Instituto Tecnológico y de Estudios
Superiores de Monterrey

Inteligencia artificial avanzada para la
ciencia de datos

Momento de Retroalimentación: Módulo 2 Análisis y Reporte sobre el desempeño del modelo.

Autor:
Carlos de Jesús Ávila González A01750220

Repositorio:
`https://github.com/Carlos1300/Retro_M2_3`

Profesor:
Jorge Adolfo Ramírez Uresti

Fecha:
13/09/2022

1. Modelo Seleccionado

El modelo seleccionado sobre el que se llevará a cabo el análisis es el Decision Tree o Árbol de Decisión. Este es un modelo de aprendizaje supervisado no paramétrico que es utilizado tanto como regresor y como clasificador. Posee una estructura de árbol jerárquica, que consta de un nodo raíz, ramas, nodos internos y nodos hojas.

Para este análisis se utilizará la paquetería *Scikit-Learn*, en específico su módulo *tree* donde se encuentra en objeto *DecisionTreeClassifier*.

Los parámetros de este clasificador son los siguientes:

- *Criterion*: Función encargada de medir la calidad de la ramificación.
- *Splitter*: Estrategia utilizada para elegir la ramificación en cada nodo.
- *Max Depth*: La profunidad máxima que puede tener el árbol.
- *Min Samples Split*: El número mínimo de conjuntos requeridos para remificar un nodo interno.
- *Min Samples Leaf*: El número mínimo de conjuntos requeridos para colocarse en un nodo final u hoja.
- *Min Weight Fraction Leaf*: La minima fracción ponderada de la suma de los pesos totales para estar en un nodo final u hoja.
- *Max Features*: El número de características que es considerado al buscar la mejor ramificación.
- *Random State*: Controla la aleatoridad del modelo.
- *Max Leaf Nodes*: Número máximo de nodos finales u hojas que puede haber.
- *Min Impurity Decrease*: Un nodo se puede ramificar si mayor o igual al número dado en este parámetro.
- *Class Weight*: Pesos relacionados con cada clase.
- *CCP Alpha*: Parámetro de complejidad.

2. División de la base de datos

El conjunto de datos utilizada en este análisis se encuentra bajo el nombre *pima-indians-diabetes.csv* en la plataforma de *Kaggle*. Para la división de nuestro conjunto de datos se utilizó el objeto *train_test_split* del módulo de Scikit-Learn *model_selection*. Los parámetros utilizados para generar la división de la base de datos fueron:

- Nuestras variables independientes del conjunto de datos.
- La variable dependiente que predecirá el modelo.
- El porcentaje en el que se quiere partir la base de datos, en este caso se optó por un 80 % en el conjunto de entrenamiento y un 20 % en el conjunto de prueba, esto debido a que la base de datos utilizada tiene muy pocas entradas.
- Un random state de 40 para que la práctica pueda ser replicada por quien lo desee.

Hecho esto se cuenta con un 80-20 para entrenar el modelo y ponerlo a prueba respectivamente.

3. Métricas a utilizar

Para evaluar el desempeño del modelo se utilizará el módulo *Metrics* de Scikit-Learn. A continuación se enumeran las métricas que serán aplicadas al modelo y que nos ayudarán para el análisis del mismo:

1. **Accuracy**: El nivel de accuracy nos indica qué tan a menudo el modelo es capaz de clasificar de manera correcta.
2. **ROC AUC**: El área bajo la curva ROC nos indica la probabilidad que tiene el modelo para poder distinguir entre las clases que se poseen.
3. **Confussion Matrix**: Una manera visual de observar cómo el modelo está clasificando los datos.

4. Ejecución del modelo

Teniendo nuestro conjunto de datos de entrenamiento y prueba se procedió a ejecutar por primera vez el modelo en nuestros datos de entrenamiento, en este caso el modelo obtuvo un **Accuracy Score** de **100 %** y un **ROC AUC** de **1.0**. También se obtuvo la siguiente matriz de confusión con la que podemos observar cómo el modelo clasificó los datos:

		Clase real		
		Negativo	Positivo	Total
Clase Predicha	Negativo	405	0	405
	Positivo	0	209	209
	Total	405	209	

Después de ejecutar el modelo con los datos de entrenamiento, se procede a ejecutarlo con los datos de prueba. En este caso se obtuvo un **Accuracy Score** de **75 %** y un **ROC AUC** de **0.72**. Además se obtiene la siguiente matriz de confusión:

		Clase real		
		Negativo	Positivo	Total
Clase Predicha	Negativo	79	16	95
	Positivo	22	37	59
	Total	101	53	

5. Nivel de Ajuste del Modelo.

como pudimos observar con anterioridad, el modelo tuvo un sobreajuste o un overfitting al momento de entrenar el modelo, pues comparando los resultados de **Accuracy** del proceso de entrenamiento y el proceso de prueba podemos observar que en el primero de ellos obtenemos un **100 %** de exactitud del modelo, esto también se ve representado en la matriz de confusión pues se puede observar que el modelo logra clasificar de manera perfecta todos los datos. Sin embargo, en la parte de prueba del modelo podemos observar que su exactitud decae hasta un **77 %** y en la matriz de confusión podemos ver que tiene diversos errores al clasificar los datos.

Lo que esto nos indica es que el modelo simplemente está memorizando los datos en lugar de aprender a generalizarlos.

6. Nivel de Sesgo del Modelo.

Tomando en cuenta los resultados obtenidos en la métrica de **Accuracy** y la **Matriz de Confusión** en el entrenamiento del modelo, así como el nivel de ajuste obtenido del modelo en la fase de entrenamiento, esto nos indica que el grado de sesgo o bias del modelo es muy bajo. Esto ocurre ya que el modelo de Árbol de Decisión tiene una tendencia bastante grande a sobreajustarse a los datos de entrenamiento pues este algoritmo iterativo no para hasta que sus nodos finales, también llamados hojas, sean puras, es decir, que se refieran a una y solo una de las clases que poseemos.

7. Nivel de Varianza del Modelo.

Por otra parte, la varianza del modelo es bastante elevada y esto lo podemos observar comparando los resultados de las métricas de **Accuracy** y **Matriz de Confusión** de la parte de entrenamiento con los de la parte de prueba, pues podemos observar que si bien el modelo se desempeña de manera excepcional en los datos de entrenamiento, este tiene carencias al momento de enfrentarse a datos que son desconocidos para él. Debido a esto podemos confirmar que el modelo en su estado actual cuenta con un grado alto de varianza.

8. Corrección del Modelo

Existen diversas técnicas para eliminar el alto grado de varianza de nuestro modelo, al igual que reducir el sobreajuste que este posee. Sin embargo, la técnica que utilizaremos en este caso es la de ajuste de hiperparámetros con el objetivo de reducir la complejidad del modelo.

Reducir la complejidad se refiere a limitar el modelo hasta cierto punto para que tenga una buena generalización de los datos, pero no un sobreajuste.

9. Ajuste de Parámetros

El parámetro en el que nos enfocaremos será en el de *max_depth* pues de este depende hasta donde llega nuestro árbol de decisión, que pudimos observar que si no regulamos podemos vernos en problemas de un sobre ajuste.

Se realizó una gráfica de ajuste la cual nos ayuda a observar cuánta exactitud tiene nuestro modelo en función de la complejidad. En este caso es la cantidad de nodos o la profundidad de nuestro modelo.

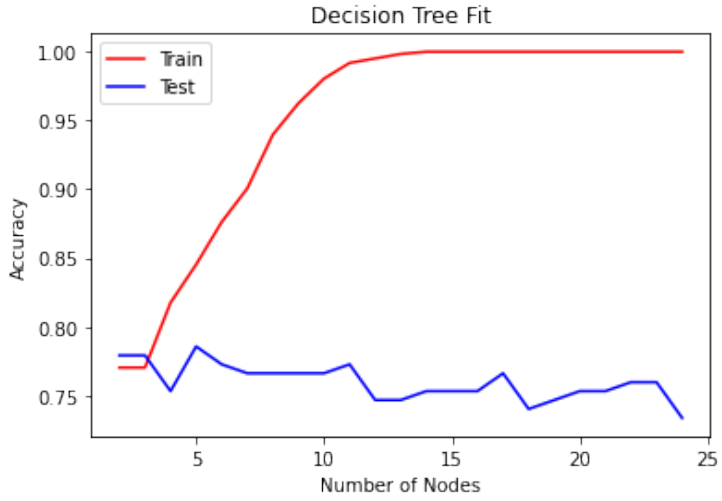

Figura 1: Gráfico de Ajuste para el Árbol de Decisión.

De la figura 1 podemos observar que el punto donde más exactitud posee nuestro modelo sin sobreajustarse es cuando cuenta con una profundidad de cinco nodos.

Con esta información procedemos a ajustar los parámetros de nuestro modelo para que cuente con cinco nodos de profundidad y le realizamos las pruebas anterior para observar si hay alguna mejora.

10. Análisis de las Mejoras

Con los parámetros ya ajustados, podemos observar que en la parte de entrenamiento se obtuvo un valor de **Accuracy** de un **85 %** y un valor de **ROC AUC** de **0.82** que si se compara con la primera ejecución redujeron significativamente, un 15 % y 0.18 respectivamente. De igual manera, se calculó la matriz de confusión para ese modelo con parámetros ajustados.

		Clase real		
		Negativo	Positivo	Total
Clase Predicha	Negativo	367	38	405
	Positivo	57	152	109
	Total	424	190	

Igualmente podemos observar que ahora el modelo en la parte de entrenamiento tuvo peor desempeño al clasificar comparado a la primera ejecución.

Sin embargo, donde tenemos que prestar atención es en la parte de la prueba del modelo pues podemos observar que se obtuvo un valor de **Accuracy** del **79 %** y un valor de **ROC AUC** de **0.75**. Su matriz de confusión fue la siguiente:

		Clase real		
		Negativo	Positivo	Total
Clase Predicha	Negativo	84	11	95
	Positivo	22	37	59
	Total	106	48	

Podemos observar que efectivamente logró clasificar de manera correcta más datos que en la primera ejecución.

El detalle importante es que obtuvo un **4 %** más de exactitud que en la primera ejecución, lo que nos indica que si se sigue haciendo un ajuste de parámetros al modelo podríamos llegar a una mejor exactitud y eliminar los problemas de sobreajuste, bias y varianza que tuvimos en un principio.