Trabajo Práctico - Segunda entrega
Organización de Datos 75.06/95.58
Cátedra Collinet

Alumnos: Rodriguez, Alejo y Vergara, Ariel

Padrones: 99869 y 97010

Corrector: Collinet, Jorge

Reporte

En este trabajo práctico los pre procesamientos no se realizaron mediante una función, sino que para cada uno de ellos se creó una clase que hereda de BaseEstimator y TransformerMixin (ambas clases de sklearn). Se hizo esto para poder utilizar el preprocesamiento como el primer paso de un Pipeline de sklearn, de manera tal que el siguiente y último paso sea el modelo. De esta manera, al hacer K Fold Cross Validation y Grid Search CV, en cada paso se puede fitear tanto el preprocesamiento (en caso de que utilice algún valor para rellenar datos o un LabelEncoder por ejemplo) como el modelo con los datos de entrenamiento, para luego realizar la predicción sobre el set de validación correspondiente. A continuación se encuentra la tabla con los pre procesamientos utilizados:

Preprocesamientos		
Nombre	**Descripción**	**Clase**
PreprocessingLE	● Agrega una columna fila_isna (que indica si el valor de la fila es o no NaN). ● Elimina columnas sin información valiosa[1]. ● Agrega una columna nombre_sede_isna (que indica si el valor de la fila es o no NaN) y rellena los NaN con la moda. ● Encodea variables categóricas mediante LabelEncoding[2]. ● Agrega una columna edad_isna (que indica si el valor de la fila es o no NaN). ● Completa los missing values de la columna edad con la media. ● Convierte en bins los valores de las columnas edad y precio_ticket.	PreprocessingLE
PreprocessingOHE	● Agrega una columna fila_isna que indica si el valor de la fila es o no NaN. ● Elimina columnas sin información valiosa[1]. ● Encodea variables categóricas mediante OneHotEncoding[2]. ● Completa los missing values de la columna edad con la media. ● Convierte en bins los valores de las columnas edad y precio_ticket.	PreprocessingOHE
PreprocessingGaussianNB1	● Elimina columnas sin información valiosa[1] y con valores categóricos[2]. ● Se agrega la columna acompaniantes y se eliminan las columnas parientes y amigos. ● Completa los missing values de la columna edad con la media.	PreprocessingGaussianNB1
PreprocessingCategoricalNB1	● Elimina columnas sin información valiosa[1] y con valores continuos o discretos[4]. ● Encodea variables categóricas mediante LabelEncoding[2].	PreprocessingCategoricalNB1
PreprocessingSE	● Agrega una columna fila_isna que indica	PreprocessingSE

	si el valor de la fila es o no NaN. ● Elimina columnas sin información valiosa[1]. ● Encodea variables categóricas mediante OneHotEncoding[2]. ● Completa los missing values de la columna edad con la media. ● Escala los valores numéricos[3] a media 0 y desvío estándar 1 con StandardScaler.	
PreprocessingSE_2	● Agrega una columna fila_isna que indica si el valor de la fila es o no NaN. ● Elimina columnas sin información valiosa[1]. ● Encodea variables categóricas mediante LabelEncoding[2]. ● Completa los missing values de la columna edad con la media. ● Escala los valores numéricos[3] a media 0 y desvío estándar 1 con StandardScaler.	PreprocesingSE_2
PreprocessingXGBoost	● Agrega una columna fila_isna que indica si el valor de la fila es o no NaN. ● Agrega una columna edad_isna que indica si el valor de la edad es o no NaN. ● Elimina columnas sin información valiosa[1]. ● Encodea variables categóricas mediante OneHotEncoding[2]. ● Completa los missing values de la columna edad con la media. ● Convierte en bins los valores de las columnas edad y precio_ticket.	PreprocesingXGBoost
PreprocessingXGBoost2	● Agrega una columna fila_isna que indica si el valor de la fila es o no NaN. ● Agrega una columna edad_isna que indica si el valor de la edad es o no NaN. ● Elimina columnas sin información valiosa[1]. ● Encodea variables categóricas mediante OneHotEncoding[2]. ● No completa missing values.	PreprocessingXGBoost2
PreprocessingCategorilaNB2	● Elimina columnas sin información valiosa[1] y con valores continuos o discretos[4]. ● Encodea variables categóricas mediante LabelEncoding[2]. ● Transforma en bins los valores de las columnas edad y precio_ticket.	PreprocessingCategoricalNB2

Aclaraciones:

- [1] fila, id_usuario y id_ticket
- [2] genero, nombre_sala, tipo_de_sala
- [3] edad, precio_ticket, parientes y amigos
- [3] edad, precio_ticket, parientes y amigos

Los preprocesamientos marcados (PreprocessingSE_2, PreprocessingXGBoost, PreprocessingXGBoost2, PreprocessingCategoricalNB2) fueron descartados luego de comparar las métricas obtenidas mediante Cross Validation con las de otros preprocesamientos, siempre utilizando los mismos modelos para cada uno.

Queda pendiente un refactor para que las diferentes transformaciones que se hacen en cada una de las clases de la tabla sean realizadas por diferentes Transformers. De esta forma, en lugar de que cada preprocesamiento sea una clase, se podría hacer que sea una función que devuelve un Pipeline de preprocesamiento con todos los steps correspondientes al mismo (ej: transformar la edad, eliminar las columnas que no aportan información, etc)

Para comparar los modelos entre sí se realizó un K Fold Cross Validation de 8 folds sobre todos los datos de entrenamiento. Una vez seleccionado el modelo, se dividieron los datos en train y test (con un test size de 15% debido a los pocos datos con los que se cuenta), se entrenó el modelo correspondiente con el train y se realizó la predicción sobre el test. Para todos los modelos, se utilizó el mismo train_test_split y para todos los Cross Validation, los mismos folds.

A partir de la predicción sobre el split de test, se obtuvieron las métricas presentadas en la siguiente tabla:

Modelos						
Nombre	**Preprocesamiento**	**Métricas**				
		AUC-ROC	**Accuracy**	**Precision**	**Recall**	**F1 Score**
1-ArbolDeDecision	PreprocessingLE	0.9084	0.8595	0.8536	0.7608	0.8045
2-RandomForest	PreprocessingLE	0.9089	0.8760	0.9428	0.7173	0.8148
3-NaiveBayes	PreprocessingGaussianNB1 - PreprocessingCategoricalNB1	0.8759	0.8181	0.8333	0.6521	0.7317
4-XGBoost	PreprocessinLE	0.8971	0.8677	0.8750	0.7608	0.8139
5-SVM	PreprocessingSE	0.8840	0.8429	0.8139	0.7607	0.7865
6-KNN	PreprocessingSE	0.8708	0.8347	0.8611	0.6739	0.7560
8-RedNeuronal	PreprocessingSE	0.8718	0.8347	0.8611	0.6739	0.7560
9-Ensambles	PreprocessingLE - PreprocessingSE	0.9127	0.8595	0.8372	0.7826	0.8089

En NaiveBayes se indican dos pre procesamientos debido a que el modelo presentado consiste en un ensamble Stacking con dos Naive Bayes, un CategoricalNB y un GaussianNB. El estimador final del ensamble corresponde a otro GaussianNB.

Por otro lado, el modelo correspondiente a Ensambles es, al igual que al anterior, un ensamble de tipo Stacking que utiliza como estimador final un GaussianNB. Los modelos utilizados en la primera capa son los correspondientes a los mejores resultados de SVM, XGBoost y RandomForest.

Conclusión

Para la entrega final, se elige el modelo correspondiente a 9-Ensambles, el cual obtuvo el mejor AUC-ROC entre todos los modelos entrenados. Este modelo no solo es un ensamble, sino que dos de los modelos que utiliza para realizarlo también lo son, siendo estos XGBoost y Random Forest. Al ser un ensamble, cada uno de los modelos que lo conforma sobre ajusta a su manera a los datos de entrenamiento, lo cual permite generar un mejor clasificador al combinarlos.

En comparación al baseline de la primera entrega, el modelo final obtenido es más complejo y puede inferir relaciones a partir de los datos que no podemos detectar a simple vista (como se hizo en la primera parte del Trabajo Práctico). El baseline, además de ser más simple, se construyó a partir de todos los datos provistos y se evaluó sobre los mismos. Por el contrario, el ensamble elegido se entrenó sobre una parte de nuestros datos elegida al azar y se validó contra la parte restante.