.UBAfiuba
FACULTAD DE INGENIERÍA

(75.06 - 95.58)
Organización de Datos

Trabajo Práctico - Parte 2

Alumnos

- Lukas Nahuel De Angelis Riva (Padrón: 103784)
- Nicolás Continanza (Padrón: 97576)

Primer Cuatrimestre 2021

Nombre Preprocesamiento	Explicación	Nombre de la función
Base	Selecciona las variables utilizadas en el baseline de la parte 1 del TP.	preprocessing_base_parte_1
StandardBase	Idem anterior, estandarizando las columnas del dataset.	standard_preprocessing_base_parte_1
Significantes **V**%	Recibe X_train, X_test y una varianza **[V]** a explicar por PCA. Devuelve X_train y X_test fiteados con PCA	preprocessing_significantes
Equilibrado	Equilibra la cantidad de muestras con alto y bajo poder adquisitivo para que haya 50% de cada una en el set de entrenamiento.	preprocessing_equilibrado
Mejores variables según Tree	Selecciona las mejores variables según el entrenamiento de un árbol.	preprocessing_mejores_por_arbol
Standard mejores variables según Tree	Idem anterior, estandarizando las columnas del dataset	standard_preprocessing_mejores_por_arbol

Modelo	Preprocesamiento	AUC-ROC	Accuracy	Precision	Recall	F1 Score
1- DecisionTreeClassifier	Equilibrado	0.9101	0.8011	0: 0.9496 1: 0.5554	0: 0.7795 1: 0.8693	0: 0.8562 1: 0.6778
2- KNeighborsClassifier	Mejores variables según Tree	0.8972	0.8551	0: 0.8687 1: 0.7870	0: 0.9533 1: 0.5452	0: 0.9090 1: 0.6441
3- MultinomialNB	Significantes 90% + MinMax_Scaler[1]	0.8656	0.7594	0: 0.7594 1: 0.0000	0: 1.0000 1: 0.0000	0: 0.8632 1: 0.0000
4- SVC	Mejores variables según Tree	0.9142	0.8641	0: 0.8718 1: 0.8243	0: 0.9626 1: 0.5533	0: 0.9150 1: 0.6622
5- RandomForest	Sin preprocessing	0.9153	0.8603	0: 0.8671 1: 0.8236	0: 0.9640 1: 0.5325	0: 0.9130 1: 0.6468
6- LogisticRegression	Standard mejores variables según Tree	0.8942	0.8455	0: 0.8702 1: 0.7347	0: 0.9363 1: 0.5581	0: 0.9020 1: 0.6343

[1] MinMax_Scaler escala los datos entre 0 y 1. No está en el archivo de preprocessing.py pues sólo lo utiliza NaiveBayes.

7- BaggingClassifier	Sin preprocessing	0.9186	0.8657	0: 0.8802 1: 0.7985	0: 0.9530 1: 0.5898	0: 0.9152 1: 0.6784
8- GradientBoosting	Base	0.9151	0.8537	0: 0.8726 1: 0.7659	0: 0.9453 1: 0.5646	0: 0.9075 1: 0.6500
9- AdaBoost (base estimator default)	Base	0.9184	0.8581	0: 0.8715 1: 0.7922	0: 0.9537 1: 0.5564	0: 0.9108 1: 0.6537
10- Redes Neuronales	StandardBase	0.8999	0.8483	0: 0.8653 1: 0.7639	0: 0.9479 1: 0.5330	0: 0.9047 1: 0.6279
11- Voting	Base	0.9106	0.8580	0: 0.8698 1: 0.7985	0: 0.9562 1: 0.5482	0: 0.9109 1: 0.6501

Conclusiones

En base a lo analizado en los notebooks y los resultados obtenidos en la tabla, recomendamos el uso de un modelo de BaggingClassifier con la siguiente combinación de hiperparámetros, y entrenado sobre el dataset sin preprocesar:

- `base_estimator = DecisionTreeClassifier(max_depth=9)`
- `n_estimators = 11`

Con ese modelo obtuvimos el valor más alto de todos según la métrica AUC-ROC, y una muy buena performance para clasificar muestras, tanto con alto poder adquisitivo como bajo. En comparación con el baseline implementado en la parte 1 de este TP, podemos decir que la mejora fue significativa, alcanzando un accuracy del 86.6%, en comparación con el 84% obtenido inicialmente. Además, tenemos una mejora muy marcada en la predicción de casos de alto poder adquisitivo, que en la primera parte había sido muy pobre.

Queremos destacar una situación que ocurrió durante la elaboración del trabajo práctico y es la alta performance en general de los modelos basados en árboles de clasificación (como pueden ser AdaBoost, Bagging, RandomForest o un árbol mismamente). Esto nos sorprendió, dado que estos modelos suelen ser más simples que los demás, pero aún así resultaron en una fuerte competencia.

Luego, queremos notar que en general los modelos no lograron obtener métricas altas para recall de unos (sin bajar demasiado el recall de ceros), y esta situación influyó notablemente a lo largo de todo el trabajo práctico, tomando el rol de cuello de botella para todos ellos. Véase que el único modelo en la tabla que obtuvo valores altos para esta métrica sacrificó otras métricas como podría ser el recall de ceros y el accuracy general y la precisión para los unos.

Dicho esto, si en lugar de mirar solamente la métrica AUC-ROC tuviéramos que elegir un modelo con el solo objetivo de minimizar la cantidad de falsos positivos, la recomendación

sería usar SVM entrenado con el preprocesamiento base y con los siguientes hiperparámetros:

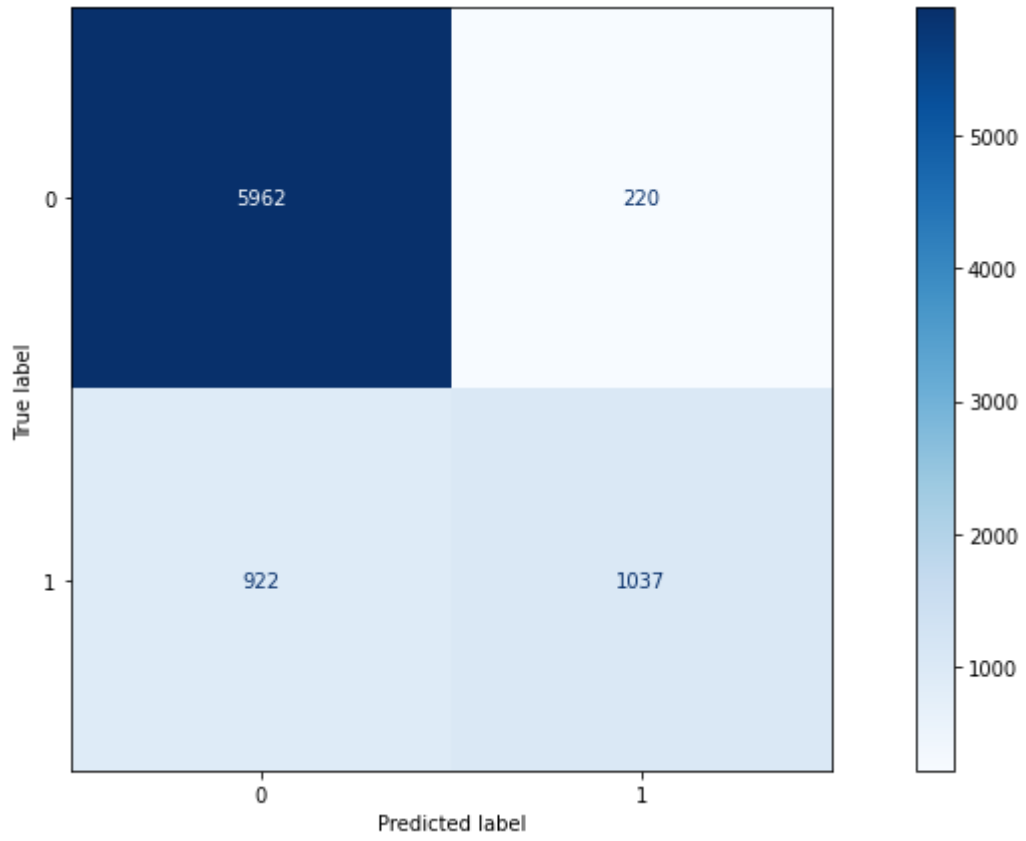
- `C = 100.0`
- `gamma = 0.0001`
- `kernel = 'rbf'`

Cuyas métricas fueron las siguientes:

Modelo	Preprocesamiento	AUC-ROC	Accuracy	Precision	Recall	F1 Score
SVC	Base	0.9053	0.8597	0: 0.8661 1: 0.8250	0: 0.9644 1: 0.5294	0: 0.9126 1: 0.6449

Vemos aquí que la precisión de unos es alta y su recall es el común para los modelos que entrenamos en el trabajo.

Podemos ver ahora la matriz de confusión para corroborar lo pedido:



Y vemos que la esquina superior derecha tiene pocas instancias.

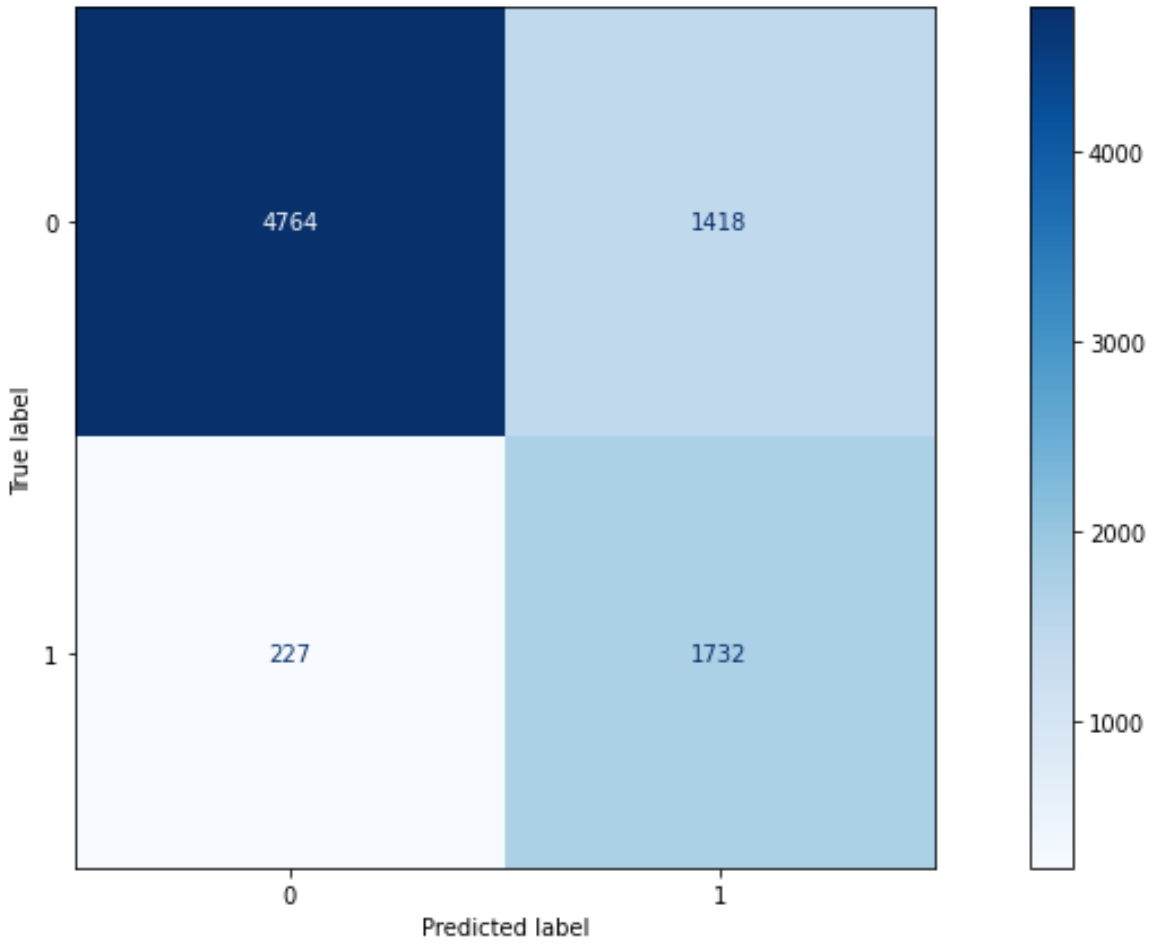
Si en cambio quisiéramos obtener una lista de aquellos individuos con un potencial poder adquisitivo alto, sin importar la cantidad de falsos positivos que allí encontremos, recomendaríamos usar un RandomForest entrenado con el preprocessing equilibrado con los siguientes hiperparámetros:

- `n_estimators = 1001`
- `criterion = 'entropy'`
- `max_depth = 15`
- `max_features = 8`

Donde obtuvimos las siguientes métricas:

Modelo	Preprocesamiento	AUC-ROC	Accuracy	Precision	Recall	F1 Score
RandomForest	Equilibrado	0.9132	0.7979	0: 0.9545 1: 0.5498	0: 0.7706 1: 0.8841	0: 0.8528 1: 0.6780

Nuevamente, para comprobar lo pedido presentamos la matriz de confusión para dicho modelo:



Vemos que la celda inferior izquierda posee pocas instancias y que la diagonal de la matriz posee la gran parte de las instancias (eso significa que sigue haciendo un buen trabajo de predicción).

Comparación parte 1 vs parte 2

Observamos una mejora notable de las métricas para los modelos de la parte 2 que, dicho sea de paso, fueron testeados sobre datos de test, mientras que en la parte 1 no.

Las métricas obtenidas para la primera parte del trabajo son:

Modelo	Preprocesamiento	Accuracy	Precision	Recall	F1 Score
Baseline	Sin preprocessing	0.8446	0: 0.8587 1: 0.7695	0: 0.9519 1: 0.5062	0: 0.9029 1: 0.6107

Vemos que nuestro clasificador supera por 2% de accuracy a este modelo, y además todas las demás métricas son superiores.

De todas formas nos sorprende que nuestro modelo que nació de la simple observación de los datos mediante gráficos aún así supere a gran parte de los modelos realizados en la elaboración de la parte 2.