367949	0.9384830	0.9309256	0.9396470	0.9376031
prec.	0.9145414	0.9015449	0.9075527	0.8534821	0.9152771
kappa	0.5501877	0.5854371	0.5645731	0.5106887	0.5522686
AUC(train)	0.9999909	0.9912596	1.0000000	0.9761057	0.9999901
prec.(train)	0.9992339	0.9552266	0.9992645	0.8771414	0.9992339
kappa(train)	0.9961610	0.8087334	0.9963316	0.5834747	0.9961601

Figure 13: Diferentes curvas ROC para los distintos muestreos con rpart.

servar el mismo patrón en la Figura 14. La variable *duration* parece tener una excepcional relevancia en la predicción final, pues parece razonable pensar que un usuario que este más tiempo hablando estará interesado en el producto y por tanto tendrá más posibilidad de acabar contratándolo.

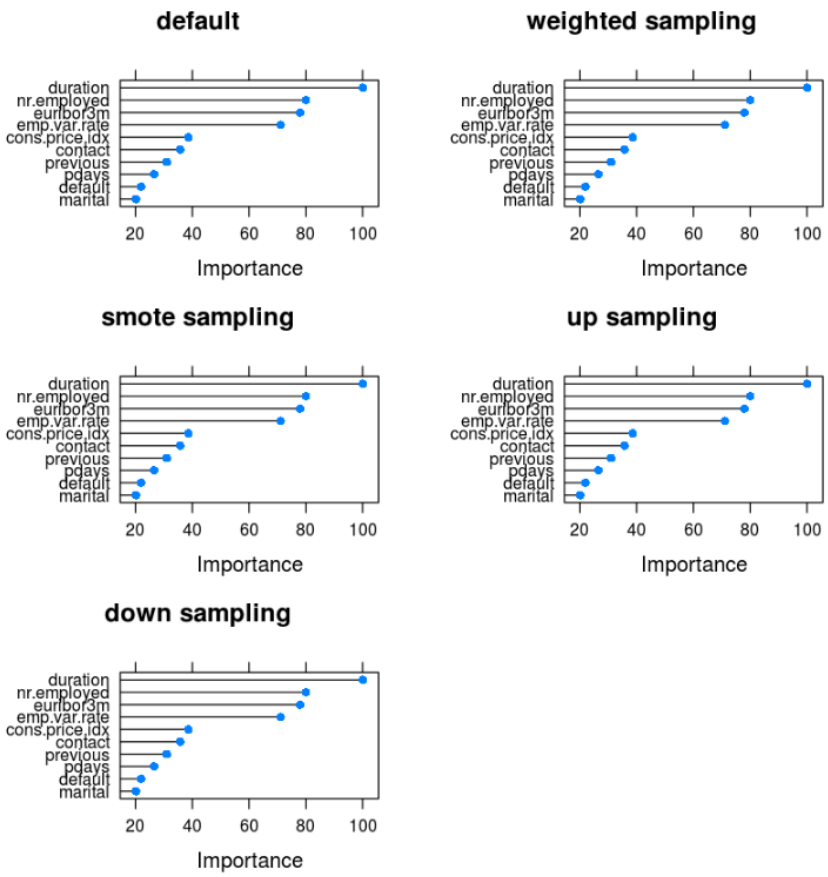

Figure 14: Gráfica comparativa con la importancia de las variables dadas por los modelos de árboles de inferencia condicional entrenados usando los distintos muestreos.

3.4 Gradient boosting models

El último modelo entrenado con árboles de regresión hace uso de la técnica de *boosting*. Para su entrenamiento se han usado las librerías *gbm* y *caret*. Los modelos entrenados por la librería *gbm* corresponden a "gradient boosting machines" que tratan de minimizar la función de perdida del modelo añadiendo clasificadores débiles, que en este caso corresponden a árboles de decisión usando una técnica similar al descenso de gradiente. En la búsqueda de parámetros se han tenido en cuenta el número de arboles del modelo (*n.trees*) y el coeficiente de aprendizaje (*shrinkage*)

Las Figuras 16, 17 muestran como usando *gbm* se consiguen mejores re-

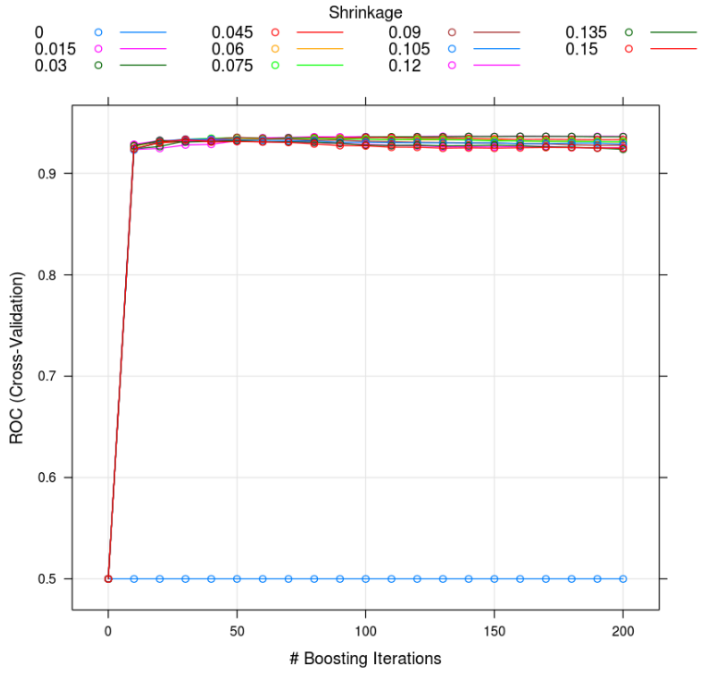

Figure 15: Ejemplo del resultado de la búsqueda de parámetros con gradient boosting machines. Solo se muestra para el caso del muestreo por defecto dado que el tamaño de las visualizaciones hacía que fuera imposible distinguir la leyenda y dificultaba su comprensión aunque en el código se hizo para todos los tipos de muestreo.

sultados del AUC que con los modelos anteriores, además el mejore resultado de precisión también es dado por el modeloo *gbm* entrenado con muestreo por defecto (aunque no es el que mayor AUC tiene). Los valores del estadístico kappa tambien aumentan respecto a los modelos anteriores, teniendo que usando muestreo *smote* tenemos un valor superior a 0.61.

En el caso de *gbm* se repite el mismo patrón en al importancia de variables. La Figura 18 muestra como la variable con mayor importancia en los *gbm* para cualquier tipo de muestreo es *duration*, que corresponde a la duración de la llamada. Se observa como hay 4 variables que tiene una importancia con un diferencia notable al resto, las cuales corresponden a *duration*, *nr.employed*, *euribor3m* y *emp.var.rate*. Además, se mantienen en ese orden en todos los modelos *gbm* para los diferentes muestreos.

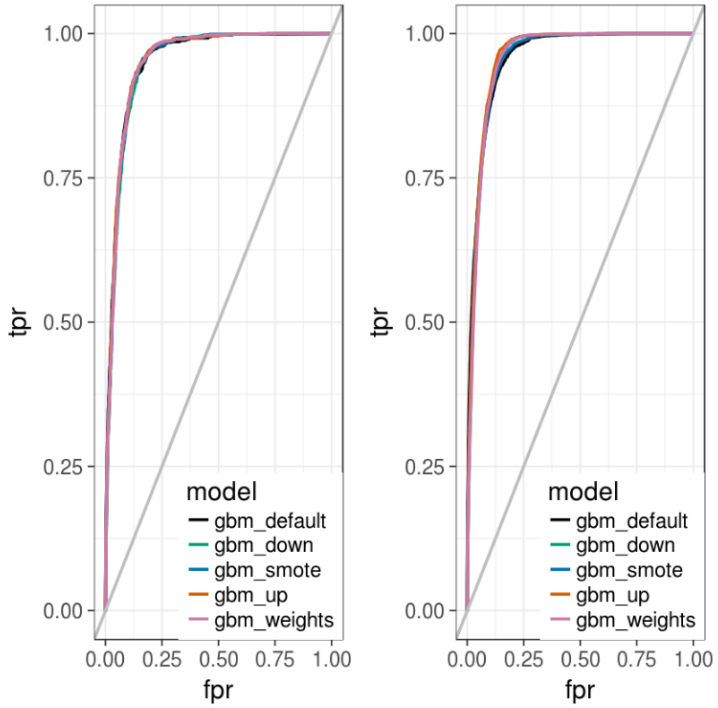

Figure 16: Visualización de las curvas ROC generados haciendo uso de bagging con árboles de decisión, la curva de la izquierda pertenece al test mientras que la derecha al entrenamiento.

	gbm_default	gbm_smote	gbm_up	gbm_down	gbm_weights
AUC	0.9514361	0.9515170	0.9524469	0.9495741	0.9516556
prec.	0.9211623	0.9111084	0.8748161	0.8602256	0.8704022
kappa	0.5686427	0.6152392	0.5595769	0.5297777	0.5497666
AUC(train)	0.9574610	0.9574572	0.9605423	0.9572357	0.9570342
prec.(train)	0.9292697	0.9145598	0.8783672	0.8649750	0.8726365
kappa(train)	0.6121333	0.6325123	0.5753957	0.5459913	0.5602377

Figure 17: Diferentes curvas ROC para los distintos muestreos con rpart.

4 Máquinas de vector soporte

Las máquinas de vector soporte buscan el hiper-plano con mayor separación entre las muestras de datos de distinta clase. Además, en algunos casos se aplica una función kernel previa a los datos con el fin de moverlas a un espacio distinto

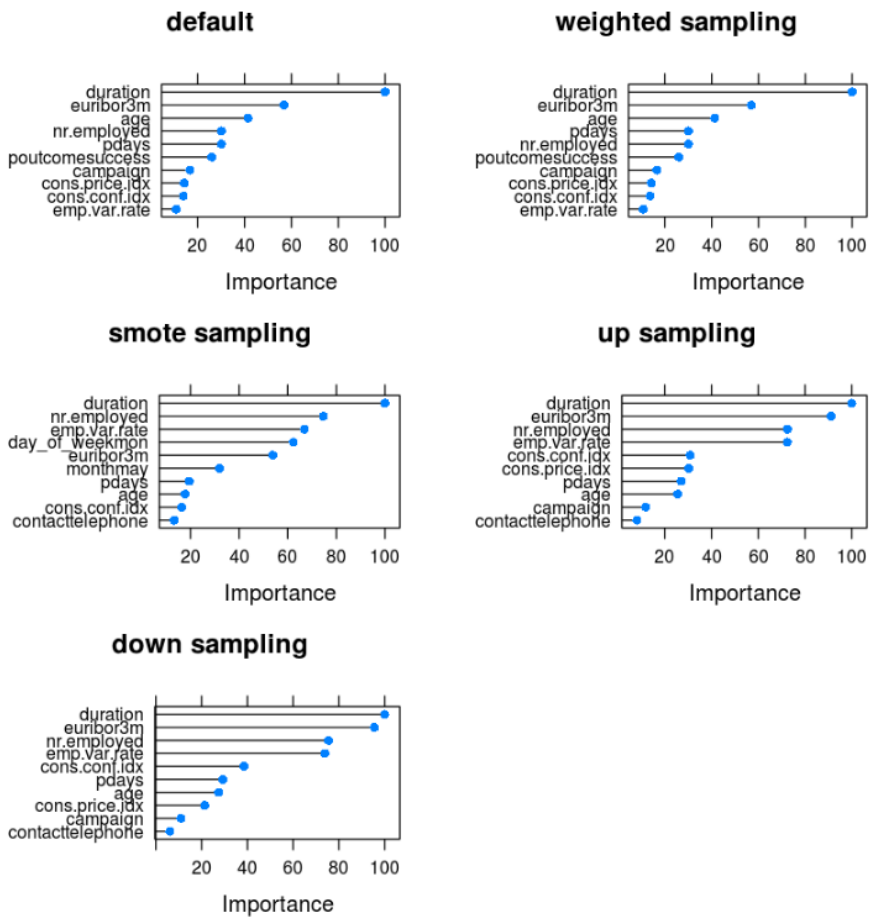

Figure 18: Gráfica comparativa con la importancia de las variables dadas por los modelos de gradient boosting machines entrenados usando los distintos muestreos.

donde es más fácil encontrar su hiper-plano separador. Para este trabajo se han explorado cuatro funciones kernel diferentes:

$$Radial(RBF) \quad \rightarrow \quad K(X,Y) = \exp(-\gamma \cdot \|X-Y\|^2), \gamma > 0 \tag{1}$$

$$Linear \quad \rightarrow \quad K(X,Y) = X^T Y \tag{2}$$

$$Polynomial \quad \rightarrow \quad K(X,Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0 \tag{3}$$

A los datos también se les aplica un escalado y centrado previo a la clasificación. Este paso se realiza principalmente para evitar que los atributos con rangos numéricos mayores dominen a los que tienen menor rango y para evi-

tar problemas numéricos a la hora de realizar cálculos (por los productos escalares entre vectores de las funciones kernel). Para la búsqueda de los mejores parámetros tanto como para el entrenamiento se ha hecho uso del paquete *caret*. En esta parte se siguen usando las técnicas de muestreo que intentan mitigar posibles sobre-ajustes de los modelos derivados de el gran desbalanceo de los datos.

En algunas imágenes aparece el modelo con pesos debido a que se ha creado una función con el fin de automatizar el proceso pero, a fin de cuentas, es un modelo similar (o igual) al por defecto dado que los pesos no tienen ningún efecto si se centran después los datos.

4.1 Kernel de base radial (RBF)

El kernel RBF mapea a un espacio dimensional mayor los datos, funciona adecuadamente cuando la relación entre las clases a predecir y los atributos son no lineales, sin embargo si el número de predictores es realmente grande otros kernels como el lineal pueden funcionar mejor.

Se tienen dos parámetros a ajustar C y σ (siguiendo la notación de *caret*). Para su búsqueda se hace uso de validación cruzada con folds. En concreto se hace uso de 8 folds y los parámetros de búsqueda son seleccionados haciendo una primera búsqueda aleatoria que indica una dirección más precisa en la que buscar. El proceso de ajuste de los parámetros con validación cruzada con 8-folds se puede ver en la Figura 27.

Tras obtener los mejores parámetros para cada técnica de muestreo entrenamos los modelos con la totalidad del subconjunto de entrenamiento y usamos el subconjunto de test para obtener los valores de las métricas visibles en la Figura 20. Los valores de AUC y precisión ahora son algo menores que los obtenidos usando *gbm* y similares a los obtenidos con *ctree*.

Si mostramos la curva ROC con todas las técnicas de muestreo vemos como el sobre-sampleo genera cierto sobre-ajuste en el modelo, teniendo un AUC de 0.96 en train). Finalmente, todos muestreos generan un resultado muy similar con AUC entre 0.93 y 0.94. En términos de la precisión obtenida vemos como el mejor modelo sería el por defecto con una precisión del 91% aproximadamente.

De nuevo si vemos la importancia de cada predictor en los modelos entrenados vemos como las variables *duration*, *nr.employed*, *euribor3m* y *emp.var.rate* son las que más influencia tienen en la decisión final siguiendo un patrón similar que el resto de modelos (ver Figura 22).

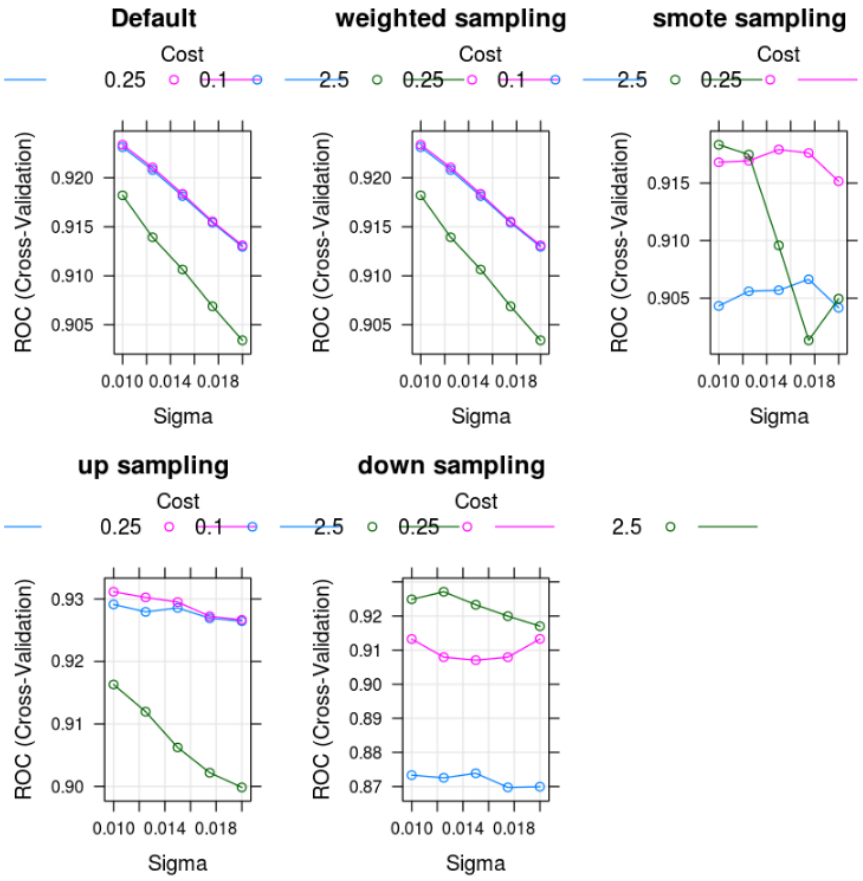

Figure 19: Resumen de la búsqueda de parámetros realizada sobre máquinas de vector soporte (SVM) aplicando la función kernel de base radial (RBF).

4.2 Kernel Lineal

El kernel linear tiende menos al sobre-ajuste que los modelos no lineales y funcionan adecuadamente cuando los datos son linealmente separables. El número de parámetros a ajustar se reduce a C. De nuevo se usa validación cruzada con 8-folds y el paquete *caret*. El resultado tras finalizar la búsqueda de parámetros se puede ver en la Figura 23

De nuevo usamos *caret* para entrenar con la totalidad del conjunto de entrenamiento y el subconjunto de test para obtener los resultados de las métricas. En este caso el kernel linear empeora ligeramente los resultados obtenidos con

	radSVM.down_default	radSVM_smote	radSVM_up	radSVM_down
AUC	0.9370406	0.9345891	0.9318694	0.9333327
prec.	0.9104953	0.8935753	0.8672143	0.8517656
kappa	0.4581711	0.5649334	0.5206552	0.4973001
AUC(train)	0.9497412	0.9453058	0.9631647	0.9380667
prec.(train)	0.9156553	0.9042847	0.8963773	0.8591087
kappa(train)	0.4855304	0.5963918	0.6187765	0.5153656

Figure 20: Tabla con los resultados de las métricas para la máquina de vector soporte con kernel RBF.

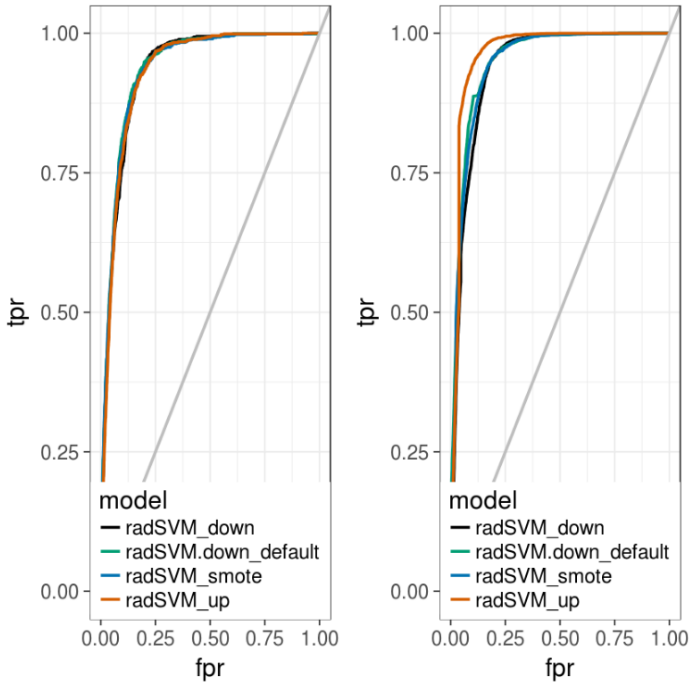

Figure 21: Gráficas con las curvas ROC de test (izquierda) y entrenamiento (derecha) de cada modelo entrenado con el kernel RBF.

el AUC. Si comparamos la precisión se observa que también obtiene valores ligeramente inferiores a los obtenidos con el kernel RBF. Si comparamos los resultados obtenidos entre entrenamiento y test se ve como los modelos no tienen prácticamente sobre-ajuste pero obtienen peores resultados. Los resultados y la curva ROC de los modelos pueden verse en las Figuras 24 y 25.

De nuevo se obtiene la importancia de las variables para realizar la predicción.

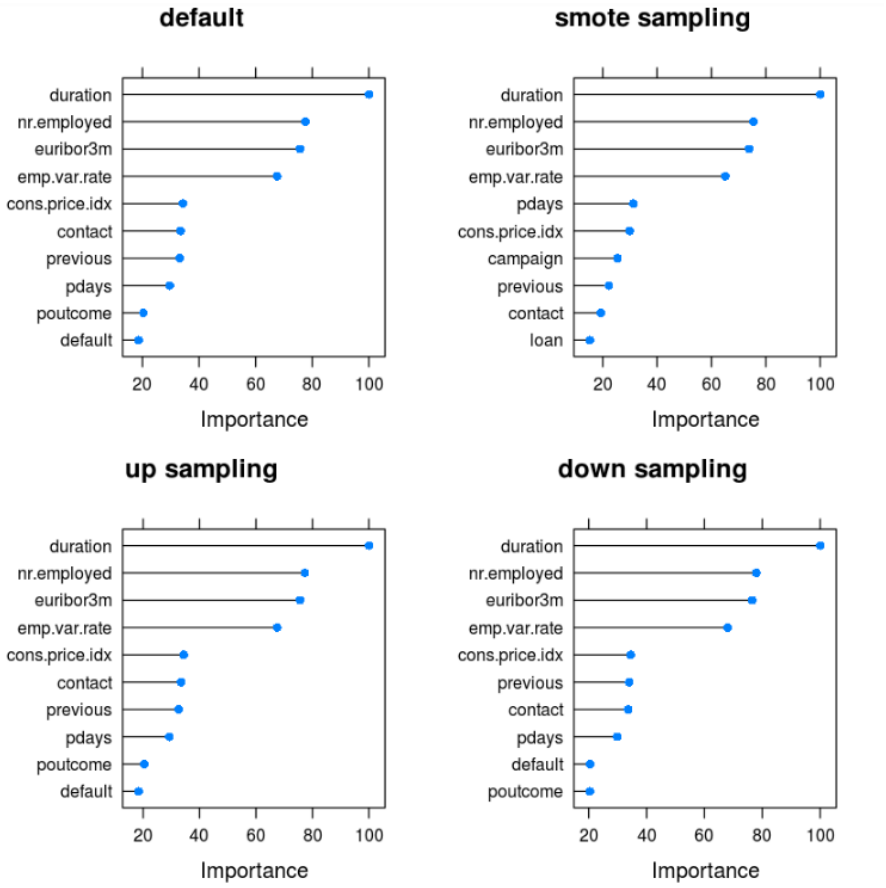

Figure 22: Importancia de las variables de los modelos entrenados con el kernel RBF.

Como en los modelos anteriores se aprecia una mayor importancia de las variables *duration*, *euribor3m*, *nr.employed* y *emp.var.rate* coincidiendo con los modelos previos tal y como se puede observar en la Figura 26.

4.3 Kernel Polinomial

El último kernel usado en las máquinas de vector soporte es el polinomial. Este kernel mapea las observaciones a un espacio no lineal lo que permite aprender este tipo de relaciones de los datos. A la hora de la búsqueda de parámetros es más complejo que los dos anteriores dado que incluye 3 parámetros a determinar: Grado, Escala y Coste. Aunque su uso habitual es para reconocimiento

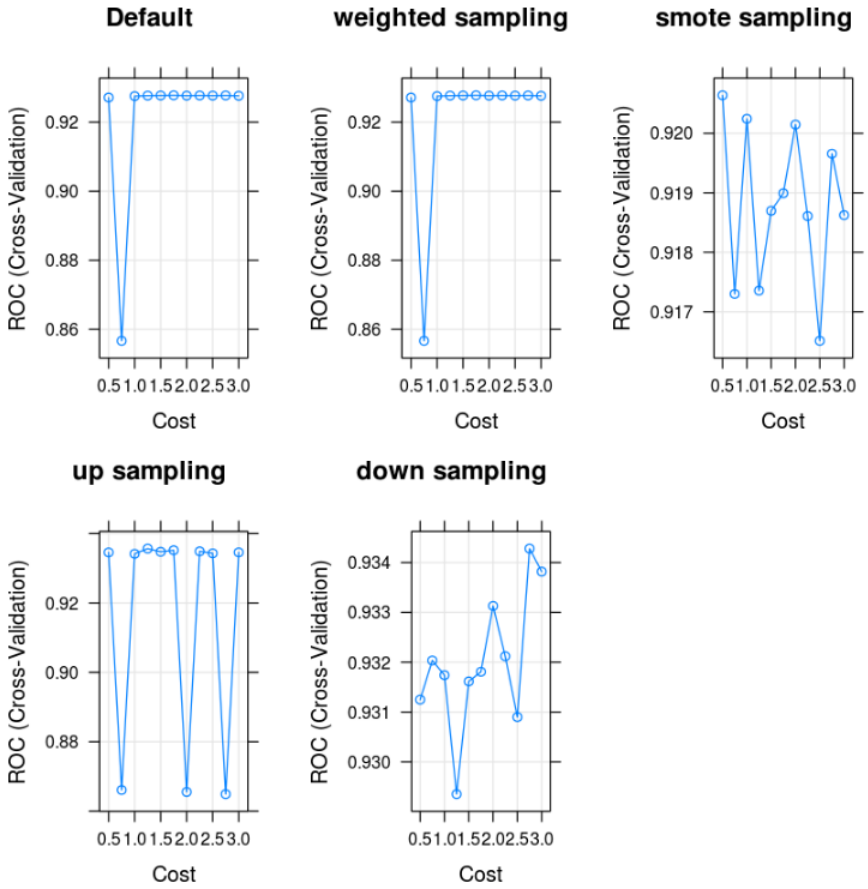

Figure 23: Resumen de la búsqueda de parámetros realizada sobre máquinas de vector soporte (SVM) aplicando la función kernel de base radial (RBF).

del lenguaje (NLP) se ha añadido a este trabajo por completitud. De nuevo, para la búsqueda de parámetros se ha hecho uso del paquete *caret*. En este caso dado el tiempo consumido se ha hecho tan solo uso de una primera búsqueda aleatoria con 5 valores de cada parámetro como se observa en la Figura 27.

Una vez se obtienen los mejores parámetros, se entrena con la totalidad de los datos de entrenamiento. Se obtiene un resultado que es similar al obtenido con el kernel RBF, pero el método gbm sigue obteniendo el mejor resultado tanto en AUC como en precisión. Las Figuras 29 y 29 muestran los resultados y la curva ROC para entrenamiento y test. Se observa que no existe casi sobre-ajuste en los modelos entrenados, esto esta relacionado con la selección del Grado con

	linSVM.down_default	linSVM_smote	linSVM_up	linSVM_down
AUC	0.9335936	0.9293725	0.9367560	0.9360834
prec.	0.9089014	0.8930848	0.8564247	0.8553212
kappa	0.4443677	0.5525000	0.5080280	0.5063361
AUC(train)	0.9352676	0.9329827	0.9378712	0.9370876
prec.(train)	0.9061542	0.8960402	0.8605186	0.8600588
kappa(train)	0.4153852	0.5546259	0.5161845	0.5151888

Figure 24: Tabla con los resultados de las métricas para la máquina de vector soporte con kernel RBF.

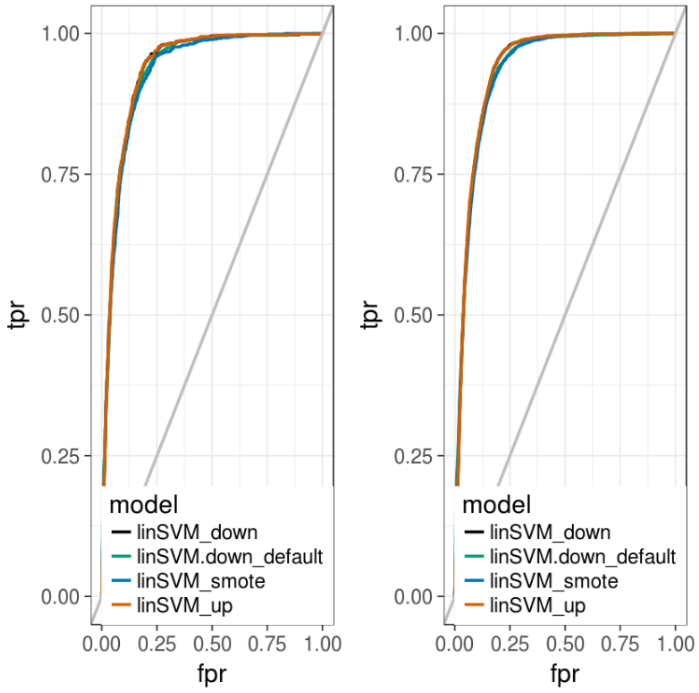

Figure 25: Gráficas con las curvas ROC de test (izquierda) y entrenamiento (derecha) de cada modelo entrenado con el kernel Lineal.

valor $d \leq 2$ en todos modelos, mayores valores pueden causar sobre-ajustes.

A la hora de observar la importancia de las variables se vuelve a repetir el mismo patrón visto para el resto de modelos con 4 variables que destacan por encima del resto. La Figura con la importancia de las variables puede observarse en 30.

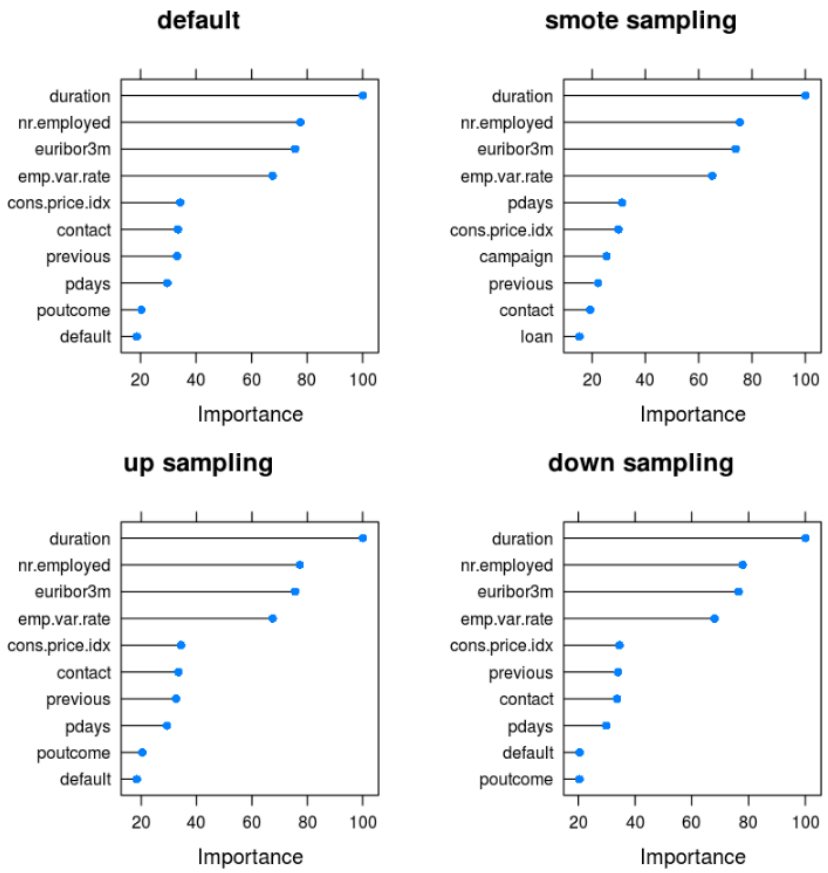

Figure 26: Importancia de las variables de los modelos entrenados con el kernel Lineal.

5 Conclusión

Si tenemos en cuenta el conjunto global de todos modelos parece que si se tiene en cuenta el area por debajo de la curva ROC (AUC) y la precisión, los modelos que hacen uso de boosting, más concretamente Gradient Boosting Machines (gbm) obtienen los mejores resultados.

El hecho de que el conjunto de datos este fuertemente desbalanceado ha hecho replantear desde el inicio las métricas a emplear así como a proponer estrategias que mitiguen dichos efectos como sobre-muestro, sub-muestro, muestro *SMOTE* o muestreo con pesos.

Finalmente se ha visto como los modelos ***gbm*** **entrenado con el muestreo**

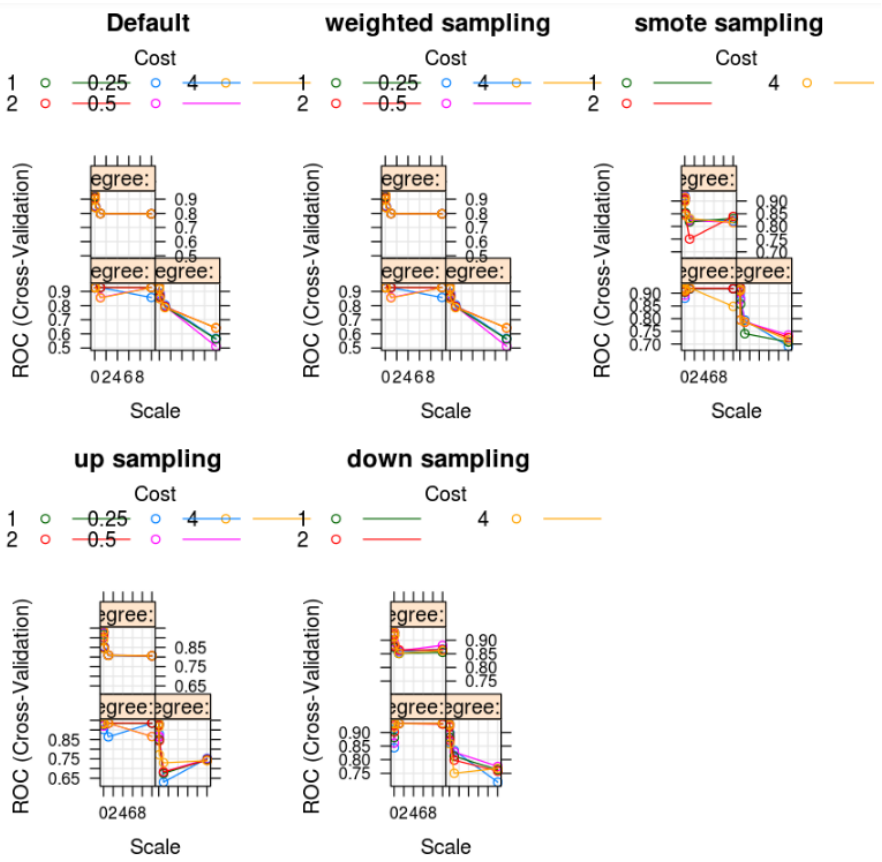

Figure 27: Resumen de la búsqueda de parámetros realizada sobre máquinas de vector soporte (SVM) aplicando la función polinomial.

	polySVM.down_default	polySVM_smote	polySVM_up	polySVM_down
AUC	0.9376650	0.9355014	0.9368666	0.9361078
prec.	0.9113536	0.8989701	0.8564247	0.8560569
kappa	0.4691676	0.5761216	0.5093060	0.5082086
AUC(train)	0.9437449	0.9418211	0.9377615	0.9370185
prec.(train)	0.9121613	0.9034572	0.8609170	0.8595072
kappa(train)	0.4607826	0.5840312	0.5166509	0.5134376

Figure 28: Tabla con los resultados de las métricas para la máquina de vector soporte con el kernel Polinomial.

estandar obtenía el mejor resultado en precisión con **AUC=0.9514**, **precisión=0.9212** y valor del estadístico **kappa=0.5686**. También cabe men-

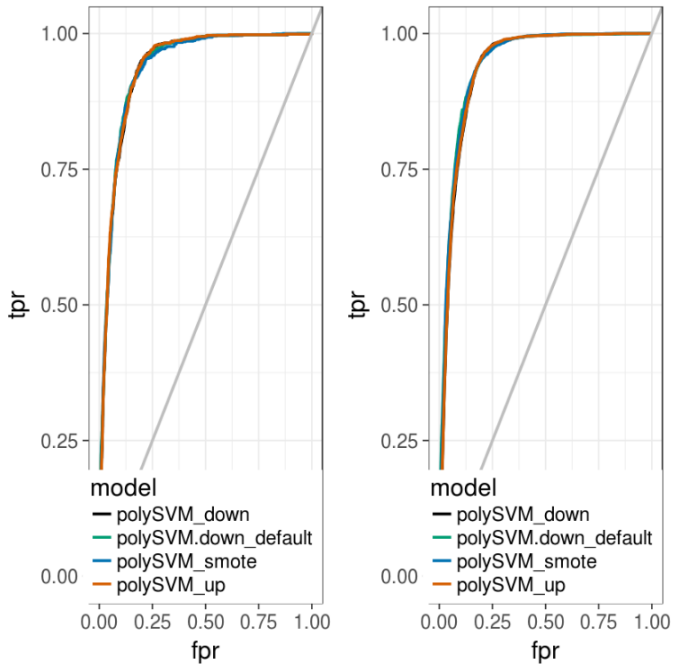

Figure 29: Gráficas con las curvas ROC de test (izquierda) y entrenamiento (derecha) de cada modelo entrenado con el kernel Polinomial.

cionar que el **muestreo *SMOTE* para el modelo *gbm*** obtiene resultados similares a los del muestreo estándar con **AUC=0.9515**, **precisión 0.9111** y valor del estadístico **kappa=0.6152** lo que incluso podría indicarnos que se trata de un **mejor modelo que el de muestreo estándar** a pesar de tener peor precisión. Aunque existen modelos con mejor valor de AUC como el muestreo por pesos del modelo *gbm* sus valores de precisión y kappa son mucho peores por lo que se decide no considerarlos en esta propuesta final.

Además, si tuviéramos que proponer a la compañía una estrategia futura a seguir basada en el análisis de los datos, es decir en los factores determinantes para que el cliente contrate dicho bono bancario. Se tendría que tener en cuenta el subgrupo de cuatro variables con alta importancia en todos modelos. Parece ser que la **duración** de la llamada es un parámetro determinante a la hora de contratar el deposito, esto se debe a que cuanto mayor interés tenga el cliente a cerca de el producto mayor tiempo estará al teléfono para informarse de ello e incluso contratarlo. Además, la variable de **número de empleados** (nr.employed) también tiene un peso crucial en los modelos, parece ser que los clientes tienen mayor confianza en un banco de acuerdo a su tamaño, lo que indirectamente puede provocar que sea más influyente y valorado. La **variación de**

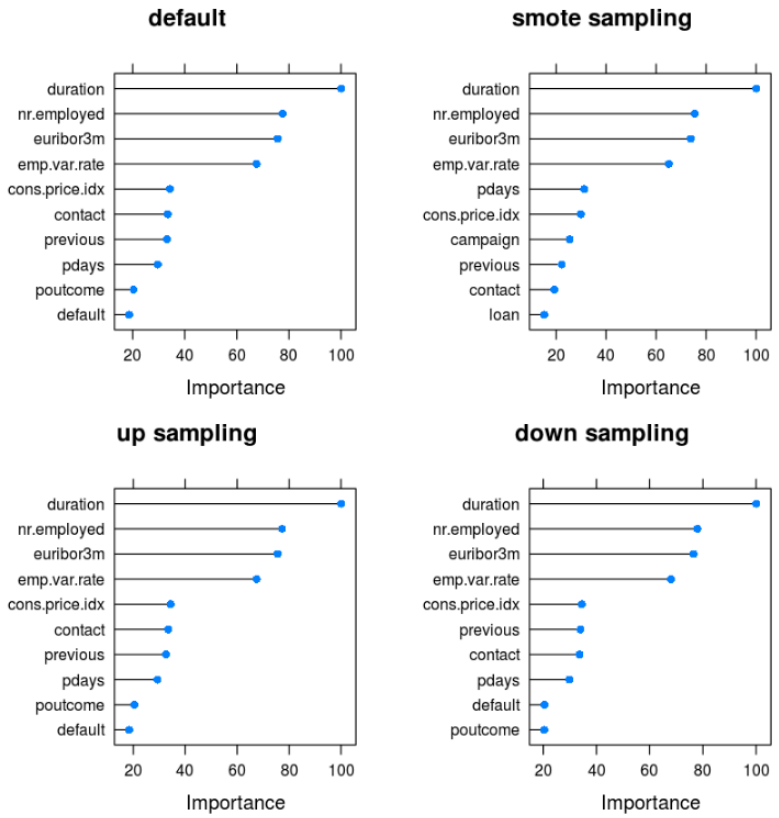

Figure 30: Importancia de las variables de los modelos entrenados con el kernel Polinomial.

empleados (emp.var.ratio), la cual causa influencia negativa en la respuesta. Cuanta mayor variación menor contratación del servicio financiero. Esto guarda sentido ya que la variación en el empleo puede marcar cierta inestabilidad haciendo que los clientes no estén seguros de si contratar el producto. Por último tenemos otro indicador como el **euribor3m**, que se trata de un interés trimestral, cuanto mayor sea el beneficio del cliente (mayor interés) mayor será su decisión para contratar dicho producto financiero.

Las recomendaciones a realizar al banco para tener un mejor resultado irían enfocadas en estas cuatro variables, intentando realizar llamadas de calidad que ”enganchen” al cliente, crear campañas de marketing en momentos de bonanza en términos del índice trimestral del euribor que permitan un mejor interés para los clientes, y en momentos de estabilidad económica donde el empleo se mantiene estable sin grandes fluctuaciones y por último, deberían contratar un mayor número de empleados, de manera que les posicione en el mercado, les de

influencia y presencia de cara a ganarse la confianza del cliente.