

Métodos de ensamblaje

Juliho Castillo Colmenares

¡Bienvenidos a la actividad práctica del módulo!

Antes de empezar

El objetivo de esta actividad es que efectúes un análisis de clasificación mediante Árboles de decisión que permita desarrollar un modelo predictivo basado en distintas métricas aplicadas a una base de datos grande.

Entregable: Un Jupyter Notebook (archivo de extensión .ipynb), archivo PDF y capturas de pantalla en el espacio de respuesta que muestren tanto el código desarrollado como la solución al problema planteado, incluyendo los comentarios que sean pertinentes a las preguntas que se plantean. Entregas sin estos elementos no serán calificadas.

¡Buena suerte!

Problema: Imagina que eres un investigador médico que recopila datos para un estudio. Has recopilado datos sobre un conjunto de pacientes, todos ellos con la misma enfermedad. Durante su tratamiento, cada paciente ha respondido a uno de los 5 medicamentos, el fármaco A, el fármaco B, el fármaco C, el fármaco X y el Y.

Parte de tu trabajo consiste en construir un modelo para averiguar qué medicamento podría ser apropiado para un futuro paciente con la misma enfermedad. Los conjuntos de características de este conjunto de datos son la edad, el sexo, la presión arterial y el colesterol de los pacientes, y el objetivo es estudiar el fármaco al que respondió cada paciente.

Se trata de un ejemplo de clasificador binario, y se puede utilizar la parte de entrenamiento del conjunto de datos para construir un árbol de decisión, y luego utilizarlo para predecir la clase de un paciente desconocido, o para prescribirlo a un nuevo paciente.

Variable a pronosticar:

Age	Sex	BP	Cholesterol	Na_to_K	Drug
23	F	HIGH	HIGH	25.355	drugY
47	M	LOW	HIGH	13.093	drugC
47	M	LOW	HIGH	10.114	drugC
28	F	NORMAL	HIGH	7.798	drugX
61	F	LOW	HIGH	18.043	drugY
22	F	NORMAL	HIGH	8.607	drugX
49	F	NORMAL	HIGH	16.275	drugY
41	M	LOW	HIGH	11.037	drugC

Paso a paso:

```
In [ ]: import warnings
warnings.filterwarnings('ignore')
```

Paso 1

Cargue la base de datos "drugs.csv" en Python e investigue cómo convertir las variables predictoras cualitativas de esta base a una escala numérica mediante la instrucción "preprocessing.LabelEncoder()". Por ejemplo, si una variable tiene 3 posibles categorías, deberá cambiar sus resultados a 0, 1 o 2.

```
In [ ]: import pandas as pd

df = pd.read_csv("drugs.csv")
df.head()
```

```
Out[ ]:   Age  Sex    BP  Cholesterol  Na_to_K  Drug
0    23   F   HIGH         HIGH    25.355  drugY
1    47   M   LOW          HIGH    13.093  drugC
2    47   M   LOW          HIGH    10.114  drugC
3    28   F  NORMAL         HIGH     7.798  drugX
4    61   F   LOW          HIGH    18.043  drugY
```

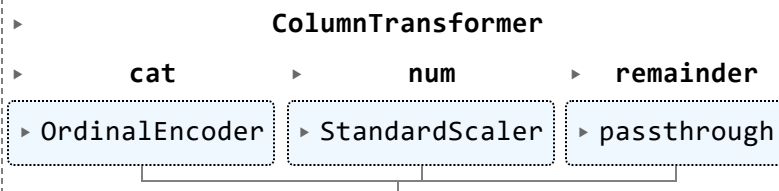
```
In [ ]: X = df.drop("Drug", axis=1)
        y = df["Drug"]
```

```
In [ ]: from preprocess import create_preprocessing_pipeline

preprocess_pipeline = create_preprocessing_pipeline(
    categorical_variables=["Sex", "BP", "Cholesterol"],
    numerical_variables=["Age", "Na_to_K"]
)
```

```
In [ ]: X_preprocessed = preprocess_pipeline.fit_transform(X)
```

```
In [ ]: preprocessor = preprocess_pipeline.named_steps['preprocessor']
preprocessor
```

```
Out[ ]: 
        The diagram shows a dashed box labeled 'ColumnTransformer'. Inside, there are three columns: 'cat', 'num', and 'remainder'. Under 'cat' is a box for 'OrdinalEncoder'. Under 'num' is a box for 'StandardScaler'. Under 'remainder' is a box for 'passthrough'. Arrows point from each of these three boxes to a common point below them, which then points to the bottom of the 'ColumnTransformer' box.
```

```
In [ ]: feature_names = preprocessor.get_feature_names_out()
feature_names
```

```
Out[ ]: array(['cat__Sex', 'cat__BP', 'cat__Cholesterol', 'num__Age',
              'num__Na_to_K'], dtype=object)
```

```
In [ ]: ordinal_encoder = preprocessor.transformers_[0][1]
        categories = ordinal_encoder.categories_
        categories
```

```
Out[ ]: [array(['F', 'M'], dtype=object),
         array(['HIGH', 'LOW', 'NORMAL'], dtype=object),
         array(['HIGH', 'NORMAL'], dtype=object)]
```

Paso 2

Use los diversos métodos de ensambles vistos en este módulo para encontrar un algoritmo óptimo de clasificación. Explique cuál sería su recomendación para este caso.

```
In [ ]: from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier, AdaBoostClassifier
        from xgboost import XGBClassifier
        from sklearn.model_selection import GridSearchCV
        from sklearn.pipeline import Pipeline

        # Definir el espacio de búsqueda de parámetros
        param_grid = [

            {
                'classifier': [GradientBoostingClassifier()],
                'classifier__n_estimators': [50, 100, 200],
                'classifier__learning_rate': [0.01, 0.1, 0.2],
                'classifier__random_state': [42]
            },
            {
```

```

        'classifier': [RandomForestClassifier()],
        'classifier__n_estimators': [50, 100, 200],
        'classifier__criterion': ['gini', 'entropy'],
        'classifier__max_depth': [1, 2, 3, 4, 5, None],
        'classifier__random_state': [42]
    },
    {
        'classifier': [AdaBoostClassifier()],
        'classifier__n_estimators': [50, 100, 200],
        'classifier__learning_rate': [0.01, 0.1, 0.2],
        'classifier__random_state': [42]
    },
    {
        'classifier': [XGBClassifier()],
        'classifier__n_estimators': [50, 100, 200],
        'classifier__learning_rate': [0.01, 0.1, 0.2],
        'classifier__max_depth': [1, 2, 3, 4, 5],
        'classifier__random_state': [42]
    }
]

# Crear un pipeline con un clasificador como paso
pipeline = Pipeline([
    ('classifier', XGBClassifier())
])

# Configurar GridSearchCV
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy')

# Ajustar GridSearchCV
grid_search.fit(X_preprocessed, y)

# Mejores parámetros y puntuación
print("Mejores parámetros:", grid_search.best_params_)
print("Mejor puntuación:", grid_search.best_score_)

```

Mejores parámetros: {'classifier': RandomForestClassifier(n_estimators=50, random_state=42), 'classifier__criterion': 'gini', 'classifier__max_depth': None, 'classifier__n_estimators': 50, 'classifier__random_state': 42}

Mejor puntuación: 0.985

```

In [ ]: from sklearn.metrics import accuracy_score

# Extraer los mejores parámetros
best_params = grid_search.best_params_
try:
    best_classifier = best_params.pop('classifier')
except:
    print("classifier not defined or already removed")

# Filtrar los parámetros para eliminar el prefijo 'classifier__'
filtered_params = {key.replace('classifier__', ''): value for key, value in best_params.items()}

# Inicializar el modelo con los mejores parámetros
# best_model = DecisionTreeClassifier(**filtered_params)
best_model = best_classifier.set_params(**filtered_params)

# Ajustar el modelo con los datos de entrenamiento
best_model.fit(X_preprocessed, y)

```

```
# Usar best_model para hacer predicciones
predictions = best_model.predict(X_preprocessed)

# Calcular la precisión del modelo
accuracy = accuracy_score(y, predictions)
print("Precisión del modelo:", accuracy)
```

Precisión del modelo: 1.0

Paso 3

¿Mejoró su poder predictivo al comparar sus resultados con los obtenidos mediante el árbol de decisión aplicado en la actividad del módulo previo? Explique.

Recordemos los mejores parámetros obtenidos en la actividad anterior, donde utilizamos el modelo `DecisionTreeClassifier`

Mejores parámetros: {'criterion': 'gini', 'max_depth': 4, 'random_state': 42}

Utilizando los resultado obtenidos por `GridSearchCV`, la exactitud se mantuvo igual. Sin embargo, al utilizar todo el conjunto de datos podemos observa que está métrica es del 100%, lo que nos indica un sobreajuste.