Aprendizaje Automático. Proyecto final

Manuel Herrera Ojea
53583380G

Ismael Marín Molina
50627728J

Contents

1 Descripción del problema **2**

2 Conjunto de datos utilizado **2**

3 Modelos empleados **3**
3.1 Random Forest 3
3.2 Motivación para RF 3
3.3 Support Vector Machine 4
3.4 Motivación para SVM 4
3.5 Combinación de ambos modelos 4

4 Desarrollo de la clasificación **5**
4.1 Tratamiento de los datos 5
4.1.1 Valores Perdidos 5
4.1.2 Desbalanceo del modelo 6
4.1.3 Valores Categóricos 9
4.2 Toma de hiperparámetros 10

5 Resultados obtenidos **11**

6 Conclusiones **14**

1 Descripción del problema

Nuestro objetivo ha sido la obtención de un predictor que nos permita, dados los datos recopilados de diferentes clientes de un banco, conocer su interés en la suscripción a un nuevo tipo de depósito.

Para ello nos hemos servido de la base de datos Bank Marketing Data Set (BMDS), alojado en el repositorio de aprendizaje automático UCI en `https://archive .ics.uci.edu/ml/datasets/bank+marketing`.

2 Conjunto de datos utilizado

El conjunto BMDS consta de 41188 instancias, cada una correspondiente a los datos obtenidos por vía telefónica de los clientes de una institución bancaria portuguesa. Durante la extracción de datos no fue posible obtener todo de todos los clientes, por lo que el conjunto de aprendizaje presenta valores nulos en ciertas características, representados por la cadena "`unknown`", que hemos tenido que tratar convenientemente para que los modelos de aprendizaje que hemos utilizado sepan qué hacer al encontrárselos. Este conjunto de datos contiene datos de varios años, pero nos hemos ceñido a los que conciernen al año más actual, 2014, pues una incluir en el aprendizaje información sobre épocas muy anteriores no nos arrojaría un resultado muy deseable ahora.

Los datos tomados por la entidad bancaria muestran 20 características de cada uno de sus clientes, teniendo en cuenta entre ellas los posibles valores nulos recién mencionados: (1) edad, (2) ocupación, (3) estado civil, (4) nivel de estudios, (5) crédito por defecto, (6) petición de un crédito inmobiliario, (7) petición de un crédito personal, (8) método de comunicación con el banco, (9) mes, (10) día de la semana y (11) duración de la última comunicación (en este caso, si la duración de la conversación es 0, es decir, no cogió el teléfono, la predicción es "no" en todos los casos), (12) cantidad de contactos realizados con el banco, (13) días transcurridos desde el último contacto, (14) veces contactado antes de la campaña para la suscripción, (15) resultado de la campaña de suscripción anterior, (16) tasa de variación de empleo, (17) índice de precios de consumo, (18) índice de confianza de consumo, (19) euribor a tres meses, y (20) cantidad de empleados. La característica a predecir es (21) Interés en el producto, que puede tomar los valores "sí" y "no".

De estas características, las número (2), (3), (4), (5), (6), (7), (8), (9),

(10) y (15) son categóricas, presentándose en formato cadena de texto, mientras que el resto son valores numéricos.

3 Modelos empleados

3.1 Random Forest

RF es un modelo de aprendizaje automático que se sirve de generar un conjunto de árboles de decisión muy simples durante la fase de aprendizaje para, durante la fase de predicción, tomar las respuestas individuales de todos los árboles para dar una respuesta final. De manera aislada, los árboles de los que está compuesto un RF ya entrenado dan una predicción poco realista, pero, combinando todas las respuestas que dan, RF termina ofreciendo una predicción considerablemente buena. En el caso concreto de su implementación en scikit-learn, RF utiliza el así llamado "valor más votado", que coincide con la moda de las respuestas de la población.

Esto arroja una predicción más fiable que la de un solo árbol de decisión, pues disminuye drásticamente la probabilidad de que caigan en un sobreajuste durante el aprendizaje, problema que suele surgir al utilizar árboles de decisión aislados, debido a la alta varianza que estos tienen.

3.2 Motivación para RF

La existencia de tantas características categóricas en nuestro conjunto de datos ha sido un gran motivador de la elección del modelo de aprendizaje RF, por ser los árboles de decisión, maquinaria interna de la predicción realizada por un RF, especialmente potentes para realizar predicciones tomando valores categóricos, pues la naturaleza discreta de sus ramificaciones se acopla muy bien a atributos cuyo dominio no sea continuo.

Otro factor que ha influido en nuestra decisión de utilizar RF es que a diferencia del resto de modelos, RF, o en concreto Decision Tree, nos da aquellas características que más efecto han tenido a la hora de clasificar el problema, lo cual, analizando la naturaleza comercial de nuestro problema, consideramos que es algo muy significativo.

Un tercer motivo que nos ha hecho optar por utilizar RF es que nos dará una muy buena aproximación a la función desconocida que queremos predecir, pero exigiendo una capacidad de computo más reducida que el

resto de modelos estudiados, por lo que no necesitará estar tanto tiempo entrenándose para alcanzar una solución aceptable.

3.3 Support Vector Machine

El modelo SVM es un clasificador lineal que se basa en encontrar la mejor separación posible entre las distintas clases del problema. En contraste con un clasificador lineal simple, para SVM dos soluciones no son equivalentes únicamente cuando obtiene con ellas la misma tasa de acierto en el conjunto de aprendizaje; busca también que el hiperplano que ha encontrado sea el que deja el "pasillo" más ancho entre las distintas clases, ampliando así su capacidad de generalización.

Para la creación de esta separación, y partiendo de que la mayoría de datos reales no son linealmente separables, ya sea por ruido en los datos o por la naturaleza del propio problema, realiza una transformación de los valores con el uso de una función, a la que llamamos *kernel*, que elegimos de forma previa al aprendizaje.

3.4 Motivación para SVM

Hemos pensado en el uso del SVM al tratarse nuestro problema de una clasificación binaria, pues su aplicación sería directa, y por contar, además, con la presencia de numerosas características numéricas.

Adicionalmente, en la información sobre el conjunto de datos que hemos utilizado facilitada en la propia web donde podemos descargarlo mencionan SVM como un modelo especialmente recomendado para tratar estos datos.

3.5 Combinación de ambos modelos

Como último hemos probado una combinación de los dos modelos mencionados. Para ello hemos razonado y procedido como exponemos a continuación.

Hemos usado RF para iniciar una primera aproximación del error sobre el conjunto de aprendizaje. Como hemos mencionado anteriormente, RF ofrece como predicción la moda de las predicciones de los árboles de decisión simples que alberga. Es por ello que aquellas instancias en las que RF haya fallado son instancias donde la mayoría de los árboles de nuestro RF han realizado una mala clasificación. Decidimos entonces eliminar estas instancias donde RF falla, por considerarla una instancia ruidosa, y reduciendo así la envergadura

del conjunto de entrenamiento que pasaremos entonces a SVM, pues este es mucho más computacionalmente pesado.

También obtenemos de este aprendizaje con RF cuáles son las características más relevantes para la clasificación de nuestro conjunto de datos. Esto es posible, como ya hemos mencionado, por la capacidad explicativa de este modelo. Al saber cuáles han sido las características relevantes tras el aprendizaje, eliminamos el resto de características de nuestro conjunto, consideradas ahora de pariticipación despreciable en la predicción.

Este conjunto resultante, con instancias problemáticas eliminadas y con la dimensionalidad reducida, es el que utilizamos para el entrnamiento con SVM.

Con esta combinación de los modelos hemos podido explotar mucho mejor las capacidades de SVM, así como crear un modelo más acertado para el problema que nos atañe.

4 Desarrollo de la clasificación

4.1 Tratamiento de los datos

Ha sido necesario un preprocesamiento de los datos facilitados por BMDS para su correcto tratamiento por las funciones implementadas en el framework scikit-learn. Primero ha sido necesario arreglar los valores perdidos. Después hemos corregido el desbalanceo entre instancias de cada clase, tras ello una recodificación de las características categóricas y finalmente una estandarización de las características numéricas. Esto ha sido realizado, de forma aislada, tanto en la partición de training como en la de test, las cuales han sido divididas en un 70% del tamaño para Training y Validación y un 30% para Test.

4.1.1 Valores Perdidos

Cuando se generó la base de datos BMDS, como hemos mencionado anteriormente, no fue posible obtener todos los datos requeridos de todos los clientes, en ocasiones ni tras haber realizado el banco varios contactos con un mismo cliente. Es por ello que hay valores desconocidos, que están registrados como la cadena de texto `'unknown'`. Esto tiene una semántica implícita muy clara, pero los modelos de `scikit-learn` no son capaces de diferenciarlos del resto de posible valores del dominio, por lo que hemos tenido que tratarlos

Table 1: Distribucion valores perdidos

Ocupación	Estado C.	Nivel de Estudios	Crédito por defecto	PCI	PCP
0.801%	0.194%	4.202%	20.872%	2.403%	2.403%

manualmente.

Inicialmente hemos hecho un conteo del número de valores perdidos en cada característica, para comprobar cómo estaban distribuidas en nuestro conjunto, y el resultado ha sido el mostrado en la tabla 1.

Como podemos comprobar en la tabla 1 la mayoría de estos se concentraban en la columna del Crédito por defecto. Con una menor representación tenemos las columnas de présgamo de crédito inmobiliario (PCI) y préstamo de crédito personal (PCP), junto con el Nivel de Estudios, con una menor cantidad tenemos la ocupación y el estado civil.

Para realizar la imputación de estos valores perdidos de la manera más eficaz posible sin realizar un excesivo cómputo hemos dividido los valores perdidos entre aquellos que tenían más de un 1% de los valores de su clase perdidos y los que tenían menos.

Aquellos con menos de esta cantidad hemos decidido imputarlo usando el valor más frecuente, el cual nos parece un mejor estimador de la realidad de la población que la media o mediana.

Para resto hemos decidido usar una táctica más sofisticada, imputándolos haciendo uso de un clasificador 3-NN, para predecir cual será su valor. Para ello hemos separado los datos según tuvieran o no el valor, hemos entrenado el modelo con los datos que no tenían valores perdidos y hemos cambiado los "unknown" por los valores predichos por el modelo, solucionando con ello los 12718 valores perdidos que teníamos.

4.1.2 Desbalanceo del modelo

En BMDS, de las 41188 instancias facilitadas, 36548 pertenecen a la clase negativa y solo 4640 a la clase positiva, cuando lo ideal es que hubiese habido una representación aproximadamente equitativa de ambas clases.
Tras el tratamiento de los valores perdidos hemos comprobado en el conjunto

de aprendizaje un fuerte desequilibrio entre ambas clases, que provoca que el modelo no pueda predecir bien la clase minoritaria, lo cual conlleva que, aunque la calidad del modelo parezca acertada, el acierto sea mínimo para la clase minoritaria. En nuestro caso el modelo acertaba el 95% de las pertenecientes a la clase 0, pero solo el 50% de aquellas que estaban en la clase 1, lo que no lo diferenciaba de un clasificador que eligiese al azar la clase a la que pertenece un ejemplo de la clase positiva.

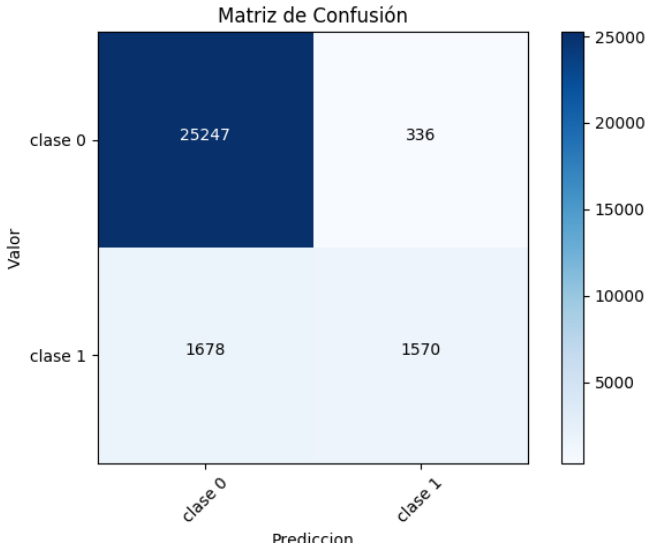

Figure 1: Matriz de Confusión Desbalanceada

Para solucionar esto hemos usado el algoritmo SMOTE, siglas de Synthetic Minority Over-sampling Technique. Es una técnica utilizada en conjuntos donde las representaciones de cada clase no están equilibradas.

Los desarrolladores de SMOTE han estudiado este problema, y su herramienta, en lugar de únicamente reducir la cantidad de instancias de la clase mayoritaria, en nuestro casos los `'no'`, hasta alcanzar una representación equitativa, ofrecen un método con el que aumentar la cantidad de instancias de la clase minoritaria. Esto es importante, pues tratar únicamente la poda de instancias negativas habría llevado a aumentar significativamente el comportamiento de nuestro predictor para favorecer los casos positivos más de lo deseado. Para aumentar la cantidad de instancias de la clase minoritaria se han servido de métodos sintéticos de generación de instancias. Según los estudios que han realizado, tanto con Ripper como con clasificadores bayesianos ingenuos,

los resultados han sido mucho más fructíferos al aplicar a conjuntos no equilibrados este tipo de poda y generación que al aplicarle únicamente podas.

El algoritmo SMOTE funciona creando ejemplos sintéticos de la clase minoritaria para equilibrar ambas clases, usando para ello un clasificador k-NN interno, que toma los k vecinos más cercanos y genera un ejemplo usando alguno de ellos. El pseudocódigo de este algoritmo es el siguiente, como ha sido presentado en el atículo de Nitesh V. Chawla y Kevin W Bowyer en 2002:

Algoritmo 1 SMOTE

Entrada: T = Poblacion de la menor clase ,N = numero de ejemplos sinteticos, k = numero de elementos del kNN.

Salida: Array con los ejemplos sinteticos.

para todo $i \in T$ **hacer**
  Computamos los k vecinos más cercanos y guardamos el indice en $nnarray$.
  Populate(N, i, $nnarray$)
fin para
mientras $N \neq 0$ **hacer**
  Elegimos un vecino aleatorio de la instancia i
  para todo Atributo en el ejemplo i **hacer**
    Compute: $dif = Sample[nnarray][nn][attr] - Sample[i][attr]$
    Compute: gap = random number entre 0 y 1
    Compute: $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
  fin para
  $newindex++$
  $N--$
fin mientras
devolver Population Synthetic

Para la implementación de este algoritmo hemos usado la librería `imbalanced learn`, con los parametrós $k = 3$ y la función de generación `borderline 1`, que es la función descrita en el pseudocódigo expuesto.

Las clases han sido equilibradas hasta igualar ambas dejando un total de 48818 instancias en el conjunto de aprendizaje. Tras volver a realizar la matriz de confusión del conjunto de entrenamiento, estos son los nuevos resultados obtenidos:

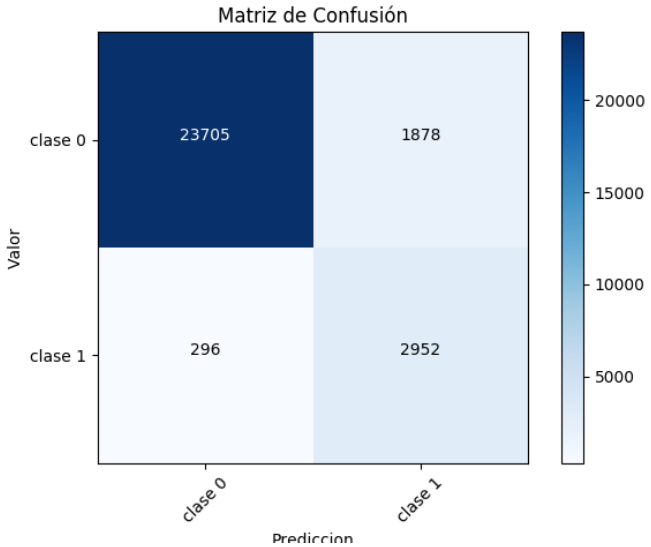

Figure 2: Matriz de Confusión Balanceada

4.1.3 Valores Categóricos

Otra convención que utiliza scikit-learn es utilizar valores numéricos enteros para la codificación de variables categóricas. Las catacterísticas catagóricas de BMDS están codificadas todas como cadenas de texto, por lo que ha sido necesaria una conversión. Para ello hemos utilizado el codificador one-hot, implementado también dentro de scikit-learn.

La forma de proceder del codificador one-hot es la siguiente. En lugar de simplemente asignar valores enteros a los posibles valores del dominio de una característica categórica, crea una nueva característica por cada valor que pueda tomar una categoría. Esto aumenta considerablemente la dimensionalidad del problema, pero arroja unos resultados mucho más satisfactorios.

El principal problema que tiene codificar los valores de una característica categórica como valores enteros es que se establece inherentemente una relación de orden entre ellos, y el modelo puede aprender que un mayor o un menor valor de dicha característica es el preferible, o buscar un valor concreto y beneficiar colateralmente a los valores colindantes. Esto es problemático porque el orden en el que se hayan codificado los valores de la característica categórica influye directamente en la respuesta que dará el predictor, cuando

una permutación de estos valores no debería influir de forma apriorística al aprendizaje.

La solución que ofrece la codificación one-hot, crear una nueva característica por cada valor posible, hace que no existan relaciones implícitas entre estos valores antes de que comience el aprendizaje. Cada dato de entrada tendrá un valor 1 en la característica correspondiente al valor que tenía antes, y 0 en el resto de características que se hayan creado que equivalgan a los demás posibles valores que podía haber tomado este dato en esta característica. Así, de las nuevas características que se crean por cada característica categórica originaria, solo en una tendrá el dato transformado el valor 1, y 0 en todos los demás.

Este proceso se conoce también como binarización de una variable categórica, donde se convierte una variable con n posibles valores en algún dominio en n variables binarias.

4.2 Toma de hiperparámetros

Para determinar los mejores hiperparámetros hemos usado el metaalgoritmo de la `metrics.gridSearch`. Con este método, determinamos los parámetros con los que queremos probar los distintos modelos sobre el conjunto de entrenamiento, y nos devuelve aquella combinación de parámetros que mejores resultados nos den.

En el caso de RF no era necesario realizarle una Cross Validation para comprobar el error ya que el internamente usa el método de bagging para estimar el error que se obtendrá, por tanto hemos probado a base de ensayo y error con distintas combinaciones de parámetros.

En el caso de la SVM los parámetros en los que nos hemos interesado han sido en la regularización, el kernel y la función de decisión. Los mejores valores se han encontrado con una regularización tipo Lasso y con la función de kernel squared-exponential kernel (RBF) con la siguiente fórmula:

$$k_{SE}(x_i, x_j) = \sigma^2 \exp \frac{(x_i - x_j)^2}{2l^2} \tag{1}$$

donde l es el rango máximo de los datos, y σ determina la distancia entre la media de tu función y la de la población.

Para el caso del RF hemos probado incrementando el número de arboles en la decisión, y las distintas funciones para estimar las variables que tomará cada árbol para ramificar. En el número de árboles vimos que con 50

obteníamos resultados parecidos a usar 100 o 150, pero siendo la capacidad de cómputo mucho menor, por lo que decidimos quedarnos con los 50 árboles. En el máximo de características tomamos $\sqrt{p}$ y $\log p$, encontrando los mejores resultados con el primero de ellos.

5 Resultados obtenidos

Como hemos explicado, nuestro primer objetivo ha sido utilizar un modelo de aprendizaje RF para poder reducir el conjunto de aprendizaje original y entrenar con el conjunto reducido un modelo SVM, obteniendo así un ensemble de modelos.

Tras aplicar RF, las características más relevantes que ha encontrado de nuestro modelo han sido las siguientes:

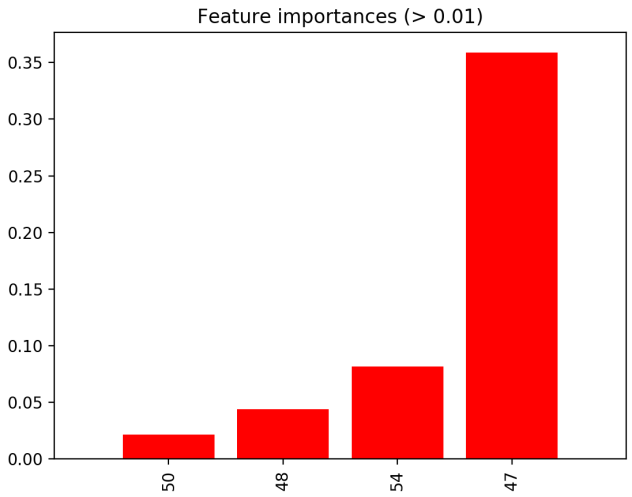

Figure 3: Importancia de las características

Las características que nuestro clasificador RF ha concluido que tienen una relevancia en la predicción mayor al 0.01 %, 50, 48, 54 y 47, son las correspondientes, tras haber realizado una binarización de las variables categóricas con el codificador one-hot, con las características iniciales (11) Duración de la última llamada al cliente, (12) cantidad de llamadas realizadas al cliente durante la campaña actual, (14) veces que el cliente fue contactado

antes de la campaña y (19) euribor a tres meses.

Tras este aprendizaje, RF nos ha proporcionado un clasificador que arroja los siguientes errores sobre el conjunto de aprendizaje con el que ha sido entrenado:

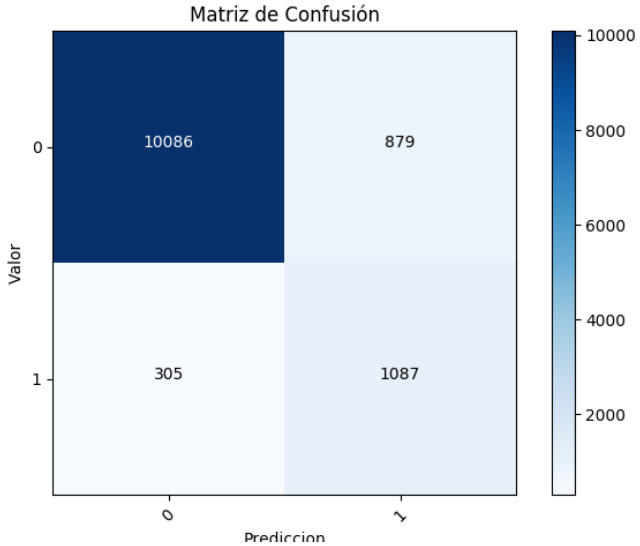

Figure 4: Matriz de confusión tras RF

Como habíamos previsto que ocurriría, el error al clasificar ejemplos pertenecientes a la clase negativa es muy pequeño, pero lo es casi del 30 % al clasificar ejemplos de la clase positiva, porcentaje que no consideramos aceptable.

Tras esto, realizamos la reducción de dimensionalidad del conjunto de datos, quedándonos con las características cuya importancia hubiese superado el 0.01 % que habíamos propuesto, y eliminamos del conjunto de aprendizaje las instancias para las que RF no realizada una buena clasificación. A este conjunto de aprendizaje resultante le aplicamos entonces SVM, y con él obtenemos los siguientes errores:

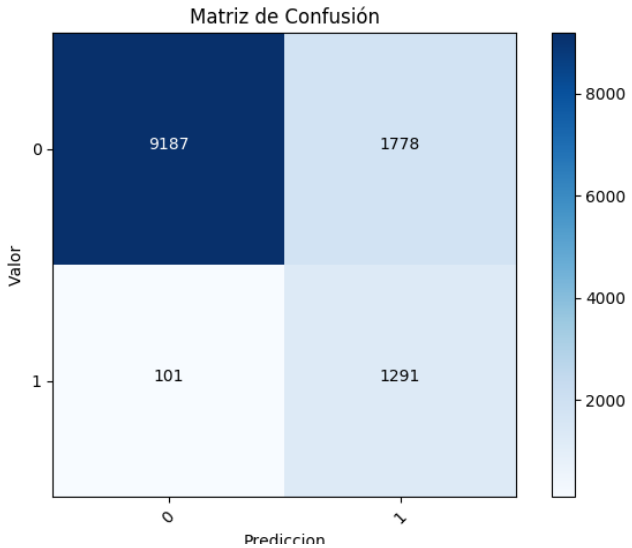

Figure 5: Matriz de confusión tras el ensemble completo

La predicción ahora ha empeorado en los casos negativos, pero ha mejorado mucho al tener en cuenta tanto los ejemplos negativos como los positivos, estando ambos cerca del 10 % de error.

Otra métrica que hemos utilizado para evaluar la bondad del ajuste de nuestro modelo ha sido la curva ROC, que ha arrojado los siguientes resultados tras haber ejecutado nuestro ensemble RF-SVM:

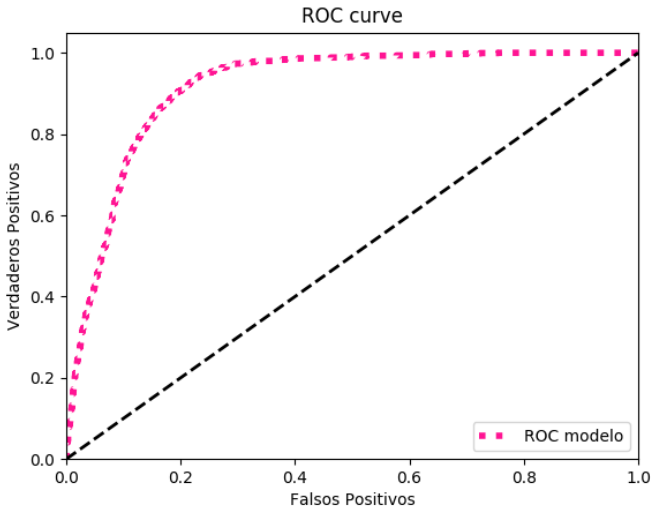

Figure 6: Curva ROC tras el ensemble completo

6 Conclusiones

Todas las características que nuestro RF ha seleccionado como pertinentes en la predicción han sido características numéricas. Esto puede ser información importante para la entidad bancaria que ha generado estos datos, pues puede apostar por aumentar su valor en futuros contactos con clientes para aumentar de forma significativa sus beneficios. Es también destacable a nivel comercial que la característica (11) Duración de la última llamada sea tan influyente en el resultado de la predicción, así como que en todos los casos en los que la duración valía 0, en decir, en los casos donde no había habido llamada telefónica al cliente, el resultado ha sido un 'no'. Es por ello que hemos decidido, a la hora de reducir el conjunto deaprendizaje para encauzarlo con SVM, eliminar del conjunto las instancias que tuviesen una duración igual a 0, por considerarlas datos ruidosos que no iban a ayudar en el aprendizaje del clasificador.

Como podemos comprobar tras haber realizado pruebas con RF y con el ensemble RF-SVM, se obtienen unos resultados mucho mejores tras haber utilizado primero un modelo de aprendizaje computacionalmente poco costoso y que nos ofrece información sobre el conjunto de aprendizaje, como lo han sido las instancias poco representativas y las características poco influyentes

en la predicción, y utilizar entonces esa información para utilizar otro modelo, más computacionalmente pesado, para aprender del conjunto de datos que obtener tras aplicar esa información.