FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2

[75.06] Organización de Datos
Curso Collinet
Primer Cuatrimestre de 2021

Alumno:	Padrón	Mail
BIANCARDI, Julián	103945	jbiancardi@fi.uba.ar
Hetrea, Joaquin	103944	jhetrea@fi.uba.ar

Índice

1. Introducción

En el siguiente trabajo entrenaremos distintos tipos de modelos, como asi tambien distintos tipos de ensambles para encontrar aquel que mejor prediga nuestra variable objetivo: si la persona tiene o no alto valor adquisitivo. Tambien realizaremos distintos preprocesamientos antes del entrenamiento para obtener el mejor modelo dentro de los de su mismo tipo.

Se nos provee un dataset de entrenamiento con el cual trabajaremos en todos los archivos que se entregan. Por otra parte se nos entrega otro dataset cuya función será la de validar el trabajo realizado.

2. Preprocesamientos

La siguiente tabla muestra todas las funciones de preprocesamientos utilizadas durante el trabajo. Las siguientes se encuentra en el archivo preprocessing.py:

Nombre	Descripción	Función
Inicializar el dataset	Inicializa el dataset recibido por parametro, realizando un tratamiento de missings y convirtiendo los tipos de datos de cada columna	init_dataset
One Hot Encoding	init: Fittea el dataset en el que se va a basar para realizar el OHE apply: Aplica OHE a las columnas categoricas del dataset recibido	init_OHE apply_OHE
Ordinal Encoder	Aplica ODE a las columnas categoricas del dataset recibido	apply_ODE
Estandarización	Estandariza los valores de las columnas numéricas	standarize
Normalización	Normaliza los valors de las columnas numéricas	normalize
Escalado	Escala los valores de las columnas numéricas en un rango especificado	scale
Reducción por Frecuencia	Agrupamos los valores categoricos que tiene una baja frecuencia de aparición en un nuevo valor categórico	reduce_by_frequency
Eliminar Features	Eliminamos columnas que consideramos que no aportan información	eliminar_features
Discretización	Transforma los valores de las columnas numéricas en valores discretos que representan intervalos mediante algoritmos de clustering	discretize_columns

3. Modelos

La siguiente tabla muestra los modelos entrenados y sus métricas obtenidas:

Nombre	Preprocesamiento	AUC ROC	Precision	Recall	F1-score	Accuracy
Boosting	OHE	0.926	0.79	0.65	0.71	0.87
Voting	OHE	0.914	0.78	0.61	0.68	0.86
Red Neuronal	OHE + Reduc. + Estandar.	0.908	0.75	0.57	0.65	0.85
Random Forest	OHE + Reducción	0.906	0.77	0.52	0.62	0.85
Arbol de Decisión	OHE	0.903	0.70	0.66	0.68	0.85
KNN	OHE + Elimin. + Reduc. + Estandar.	0.886	0.70	0.51	0.59	0.83
Regresión Logistica	OHE + Escalado	0.893	0.71	0.57	0.63	0.84
EnsamblelNB	scale / Reduc. + ODE	0.888	0.74	0.46	0.57	0.83

En cada uno de los modelos inicialmente se aplica la función de preprocesado init_dataset, por

lo tanto no se indica en la tabla. Esta función realiza un tratamiento de missings que pueden ser: transformarlos en una nueva categoría o removerlos. Para el entrenamiento de todos los modelos optamos por generar una nueva categoría. Ademas se elimina la columna educacion_alcanzada debido al análisis realizado en el primer trabajo práctico, donde se veia una relación lineal. Por ultimo realiza una conversion de los tipos de datos correspondiente para cada columna.

Por otra parte, cuando se menciona el preprocesamiento OHE se aplican las funciones init_OHE y apply_OHE. Estas funciones tambien deben aplicarse a los datos de Holdout como se ve en los notebooks, ya que los modelos esperan una determinada cantidad de features.

Las distintas busquedas de los hiperparametros para los modelos se realizaron primero con RandomSearchCV, para identificar el rango de posibles valores, y luego una busqueda más exhaustiva con GridSearchCV.

Las particiones realizadas para el entrenamiento y validación de todos los modelos son de 80 % y 20 % respectivamente. Estas particiones fueron realizadas con el metodo "stratify"para la variable objetivo puesto que mediante el análisis realizado en el primer trabajo se vio que la misma se encontraba desbalanceada.

Para el modelo 'EnsambleNB' realizamos un ensamble de tipo 'Voting' con los modelos Gaussian Naive Bayes y Categorical Naive Bayes, los cuales entrenamos previamente con distintos preprocesados para hallar la mejor combinación de ambos. Si bien este ensamble no obtuvo buenas métricas comparado con los demás modelos, sus métricas fueron mejores que la de ambos Naive Bayes por sepadado.

Por último, las métricas Precision, Recall y F1-score son calculadas para la clase 1: tiene alto valor adquisitivo.

4. Conclusiones

Luego de haber entrenado los distintos modelos con los diferentes hiperparametros y preprocesados, concluimos que el mejor modelo bajo la métrica AUC_ROC es Boosting por lo que recomendamos este ensamble para realizar las predicciones.

Para obtener este resultado se inicializó el dataset de la siguiente manera: tratamos los missings como una categoria nueva, eliminamos la columna educacion alcanzada y luego aplicamos OHE a las columnas categóricas. Para la busqueda de los hiperparámetros realizamos Random Search con distintos intervalos de parámetros, y luego una busqueda final mas exaustiva e intervalos mas reducidos con GridSearch.

Es de esperar este resultado pues estamos hablando de un ensamble y una de las principales caracteristicas de estos es que se acercan mas al concepto del problema que queremos solucionar. Ademas, el ensamble Boosting le da un peso mayor a los casos predichos erroneamente en los arboles anteriores, por lo que pensamos que ayudaría con el problema general de los modelos que analizamos previamente, que es predecir los casos con alto valor adquisitivo correctamente.

Para el baseline realizado en el primer trabajo práctico podemos ver que no hicimos uso de las buenas practicas para entrenar un modelo:

- No hicimos la particion de nuestros datos para entrenar y luego validar. En este punto tenemos problemas de sesgo y posiblemente overfitting
- Utilizamos unicamente como métrica para medir nuestro modelo el accuracy cuando nuestro problema estaba desbalanceado. Deberiamos haber utilizado distintas metricas como AUC_ROC, precision, recall, etc.
- Para el entrenamiento del modelo nos basamos en gráficos y proporciones de nuestra variable objetivo, similar a como trabajo un arbol de decision, excepto que sin el respaldo de los cálculos matemáticos que realizan los arboles.
- No realizamos ningún tipo de preprocesamiento además del tratamiento de missings.

El mejor modelo a la hora de tener la menor cantidad de falsos positivos es Bosting: tiene una precision de la clase 1 de 0.79. Cabe destacar que si bien es el modelo con menor cantidad de falsos positivos, el modelo tambien tiene una baja cantidad de falsos negativos.

Si necesitaramos una lista de todos los que potencialmente son de valor adquisitivo sin preocuparnos demasiado si metemos en la misma personas que realmente no tienen alto valor adquisitivo, lo que estamos buscando es un modelo con alto recall de la clase 1. De los modelos que entrenamos, el que mejor cumple con dicha condición es: Arbol de Decision con un recall de la clase 1 de 0.66.