UNIVERSIDAD TECNOLÓGICA DE QUERÉTARO

Nombre del trabajo:

2a. Evaluación DE

Nombre de la materia:

Extracción de conocimiento

INGENIERÍA EN GESTIÓN Y DESARROLLO DE SOFTWARE

Presenta:

Santiago Garcia Abel Alejandro

Matricula:

2021171016

Profesor

Filiberto Ruiz Hernández

Santiago de Querétaro, Qro. 30 de noviembre 2023

DE

https://github.com/alexjamesmx/edc_2
Nota: Gráfica personalizada e interpretación de resultados se encuentran en el código de cada ejercicio.

1. Basado en el conjunto de datos "diabetes_indiana.csv". usando otro algoritmo de clasificación.

Justificación del algoritmo
RandomForest es un algoritmo de aprendizaje en conjunto que combina múltiples modelos de árboles de decisión, cada árbol se entrena de manera independiente en diferentes subconjuntos de datos y, luego, las predicciones se combinan para mejorar la precisión y generalización del modelo. Tiene la capacidad de manejar overfitting, un problema común en árboles de decisión individuales.

RandomForest es paralelizable, lo que significa que puede beneficiarse de la capacidad de cómputo en paralelo. Esto lo hace eficiente en términos de tiempo de entrenamiento, especialmente en comparación con algoritmos más complejos.

RandomForestClassifier es una opción robusta y versátil para problemas de clasificación, especialmente cuando se buscan buen rendimiento predictivo y capacidad de manejar conjuntos de datos diversos.

Descripción de los modelos

```
# usando otro algoritmo de clasificación
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# crear modelo
modelo_rf = RandomForestClassifier(random_state=42, class_weight='balanced')

# entrenar modelo
modelo_rf.fit(X_train_scaled, y_train)
```

Evaluación del modelo
Similar a logistic regression, el modelo fue optimizado para encontrar los mejores hiper parámetros por medio de una búsqueda aleatoria en rejilla, la cual permite probar con distintos parámetros de forma random dentro de límites definidos.

El performance general mejoró, ya que este era el accuracy original:
0.7402597402597403 vs el optimizado 0.7207792207792207
junto con valores más estables en f-score, rondando entre 80 de puntuación.

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

# Definir el espacio de búsqueda
param_dist = {
    'n_estimators': randint(10, 200),
    'max_depth': [None] + list(randint(5, 50).rvs(5)),
    'min_samples_split': randint(2, 20),
    'min_samples_leaf': randint(1, 20),
}

# Create a random search object
randomized_search = RandomizedSearchCV(
    estimator=RandomForestClassifier(random_state=42, class_weight='balanced'),
    param_distributions=param_dist,
    n_iter=10,
    cv=5,
    scoring='accuracy',
    random_state=42,
    n_jobs=-1  # Use parallel processing
)

# Entrene el modelo en los datos de entrenamiento escalados
randomized_search.fit(X_train_scaled, y_train)

print("Best Parameters:", randomized_search.best_params_)

# Usar el mejor modelo
best_model = randomized_search.best_estimator_
```