Campus:
Monterrey

Inteligencia Artificial Avanzada para la Ciencia de Datos (Gpo 102)

Curso:
TC3006C.102

Portafolio Análisis

Estudiante:

Jose David de la Garza Salas | A00834760

Lugar y Fecha:
Monterrey, Nuevo León
07/09/2024

1. Introducción

En este portafolio se desarrolla el proceso de analisis y evaluacion de un modelo que machine learning, Random Forest, junto al set de datos de Astronomical Data, acerca de la información de distintas estrellas y como las categorizamos dependiendo de su características esenciales, como este reporte solo se enfoca en el diagnóstico de los resultados y porque se decidió haces cambios en el modelo en base a los resultados anteriores se da por sentado la limpieza y el procesamiento de los datos en el modelo. Conociendo esto se dividió el reporte en tres secciones principales, primero los resultados de predicción de nuestro modelo sin la aplicación de técnicas de regularización ni ajuste de hiperparametros, una vez conociendo estos resultados pasamos a la aplicación de los cambios de optimización y por último vemos una comparativa a los resultados obtenidos con nuestro modelo refinado y los mejores resultados obtenidos.

2. Análisis

2.1 Separación y evaluación del modelo con un conjunto de prueba y un conjunto de validación (Train/Test/Validation).

Para conocer que tan bueno es nuestro modelo debemos poder evaluarlo y para esto tenemos que probar el modelo entrenado con datos reales, para esto debemos hacer una división de nuestro data set en tres partes: train, test y validation.

Una vez entrenado el modelo los resultados de validación fueron:

```
Score de exactitud en validación: 1.0
Score de precisión en validación: 1.0
Score f1 en validación: 1.0
Score de recall en validación: 1.0
```

En comparación con nuestro conjunto de prueba podemos notar una diferencia en las mediciones de exactitud. precisión y recall:

```
Score de exactitud en prueba: 0.9772727272727273
Score de precisión en prueba: 0.9801136363636364
Score f1 en prueba: 0.9772727272727273
Score de recall en prueba: 0.9772727272727273

Matriz de confusión en prueba:
[[5 0 0 0 0 0]
 [0 9 0 0 0 0]
 [0 0 6 0 0 0]
 [0 0 0 7 0 0]
 [0 0 0 1 7 0]
 [0 0 0 0 0 9]]
Reporte de clasificación en prueba:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00         5
         1.0       1.00      1.00      1.00         9
         2.0       1.00      1.00      1.00         6
         3.0       0.88      1.00      0.93         7
         4.0       1.00      0.88      0.93         8
...
   accuracy                           0.98        44
   macro avg       0.98      0.98      0.98        44
weighted avg       0.98      0.98      0.98        44
```

2.2 Diagnóstico y explicación el grado de bias o sesgo: bajo medio alto

Conforme a los resultados de nuestro modelo en el conjunto de validación podemos observar con respecto a la precisión, exactitud y recall los resultados son los mejores esperados o de muy poco error por lo que se considera que el modelo entrenado es de muy bajo sesgo.

2.3 Diagnóstico y explicación el grado de varianza: bajo medio alto

Este código define una función accuracy que evalúa el rendimiento de un modelo de aprendizaje automático tanto en el conjunto de entrenamiento como en el conjunto de prueba.

```
from sklearn.metrics import accuracy_score

def accuracy(y_train, y_pred, x_train, model):
    # Performance en el conjunto de entrenamiento
    y_train_pred = model.predict(x_train)
    train_accuracy = accuracy_score(y_train, y_train_pred)

    # Performance en el conjunto de prueba
    test_accuracy = accuracy_score(y_test, y_pred)

    print(f"Accuracy en el conjunto de entrenamiento:
{train_accuracy:.4f}")
    print(f"Accuracy en el conjunto de prueba: {test_accuracy:.4f}")

    # Evaluar el sesgo y la varianza
    if train_accuracy > test_accuracy:
        print("Hay una posible alta varianza (overfitting).")
    elif train_accuracy < test_accuracy:
        print("Hay un posible alto sesgo (underfitting).")
    else:
        print("El modelo tiene un buen equilibrio entre sesgo y
varianza.")

accuracy(y_train, y_pred, x_train, model)
```

En este modelo:

```
Accuracy en el conjunto de entrenamiento: 1.0000
Accuracy en el conjunto de prueba: 0.9773
Hay una posible alta varianza (overfitting).
```

2.4 Diagnóstico y explicación el nivel de ajuste del modelo: underfitt fitt overfitt

Por el desnivel entre los resultados de el conjunto de prueba y entrenamiento el modelo puede ser mejorado para esto es necesario conocer que ell modelo de **Random Forest** combina múltiples árboles de decisión para mejorar la precisión y reducir la varianza, evitando así el sobreajuste en comparación con un solo árbol.

Hiperparámetros Seleccionados:

- **n_estimators = 100**: Se seleccionan 100 árboles para lograr un equilibrio entre rendimiento y eficiencia computacional. Este número es suficiente para reducir la varianza sin incrementar excesivamente el tiempo de entrenamiento.
- **max_depth = None**: Permite a los árboles crecer hasta su máxima profundidad. Esto permite al modelo capturar todas las relaciones posibles en los datos, lo cual es útil cuando no hay preocupaciones inmediatas de sobreajuste o cuando se espera que los datos tengan patrones complejos.
- **min_samples_split = 2**: Permite que los nodos se dividan si tienen al menos 2 muestras, fomentando que el árbol sea lo más profundo posible y capturando la mayor cantidad de detalles de los datos. Es útil para encontrar relaciones complejas pero puede llevar a sobreajuste si no se controla adecuadamente.
- **min_samples_leaf = 1**: Permite que cada hoja contenga al menos una muestra, lo que maximiza la complejidad de cada árbol. Esto puede ser beneficioso para detectar patrones muy específicos o detalles menores, pero también aumenta el riesgo de sobreajuste.
- **max_features = 'sqrt'**: Selecciona un subconjunto aleatorio de características igual a la raíz cuadrada del número total de características para cada división. Esto introduce diversidad entre los árboles, reduce la correlación y mejora la capacidad de generalización del modelo.
- **random_state = 42**: Fija la semilla del generador de números aleatorios para garantizar la reproducibilidad de los resultados. Esto permite que los experimentos sean consistentes y comparables en diferentes ejecuciones.

Estos hiperparámetros se optimizaron mediante pruebas empíricas y técnicas de búsqueda automatizada, como búsqueda en cuadrícula y aleatoria, con validación cruzada, para maximizar la precisión y generalización del modelo.

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import GridSearchCV



# Definir el espacio de hiperparámetros a explorar

param_grid = {

    'n_estimators': [100, 200, 300],

    'max_depth': [None, 10, 20],

    'min_samples_split': [2, 5, 10],

    'min_samples_leaf': [1, 2, 4],

    'max_features': ['auto', 'sqrt', 'log2']
```

```
}



# Crear el modelo de Random Forest

random_forest = RandomForestClassifier(random_state=42)



# Configurar la búsqueda en rejilla

grid_search = GridSearchCV(estimator=random_forest,
param_grid=param_grid, cv=5, n_jobs=-1, scoring='accuracy')



# Ajustar el modelo a los datos

grid_search.fit(x_train, y_train)



# Obtener los mejores hiperparámetros

best_params = grid_search.best_params_

print("Mejores hiperparámetros encontrados:", best_params)
```

3. Evaluación

Una vez seleccionados los mejores hiperparametros y aplicarlos al modelo podemos volver a cargar los datos y ahora observamos que:

```
Score de exactitud en validación: 1.0
Score de precisión en validación: 1.0
Score f1 en validación: 1.0
Score de recall en validación: 1.0

Score de exactitud en prueba: 1.0
Score de precisión en prueba: 1.0
```

```
Score f1 en prueba: 1.0
Score de recall en prueba: 1.0

Matriz de confusión en prueba:
[[5 0 0 0 0 0]
 [0 9 0 0 0 0]
 [0 0 6 0 0 0]
 [0 0 0 7 0 0]
 [0 0 0 0 8 0]
 [0 0 0 0 0 9]]
Reporte de clasificación en prueba:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00         5
         1.0       1.00      1.00      1.00         9
         2.0       1.00      1.00      1.00         6
         3.0       1.00      1.00      1.00         7
         4.0       1.00      1.00      1.00         8
...
   accuracy                           1.00        44
   macro avg       1.00      1.00      1.00        44
weighted avg       1.00      1.00      1.00        44
```

Ahora si volvemos a comparar su precisión:

```
Accuracy en el conjunto de entrenamiento: 1.0000
Accuracy en el conjunto de prueba: 1.0000
El modelo tiene un buen equilibrio entre sesgo y varianza.
```

Por ultimo hacemos una validación de cruz o cross validation para para evaluar la capacidad de generalización del modelo. Aquí podemos observar una comparación antes y después de agregar los nuevos hiperparametros

```python
scores = cross_val_score(model_op, x, y, cv=5)
print(f'Cross-Validation Scores: {scores}')
print(f'Average Cross-Validation Score: {scores.mean()}')
```

Resultados del primer modelo(sin optimizar):

```
Cross-Validation Scores: [1.      0.90909091 1.       1.       1.     ]
Average Cross-Validation Score: 0.9818181818181818
```

Resultados del segundo modelo:

```
Cross-Validation Scores: [1.      0.95454545 1.       1.       1.     ]
Average Cross-Validation Score: 0.990909090909091
```

Los Cross-Validation Scores y el Average Cross-Validation Score indican que el modelo tiene un desempeño excepcionalmente bueno en diferentes subconjuntos de datos, con una precisión promedio de aproximadamente el 99,0 %. Esto sugiere que el modelo es sólido y se generaliza bien a los datos.