Clasificación de ingreso anual mayor a 50k

José Pablo Martínez Valdivia

September 7, 2024

Abstract

This paper presents a predictive model based on the Random Forest algorithm to classify income levels using the Census Income (50k) dataset. The objective of the study is to predict whether an individual's income exceeds $50,000 per year based on demographic and work-related attributes. Furthermore, I discuss the various preprocessing decisions made, such as handling encoding categorical variables, and the choice of hyperparameters to optimize model performance.

1. Introducción

El dataset "Census Income" fue proveido por el "UCI Machine Learning Repository". Contiene 14 variables y 48842 instancias de datos provenientes de un censo realizado en 1994 para determinar si un adulto genera un ingreso anual mayor a $50k. Emplearemos el modelo de bosque aleatorio (random forest) para emplear una predicción categórica y disminuir el overfit en el modelo.

2. ETL

2.1. Extracción

Los datos fueron descargados por medio de la librería de "ucimlrepo". Estos proveen datos demográficos como edad, estado civil, educación, ocupación, raza, sexo, país de origen, etc.

Los datos se encuentran completos en su mayoría, pero hay datos faltantes en las columnas de tipo de trabajo, ocupación y país de origen como se puede ver en 1. Se tomó la decisión de deshacernos de las filas con datos faltantes las cuales representan 1221 instancias, lo cual es el 2.4 % de los datos; sobrándonos 47621 instancias. Esta decisión se tomó considerando que algún tipo de imputación podría introducir errores para la tarea de clasificación.

Por otro lado la columna con las clases objetivo consta de strings que representan si la ganancia fue menor o igual a 50k dolares $\leq 50k$ o mayor $50k$. Esta columna tiene un problema con la imputación ya que en ambas clases estas contien un punto "." en unas instancias y no en otras, generándonos 4 clases en lugar de dos. Después de revisar la documentación del dataset y confirmar que no se menciona nada sobre la significancia de este punto en los registros se tomó la decisión de asumir que el punto no es significativo y transformar los datos para removerlo de las clases correspondientes.

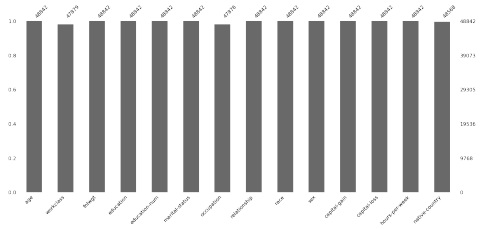

Figura 1: Valores faltantes.

2.2. Transformación

Para poder general el bosque aleatorio necesitamos convertir las columnas categóricas en valores numéricos. Para esto haremos uso de *one-hot encoding*, esta técnica consta de tomar todas las categorías en una columna y crear una columna por categoría, dejando un uno en la categoría que presenta la instancia y ceros en el resto.

Colour
Green
Red
Blue

Green	Red	Blue
0	1	1
1	1	1
1	0	1
0	0	0
0	1	0

Figura 2: Método de one-hot encoding [3].

3. Marco teórico

3.1. Árboles de decisión

Un árbol de decisión, específicamente para clasificación, es un modelo que nos permite producir una salida discreta. Este modelo consta de una estructura de árbol binario en la cual cada nodo padre presenta una comparación lógica de una de las entradas y las hojas representan una clase de salida.

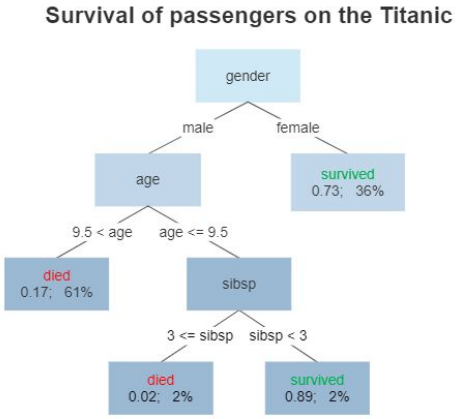

Figura 3: Estructura de un árbol de decisión [4].

Este modelo nos trae una serie de problemas. Para empezar, la complejidad de tiempo para obtener el resultado de un árbol es $O(n \log n)$ [2] dónde n se refiere al número de muestras con el que hayamos entrenado el árbol, por otro lado la complejidad para crear el árbol es de [2]:

$$O(n_{muestras} n_{variables} \log n_{muestras}) \qquad (1)$$

lo cual indica que los tiempos de entrenamiento escalaran $O(n \log n)$ [2] con base a la cantidad de muestras que tengamos en el dataset. Esto quizá no presenta un problema tan grave, ya que solo contamos con 48 mil entradas, pero el problema más grande de los árboles de decisión es que el generar un árbol muy complejo está sujeto sobre ajuste, es decir, que el modelo se adapte a predecir los datos de entrenamiento y por lo tanto no generalice y tenga una menor precisión al predecir datos nuevos.

3.2. Bosque Aleatorio

Una de las formas de combatir el sobre ajuste inevitable de los árboles de decisión es haciendo uso de un bosque aleatorio, este es un método de ensamble en el cual, en lugar de generar un solo árbol complejo [5]. Generamos muchos árboles pequeños y tomamos como salida la clase que haya sido más votada por todos los árboles. Este método nos permite en general mitigar el sobre ajuste dada la aleatoriedad de cada árbol.

4. Entrenamiento

El set de datos fue separado en tres conjuntos: entrenamiento, validación y pruebas constando respectivamente del 60 %, 20 % y 20 % de los datos del set. Para el modelo se decidió usar los siguientes hiperparámetros:

- n_estimators: 300
- max_depth: None
- min_samples_split: 4
- min_samples_leaf: 1
- max_features: sqrt

Estos parámetros serán usados inicialmente para probar como se desempeña el modelo contra los datos de validación.

La prueba inicial nos regresa una R^2 de 0,979132 para el conjunto de entrenamiento y 0,857953 para prueba. Podemos analizar la matriz de confusión 4 y vemos que la mayoría de las predicciones se encuentran en la clase ≤ 50 con 870 predicciones incorrectas.

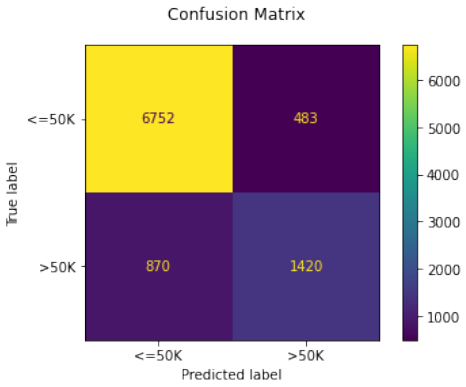

Figura 4: Matriz de confusión.

Comparando los resultados de los errores podemos notar un claro sobre ajuste a los datos de entrenamiento, esto se debe a que la profundidad máxima sea None, igualmente podemos analizar el dataset para ver la cantidad de instancias que tenemos para cada clase:

- $\leq 50K$: 37155
- $> 50K$: 11687

La gran profundidad de los árboles está causando que se aprendan los datos del conjunto de entrenamiento y no puedan generalizar lo suficiente para responder con datos que no hayan visto antes.

Se realizaron pruebas posteriores cambiando los hiperparámetros y comparando con un conjunto de validación y prueba para tener una mejor visión a la precisión del modelo. Posteriormente pasamos a hacer pruebas con diferentes parámetros para intentar mejorar la precisión del modelo.

5. Resultados

Después de realizar las pruebas con los diferentes hiperparámetros como se ve en la tabla 1 se llegó a una precisión de:

- Entrenamiento: 0.874449
- Validación: 0.875459
- Pruebas: 0.862047

con los siguientes hiperparámetros:

- n_estimators: 200
- max_depth: 40
- min_samples_split: 2
- min_samples_leaf: 5
- max_features: sqrt

se decidió por estos, ya que fueron los que produjeron un nivel bajo de sobre ajuste y a la vez un buen resultado de precisión del modelo.

Analizando la matriz de confusión de nuestro modelo ajustado 5 podemos observar un ligero incremento en las predicciones correctas para ≤ 50 y una pequeña reducción en las respuestas incorrectas para > 50. Se decidió tomar el error para el conjunto de validación 0,862047 como el mejor que se pudo lograr dada la complejidad del problema y el método de prueba.

Otros métodos para realizar pruebas automatizadas como lo es el GridSearch fueron considerados, el cual consta de especificar diferentes valores de prueba para cada parámetro y posteriormente entrenar el modelo y medir su precisión, resultando en un modelo con los mejores parámetros posibles de todos los probados. Sin embargo, este fue descartado dado el tiempo que tarda este algoritmo escala basado en todas las combinaciones posibles de los datos que se le proveen.

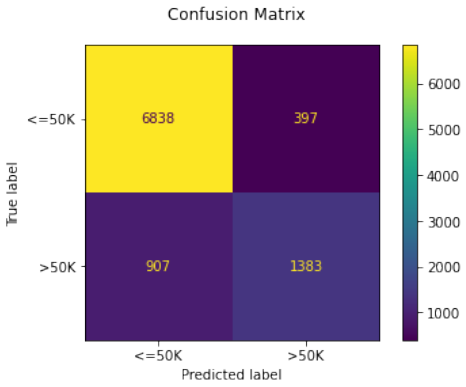

Figura 5: Matriz de confusión final.

6. Conclusión

El modelo en general se puede clasificar con un grado de sesgo medio, producido por la cantidad de observaciones apuntando a personas con ingresos menores a 50 mil dólares. Igualmente la varianza se puede considerar como media dado a la escala del problema ya que este considera solo clases entre menor o igual a 50 mil dólares y mayor a los mismos, este es un espectro bastante amplio y las predicciones al nivel final de precisión pueden ser muy malas en el caso de predecir erróneamente la clase de mayor a 50 mil.

Referencias

[1] Becker, B., & Kohavi, R. (n.d.). UCI Machine Learning Repository. https://doi.org/10.24432/C5XW20

[2] 1.10. Decision Trees. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/tree.html

[3] O, C. (2023, August 16). Learn one hot encoding in 3 minutes - Dev Genius. Medium. https://blog.devgenius.io/learn-one-hot-encoding-in-3-minutes-d1ef270d6e86

[4] Wikipedia contributors. (2024, July 16). Decision tree learning. Wikipedia. https://en.wikipedia.org/wiki/Decision_tree_learning

[5] Wikipedia contributors. (2024b, August 23). Random forest. Wikipedia. https://en.wikipedia.org/wiki/Random_forest

estimators	depth	samples_split	samples_leaf	features	Train	Validation	Test
400	20	2	1	sqrt	0.910174	0.910761	0.862257
300	None	4	1	sqrt	0.979132	0.979265	0.857953
300	None	4	4	6	0.868858	0.871391	0.857113
200	40	3	2	sqrt	0.901039	0.900656	0.864462
200	30	2	5	sqrt	0.874449	0.875459	0.862047
100	20	5	2	sqrt	0.882219	0.883333	0.862782

Cuadro 1: Random Forest Classifier Results with Different Hyperparameters