Datos desafiantes – Demasiados, muy pocos, muy complejos, parte I

Los datos desafiantes adoptan diversas formas a lo largo de un proyecto de aprendizaje automático, y el recorrido de cada nuevo proyecto representa una aventura que requiere un espíritu pionero. Iniciar con datos inexplorados que deben explorarse y luego se procesan antes de poder utilizarlos con el algoritmo de aprendizaje.

Aun así, puede que aún existan aspectos complejos que deban controlarse para que el proyecto tenga éxito. Es necesario descartar información superflua, cultivar pequeños pero importantes detalles y despejar las marañas de complejidad del camino del aprendiz.

En la era del big data, la sabiduría popular sugiere que los datos son un tesoro, pero como dice el refrán, uno puede tener "demasiado de algo bueno". La mayoría de los algoritmos de aprendizaje automático aceptarán con gusto tantos datos como se les proporcione, lo que genera una serie de problemas similares a la sobrealimentación.

Una gran cantidad de datos puede saturar al aprendiz con información innecesaria, ocultar patrones importantes y desviar su atención de los detalles importantes a los obvios. Por lo tanto, conviene evitar la mentalidad de "cuanto más, mejor" y, en su lugar, encontrar un equilibrio entre cantidad y calidad.

El propósito de este documento es considerar técnicas que permiten adaptarse a la relación señal-ruido de un conjunto de datos. Aprenderás:

- Cómo gestionar conjuntos de datos con una cantidad 'abrumadora' (*overwhelming*) de características.
- Métodos para aprovechar valores de características que faltan o aparecen con muy poca frecuencia.
- Enfoques para modelar resultados objetivo poco frecuentes.

Descubrirás que algunos algoritmos de aprendizaje están mejor equipados para ejecutar estas técnicas de forma independiente, mientras que otros requerirán una intervención más exhaustiva en el proceso.

En cualquier caso, debido a la prevalencia de este tipo de problemas con los datos y a que se consideran algunos de los más complejos del aprendizaje automático, es importante comprender cómo solucionarlos.

El desafío de los datos de alta dimensión

Si alguien dice tener dificultades para gestionar el tamaño de un conjunto de datos, es fácil asumir que se refiere a demasiadas filas o a que los datos utilizan demasiada memoria o espacio de almacenamiento.

De hecho, estos son problemas comunes que dificultan a los nuevos profesionales del aprendizaje automático. En este escenario, las soluciones tienden a ser técnicas más que metodológicas; generalmente se elige un algoritmo más eficiente o se utiliza hardware o una plataforma de computación en la nube capaz de consumir grandes conjuntos de datos. En el peor de los casos, se puede tomar una muestra aleatoria y simplemente descartar algunas de las filas sobrantes.

El desafío de tener demasiados datos también puede aplicarse a las columnas de un conjunto de datos, lo que hace que este sea excesivamente amplio en lugar de excesivamente largo. Puede requerir un poco de pensamiento creativo para imaginar por qué ocurre esto, o por qué es un problema, ya que rara vez se encuentra en los límites ordenados de los ejemplos de enseñanza. Incluso en la práctica real, puede pasar bastante tiempo antes de que alguien se encuentre con este problema, ya que los predictores útiles pueden ser escasos y los conjuntos de datos a menudo se buscan pieza por pieza. Para este tipo de proyectos, tener demasiados predictores sería un problema.

Sin embargo, imaginemos una situación en la que una organización basada en datos, plenamente consciente de la ventaja competitiva del big data, ha acumulado una gran cantidad de información procedente de diversas fuentes. Quizás recopilaron algunos datos directamente en el curso normal de sus operaciones, adquirieron datos complementarios de proveedores y recopilaron otros mediante sensores adicionales o interacciones indirectas y pasivas a través de internet. Todas estas fuentes se fusionan en una única tabla que proporciona un conjunto de características rico, pero muy complejo y variado. La tabla resultante no se construyó cuidadosamente pieza por pieza, sino mediante una combinación de elementos de datos, algunos de los cuales serán más útiles que otros.

Hoy en día, este tipo de tesoro de datos se encuentra predominantemente en organizaciones muy grandes o con un gran conocimiento de los datos, pero es probable que un número cada vez mayor tenga acceso a conjuntos de datos similares en el futuro. Los conjuntos de datos son cada vez más amplios con el tiempo, incluso antes de considerar fuentes inherentemente ricas en características, como texto, audio, imágenes o datos genéticos.

En resumen, el desafío de este tipo de conjuntos de datos de alta dimensión tiene mucho que ver con el hecho de que se han recopilado más puntos de datos de los que realmente se necesitan para representar el patrón subyacente.

Los puntos de datos adicionales añaden ruido o variaciones sutiles entre los ejemplos y pueden distraer al algoritmo de aprendizaje de las tendencias importantes. Esto describe la **maldición de la dimensionalidad**, en la que los aprendices fracasan a medida que aumenta el número de características. Si imaginamos cada característica adicional como una nueva dimensión del ejemplo —y aquí, la palabra "dimensión" se usa tanto en sentido literal como metafórico—, a medida que aumentan las dimensiones, aumenta la riqueza de nuestra comprensión de cualquier ejemplo dado, pero también lo hace su singularidad relativa. En un espacio de dimensiones suficientemente altas, cada ejemplo es único, ya que se compone de su propia combinación distintiva de valores de características.

Consideremos una analogía. Una huella dactilar identifica de forma única a las personas, pero no es necesario almacenar todos sus detalles para lograr una coincidencia precisa. De hecho, de los ilimitados detalles encontrados en cada huella, un investigador forense puede usar solo de 12 a 20 puntos distintos para confirmar una coincidencia; incluso los escáneres de huellas dactilares computarizados usan solo de 60 a 80 puntos. Cualquier detalle adicional probablemente sea superfluo y reste calidad a la coincidencia, y llevado al extremo, podría causar coincidencias fallidas, ¡incluso si las huellas dactilares pertenecen a la misma persona!

Por ejemplo, incluir demasiados detalles podría generar falsos negativos, ya que el algoritmo de aprendizaje se distrae con la orientación de la huella o la calidad de la imagen, pero incluir muy pocos detalles puede generar falsos positivos, ya que el algoritmo tiene muy pocas características para distinguir entre candidatos similares. Claramente, es importante encontrar un equilibrio entre demasiado y muy poco detalle. Este es, en esencia, el objetivo de la **reducción de dimensionalidad**, que busca remediar la maldición de la dimensionalidad identificando los detalles importantes.

Figura 1: La reducción de dimensionalidad ayuda a ignorar el ruido y a enfatizar los detalles clave que serán útiles para aprender el patrón subyacente.

A diferencia del problema de los conjuntos de datos muy largos, las soluciones necesarias para aprender de conjuntos de datos amplios son completamente diferentes y son tanto

conceptuales como prácticas. No se pueden descartar columnas aleatoriamente, como sí se hace con las filas, porque algunas columnas son más útiles que otras. En su lugar, se adopta un enfoque sistemático, que a menudo colabora con el propio algoritmo de aprendizaje para encontrar el equilibrio entre el exceso y la falta de detalle. Como aprenderás en las siguientes secciones, algunos de estos métodos se integran en el proceso de aprendizaje, mientras que otros requieren un enfoque más práctico.

Aplicación de la selección de características

En el contexto del aprendizaje automático supervisado, el objetivo de la selección de características es mitigar el problema de la dimensionalidad seleccionando únicamente los predictores más importantes. La selección de características también puede ser beneficiosa, incluso en el caso del aprendizaje no supervisado, debido a su capacidad para simplificar los conjuntos de datos al eliminar información redundante o inútil. Además del objetivo principal de la selección de características, que consiste en facilitar la separación de la señal del ruido por parte de un algoritmo de aprendizaje, esta práctica ofrece beneficios adicionales como:

- Reducir el tamaño del conjunto de datos y los requisitos de almacenamiento.
- Reducir el tiempo y el costo computacional del entrenamiento del modelo.
- Permitir a los científicos de datos centrarse en menos características para la exploración y visualización de datos.

En lugar de intentar encontrar el conjunto completo de predictores 'más óptimo', lo cual puede resultar muy costoso computacionalmente, la selección de características tiende a centrarse en la identificación de características individuales o subconjuntos de características útiles. Para ello, la selección de características suele basarse en heurísticas que reducen el número de subconjuntos buscados. Esto reduce el costo computacional, pero puede llevar a perder la mejor solución posible.

Buscar subconjuntos de características útiles implica asumir que algunos predictores son inútiles, o al menos 'menos' útiles que otros. Sin embargo, a pesar de la validez de esta premisa, no siempre está claro qué hace que algunas características sean útiles y otras no. Por supuesto, puede haber características obviamente irrelevantes que no aporten valor predictivo, pero también puede haber características útiles que sean redundantes y, por lo tanto, innecesarias para el algoritmo de aprendizaje. La clave está en reconocer que algo que parece redundante en un contexto puede ser útil en otro.

La siguiente figura ilustra la capacidad de las características útiles para disfrazarse de predictores aparentemente inútiles y redundantes. El diagrama de dispersión representa una

relación entre dos características hipotéticas, cada una con valores en el rango aproximado de -50 a 50, y que se utilizan para predecir un resultado binario: triángulos frente a círculos.

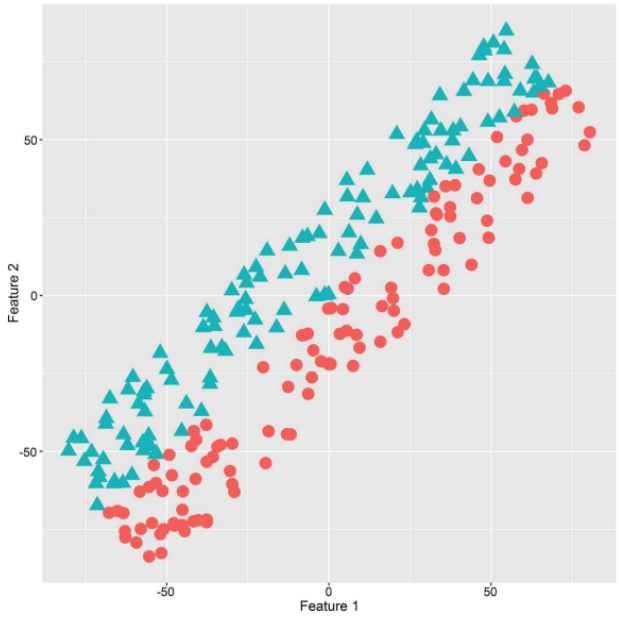

Figura 2: Las características 1 y 2 son aparentemente inútiles y redundantes, pero tienen valor predictivo cuando se utilizan juntas.

Conocer el valor de la característica 1 o 2 por sí solo prácticamente no aporta valor alguno para predecir el resultado del objetivo, ya que los círculos y triángulos se dividen casi por completo para cualquier valor de cualquiera de las características. En términos cuantitativos, esto se demuestra por una correlación muy débil entre las características y el resultado. Un algoritmo simple de selección de características que examine únicamente la relación entre una característica y el resultado podría determinar que ninguna de las dos es útil para la predicción. Además, dado que la correlación entre las dos características es de aproximadamente 0.90, un algoritmo de selección de características más sofisticado que considere simultáneamente el par podría excluir inadvertidamente una de las dos debido a la aparente redundancia.

A pesar de la naturaleza aparentemente inútil y redundante de las dos características, el diagrama de dispersión muestra claramente su capacidad predictiva cuando se utilizan juntas: si la característica 2 es mayor que la característica 1, se predice un triángulo; de lo contrario, se predice un círculo. Un método de selección de características útil debería ser capaz de reconocer este tipo de patrones; de lo contrario, se corre el riesgo de excluir predictores importantes del algoritmo de aprendizaje. Sin embargo, la técnica de selección de características también debe considerar la eficiencia computacional, ya que examinar cada combinación potencial de características es inviable, excepto para los conjuntos de datos más pequeños.

La necesidad de equilibrar la búsqueda de características útiles y no redundantes con la posibilidad de que las características solo sean útiles en combinación con otras es parte de la razón por la que no existe un enfoque único para la selección de características. Dependiendo del caso de uso y del algoritmo de aprendizaje elegido, se pueden aplicar diferentes técnicas que realizan una búsqueda menos rigurosa o más exhaustiva de las características.

Para una mayor profundidad en la selección de características, consultar el artículo: An Introduction to Variable and Feature Selection, 2003, Guyon, I. and Elisseeff, A., Journal of Machine Learning Research, vol. 3, pp. 1157-1182.

Métodos de filtro

Quizás la forma más accesible de selección de características sea la categoría de **métodos de filtro**, que utilizan una función de puntuación relativamente simple para medir la importancia de cada característica. Las puntuaciones resultantes se pueden utilizar para clasificar las características y limitar el número utilizado en el modelo predictivo.

Debido a la simplicidad de este enfoque, los métodos de filtro se utilizan a menudo como primer paso en un proceso iterativo de exploración de datos, ingeniería de características y construcción de modelos. Inicialmente, se podría aplicar un filtro rudimentario para identificar las características candidatas más interesantes para una exploración y visualización exhaustivas, y posteriormente aplicar métodos de selección de características más rigurosos si se desea una mayor reducción.

Una característica distintiva de los métodos de filtro es el uso de una medida proxy de la importancia de la característica. Esta medida es proxy porque sustituye lo que realmente nos importa: la capacidad predictiva de la característica. Sin embargo, no podemos saberlo sin construir primero el modelo predictivo. En su lugar, elegimos una métrica mucho más simple, que esperamos refleje la utilidad de la característica cuando se incorpore posteriormente al modelo.

Por ejemplo, en un modelo de predicción numérica, se podrían calcular correlaciones bivariadas entre cada característica y el objetivo y seleccionar solo las características que estén sustancialmente correlacionadas con este. Para un objetivo binario o categórico, un enfoque comparable podría implicar la construcción de un clasificador univariable, el examen de tablas de contingencia para detectar relaciones bivariadas sólidas entre las características y el objetivo, o el uso de una métrica como la ganancia de información, descrita en el tema “Divide y vencerás: Clasificación mediante árboles de decisión y reglas”.

Una ventaja de este tipo de métricas simples de selección de características es que es improbable que contribuyan al sobreajuste, ya que las medidas proxy utilizan un enfoque diferente y realizan suposiciones sobre los datos distintas a las del algoritmo de aprendizaje.

La mayor ventaja de los métodos de filtro puede ser su escalabilidad incluso para conjuntos de datos con un gran número de características. Esta eficiencia se debe a que el método de filtro solo calcula una puntuación de importancia para cada característica y luego ordena los predictores según estas puntuaciones, de mayor a menor importancia. Por lo tanto, a medida que aumenta el número de características, el gasto computacional crece de forma relativamente lenta y en proporción directa al número de predictores.

Cabe destacar que el resultado de este enfoque es una lista de características ordenadas por rango, en lugar de un único conjunto óptimo de características; por lo tanto, se requiere un juicio subjetivo para determinar el límite óptimo entre características importantes y no importantes.

Aunque los métodos de filtro son computacionalmente eficientes, carecen de la capacidad de considerar grupos de características, lo que significa que los predictores importantes pueden excluirse si solo son útiles en combinación con otros. Además, la improbabilidad de que los métodos de filtro contribuyan al sobreajuste conlleva la posible desventaja de que podrían no generar el conjunto de características más adecuado para el algoritmo de aprendizaje deseado. El método de selección de características descrito en la siguiente sección sacrifica la eficiencia computacional para abordar cada uno de estos problemas.

Métodos de envoltura (*wrapper*) y métodos integrados

A diferencia de los métodos de filtro, que utilizan una medida proxy de la importancia de la variable, los **métodos de envoltura** utilizan el propio algoritmo de aprendizaje automático para identificar la importancia de las variables o subconjuntos de ellas. Los métodos de envoltura se basan en la simple idea de que, a medida que se proporcionan más características importantes al algoritmo, su capacidad para realizar la tarea de aprendizaje debería mejorar. En otras palabras, su tasa de error debería reducirse al incluir predictores importantes o las combinaciones correctas.

Por lo tanto, al construir iterativamente modelos compuestos por diferentes combinaciones de características y examinar cómo cambia su rendimiento, es posible identificar los predictores y conjuntos de predictores importantes. Al probar sistemáticamente todas las combinaciones posibles de características, incluso es posible identificar el mejor conjunto de predictores.

Sin embargo, como cabría esperar, el proceso de probar todas las combinaciones posibles de características es extremadamente ineficiente desde el punto de vista computacional. Para un conjunto de datos con p predictores, hay 2^p conjuntos potenciales de predictores que deben probarse, lo que provoca que el costo computacional de esta técnica crezca con relativa rapidez a medida que se añaden características adicionales. Por ejemplo, un conjunto de datos con solo 10 predictores requeriría la evaluación de $2^{10} = 1{,}024$ modelos diferentes, mientras que un conjunto de datos que añadiera solo cinco predictores más requeriría $2^{15} = 32{,}768$ modelos, ¡lo que supone más de 30 veces el costo computacional!

Claramente, este enfoque no es viable excepto para los conjuntos de datos más pequeños y los algoritmos de aprendizaje automático más sencillos. Una solución a este problema podría ser reducir primero el número de características mediante un método de filtro, pero esto no solo corre el riesgo de perder combinaciones importantes de características, sino que también requeriría una reducción tal en la dimensionalidad que podría anular muchos de los beneficios de los métodos wrapper.

En lugar de permitir que su ineficiencia nos impida aprovechar sus ventajas, podemos usar heurísticas para evitar buscar cada combinación de características. En particular, el enfoque "codicioso" descrito en el tema, "Divide y vencerás: Clasificación mediante árboles de decisión y reglas", que ayudó a desarrollar árboles de decisión eficientemente, también se puede utilizar aquí.

Recordarás que la idea de un algoritmo codicioso (*greedy*) es utilizar los datos por orden de llegada, utilizando primero las características más predictivas. Aunque esta técnica no garantiza encontrar la solución óptima, reduce drásticamente el número de combinaciones que deben probarse.

Existen dos enfoques básicos para adaptar los métodos wrapper para la selección codiciosa de características. Ambos implican probar el algoritmo de aprendizaje modificando una variable a la vez. La técnica de **selección progresiva** (*forward selection*) comienza introduciendo cada característica en el modelo una por una, para determinar cuál de ellas resulta en el mejor modelo de un solo predictor. La siguiente iteración de selección hacia adelante conserva el primer mejor predictor del modelo y prueba las características restantes para identificar cuál constituye el mejor modelo de dos predictores. Como era de esperar, este proceso puede continuar seleccionando el mejor modelo de tres predictores, el modelo de cuatro predictores, y así sucesivamente, hasta que se hayan seleccionado todas las características.

Sin embargo, dado que el objetivo de la selección de características no es seleccionar el conjunto completo, el proceso de selección hacia adelante se detiene prematuramente cuando

añadir características adicionales ya no mejora el rendimiento del modelo más allá de un umbral específico.

La técnica similar de **eliminación hacia atrás** (*backward elimination*) funciona de la misma manera, pero a la inversa. Partiendo de un modelo que contiene todas las características, el modelo itera repetidamente, eliminando cada vez la característica menos predictiva, hasta que se detiene cuando la eliminación de una característica disminuye el rendimiento del modelo más allá del umbral deseado.

Los algoritmos de aprendizaje conocidos como **métodos integrados** (*embedded methods*) tienen una especie de envoltorios integrados muy similares a la selección hacia adelante. Estos métodos seleccionan las mejores características automáticamente durante el proceso de entrenamiento del modelo. Ya conoces uno de estos métodos, los árboles de decisión, que utiliza la selección hacia adelante voraz para determinar el mejor subconjunto de características. La mayoría de las técnicas de aprendizaje automático no incorporan la selección de características; las dimensiones deben reducirse previamente. La siguiente sección demuestra cómo se pueden aplicar estos métodos en R mediante una variante del algoritmo de aprendizaje automático presentado en el tema, Pronóstico de Datos Numéricos: Métodos de Regresión.

Ejemplo - Uso de la regresión por pasos para la selección de características

Una implementación ampliamente conocida de los métodos de envoltura es **la regresión por pasos** (*stepwise regression*), que utiliza la selección progresiva o regresiva para identificar un conjunto de características para un modelo de regresión. Para demostrar esta técnica, revisaremos el conjunto de datos de pasajeros del Titanic utilizado en temas anteriores y construiremos un modelo de regresión logística que predice si cada pasajero sobrevivió al desafortunado viaje. Para comenzar, utilizaremos tidyverse para leer los datos y aplicar algunos pasos sencillos de preparación de datos.

La siguiente secuencia de comandos crea un indicador de valor faltante para la edad, imputa la edad promedio para los valores faltantes de edad, imputa X para los valores faltantes de cabina y embarcado, y convierte el sexo en un factor:

```
> titanic_train <- read_csv("titanic_train.csv") |>
+   mutate(
+     Age_MVI = if_else(is.na(Age), 1, 0),
+     Age = if_else(is.na(Age), mean(Age, na.rm = TRUE), Age),
+     Cabin = if_else(is.na(Cabin), "X", Cabin),
+     Embarked = factor(if_else(is.na(Embarked), "X", Embarked)),
+     Sex = factor(Sex)
```

+)

```
Rows: 891 Columns: 12
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (5): Name, Sex, Ticket, Cabin, Embarked
dbl (7): PassengerId, Survived, Pclass, Age, SibSp, Parch, Fare

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

El proceso paso a paso necesita conocer las condiciones iniciales y finales para la selección de características, o el conjunto mínimo y máximo de variables que se pueden incluir. En nuestro caso, definiremos el modelo más simple posible como uno sin variables: un modelo con solo un término de intersección constante.

Para definir este modelo en R, usaremos la función glm() para modelar la supervivencia en función de una intersección constante mediante la fórmula Survived ~ 1. Al establecer el parámetro family en binomial, se define un modelo de regresión logística:

```
> simple_model <- glm(Survived ~ 1, family = binomial,
+   data = titanic_train)
```

El modelo completo sigue utilizando regresión logística, pero incluye muchos más predictores:

```
> full_model <- glm(Survived ~ Age + Age_MVI + Embarked +
+   Sex + Pclass + SibSp + Fare,
+   family = binomial, data = titanic_train)
```

La selección hacia adelante comenzará con el modelo simple y determinará qué características del modelo completo vale la pena incluir en el modelo final. La función step() del paquete básico stats de R proporciona esta funcionalidad; sin embargo, dado que otros paquetes también tienen funciones step(), especificar stats::step() garantiza que se utilice la función correcta.

El primer argumento de la función proporciona el modelo inicial, el parámetro scope requiere la función formula() del modelo completo y la dirección se establece en regresión gradual hacia adelante:

```
> sw_forward <- stats::step(simple_model,
+   scope = formula(full_model),
+   direction = "forward")
```

Este comando genera un conjunto de resultados para cada iteración del proceso gradual, pero solo se incluyen la primera y la última iteración para abreviar.

Si seleccionas entre un gran número de variables, establece trace = 0 en la función step() para desactivar la salida en cada iteración.

Al inicio del proceso gradual, se parte del modelo simple utilizando la fórmula Survived ~ 1, que modela la supervivencia utilizando únicamente un término de intersección constante. El primer bloque de resultados muestra la calidad del modelo al inicio y después de evaluar otros siete modelos candidatos, cada uno con un predictor adicional. La fila etiquetada como <none> se refiere a la calidad del modelo al inicio de esta iteración y a su clasificación en comparación con los otros siete candidatos:

```
Start:  AIC=1188.66
Survived ~ 1

           Df Deviance    AIC
+ Sex       1    917.8  921.8
+ Pclass    1   1084.4 1088.4
+ Fare      1   1117.6 1121.6
+ Embarked  3   1157.0 1165.0
+ Age_MVI   1   1178.9 1182.9
+ Age       1   1182.3 1186.3
<none>          1186.7 1188.7
+ SibSp     1   1185.5 1189.5
```

La medida de calidad utilizada, AIC, mide la calidad relativa de un modelo en comparación con otros modelos. En particular, se refiere al **criterio de información de Akaike**. Si bien una definición formal del AIC queda fuera del alcance de este documento, la medida busca equilibrar la complejidad y el ajuste del modelo. Los valores de AIC más bajos son mejores. Por lo tanto, el modelo que incluye Sexo es el mejor de los otros seis modelos candidatos, además del modelo original. En la iteración final, el modelo base utiliza Sexo, Pclass, Edad y SibSp, y ninguna característica adicional reduce aún más el AIC. La fila <none> se clasifica por encima de los modelos candidatos que añaden las características Embarked, Fare y Age_MVI:

```
Step:  AIC=800.84
Survived ~ Sex + Pclass + Age + SibSp

           Df Deviance    AIC
<none>          790.84 800.84
+ Embarked  3   785.27 801.27
+ Fare      1   789.65 801.65
+ Age_MVI   1   790.59 802.59
```

En este punto, el proceso de selección hacia adelante se detiene. Podemos obtener la fórmula para el modelo final:

```
> formula(sw_forward)
Survived ~ Sex + Pclass + Age + SibSp
```

También podemos obtener los coeficientes de regresión estimados del modelo final:

```
> sw_forward$coefficients
(Intercept)        Sexmale        Pclass           Age         SibSp
 5.19197585   -2.73980616   -1.17239094   -0.03979317   -0.35778841
```

La eliminación hacia atrás es aún más sencilla de ejecutar. Al proporcionar un modelo con el conjunto completo de características para probar y establecer direction = "backward", el modelo iterará y eliminará sistemáticamente cualquier característica que resulte en un mejor AIC. Por ejemplo, el primer paso comienza con un conjunto completo de predictores, pero la eliminación de las características Fare, Age_MVI o Embarked resulta en un AIC más bajo:

```
> sw_backward <- stats::step(full_model, direction = "backward")
```

```
Start:  AIC=803.49
Survived ~ Age + Age_MVI + Embarked + Sex + Pclass + SibSp +
    Fare

           Df Deviance     AIC
- Fare      1   783.88  801.88
- Age_MVI   1   784.81  802.81
- Embarked  3   789.42  803.42
<none>          783.49  803.49
- SibSp     1   796.34  814.34
- Age       1   810.97  828.97
- Pclass    1   844.74  862.74
- Sex       1  1016.36 1034.36
```

En cada iteración, se elimina la peor característica, pero en el paso final, la eliminación de cualquiera de las características restantes resulta en un AIC más alto y, por lo tanto, en un modelo de menor calidad que el modelo base. Por lo tanto, el proceso se detiene aquí:

```
Step:  AIC=800.84
Survived ~ Age + Sex + Pclass + SibSp

         Df Deviance     AIC
<none>        790.84  800.84
- SibSp   1   805.33  813.33
- Age     1   819.32  827.32
- Pclass  1   901.80  909.80
- Sex     1  1044.10 1052.10
```

En este caso, la selección hacia adelante y la eliminación hacia atrás resultaron en el mismo conjunto de predictores, pero esto no siempre es así. Pueden surgir diferencias si ciertas características funcionan mejor en grupos o si están interrelacionadas de alguna otra manera.

Como se mencionó anteriormente, una de las desventajas de las heurísticas utilizadas por los métodos wrapper es que no garantizan encontrar el conjunto de predictores 'más óptimo'; sin embargo, esta deficiencia es precisamente lo que hace que el proceso de selección de características sea computacionalmente viable.

Ejemplo - Uso de Boruta para la selección de características

Para un método de selección de características más robusto, pero con un uso computacional mucho más intensivo, el paquete Boruta implementa un envoltorio alrededor del algoritmo de bosque aleatorio. Por ahora, basta con saber que los bosques aleatorios son una variante de los árboles de decisión, que proporcionan una medida de la importancia de las variables. Al probar sistemáticamente subconjuntos aleatorios de variables repetidamente, es posible determinar si una característica es significativamente más o menos importante que otras mediante técnicas estadísticas de prueba de hipótesis.

Dada su gran dependencia de la técnica del bosque aleatorio, no sorprende que la técnica comparta su nombre con Boruta, una criatura mitológica eslava que se cree que habita en pantanos y bosques.
Para obtener más información sobre los detalles de la implementación de Boruta, consulta Feature Selection with the Boruta Package. Kursa, M. B. y Rudnicki, W. R., 2010, Journal of Statistical Software, Vol. 36, Núm. 11.

La técnica Boruta emplea un ingenioso truco que utiliza las llamadas "características de sombra" para determinar la importancia de una variable. Estas características de sombra son copias de las características originales del conjunto de datos, pero con valores aleatorios que rompen cualquier asociación entre la característica y el resultado objetivo. Por lo tanto, estas características de sombra son, por definición, irrelevantes y sin importancia, y no deberían aportar ningún beneficio predictivo al modelo, salvo por casualidad. Sirven como base para evaluar las demás características.

Tras ejecutar las características originales y las características de sombra mediante el proceso de modelado de bosque aleatorio, se compara la importancia de cada característica original con la característica de sombra más importante. Las características significativamente mejores que la característica de sombra se consideran importantes; las significativamente peores se consideran irrelevantes y se eliminan permanentemente. El algoritmo itera repetidamente hasta que todas las características se consideran importantes o irrelevantes, o hasta que el proceso alcanza un límite predeterminado de iteraciones.

Para ver esto en acción, apliquemos el algoritmo Boruta al mismo conjunto de datos de entrenamiento del Titanic, usado en la sección anterior. Para demostrar que el algoritmo puede detectar características realmente inútiles, a modo de demostración, podemos añadir

una al conjunto de datos. Primero, estableceremos la semilla aleatoria en el número arbitrario 12345 para garantizar que los resultados coincidan con los que se muestran aquí. Después, asignaremos a cada uno de los 891 ejemplos de entrenamiento un valor aleatorio entre 1 y 100.

Dado que los números son completamente aleatorios, esta característica debería resultar inútil casi con toda seguridad, salvo por pura suerte:

```
> set.seed(12345)
> titanic_train$rand_vals <- runif(n = 891, min = 1, max = 100)
```

A continuación, cargaremos el paquete Boruta y lo aplicaremos al conjunto de datos Titanic. La sintaxis es similar a la del entrenamiento de un modelo de aprendizaje automático; aquí, especificamos el modelo mediante la interfaz de fórmula para listar el objetivo y los predictores:

```
> library(Boruta)
> titanic_boruta <- Boruta(Survived ~ PassengerId + Age +
+   Sex + Pclass + SibSp + rand_vals,
+   data = titanic_train, doTrace = 1)
```

El parámetro doTrace se establece en 1 para solicitar una salida detallada, que produce una actualización de estado a medida que el algoritmo alcanza puntos clave en el proceso de iteración. Aquí vemos el resultado tras 10 iteraciones, que muestra que, como era de esperar, la función rand_vals se ha rechazado por considerarse irrelevante, mientras que cuatro se han confirmado como importantes y una permanece indeterminada:

```
After 10 iterations, +4.9 secs:
 confirmed 4 attributes: Age, Pclass, Sex, SibSp;
 rejected 1 attribute: rand_vals;
 still have 1 attribute left.
```

Una vez completado el algoritmo, escriba el nombre del objeto para ver los resultados:

```
> titanic_boruta
```

```
Boruta performed 99 iterations in 22.38234 secs.
 4 attributes confirmed important: Age, Pclass, Sex, SibSp;
 1 attributes confirmed unimportant: rand_vals;
 1 tentative attributes left: PassengerId;
```

La función Boruta() tiene un límite de 100 ejecuciones por defecto, que alcanzó tras 99 iteraciones en aproximadamente 22.38 segundos. Antes de detenerse, cuatro funciones se

consideraron importantes y una irrelevante. La función PassengerId, que figura como provisional, no se pudo confirmar como importante o irrelevante. Establecer el parámetro maxRuns en un valor superior a 100 puede ayudar a llegar a una conclusión; en este caso, establecer maxRuns = 500 confirmará que PassengerId no es importante después de 486 iteraciones.

También es posible graficar la importancia relativa de las características:

```
> plot(titanic_boruta)
```

La visualización resultante se muestra en la figura 3. Para cada una de las seis características, así como para las características de sombra de rendimiento máximo, medio (promedio) y mínimo, un diagrama de caja muestra la distribución de las métricas de importancia para cada característica. Con estos resultados, podemos confirmar que PassengerId es ligeramente menos importante que la característica de sombra máxima, y que rand_vals es aún menos importante:

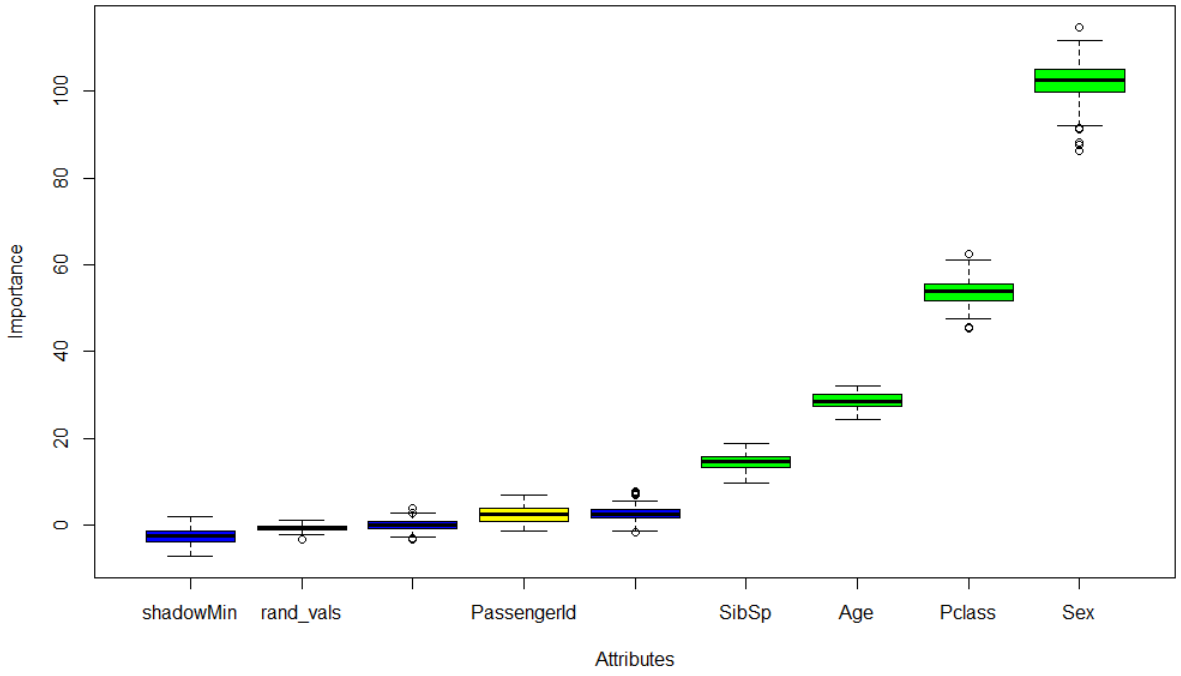

Figura 13.3: El gráfico de la salida de Boruta muestra la importancia relativa de las características en comparación entre sí y con las características de sombra (después de rand_vals estaría shadowMean y después de PassengerId iría shadowMax).

Basándonos en la exploración del conjunto de datos del Titanic (que podrías realizar como ejercicio), la alta importancia de las características Sex y Pclass no es sorprendente. Asimismo, no esperaríamos que PassengerId fuera importante, a menos que los

identificadores estuvieran vinculados de alguna manera con la supervivencia del Titanic en lugar de asignarse aleatoriamente. Dicho esto, aunque los resultados de este proceso de selección de características no revelaron nuevos conocimientos, la técnica sería mucho más útil para conjuntos de datos que no son tan fáciles de explorar manualmente o donde se desconoce el significado real de las características.

Por supuesto, este es solo un enfoque para abordar un gran número de características de importancia indeterminada; la siguiente sección describe una alternativa que podría tener un mejor rendimiento, especialmente si muchas de las características están correlacionadas.

La técnica Boruta puede requerir un gran esfuerzo computacional y, en conjuntos de datos reales, generalmente tardará minutos o incluso horas en completarse, en lugar de segundos, como con los datos del Titanic.

Los autores del paquete estiman que, en una computadora moderna, se necesita aproximadamente una hora por cada millón de combinaciones de características y ejemplos. Por ejemplo, un conjunto de datos con 10,000 filas y 50 características tardará aproximadamente media hora en completarse. Aumentar el tamaño de este conjunto de datos a 100,000 filas requeriría unas cinco horas de procesamiento.

Extracción de características

La selección de características no es el único enfoque disponible para reducir la dimensionalidad de un conjunto de datos altamente dimensional. Otra posibilidad es sintetizar un número menor de predictores compuestos.

Este es el objetivo de la **extracción de características**, una técnica de reducción de la dimensionalidad que crea nuevas características en lugar de seleccionar un subconjunto de las existentes. Las características extraídas se construyen de forma que reduzcan la cantidad de información redundante, conservando al mismo tiempo la mayor cantidad de información útil posible. Por supuesto, encontrar el equilibrio ideal entre demasiada y muy poca información es un desafío en sí mismo.

Comprendiendo el análisis de componentes principales

Para comenzar a comprender la extracción de características, imagina un conjunto de datos con una gran cantidad de características. Por ejemplo, para predecir la probabilidad de impago de un préstamo, un conjunto de datos puede incluir cientos de atributos del solicitante. Obviamente, algunas características predecirán el resultado objetivo, pero es probable que muchas también se predigan entre sí. Por ejemplo, la edad, el nivel educativo, los ingresos, el código postal y la ocupación de una persona predicen su probabilidad de pagar un préstamo, pero también se predicen entre sí en distintos grados.

Su interrelación sugiere que existe cierto grado de superposición o dependencia conjunta entre ellos, lo cual se refleja en su covarianza y correlación.

Es posible que la razón por la que estos cinco atributos de los solicitantes de préstamos estén relacionados sea que son componentes de un número menor de atributos que son los verdaderos impulsores subyacentes del comportamiento de pago de préstamos. En particular, podríamos creer que la probabilidad de pago de un préstamo se basa en la responsabilidad y la riqueza del solicitante, pero debido a que estos conceptos son difíciles de medir directamente, utilizamos múltiples medidas proxy fácilmente disponibles.

La siguiente figura ilustra cómo cada una de las cinco características podría capturar aspectos de las dos dimensiones ocultas de interés. Cabe destacar que ninguna de las características captura completamente ninguna de las dimensiones del componente, sino que cada dimensión del componente es una combinación de varias características.

Por ejemplo, el nivel de responsabilidad de una persona podría reflejarse en su edad y nivel educativo, mientras que su nivel económico podría reflejarse en sus ingresos, ocupación y código postal.

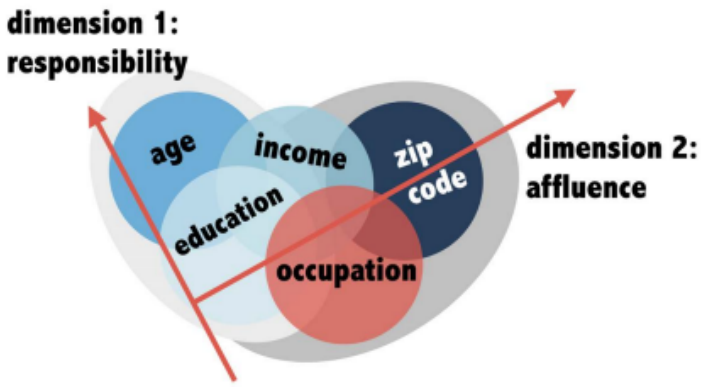

Figura 4: Cinco atributos hipotéticos de los solicitantes de préstamos podrían expresarse de forma más sencilla en dos dimensiones creadas a partir de la covarianza de cada atributo.

El objetivo del **análisis de componentes principales** (PCA, *Principal component analysis*) es extraer un número menor de dimensiones subyacentes de un mayor número de características, expresando la covarianza de múltiples atributos correlacionados como un único vector.

En pocas palabras, la covarianza se refiere al grado en que los atributos varían en conjunto. Cuando uno aumenta o disminuye, el otro tiende a aumentar o disminuir. Los vectores resultantes se conocen como **componentes principales** y se construyen como combinaciones ponderadas de los atributos originales.

Al aplicarlo a un conjunto de datos con muchas características correlacionadas, un número mucho menor de componentes principales puede ser capaz de expresar gran parte de la varianza total del conjunto de datos de mayor dimensión. Aunque esto parezca mucha jerga técnica, y los cálculos necesarios para implementar el PCA exceden el alcance de este documento, trabajaremos para comprender conceptualmente el proceso.

El análisis de componentes principales está estrechamente relacionado con otra técnica, **el análisis factorial**, que es un enfoque más formal para explorar las relaciones entre factores observados y no observados (latentes), como los que se muestran en las figuras.

En la práctica, ambos pueden aplicarse de forma similar, pero el PCA es más sencillo y evita la construcción de un modelo formal; simplemente reduce el número de dimensiones, manteniendo la variación máxima.

Para profundizar en las numerosas distinciones sutiles, consulta el siguiente hilo de Stack Exchange:
https://stats.stackexchange.com/questions/1576/what-are-the-differences-between-factor-analysis-and-principal-component-analysi/.

Revisando la figura 13.4, cada círculo representa las relaciones entre cada una de las cinco características. Los círculos con mayor superposición representan características correlacionadas que pueden medir un concepto subyacente similar. Ten en cuenta que esta es una representación muy simplificada que no representa los puntos de datos individuales que se utilizarían para calcular las correlaciones entre las características.

En realidad, estos puntos de datos individuales representarían a los solicitantes de préstamos individuales y se ubicarían en un espacio de cinco dimensiones con coordenadas determinadas por los cinco valores de las características de cada solicitante.

Por supuesto, esto es difícil de representar en las dos dimensiones de las páginas de este documento impreso, por lo que los círculos en esta representación simplificada deben entenderse como una masa de personas con valores altos del atributo, similar a una nube. En este caso, si dos características están altamente correlacionadas, como los ingresos y la educación, las dos nubes se superpondrán, ya que las personas con valores altos de un atributo tenderán a tener valores altos del otro. La figura 5 representa esta relación:

Al examinar la figura 5, observa que la flecha diagonal que representa la relación entre los ingresos y la educación refleja la covarianza entre las dos características. Saber si un punto está más cerca del inicio o del final de la flecha proporcionaría una buena estimación tanto de los ingresos como de la educación. Por lo tanto, es probable que las características altamente covariantes expresen atributos subyacentes similares y, por lo tanto, puedan ser redundantes. De esta manera, la información expresada por dos dimensiones, ingresos y

educación, podría expresarse de forma más sencilla en una sola dimensión, que sería el componente principal de estas dos características.

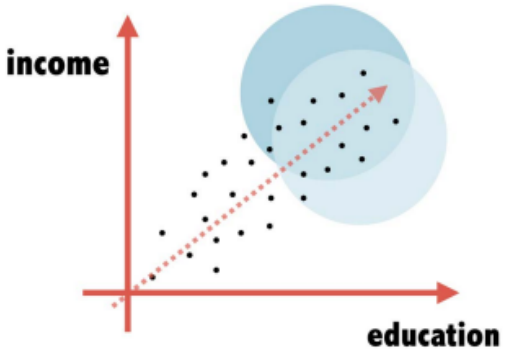

Figura 5: Cuando dos características están altamente correlacionadas, los puntos con valores altos de una tienden a tener valores altos de la otra.

Aplicando esta relación a un diagrama tridimensional, podríamos imaginar este componente principal como la dimensión z en la siguiente figura:

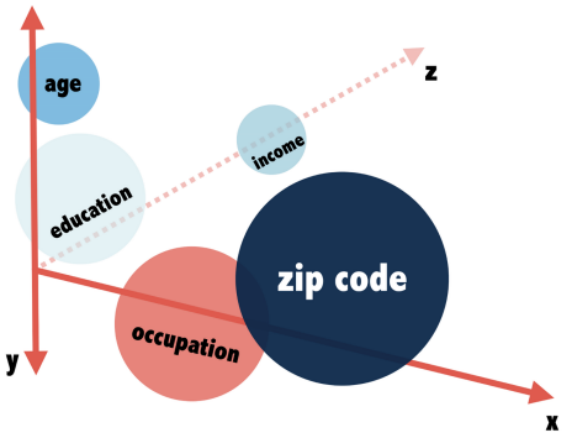

Figura 6: Cinco atributos con distintos grados de covarianza en tres dimensiones.

Al igual que en el caso bidimensional, la posición de los círculos representa la covarianza entre las características; el tamaño de los círculos representa la profundidad, con círculos más grandes o más pequeños más cerca del frente o del fondo del espacio. En las tres dimensiones, la edad y la educación son similares en una dimensión, la ocupación y el código postal son similares en otra, y los ingresos varían en una tercera dimensión.

Si esperáramos capturar la mayor parte de la varianza y al mismo tiempo reducir el número de dimensiones de tres a dos, podríamos proyectar este gráfico tridimensional en un gráfico bidimensional de la siguiente manera:

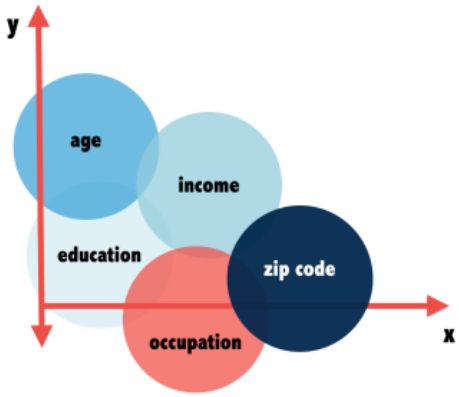

Figura 7: El análisis de componentes principales reduce muchas dimensiones a un número menor de componentes clave.

Con estas dos dimensiones, hemos construido los dos componentes principales del conjunto de datos y, al hacerlo, hemos reducido su dimensionalidad de cinco dimensiones con significado real a dos dimensiones, *x* e *y*, sin conexión inherente con el mundo real. En cambio, las dos dimensiones resultantes reflejan ahora combinaciones lineales de los puntos de datos subyacentes; son resúmenes útiles de los datos subyacentes, pero no son fácilmente interpretables.

Podríamos reducir la dimensionalidad aún más proyectando el conjunto de datos sobre una línea para crear un único componente principal, como se ilustra en la figura 8:

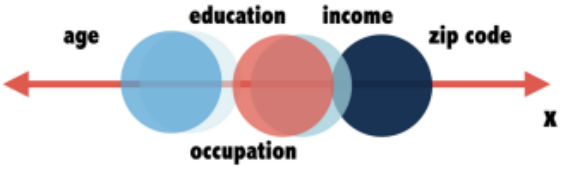

Figura 8: El primer componente principal captura la dimensión con la mayor varianza.

En este ejemplo, el enfoque PCA extrajo una característica híbrida de las cinco dimensiones originales del conjunto de datos. La edad y la educación se consideran algo redundantes, al igual que la ocupación y los ingresos.

Además, la edad y la educación tienen un impacto opuesto en la nueva característica que el código postal: están desplazando el valor de *x* en direcciones opuestas. Si esta representación unidimensional pierde demasiada información almacenada en las cinco características originales, se podrían utilizar los enfoques anteriores con dos o tres componentes. Como ocurre con muchas técnicas de aprendizaje automático, existe un equilibrio entre el sobreajuste y el subajuste de los datos. Veremos esto reflejado en un ejemplo real en breve.

Antes de aplicar el PCA, es importante saber que los componentes principales se identifican mediante un algoritmo determinista, lo que significa que la solución es consistente cada vez que se completa el proceso en un conjunto de datos determinado.

Cada vector componente es siempre ortogonal, o perpendicular, a todos los vectores componentes anteriores. El primer componente principal captura la dimensión de mayor varianza, el siguiente captura la siguiente mayor, y así sucesivamente, hasta que se haya construido un componente principal para cada uno en el conjunto de datos original, o el algoritmo se detenga prematuramente cuando se alcance el número deseado de componentes.

Ejemplo - Uso de PCA para reducir datos de redes sociales altamente dimensionales

Como se mencionó anteriormente, PCA es una técnica de extracción de características que reduce la dimensionalidad de un conjunto de datos sintetizando un conjunto más pequeño de características del conjunto completo. Aplicaremos esta técnica a los datos de redes sociales, este conjunto de datos incluye 36 palabras diferentes que aparecieron en las redes sociales de 30,000 adolescentes en Estados Unidos. Las palabras reflejan diversos intereses y actividades, como deportes, música, religión y compras. Si bien 36 no es una cantidad descabellada para la mayoría de los algoritmos de aprendizaje automático, si tuviéramos más, quizás cientos de características, algunos algoritmos podrían comenzar a tener dificultades con la dimensionalidad.

Utilizaremos el conjunto de funciones tidyverse para leer y preparar los datos. Primero, cargaremos el paquete y usaremos su función read_csv() para leer los datos de redes sociales como un tibble:

```
> library(tidyverse)
> sns_data <- read_csv("snsdata.csv")
```

A continuación, usaremos select() solo en las columnas correspondientes a las características que registran el número de veces que se usaron 36 palabras en cada perfil de redes sociales. Esta notación selecciona desde la columna basketball hasta la columna drugs y se guarda el resultado en un nuevo tibble llamado sns_terms:

```
> sns_terms <- sns_data |> select(basketball:drugs)
```

La técnica de PCA solo funciona con una matriz de datos numéricos. Sin embargo, dado que cada una de las 36 columnas resultantes es un recuento, no se requiere más preparación de datos. Si el conjunto de datos incluyera características categóricas, sería necesario convertirlas a numéricas antes de continuar.

La base de R incluye una función de PCA integrada llamada prcomp(), cuya ejecución se ralentiza a medida que los conjuntos de datos aumentan de tamaño. Usaremos una función de sustitución directa del paquete irlba de Bryan W. Lewis, que puede detenerse antes de tiempo para devolver solo un subconjunto del conjunto completo de posibles componentes principales.

Este enfoque truncado, sumado al uso de un algoritmo generalmente más eficiente, hace que la función irlba_prcomp() sea mucho más rápida que prcomp() en conjuntos de datos más grandes, manteniendo la sintaxis y la compatibilidad prácticamente idénticas a las de la función base, en caso de que esté siguiendo tutoriales en línea antiguos.

El paquete irlba recibe su nombre, aunque parezca extraño, de la técnica que utiliza: el "algoritmo de bidiagonalización de Lanczos reiniciado implícitamente", desarrollado por Jim Baglama y Lothar Reichel. Para más información sobre este enfoque, consulta el paquete vignette con el comando vignette("irlba").

Antes de comenzar, estableceremos la semilla aleatoria en un valor arbitrario de 2023 para garantizar que los resultados coincidan con el documento. Luego, tras cargar el paquete requerido, canalizaremos el conjunto de datos sns_terms a la función PCA. Los tres parámetros nos permiten limitar el resultado a los primeros 10 componentes principales, a la vez que estandarizamos los datos centrando cada característica alrededor de cero y escalándolas para que tengan una varianza de uno.

Esto suele ser deseable por la misma razón que en el enfoque de k-Vecinos Más Próximos: evita que las características con mayor varianza dominen los componentes principales. Los resultados se guardan como un objeto llamado sns_pca:

```
> set.seed(2023)
> library(irlba)
> sns_pca <- sns_terms |>
+   prcomp_irlba(n = 10, center = TRUE, scale = TRUE)
```

Aunque PCA es un algoritmo determinista, el signo (positivo o negativo) es arbitrario y puede variar entre ejecuciones, de ahí la necesidad de establecer la semilla aleatoria de antemano para garantizar la reproducibilidad.
Este hilo de Stack Exchange contiene más información sobre este fenómeno: https://stats.stackexchange.com/questions/88880/

Recuerda que cada componente del PCA captura una cantidad decreciente de la varianza del conjunto de datos y que solicitamos 10 de los 36 componentes posibles. Un gráfico scree, llamado así por los patrones de deslizamientos de tierra que se forman en el fondo de los acantilados, ayuda a visualizar la cantidad de varianza capturada por cada componente y, por

lo tanto, puede ayudar a determinar el número óptimo de componentes a utilizar. La función screeplot() de R se puede aplicar a nuestro resultado para crear dicho gráfico. Los cuatro parámetros que proporcionan nuestro resultado de PCA indican que queremos representar gráficamente los 10 componentes, usar un gráfico de líneas en lugar de un gráfico de barras y aplicar el título del gráfico:

```
> screeplot(sns_pca, npcs = 10, type = "lines",
+   main = "Scree Plot of SNS Data Principal Components")
```

El gráfico resultante se ve así:

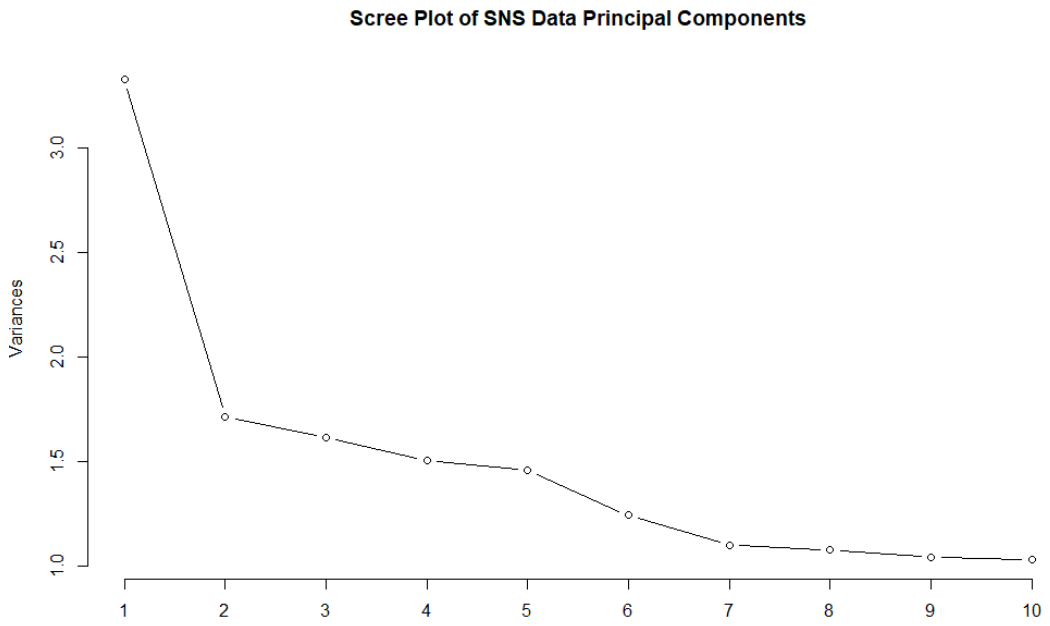

Figura 9: El gráfico scree que representa la varianza de los primeros 10 componentes principales del conjunto de datos de redes sociales.

El gráfico scree muestra una disminución sustancial en la varianza capturada entre el primer y el segundo componente. Los componentes del segundo al quinto capturan aproximadamente la misma cantidad de varianza, y posteriormente se observan descensos sustanciales adicionales entre los componentes quinto y sexto, y entre los componentes sexto y séptimo. Los componentes del séptimo al décimo capturan aproximadamente la misma cantidad de varianza. Con base en este resultado, podríamos decidir utilizar uno, cinco o seis componentes principales como nuestro conjunto de datos de dimensionalidad reducida. Podemos visualizarlo numéricamente aplicando la función summary() a nuestro objeto de resultados de PCA:

```
> summary(sns_pca)
```

```
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10
Standard deviation     1.82375 1.30885 1.27008 1.22642 1.20854 1.11506 1.04948 1.03828 1.02163 1.01638
Proportion of Variance 0.09239 0.04759 0.04481 0.04178 0.04057 0.03454 0.03059 0.02995 0.02899 0.02869
Cumulative Proportion  0.09239 0.13998 0.18478 0.22657 0.26714 0.30167 0.33227 0.36221 0.39121 0.41990
```

El resultado muestra la desviación estándar, la proporción de la varianza total y la proporción acumulada de la varianza para cada uno de los 10 componentes (etiquetados del PC1 al PC10). Dado que la desviación estándar es la raíz cuadrada de la varianza, al elevar al cuadrado las desviaciones estándar se obtienen los valores de varianza representados en el diagrama de sedimentación; por ejemplo, $1.82375^2 = 3.326064$, que es el valor mostrado para el primer componente en el diagrama de sedimentación.
La proporción de varianza de un componente es su varianza respecto al total de todos los componentes: no solo los 10 que se muestran aquí, sino también los 26 restantes que podríamos haber creado. Por lo tanto, la proporción acumulada de varianza alcanza un máximo del 41.99 %, en lugar del 100 %, que se explicaría por los 36 componentes.

El uso del PCA como técnica de reducción de dimensionalidad requiere que el usuario determine cuántos componentes conservar. En este caso, si elegimos cinco componentes, capturaremos el 26.7 % de la varianza, o una cuarta parte de la información total de los datos originales. Que esto sea suficiente depende de qué parte del 73.3 % restante de la varianza sea señal o ruido, algo que solo podemos determinar mediante la creación de un algoritmo de aprendizaje útil.

Un aspecto que facilita este proceso es que, independientemente de la cantidad de componentes que elijamos, nuestro proceso de PCA está completo; podemos usar tantos o tan pocos de los 10 componentes como deseemos. Por ejemplo, no es necesario volver a ejecutar el algoritmo para obtener los tres mejores componentes en lugar de los siete mejores; al encontrar los primeros siete componentes, se incluirán naturalmente los tres mejores y los resultados serán idénticos. En una aplicación práctica de PCA, puede ser conveniente probar varios puntos de corte diferentes.

Para simplificar, reduciremos el conjunto de datos original de 36 dimensiones a cinco componentes principales. Por defecto, la función irlba_prcomp() guarda automáticamente una versión del conjunto de datos original transformada al espacio de menor dimensión. Esta se encuentra en el objeto de lista sns_pca resultante, llamado x, que podemos examinar con el comando str():

```
> str(sns_pca$x)
```

```
num [1:30000, 1:10] -1.448 3.492 -0.646 -1.041 4.322 ...
- attr(*, "dimnames")=List of 2
 ..$ : NULL
 ..$ : chr [1:10] "PC1" "PC2" "PC3" "PC4" ...
```

El conjunto de datos transformado es una matriz numérica con 30,000 filas, al igual que el conjunto de datos original, pero con 10 columnas en lugar de 36, cuyos nombres van del PC1 al PC10. Podemos observar esto con mayor claridad utilizando la salida del comando head() para ver las primeras filas:

```
> head(sns_pca$x)
```

```
            PC1         PC2       PC3        PC4         PC5         PC6          PC7          PC8         PC9        PC10
[1,] -1.4477620  0.07976310 0.3357330 -0.3636082  0.03833596 -0.01559079 -0.007278589 -0.004582346  0.19226144 -0.08086065
[2,]  3.4922144  0.36554520 0.7966735 -0.1871626  0.57126163  3.02758235  0.306304037 -1.142422251  0.72992534 -0.11203923
[3,] -0.6459385 -0.67798166 0.8000251  0.6243070  0.25122261 -0.40751994 -0.454614417  0.704544996 -0.43734980  0.07735574
[4,] -1.0405145  0.08118501 0.4099638 -0.2555128 -0.02620989  0.27837411 -0.462898314 -0.175251793 -0.08843005 -0.26784326
[5,]  4.3216304 -1.01754361 3.4112730 -1.9209916 -0.43409869 -1.11734548  2.122420077 -2.287638056  2.19992650  0.26536161
[6,]  0.2131225 -0.65882053 1.6215828  0.9372545  1.47217369  0.04614790  0.654207687  0.285263646  0.69439745  0.89649127
```

Recuerda que en el conjunto de datos original, cada una de las 36 columnas indicaba el número de veces que aparecía una palabra en el texto del perfil de redes sociales. Si estandarizamos los datos para obtener una media de cero y como se ha hecho aquí para los componentes principales, los valores positivos y negativos indican perfiles con valores superiores o inferiores a la media, respectivamente.

La clave reside en que cada una de las 36 columnas originales tenía una interpretación obvia, mientras que los resultados del PCA carecen de significado aparente.

Podemos intentar comprender los componentes visualizando las cargas (**loadings**) del PCA, o los pesos que transforman los datos originales en cada uno de los componentes principales. Las cargas altas son más importantes para un componente en particular. Estas cargas se encuentran en el objeto de lista sns_pca llamado rotation.

Esta es una matriz numérica con 36 filas que corresponden a cada una de las columnas originales del conjunto de datos y 10 columnas que proporcionan las cargas de los componentes principales. Para construir nuestra visualización, necesitaremos pivotar estos datos de forma que tengan una fila por término de redes sociales por componente principal; es decir, tendremos 36 * 10 = 360 filas en la versión más larga del conjunto de datos.

El siguiente comando utiliza dos pasos para crear el conjunto de datos largo requerido. El primer paso crea un tibble que incluye una columna SNS_Term con una fila para cada uno de los 36 términos, así como la matriz sns_pca$rotation, que se convierte en un tibble mediante as_tibble(). El tibble combinado, con 11 columnas y 36 filas, se canaliza a la función pivot_longer(), que pivota la tabla de formato ancho a largo.

Los tres parámetros indican a la función que pivotee las 10 columnas de PC1 a PC10. Los nombres de columna anteriores se convierten ahora en las filas de una columna llamada PC

y los valores de columna anteriores en los valores de fila de una columna llamada Contribution.

El comando completo crea un tibble con 3 columnas y 360 filas:

```
> sns_pca_long <- tibble(SNS_Term = colnames(sns_terms),
+     as_tibble(sns_pca$rotation)) |>
+ pivot_longer(PC1:PC10, names_to = "PC", values_to = "Contribution")
```

La función ggplot() ahora puede usarse para representar gráficamente los términos contribuyentes más importantes para un componente principal dado. Por ejemplo, para examinar el tercer componente principal, filtraremos las filas para limitarlas solo a PC3, seleccionaremos los 15 valores de contribución más altos (considerando tanto los valores positivos como los negativos utilizando la función de valor absoluto abs()) y modificaremos SNS_Term para reordenarlos según la cantidad de contribución.

Finalmente, esto se canaliza a ggplot() con varios ajustes de formato:

```
> sns_pca_long |>
+   filter(PC == "PC3") |>
+   top_n(15, abs(Contribution)) |>
+   mutate(SNS_Term = reorder(SNS_Term, Contribution)) |>
+   ggplot(aes(SNS_Term, Contribution, fill = SNS_Term)) +
+     geom_col(show.legend = FALSE, alpha = 0.8) +
+     theme(axis.text.x = element_text(angle = 90, hjust = 1,
+       vjust = 0.5), axis.ticks.x = element_blank()) +
+     labs(x = "Social Media Term",
+       y = "Relative Importance to Principal Component",
+       title = "Top 15 Contributors to PC3")
```

El resultado se muestra en el gráfico a continuación. Dado que los términos con impactos positivos y negativos parecen estar divididos entre temas relacionados con sexo, drogas y rock and roll, se podría argumentar que este componente principal ha identificado una dimensión estereotipada de la identidad adolescente:

Al repetir el código ggplot anterior para los otros cuatro componentes principales entre los cinco primeros, observamos distinciones similares, como se muestra en la figura 11:

```
> # crea una función para visualizar los otros cuatro componentes
> plot_pca <- function(component) {
+   sns_pca_long |>
+     filter(PC == component) |>
```

```
+     top_n(15, abs(Contribution)) |>
+     mutate(SNS_Term = reorder(SNS_Term, Contribution)) |>
+     ggplot(aes(SNS_Term, Contribution, fill = SNS_Term)) +
+     geom_col(show.legend = FALSE, alpha = 0.8) +
+     theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
+       axis.ticks.x = element_blank()) +
+     labs(x = "Social Media Term",
+       y = "Relative Importance to Principal Component",
+       title = paste("Top 15 Contributors to", component))
+ }
```

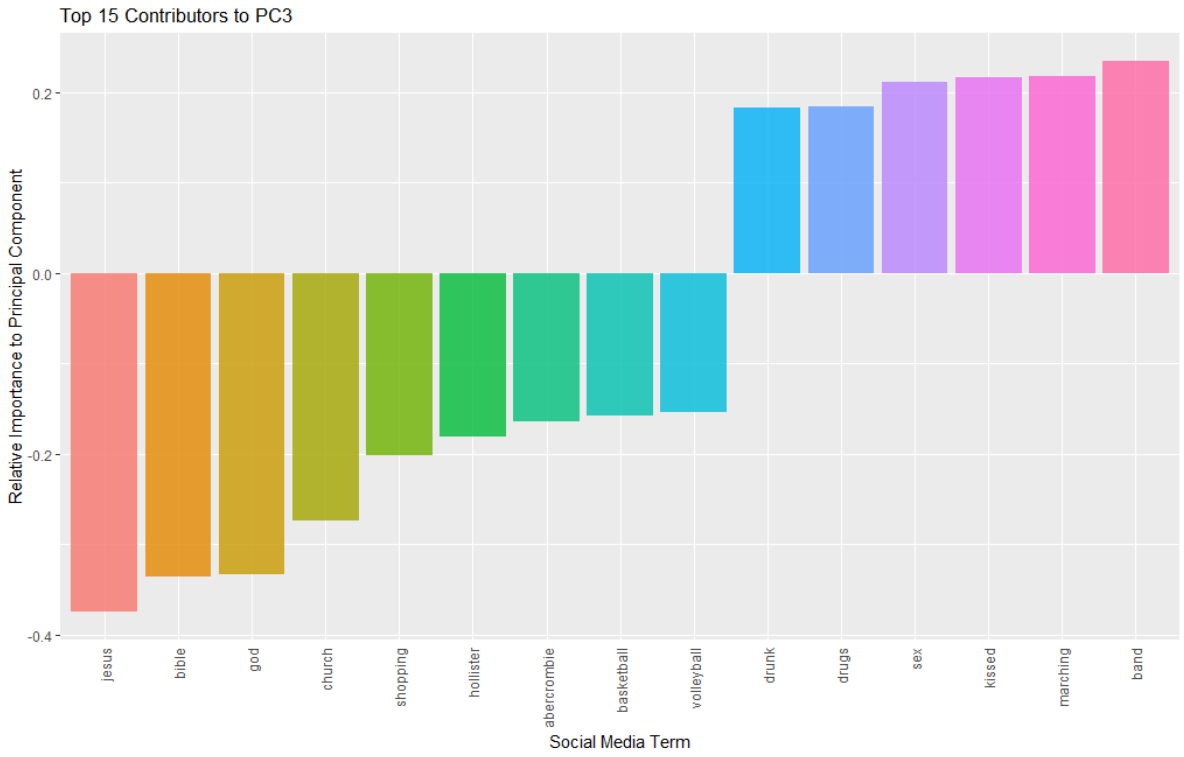

Figura 10: Los 15 términos principales que contribuyen al PC3.

```
> plot_pca("PC1")
> plot_pca("PC2")
> plot_pca("PC4")
> plot_pca("PC5")
```

PC1 es particularmente interesante, ya que cada término tiene un impacto positivo; esto podría estar diferenciando a las personas que tienen algo y nada en sus perfiles de redes sociales. PC2 parece favorecer los términos relacionados con las compras, mientras que PC4 parece ser una combinación de música y deportes, sin sexo ni drogas. Por último, parece que PC5 podría estar diferenciando entre términos deportivos y no deportivos. Examinar los

gráficos de esta manera ayudará a comprender el impacto de cada componente en el modelo predictivo.

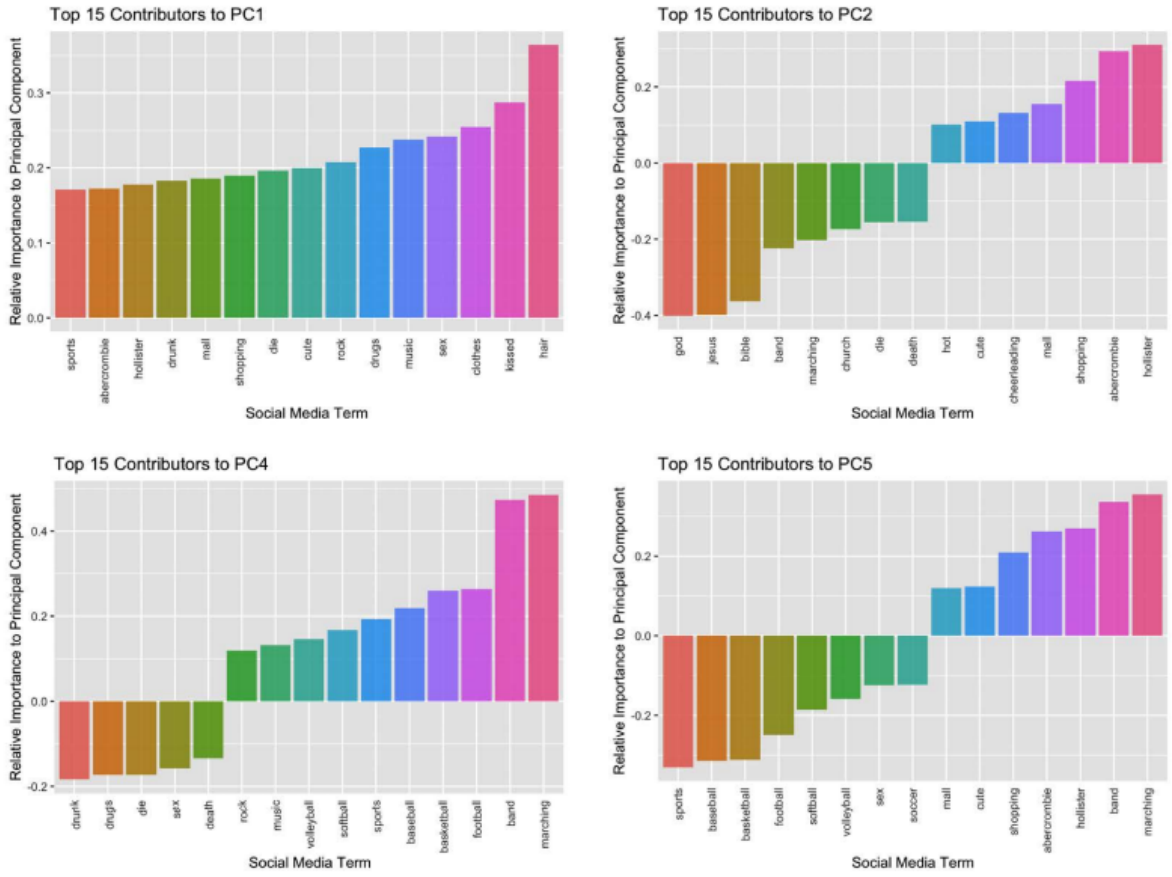

Figura 11: Los 15 términos principales que contribuyen a los otros cuatro componentes principales.

Comprender el análisis de componentes principales es de poca utilidad si la técnica no resulta útil para construir modelos de aprendizaje automático. En el ejemplo anterior, redujimos la dimensionalidad de un conjunto de datos de redes sociales de 36 a 10 componentes o menos.

Al fusionar estos componentes con el conjunto de datos original, podemos usarlos para realizar predicciones sobre el género o el número de amigos de un perfil. Comenzaremos utilizando la función cbind() para combinar las primeras cuatro columnas del frame de datos original con los datos del perfil transformados a partir del resultado del PCA:

```
> sns_data_pca <- cbind(sns_data[1:4], sns_pca$x)
```

A continuación, construiremos un modelo de regresión lineal que predice el número de amigos en redes sociales en función de los primeros cinco componentes principales. Este

enfoque de modelado se presentó en el tema, Pronóstico de Datos Numéricos - Métodos de Regresión. El resultado es el siguiente:

```
> m <- lm(friends ~ PC1 + PC2 + PC3 + PC4 + PC5, data = sns_data_pca)
> m

Call:
lm(formula = friends ~ PC1 + PC2 + PC3 + PC4 + PC5, data = sns_data_pca)

Coefficients:
(Intercept)          PC1          PC2          PC3          PC4          PC5
    30.1795       1.9857       0.9748      -2.5230       1.1160       0.8780
```

Dado que el valor de la intersección es aproximadamente 30.18, la persona promedio en este conjunto de datos tiene unos 30 amigos. Se espera que las personas con valores más altos de PC1, PC2, PC4 y PC5 tengan más amigos, mientras que valores más altos de PC3 se asocian con menos amigos, asumiendo que todo lo demás se mantiene igual.
Por ejemplo, por cada unidad de aumento en PC2, anticipamos aproximadamente un amigo adicional en promedio.

Dada nuestra comprensión de los componentes, estos hallazgos son coherentes; los valores positivos de PC2, PC3 y PC5 se asociaron con más actividades sociales. En cambio, PC3 se relacionaba con sexo, drogas y rock and roll, lo cual puede ser algo antisocial.
Aunque este es un ejemplo muy simple, el PCA puede utilizarse de la misma manera con conjuntos de datos mucho más grandes.

Además de mitigar el problema de la dimensionalidad, también tiene la ventaja de reducir la complejidad. Por ejemplo, un conjunto de datos con una gran cantidad de predictores puede ser demasiado costoso computacionalmente para que un algoritmo de k-vecinos más cercanos o una red neuronal artificial lo ejecuten tal cual, pero al seleccionar un número menor de componentes principales, estas técnicas podrían ser viables.

Experimente con el PCA y compara este tipo de extracción de características con otros métodos de selección de características, como filtros y envoltorios; es posible que tengas más éxito con uno u otro enfoque. Incluso si decides no utilizar la reducción de dimensionalidad, existen otros problemas con los datos altamente dimensionales que descubrirás en la siguiente sección.

Uso de datos dispersos

A medida que los conjuntos de datos aumentan en dimensión, es probable que algunos atributos sean **dispersos** (*sparse*), lo que significa que la mayoría de las observaciones no

comparten los valores del atributo. Esto es una consecuencia natural de la maldición de la dimensionalidad, donde este detalle cada vez mayor convierte las observaciones en valores atípicos identificados por su combinación única de atributos. Es muy poco común que los datos dispersos tengan un valor específico, o incluso ningún valor, como fue el caso de las matrices dispersas para datos de texto del tema, Aprendizaje Probabilístico: Clasificación con Bayes Naive, y las matrices dispersas para datos de carritos de compra del tema, Búsqueda de Patrones: Análisis de la Cesta de la Compra con Reglas de Asociación. Esto no es lo mismo que los datos faltantes, donde normalmente se desconoce una porción relativamente pequeña de valores.

En los datos dispersos, se conocen la mayoría de los valores, pero la cantidad de valores interesantes y significativos se ve eclipsada por una cantidad abrumadora de valores que aportan poco valor a la tarea de aprendizaje. Con datos faltantes, los algoritmos de aprendizaje automático tienen dificultades para aprender algo de la nada; con datos dispersos, les cuesta encontrar la aguja en el pajar.

Identificación de datos dispersos

Los datos dispersos pueden presentarse en diversas formas interrelacionadas. Quizás la forma más común sea la categórica, en la que una sola característica tiene una gran cantidad de niveles o categorías, y algunas tienen recuentos extremadamente bajos en comparación con las demás. Se dice que características como esta tienen una alta **cardinalidad** y generarán problemas de datos dispersos al introducirse en un algoritmo de aprendizaje.

Un ejemplo de esto son los códigos postales; en Estados Unidos, hay más de 40,000 códigos postales, algunos con más de 100,000 habitantes y otros con menos de 100. Por consiguiente, si se incluye una característica de código postal disperso en un proyecto de modelado, es probable que el algoritmo de aprendizaje tenga dificultades para encontrar el equilibrio entre ignorar y sobre enfatizar las áreas con pocos residentes.

Las características categóricas con muchos niveles se expresan a menudo como una serie de características binarias con una característica por nivel. Hemos utilizado estas características en numerosas ocasiones al construir manualmente variables binarias ficticias, y muchos algoritmos de aprendizaje hacen lo mismo automáticamente con datos categóricos. Esto puede llevar a una situación en la que las características binarias sean escasas, ya que los valores 1 se ven superados por los valores 0.

Por ejemplo, en un conjunto de datos de códigos postales de la población estadounidense, una pequeña fracción de los 330 millones de residentes corresponderá a cada uno de los 40,000 códigos postales, lo que hace que cada característica binaria de código postal sea muy escasa y difícil de usar para un algoritmo de aprendizaje.

Muchas formas de los llamados “big” data son inherentemente muy dimensionales y escasas. La escasez está estrechamente relacionada con la maldición de la dimensionalidad. Al igual que el universo en constante expansión crea mayores vacíos entre los objetos, se podría argumentar que cada conjunto de datos se vuelve disperso a medida que se añaden más dimensiones. Los datos de texto suelen ser dispersos porque cada palabra puede considerarse una dimensión y pueden aparecer innumerables palabras, cada una con una baja probabilidad de aparecer en un documento específico.

Otras formas de big data, como los datos de ADN, los datos de la cesta de la compra transaccional y los datos de imágenes, también suelen presentar el problema de la escasez. A menos que se aumente la densidad del conjunto de datos, muchos algoritmos de aprendizaje tendrán dificultades para aprovechar un conjunto de datos extenso y complejo.

Incluso un rango numérico simple puede ser disperso. Esto ocurre cuando la distribución de valores numéricos es amplia, lo que lleva a que algunos rangos de la distribución tengan una densidad muy baja. Los ingresos son un ejemplo de esto, ya que los valores generalmente se vuelven cada vez más dispersos a medida que aumentan los ingresos.

Esto está estrechamente relacionado con el problema de los valores atípicos, pero aquí claramente esperamos modelar los valores atípicos. Los datos numéricos dispersos también pueden encontrarse cuando los números se han almacenado con un grado de precisión excesivamente específico. Por ejemplo, si los valores de edad se almacenan con decimales en lugar de enteros, como 24.9167 y 36.4167 en lugar de simplemente 24 y 36, se crea un vacío implícito entre los números que algunos algoritmos de aprendizaje pueden tener dificultades para ignorar. Por ejemplo, un árbol de decisión podría distinguir entre personas de 24.92 y 24.90 años, lo que probablemente esté más relacionado con el sobreajuste que con una distinción significativa en el mundo real.

Reducir manualmente la escasez de un conjunto de datos puede ayudar a un algoritmo de aprendizaje a identificar las señales importantes e ignorar el ruido. El enfoque utilizado depende del tipo y el grado de escasez de datos, así como del algoritmo de modelado empleado. Algunos algoritmos son mejores que otros para manejar ciertos tipos de datos dispersos. Por ejemplo, el método naïve Bayes funciona relativamente bien con datos categóricos dispersos, los métodos de regresión funcionan relativamente bien con datos numéricos dispersos, y los árboles de decisión tienden a tener dificultades con datos dispersos en general debido a su preferencia por características con un mayor número de categorías.

Métodos más sofisticados, como las redes neuronales profundas y el boosting, pueden ser útiles, pero en general, es mejor si el conjunto de datos se puede densificar antes del proceso de aprendizaje.