.UBAfiuba
FACULTAD DE INGENIERÍA

Trabajo Práctico 2

[75.06/95.58] Organización de Datos
Cátedra Collinet
Primer cuatrimestre de 2021

Alumnos	Padrón	Correo electrónico
Craviotto Roitbarg, Mateo Exequiel	106255	macraviotto@fi.uba.ar
Gómez, Joaquín	103735	joagomez@fi.uba.ar

Índice

1. Introducción **2**

2. Objetivos **2**

3. Archivos Presentados **2**

4. Tablas **3**
4.1. Tabla de preprocesamientos 3
4.2. Tabla de métricas 3

5. Conclusiones **3**

1. Introducción

En el presente trabajo se propone realizar una expansión del análisis realizado a los datos recolectados para la primera parte.. Gracias al éxito logrado en la primera campaña la organización tiene más confianza y está ansiosa por probar las avanzadas técnicas de inteligencia artificial.

2. Objetivos

Los objetivos del trabajo práctico son:

- Entender los diferentes modelos de machine learning vistos a lo largo de la cursada
- Entender las diferentes métricas utilizadas para medir la efectividad de los modelos
- Entender y aprender diferente formas de preprocesar datos.

3. Archivos Presentados

Se presentan los siguientes notebooks, cada uno para un modelo o ensamble distinto:

- ArbolDeDecision.ipynb
- Boosting.ipynb
- KNN.ipynb
- NaiveBayes.ipynb
- RandomForest.ipynb
- RedesNeuronales.ipynb
- Stacking.ipynb
- SVM.ipynb
- Voting.ipynb

Además agregamos los archivos:

- Baseline.ipynb : Contiene el baseline del TP1 con las métricas usadas en este trabajo.
- funciones_auxiliares.py : Contiene funciones que se utilizan en todos los notebooks.
- preprocessing.py : Contiene los preprocesamientos utilizados en los distintos notebooks.
- requirements.txt : Contiene los requerimientos para correr los notebooks.
- Carpeta predicciones: Habra un csv por cada modelo o ensamble con la prediccion de nuestros modelos sobre el dataset de predicciones nuevo.

4. Tablas

4.1. Tabla de preprocesamientos

Preprocesamiento	Explicación	Nombre Función
dividirDataset	Se encarga de separar los datos que se van a utilizar en los modelos de la variable target.	dividir_dataset
preparar dataset	Se encarga de aplicar el feature engineering visto en el TP1.	preparar_dataset
One Hot Encoding	Se encarga de aplicar One Hot Encoding sobre las variables categoricas.	aplicar_one_hot_encoding
Normalizacion	Se encarga de normalizar los valores de los features numéricos.	normalizar_datos
Categóricas	Se queda con las variables categóricas	traer_variables_categoricas
Discretas	Se queda con las features numéricas discretas	traer_variables_discretas
Numéricas	Se queda con las features numéricas	traer_variables_numericas
Expansión	Se encarga de crear nuevas features a partir del dataset original, mediante clustering.	expandir_dataset
DF Predicciones	Llama a a preparar dataset, elimina la columna 'representatividad_poblacional' y separa los ids de cada registro.	preparar_df_predicciones

Aclaración: En los notebooks, al hacer el split del dataset se lo hace estratificado, ya que se trata de un problema desbalanceado respecto a la variable a predecir, por lo que se quieren mantener las proporciones originales. De la misma manera, se utiliza Stratified K-Folds cross validation para la búsqueda de hiperparámetros.

4.2. Tabla de métricas

Presentamos la tabla de métricas ordenadas segun AUC-ROC de forma descendente. A todos los modelos se les aplico el procesamiento 'preparar dataset' en el que se aplica el feature engineering visto el TP1. Además todos los metodos tienen aplicado one hot encoding, por lo que no se va a reflejar en la tabla

Modelo	Preprocesamiento	AUC-ROC	Accuracy	Precision	Recall	F1-Score
Boosting	Expansión	0.925	0.871	0.793	0.626	0.700
Stacking	Expansión y normalización	0.917	0.864	0.780	0.606	0.682
Random Forest	Expansión	0.915	0.861	0.793	0.572	0.664
Voting	Expansión y normalización	0.906	0.847	0.742	0.560	0.638
Redes Neuronales	Expansión	0.904	0.847	0.730	0.581	0.647
SVM	Expansión y normalización	0.896	0.842	0.738	0.531	0.618
KNN	Expansión y normalización	0.889	0.838	0.700	0.574	0.631
Naive Bayes	Catégoricas y numéricas	0.873	0.822	0.637	0.609	0.622
Arbol de decisión	—	0.847	0.814	0.709	0.389	0.502
Baseline	—	0.592	0.800	0.908	0.191	0.315

5. Conclusiones

Nuestras conclusiones del trabajo práctico comienzan por observar que la mayoría de los modelos tuvo un buen score en la métrica solicitada (AUC-ROC). Es decir, muchos de los modelos tuvieron resultados por encima del 0.80. Esto es una buena señal, ya que indica que podríamos

recomendar casi cualquiera de nuestros modelos a la agencia FIUFIP.

Sin embargo, observando más detalladamente cada modelo y su respectivo score, los que aparecen como destacados son Voting, Stacking, Boosting y RandomForest. Se podría decir que estos resultados eran los esperados, pues estos primeros tres son ensambles de los modelos que mejores métricas proporcionaron, y el RandomForest es un ensamble de árboles de decisión, que produjo un score bastante alto. También podemos ver que ninguno de los modelos en general logró un muy alto f1-score para las instancias con alto valor adquisitivo. Es decir, las métricas precision y recall para esas instancias fueron bajas en general. Por lo tanto, a los modelos le cuesta más clasificar el alto valor adquisitivo dentro del dataset, más que el bajo. Otro factor a tener en cuenta es que la mayoría de los modelos logró una pequeña mejora al utilizar el dataset expandido mediante k-means, en comparación al dataset original. Es decir, los nuevos features generados por clustering se tuvieron en cuenta por los modelos y así se logró un mejor rendimiento, en términos generales.

Al tener que recomendar un único modelo, elegimos recomendar Boosting, pues este logró un AUC score de 0.925. Como explicamos, este es un ensamble, por lo que es esperable que resulte el mejor.

Comparando Boosting con el baseline hecho en el informe anterior, podemos notar que se logró un accuracy superior. Concretamente, viendo el informe de Baseline.ipynb, se puede ver que se pasó de un 0.800 de accuracy en el baseline a un 0.871 en este modelo. Como el accuracy no suele ser la mejor métrica para comparar modelos, decidimos predecir sobre el dataset original con el baseline del TP1 (ver Baseline.ipynb). Allí, podemos ver que el baseline tuvo un AUC-ROC muy bajo, para el cuál se debe tener en cuenta que no se usó predict proba, como se explica en el notebook. Por otra parte, la precision sobre las instancias de alto valor adquisitivo fue la más alta de todas con diferencia, con un valor de 0.908. Sin embargo, el problema del baseline está en el recall, donde se tiene un 0.191, extremadamente bajo. Esto puede haber ocurrido porque el baseline tuvo una tendencia a predecir la mayoría de las instancias como de bajo valor adquisitivo. A causa de esto, se obtiene un F1-Score muy bajo (0.315), en comparación a nuestro mejor modelo elegido, Boosting que obtuvo un 0.700. Allí es donde se ve la mejoría más claramente. Se debe tener en cuenta que el baseline hecho previamente fue evaluado sobre el dataset en su totalidad, sin hacer particiones en entrenamiento y validation.

Pasando al modelo que realice una predicción con la menor cantidad de falsos positivos, creemos que la elección correcta sería RandomForest. No solo proporciona la menor cantidad de falsos positivos, con un recall de 0.953 en el dataset expandido, sino que además tiene un muy buen AUC score, 0.915. Viéndolo de otra forma, se quiere la mayor precision sobre los que tienen alto valor adquisitivo. Esto confirma que ese modelo es Random Forest, pues tiene una precision de 0.793, como se puede ver en la tabla.

Por otra parte, si se necesita tener una lista de todos los que potencialmente son de valor adquisitivo, sin preocuparse demasiado si ahí se incluye a personas que realmente no tienen alto valor adquisitivo, recomendamos utilizar Boosting. Este modelo es el que tiene el recall más alto para las instancias que tienen alto valor adquisitivo (0.626).