Aprendizaje Automático. Proyecto final

Manuel Herrera Ojea
53583380G

Ismael Marín Molina
AAAAAAAAB

Contents

1 Descripción del problema

Nuestro objetivo ha sido la obtención de un predictor que nos permita, dados los datos de un cliente de un banco, saber si es seguro permitir un préstamo a dicho cliente.

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`.

Hemos utilizado como modelo de aprendizaje el algoritmo Random Forest (RF) para generar el predictor, utilizando para ello su implementación en el framwork scikit-learn.

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

todos de todos los clientes, y algunos fueron introducidos a partir de cierto momento, con la toma de datos ya iniciada, por lo que el conjunto de aprendizaje presenta valores nulos en ciertas características, que hemos codificado como la cadena `'numpy.NaN'` para que las funciones de scikit-learn sepan que se trata de valores nulos y puedan ignorarlos.

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, (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.

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 toma valores numéricos dentro un intervalo.

Poe qué RF. Categóricos
categóricos, numéricos
Tratamiento que hemos tenido que hacer de los datos

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 durante la fase de aprendizaje para, durante la fase de predicción, realizar una predicción del dato de entrada pertinente en cada uno de los árboles de dicho conjunto y tomar como respuesta final la moda de las respuestas que han dado los árboles individuales.

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, sobre todo si el árbol es muy profundo. Los RF son también un modelo muy robusto ante datos con una muy alta varianza dentro del

conjunto de aprendizaje.

Con este modelo perdemos la ventaja que nos ofrecían los árboles de decisión simples de poder tener una explicación de por qué se ha predicho el valor que se ha predicho, pues, al aumentar el número de árboles resultantes en el RF, la explicación es tan extensa que deja de ser útil en la práctica.

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 para tratar los ejemplos contenidos en la base de datos BMDS es que nuestro problema consistiese en realizar una clasificación. Al igual que lo mencionado en el párrafo anterior, los árboles de decisión son un modelo de aprendizaje automático de equivalencia directa con un problema de clasificación, pues cada una de sus ramas equivale a ofrecer una respuesta discreta a una pregunta. Todos sus nodos son, por tanto, clasificadores, y todas sus hojas corresponden a una clasificación del dato de entrada.

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 modificar por `'numpy.NaN'` los valores nulos de las entradas. Después hemos aplicado SMOTE para un correcto equilibrio entre instancias de cada clase, tras ello una codificación one-hot para transformar las características categóricas de tipo cadena de texto en valores numéricos discretos y finalmente una normalización de las características numéricas.

Para generar la base de datos BMDS, como hemos mencionado anteriormente, no ha sido 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 utilizan otra convención, consistente en etiquetar con la constante `NaN`, de la biblioteca `numpy`, todos los valores que queramos que sean ignorados. Es por ello que una primera transformación que hemos hecho de los datos ha sido modificar todos los `'unknown'` por `numpy.NaN`.

Lo siguiente que hemos aplicado ha sido SMOTE, siglas de Synthetic Minority Over-sampling Technique. Es una técnica utilizada en conjuntos donde las representaciones de cada clase no están equilibradas. 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. Esta descompensación tan grande, donde un 89 % de los datos pertenece a una de las dos clases y solo el 11 % pertenece a la otra, puede causar problemas en el aprendizaje, por lo que ha sido necesario utilizar un método de equilibrado de participación de instancias de cada clase. Por suerte o por desgracia, una representación tan descompensada es una situación común en los casos reales, hecho que ha motivado el surgimiento de herramientas que palien este problema, como lo es de la que nos hemos servido, SMOTE.

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.

Otra convención que utiliza scikit-learn es utilizar valores numéricos enteros para la codificación de variables categóricas. Los valores de 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 permitación 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 trendrá 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.