Instituto Tecnológico y de Estudios Superiores de Monterrey

Análisis y reporte sobre el desempeño del modelo.

Israel Sánchez Miranda A01378705

Profesor Jorge Adolfo Ramirez Uresti

13 de septiembre de 2022

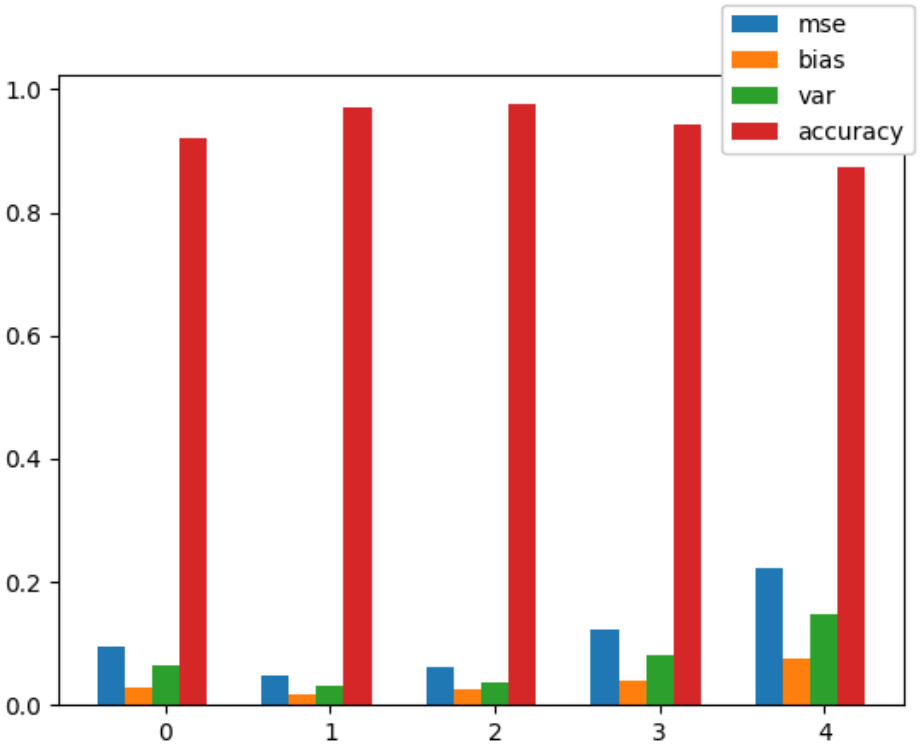
1. **Sobre el modelo.**
   Para esta actividad se realizó un modelo de *Random Forest* para categorizar la clase de un tipo de vino basado en el dataset wine.csv, el cual se encuentra en el repositorio de Github.
   a. **Descripción del modelo.**
      El modelo lee los datos del archivo wine.csv para posteriormente separarlos en sets de prueba y sets de entrenamiento, la proporción prueba-entrenamiento es solicitada al usuario.
      Una vez separados los datos se crea el modelo de *Random Forest* con n árboles y j nodos hoja como límite. Ambos valores también son proporcionados por el usuario. Cabe destacar que se mantiene una semilla aleatoria constante para reducir la variación de las respuestas y tener un mejor análisis y entendimiento de cómo es que está operando el modelo basado en los hiper-parámetros proporcionados.
      Ya que se creó el modelo este es entrenado con los datos correspondientes para después realizar predicciones con el set de pruebas y así obtener el porcentaje de precisión.
      Posterior a esto, el usuario puede proporcionar valores para cada columna x dentro del data frame para que así el mismo modelo prediga qué clase de vino será el que el usuario ingresó.
      Finalmente, el programa muestra un árbol de decisión aleatorio que conforma el bosque para que el usuario pueda apreciar cómo es que el bosque opera.
   b. **Hiper-parámetros del modelo.**
      - `n_estimators`: Número de estimadores (árboles de decisiones) que conforman el bosque.
      - `max_leaf_nodes`: Máximo número de nodos hoja que puede tener cada árbol.
      - `RATIO`: Proporción prueba-entrenamiento, indica qué porcentaje del dataset va a ser usado para pruebas, el porcentaje restante será el set de datos de entrenamiento.
   c. **Descripción del algoritmo.**
      Se genera un dataset de bagging con registros aleatorios del dataset para generar los árboles del bosque.
      Posterior a esto se crea un árbol de decisión para este dataset con un subset de características (columnas) en cada paso de la creación del árbol.
      Se sigue repitiendo este procedimiento hasta llenar el bosque con el número de árboles deseado.
      Cada árbol aprenderá y dará una respuesta diferente dependiendo de cómo esté conformado y todo el bosque tomará una decisión de clasificación basado en un promedio o voto por mayoría.
2. **Data sets del modelo.**
   Para este modelo se usó el módulo `train_test_split` de la librería `sklearn.model_selection` para dividir todo el dataset en sets de entrenamiento y

validación. Esto ayuda a que el mismo modelo pueda usar diferentes tipos de datos (escogidos aleatoriamente) para entrenarse y poder determinar la mejor aproximación a la respuesta.

Como se mencionó anteriormente, la proporción del set de entrenamiento contra la del set de validación se ve determinada por el usuario mediante el hiper parámetro `RATIO`, es decir, si el usuario usa un `RATIO=70` el 70% del dataset será destinado a entrenamiento, mientras que el 30% restante será el dataset de validación.

3. **Bias, varianza y ajuste del modelo.**

Para analizar el bias, la varianza y el ajuste del modelo se corrieron 5 pruebas con los siguientes hiper parámetros:
`RATIO=[50,60,70,80,90],n_estimators=[5,10,15,20,5],max_leaf_nodes=[5,10,16,18,25]` para así obtener los siguientes resultados en cada corrida respectivamente:



En general se puede observar que el modelo tiene un buen ajuste (accuracy) a los datos que se le presentan, obteniendo valores muy cercanos a 1.0, esto incluso aunque solo se le proporcione el 50% de los datos para entrenar. En la gráfica se puede observar cómo es que si se ocupa un porcentaje entre el 60%-80% se obtiene un accuracy muy bueno, sin embargo esto podría deberse a que existe overfitting en el modelo, por lo que para ello se necesita analizar el bias y la varianza.

Sabiendo que un modelo con varianza alta tiende a hacer overfitting se puede hacer su comparación respecto al bias y ver cuales son los problemas que el modelo puede presentar.

Si volvemos a la gráfica se puede observar que en todas las iteraciones existe una varianza relativamente alta lo que puede indicar riesgo de overfitting. La varianza es

especialmente alta cuando se usa un set de entrenamiento con el 90% de los datos y se tienen varios nodos hoja, lo que indicaría que gracias a ello el modelo puede tender a hacer overfitting en la última iteración.

Las corridas en donde la varianza se mantuvo más baja fueron aquellas en donde se tuvieron porcentajes de entrenamiento entre el 60% y 70% con pocos nodos hoja y entre 15 y 20 árboles de decisión por lo que se podría decir que estos son los mejores hiper parámetros para lograr un modelo preciso y no sobre ajustado.

Del lado del bias no existe una gran preocupación ya que siempre se mantiene bajo por lo que se puede concluir que el modelo no se esta sub ajustando ni está haciendo una generalización muy simple de las variables que afectan el resultado. Sin embargo, es importante notar cómo es que los valores del bias suben considerablemente en la última iteración justamente debido a las mismas razones por las que la varianza sube.

Con el MSE ocurre exactamente el mismo comportamiento, por lo general suele mantenerse relativamente alto, sin embargo en la última iteración el valor se dispara lo que hace plantearse si realmente las estimaciones hechas son mejores o más precisas que iteraciones anteriores. Cabe destacar que en las mejores corridas (2 y 3) el MSE se mantiene muy bajo, reforzando que la mejor combinación de parámetros se encuentra entre estas dos iteraciones.

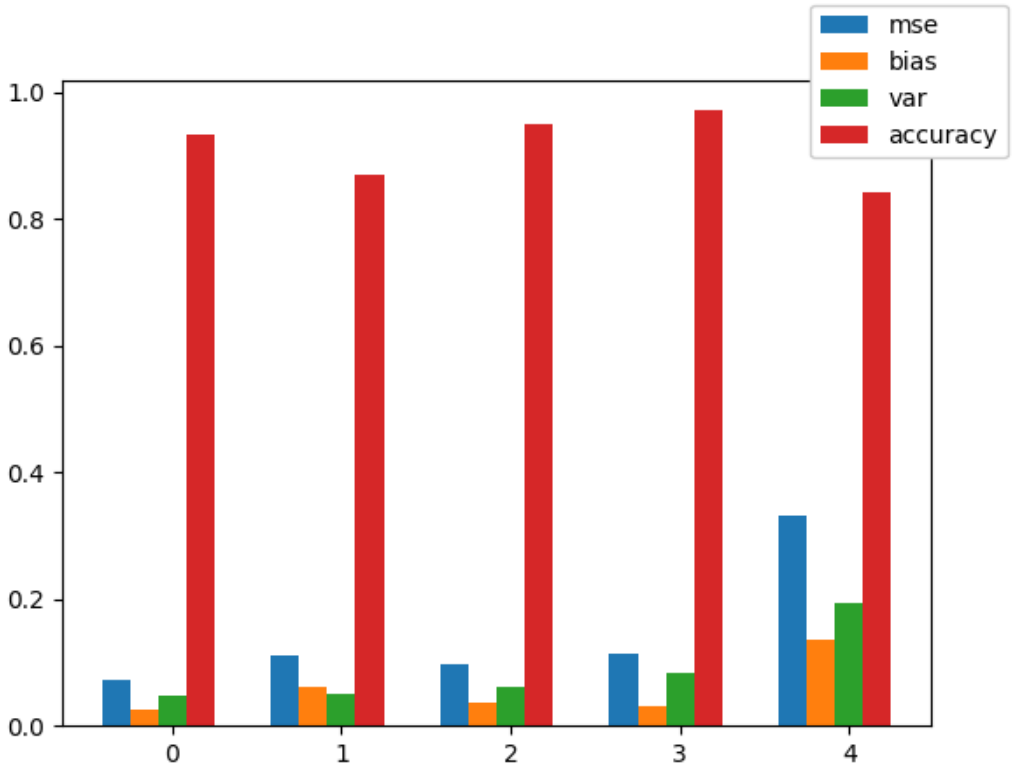
4. **Regularización del modelo.**

   Ahora bien, a pesar de que se tiene una serie de hiper parámetros que dan un buen modelo, se puede buscar mejorarlo agregando más hiper parámetros, cambiando los actuales o aplicando técnicas de regularización al modelo.

   Ahora bien, debido a que los árboles de decisiones son algoritmos heurísticos es algo complicado aplicar alguna técnica de regularización conocida en los mismos. Por lo que la solución para evitar el overfitting que se plantea es la siguiente:

   - Limitar la máxima profundidad de los árboles usando el parámetro `max_depth`.
   - Buscar un número entre 10 y 20 que de el mejor resultado de precisión con respecto al MSE, varianza y bias.
   - Poner criterios más estrictos de cuando un nodo hoja debe de partirse usando el parámetro `min_samples_split` y `min_impurity_decrease`.

   Así, se mejoró el código, haciendo otras 5 corridas con los mismos valores para `RATIO` y `max_leaf_nodes`, mejorando el valor de `n_estimators` a `[10, 15, 20, 12, 17]` y usando los valores de `max_depth=[10, 20, 30, 40, 40],min_samples_split=[4,3,5,3,5],min_impurity_decrease[0.1,0.2,0.2,0.1,0.2]` y se obtuvieron los siguientes resultados:



Ahora bien, a pesar de que se ve un decremento en el accuracy se puede observar que en las iteraciones que contiene los valores que se consideran como las mejores combinaciones (iteraciones 2 a 4) se obtienen valores estables del bias y la varianza haciendo que el modelo se ajuste bien pero no tienda al overfitting y generalice bien sin hacer un sub ajuste del modelo.

Realmente cuando se utiliza un gran porcentaje de pruebas y se deja que los árboles crezcan mucho (última iteración) es cuando se obtienen los peores resultados, por lo que si se genera un bosque aleatorio con valores de porcentaje entre el 60%-80% con entre 10-20 árboles cada uno con un máximo de 18 nodos hoja, profundidad de 40 con un `min_samples_split` de entre 3 y 4 y un `min_impurity_decrease` de 0.1-0.2 se puede obtener un modelo que haga predicciones lo suficientemente precisas así como lo suficientemente generales.

5. **Evaluación general del modelo.**

   Como se pudo observar, el ajuste de los hiper parámetros del modelo es sumamente importante para llegar a un balance entre el bias y la varianza. En el primer modelo, a pesar de que se tenían predicciones con una precisión muy alta, existía el riesgo de hacer overfitting y el porcentaje de entrenamiento tenía un impacto muy alto dentro de la ejecución del modelo.

   Ahora, después de aplicar las estrategias para regularizar el modelo, se logró un balance y mayor estabilidad en el mismo en el que, a pesar de tener predicciones con un poco menos de precisión, se tiene un bias y una varianza lo suficientemente estables, indicando que el modelo puede tener buena precisión sin hacer overfitting de los datos y puede generalizar sin simplificar la modelación de los datos de entrada.