Informe del desafío técnico

Mauro Bringas
24 de Marzo de 2025

Introducción:

Se nos pide entrenar un algoritmo de Machine Learning para detectar transacciones fraudulentas y separarlas de transacciones válidas. Este tipo de problemas suele tener la característica de un fuerte desbalance de clases; es decir, encontraremos pocos casos de fraude en relación a las transacciones no fraudulentas.

La métrica que guiará la selección del modelo adecuado remite directamente a la ganancia generada por las transacciones válidas que son clasificadas como no-fraudulentas (el 25% del monto de la transacción, negativos verdaderos) y la pérdida del 100% del valor de las transacciones fraudulentas que son clasificadas como válidas (falsos negativos). El objetivo es maximizar esta ganancia.

El dataset provisto contiene 18 variables explicativas y la variable objetivo, con lo cual el problema se puede encarar desde una perspectiva de aprendizaje supervisado.

Hipótesis:

Se podrá mejorar la ganancia respecto de un modelo baseline (en este caso, un clasificador tipo Naive Bayes) incorporando una estrategia de balanceo de clases o un algoritmo de machine learning que tenga en sus fundamentos incorporada la posibilidad de lidiar con el desbalance (en este caso, Random Forest).

Análisis y transformaciones del dataset

La variable objetivo, llamada “fraude” es dicotómica y está presente en los 150 mil registros provistos. Dentro de las 18 variables explicativas podemos encontrar 13 numéricas, 4 categóricas y una fecha (presentada como string).

Las variables numéricas se graficaron en forma de histograma y de boxplot. Las variables con valores de punto flotante muestran, en su mayoría, una distribución asimétrica pesada a izquierda (abundantes valores pequeños). Según el criterio de las distancias intercuartiles (IQR), que sugiere considerar “atípicos” los valores que distan en más de 1.5 IQR del tercer cuartil, todas las variables numéricas de punto flotante contienen casos atípicos. Sin embargo, por la naturaleza asimétrica de las distribuciones parece razonable no tratarlos de manera distintiva. Los graficos bidimensionales generados de a pares no presentan rasgos destacables en su distribución, incluso cuando se diferencia visualmente los casos de fraude y de transacciones válidas.

Dos de estas variables, “b” y “c” presentan un 10% de valores faltantes. Otras cuatro, “d”,”f”,”l” y “m” tienen menos de 5% de valores restantes. Para no utilizar estrategias de imputación que dependan de otros valores en el dataset, se decidió generar una flag por cada una que indique si el valor ha sido imputado y en lugar de los valores perdidos se colocó un número que se encuentra fuera del rango observado para estas variables (un número negativo). Vale aclarar que esto tendrá menos repercusión en los algoritmos de la familia de los árboles de decisión que en el resto, dado que las variables flag permiten fácilmente descartar los registros con valores imputados.

Finalmente, estas variables numéricas fueron incrementadas (a modo de ingeniería de variables) multiplicándolas entre sí de a pares (luego de la imputación) para permitir cierta no linealidad en la toma de decisiones de los algoritmos utilizados.

La variable fecha se presenta en formato ISO (19 caracteres, YYYY-MM-DDThh:mmTZD) y no presenta casos donde el string esté incompleto. Se pre-trató el atributo considerando su formato y se generaron variables adicionales que toman en cuenta el año, mes, día de la semana y hora de la transacción. La variable fecha *per se* no se utilizó para entrenar los modelos de Machine Learning.

Las variables categóricas fueron tratadas cada una en distinta forma. Una de ellas, llamada "j", parece hacer referencia a un identificador y fue descartada. La variable "g" parece hacer referencia al país donde se realizó la transacción. Con el objetivo de generar variables dicotómicas para codificar estos valores, agrupamos los países menos abundantes (48 países con menos de 1000 transacciones c/u en una nueva categoría "OTH" (otros). De esta forma se reemplazó este atributo categórico por otros 5 de naturaleza categórica, que hacen referencia a los siguientes "países": Argentina, Brasil, Uruguay, Estados Unidos u otros.

Las otras tres variables categóricas son de tipo Y/N. Dos de ellas no presentaron valores faltantes y fueron codificadas como dummies booleanos. La tercera de ellas presentó una alta proporción de valores faltantes (78% de los casos). Ante la disyuntiva de descartar el atributo y generar una categoría "UNK" haciendo referencia al valor desconocido, se decidió tomar el segundo enfoque y generar variables dummy para los niveles Y, N y UNK.

Modelos utilizados:

Aproximación de orden cero

No es exactamente un "modelo de machine learning", pero es una estrategia frecuente para saber qué performance se puede lograr sin utilizar ningun modelo de machine learning. Esta aproximación se basa en asignar la clase mayoritaria a todos los eventos en el dataset de testeo y calcular la performance de las predicciones bajo este supuesto.

Clasificador Naive Bayes

Este clasificador aprende la probabilidad condicional de pertenecer a cada clase según las chances de observar ciertos valores en los datos de entrenamiento. Se utiliza como modelo de referencia o baseline por su simpleza a la hora del entrenamiento, dado que no requiere ajustar hiperparámetros, y su relativa buena performance en tareas de clasificación binaria.

Random Forest Classifier

Este modelo es uno de los denominados "de ensamble" pues su funcionamiento se basa en elegir la categoría de clasificación por "votación" de muchos clasificadores más simples. En el caso del random forest, la unidad es el árbol de decisión. Estos clasificadores intentan separar los casos correspondientes a distintas clases mediante una sucesiva partición binaria del espacio generado por los atributos de la muestra. Es decir, van "partiendo el espacio" con rectas fijadas para un dado valor de algún atributo, intentando optimizar la separación de casos.

La fortaleza de los modelos de ensamble se basa en la diversidad con la cual cada modelo sencillo realiza la separación de casos, llevando usualmente a decisiones más robustas y menor sobre-ajuste.

La elección de modelos se llevó adelante probando distintos valores en algunos hiperparámetros de la familia de los modelos y se realizó por Grid Search (probar todas las combinaciones) con una estrategia de Cross Validation de 5 partes (se generan 5 conjuntos de test y de validación a partir de los datos de entrenamiento para hacernos una idea más robusta de la performance. Se informa la performance en validación como promedio de la performance de los 5 modelos entrenados).

Evaluación

Realizamos una separación del dataset en 120.000 registros para entrenamiento/ validación y 30.000 para evaluación (testeo). Reportamos la performance de los distintos modelos evaluandolos con este último dataset.

El primer chequeo que se realiza es cuál sería la ganancia si asignamos a todos los registros la clase mayoritaria (no-fraude). En este caso la ganancia es de 195 mil pesos.

El modelo baseline propuesto es un Naive Bayes entrenado con la totalidad del set de entrenamiento (120.000 registros y 202 atributos). Éste devuelve una ganancia total de 40.387 pesos. Esto es una muy mala performance, incluso respecto de “no realizar una clasificación” y simplemente asignar la clase mayoritaria.

Se realizó una búsqueda de hiperparámetros en la familia de los Random Forest explorando según la Tabla 1. Seleccionamos el mejor y peor modelo de estos explorados según su performance en cross validation de 5 folds y reportamos la ganancia obtenida con el dataset de testeo. Los valores correspondientes son 181 mil y 195 mil. Podemos ver de las matrices de confusión (Figura 1.b.) que se pierden muchos casos de fraude pero también hay pocos falsos positivos (donde se pierde la totalidad del monto de la transferencia). La performance de este modelo termina siendo comparable a asignar la clase mayoritaria a todos los registros en test.

Tabla 1. Valores de hiperparámetros de Random Forest que se utilizaron para la búsqueda con validación cruzada.

Hiperparámetro	Valores
n_classifiers	10,20,50
max_depth	5,10,15
min_sample_split	5,10,20

Cuando activamos el hiperparámetro de los random forest que tiene particular cuidado con el desbalanceo de clase, llamado class_weight en la implementación de scikit-learn para Python (ref), el mejor modelo en validación da una ganancia de 225 mil pesos en testeo. En la matriz de confusión (Figura 1.c.) que surge del testeo de este último puede verse que se combina una buena detección de los casos de fraude (Verdaderos Positivos) con una baja proporción (en relación a lo visto en modelos anteriores) de Falsos Positivos.

Probamos una segunda estrategia de mitigación para el desbalance, principalmente para determinar si mejora la performance del modelo Naive Bayes. Procedemos

modificando el dataset de entrenamiento y validación. Realizamos un submuestreo de transacciones válidas a un 10%, de forma tal que la composición final de dicho dataset sea 66% de transacciones válidas y 22% de casos de fraude. Luego del entrenamiento del modelo, presentamos la ganancia obtenida en el dataset de testeo, que tiene la composición correcta (solo 5% de fraude). Esta ganancia es de 62 mil pesos. Podemos ver que luego de este entrenamiento (Figura 1.d.), mejora la detección de casos de fraude (incrementa los Verdaderos positivos) pero sigue dando una enorme cantidad de Falsos Positivos. Este intento de corregir el desbalance no permite alcanzar siquiera la performance de la aproximación de orden cero (asignar no-fraude a todos los casos de test).

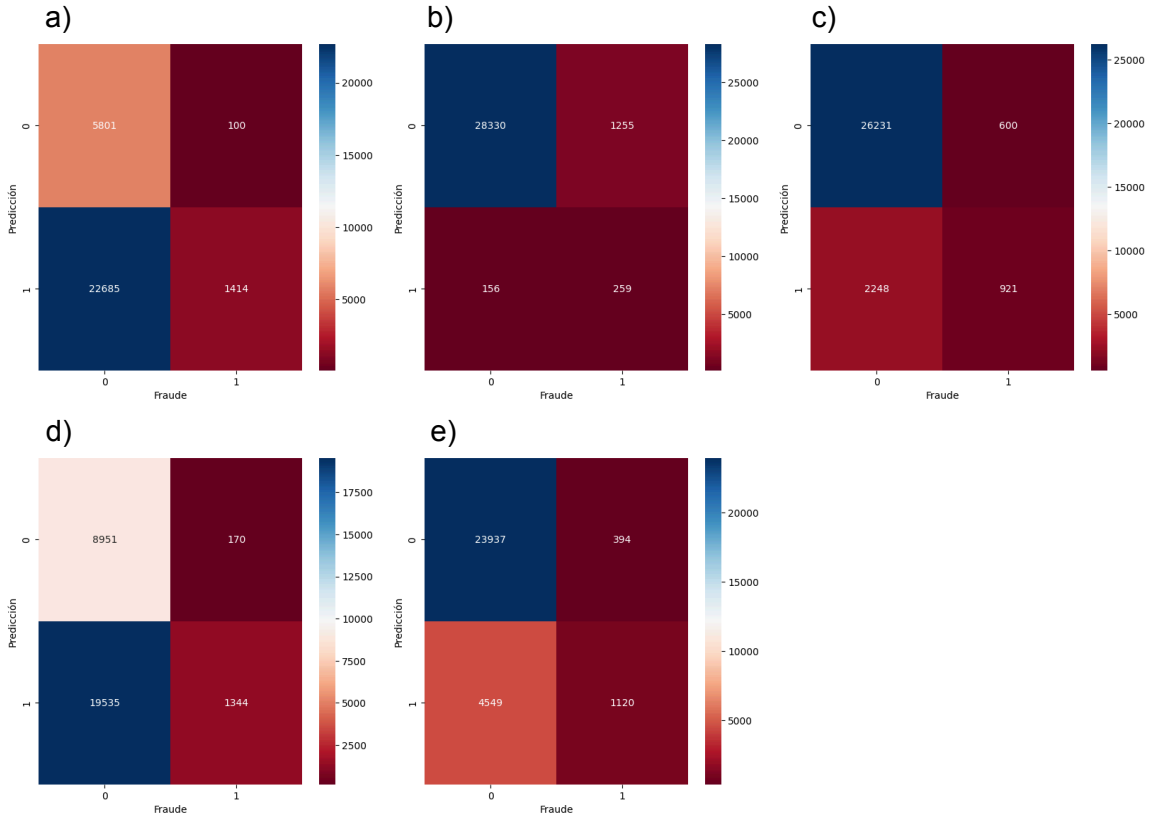

Figura 1, Matrices de confusión sobre el dataset de testing del modelo Naive Bayes (a), del Random Forest sin balanceo de clases (b), del Random Forest con pesado relativo de las clases (c), del modelo de Naive Bayes con submuestreo del train (d) y del Random Forest con submuestreo del test (e)

Por último, casi como curiosidad, realizamos la misma búsqueda de hiperparámetros que la realizada con los datasets de train-val y test originales pero realizando el submuestreo en train-val. En este caso la performance de validación es muy mala, obteniendo una ganancia cercana a cero. Sin embargo, tomando el mejor modelo en validación y evaluándolo sobre el dataset de testeo la ganancia obtenida es cercana a 207 mil pesos. Esta discrepancia es esperable dado que para la ganancia pesa mucho las predicciones correctas/erróneas de transacciones válidas, que en el dataset submuestreado son menos. Esta relativamente “buena” performance es esperable, ya que los Random Forest tienen una buena capacidad de ajuste de las observaciones de entrenamiento, pero

es peor al modelo entrenado con la totalidad de los datos - por el simple hecho de que tiene menos datos para entrenarse.

Conclusión

Hemos verificado que el modelo llamado “baseline” funciona bastante peor que simplemente asignar la clase mayoritaria a todos los casos de test. Llos modelos de la familia Random Forest presentan una gran mejora (aproximadamente 4 o 5 veces mayor ganancia en testeo) respecto del modelo baseline, pero no todos son mejores que la aproximación de orden cero. Esto quiere decir que no cualquier modelo resulta mejor que no tener un modelo.

Hemos contrastado la ventaja obtenida por el uso de un modelo robusto ante el desbalance de clases (RF) con la mejora asociada a la estrategia de sub-muestreo de clase mayoritaria, preservando el modelo Naive Bayes y también en la familia de los Random Forest. Nuestros resultados indican que el submuestreo provoca una mejora en performance respecto del dataset original de entrenamiento (con extremo desbalance) en Naive Bayes, pero queda por debajo en desempeño que el modelo robusto al desbalance e incluso de la aproximación de orden cero.

El modelo que se reporta como óptimo, dentro de las posibilidades evaluadas, es un random forest que toma en consideración la cantidad relativa de observaciones con las distintas categorias a clasificar. Este random forest cuenta con 50 estimadores, profundidad máxima de 10 y cantidad mínima de registros para realizar un sucesivo split de 10.

Para exploraciones futuras queda evaluar la posible mejora de los modelos potenciados por gradiente (tipo XGboost y ligthGBM), que también suelen tener excelente performance en datos tabulares pero su costo computacional de entrenamiento es mayor. Por otra parte, también vale la pena explorar estrategias que busquen aislar casos atípicos (por ejemplo, con el uso de bosques de aislamiento).