Práctica: Configuración manual de hiperparámetros

Nombre del estudiante: Jesús Alberto Aréchiga Carrillo

El objetivo de esta práctica es realizar una búsqueda exhaustiva para encontrar la mejor combinación de parámetros que maximice el rendimiento de generalización del modelo.

Aquí usamos un pequeño subconjunto del conjunto de datos Adult Census para hacer que el código se ejecute más rápido. Una vez que tu código funcione en el subconjunto pequeño, intenta cambiar `train_size` a un valor más grande (p. ej., 0.8 para 80% en lugar de 20%).

```
import pandas as pd

from sklearn.model_selection import train_test_split

adult_census = pd.read_csv("adult_census.csv")

target_name = "class"
target = adult_census[target_name]
data = adult_census.drop(columns=[target_name, "education-num"])

data_train, data_test, target_train, target_test = train_test_split(
    data, target, train_size=0.2, random_state=42
)
```

```
from sklearn.compose import ColumnTransformer
from sklearn.compose import make_column_selector as selector
from sklearn.preprocessing import OrdinalEncoder

categorical_preprocessor = OrdinalEncoder(
    handle_unknown="use_encoded_value", unknown_value=-1
)
preprocessor = ColumnTransformer(
    [
        (
            "cat_preprocessor",
            categorical_preprocessor,
            selector(dtype_include=object),
        )
    ],
    remainder="passthrough",
)

from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.pipeline import Pipeline

model = Pipeline(
    [
        ("preprocessor", preprocessor),
        ("classifier", HistGradientBoostingClassifier(random_state=42)),
    ]
)
```

Usa el modelo definido anteriormente (llamado `model`) y, utilizando dos bucles `for` anidados, realiza una búsqueda de las mejores combinaciones de los parámetros `learning_rate` y `max_leaf_nodes`. En este sentido, tienes que entrenar y probar el modelo estableciendo los parámetros. La evaluación del modelo debe realizarse usando `cross_val_score` en el conjunto de entrenamiento. Utiliza la siguiente búsqueda de parámetros:

- `learning_rate` para los valores 0.01, 0.1, 1 y 10. Este parámetro controla la capacidad de un nuevo árbol para corregir el error de la secuencia anterior de árboles.
- `max_leaf_nodes` para los valores 3, 10, 30. Este parámetro controla la profundidad de cada árbol.

```
# Escribe tu código aquí.
from sklearn.model_selection import cross_val_score
import numpy as np
learning_rate = [0.01, 0.1, 1, 10]
```

```
max_leaf_nodes = [3, 10, 30]
best_score = 0
best_params = {}
for lr in learning_rate:
    for mln in max_leaf_nodes:
        model.set_params(classifier__learning_rate=lr, classifier__max_leaf_nodes=mln)
        scores = cross_val_score(model, data_train, target_train, cv=2)
        mean_score = scores.mean()

        print(f"learning_rate={lr}, max_leaf_nodes={mln}: score={mean_score:.4f}")

        # Actualizar si encontramos mejores parámetros
        if mean_score > best_score:
            best_score = mean_score
            best_params = {"learning_rate": lr, "max_leaf_nodes": mln}

print(f"\nMejores parámetros: {best_params}")
print(f"Mejor score en validación cruzada: {best_score:.4f}")
```

```
learning_rate=0.01, max_leaf_nodes=3: score=0.7892
learning_rate=0.01, max_leaf_nodes=10: score=0.8134
learning_rate=0.01, max_leaf_nodes=30: score=0.8417
learning_rate=0.1, max_leaf_nodes=3: score=0.8475
learning_rate=0.1, max_leaf_nodes=10: score=0.8584
learning_rate=0.1, max_leaf_nodes=30: score=0.8533
learning_rate=1, max_leaf_nodes=3: score=0.8517
learning_rate=1, max_leaf_nodes=10: score=0.8259
learning_rate=1, max_leaf_nodes=30: score=0.7985
learning_rate=10, max_leaf_nodes=3: score=0.2883
learning_rate=10, max_leaf_nodes=10: score=0.4814
learning_rate=10, max_leaf_nodes=30: score=0.6991

Mejores parámetros: {'learning_rate': 0.1, 'max_leaf_nodes': 10}
Mejor score en validación cruzada: 0.8584
```

Ahora usa el conjunto de prueba (test set) para puntuar el modelo usando los mejores parámetros que encontramos mediante validación cruzada. Tendrás que reajustar (refit) el modelo sobre el conjunto de entrenamiento completo.

```
# Escribe tu código aquí.
# Reentrenar con los mejores parámetros en todo el conjunto de entrenamiento
model.set_params(
    classifier__learning_rate=best_params["learning_rate"],
    classifier__max_leaf_nodes=best_params["max_leaf_nodes"]
)

# Ajustar el modelo al conjunto de entrenamiento completo
model.fit(data_train, target_train)

# Evaluar en el conjunto de test
test_score = model.score(data_test, target_test)
print(f"Score en el conjunto de test: {test_score:.4f}")
```

```
Score en el conjunto de test: 0.8691
```