Análisis y Reporte sobre el desempeño del modelo

Elisa María Bonilla Martín
A01028576

Instituto Tecnológico de Estudios Superiores de Monterrey
Campus Estado de México

1 Introducción

Random Forest es un algoritmo de aprendizaje supervisado. Para esta entrega decidí usarlo debido a que probé diferentes datasets y es un algoritmo flexible y fácil de utilizar. Básicamente el algoritmo crea árboles de decisión con datos que se seleccionan aleatoriamente.

2 Tratamiento de los datos

La base de datos que se utiliza en esta entrega es sacada de la plataforma Kaggle, esta habla acerca de si un paciente muere debido a un problema cardíaco. La liga al problema y a la base de datos es la siguiente: https://www.kaggle.com/datasets/andrewmvd/heart-failure-clinical-data. La base de datos no contaba con valores nulos ni con variables categóricas, debido a esto fue bastante sencillo utilizarla.

Para entrenar el modelo y evaluarlo se creó un conjunto de prueba y un conjunto de validación con la librería sklearn. A continuación se muestra como se hizo.

```
1 from sklearn.model_selection import train_test_split
2
3 X = data.drop(['DEATH_EVENT'],axis=1)
4 Y = data['DEATH_EVENT']
5
6 # Divido el dataset en training y test
7 X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3) # 70% training y 30% test
8
9
```

3 Modelo

Para esta entrega utilicé el algoritmo Random Forest de la librería de Python *Sklearn.*

Para la primera prueba utilicé los siguientes parámetros:

- n_estimators = 100
- max_leaf_nodes = 10
- n_jobs = 1
- random_state = 50
- warm_start = True

Y la precisión del modelo fue la siguiente:

```
Precisión: 0.8111111111111111
```

En la segunda prueba decidí eliminar la mayoría de los hiperparametros y solamente deje n_estimators = 100, la precisión del modelo fue la siguiente:

```
Precisión: 0.7888888888888889
```

Para la tercera y última prueba cambie la función que mide la calidad de un split en el algoritmo, en default es 'gini' pero la cambie a 'entropy'. La precisión resultante fue la más alta, con eso podemos llegar a la conclusión que modificar los parámetros si cambia la precisión del modelo, es importante ver todos los parámetros que se pueden modificar dentro del modelo ya programado.

```
Precisión: 0.8222222222222222
```

3 Conclusión

Se puede llegar a la conclusión de que cuando tuneamos el modelo nos da mejores resultados, por lo que es mejor realizar un grid search para encontrar las mejores hiperparametros del modelo.

Para este tipo de problemas de clasificación el modelo Random Forest normalmente tiene un buen desempeño como lo pudimos observar en la precisión del modelo, es de suma importancia determinar qué modelo se ajusta mejor a tus datos y no solo utilizarlo porque es el más sencillo.

Es mucho más fácil trabajar con las librerías ya creadas en python debido a que pierdes mucho tiempo implementando los algoritmos a mano, por lo que yo recomendaría siempre utilizarlas.