David Rodriguez Fragoso

A01748760

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

El objetivo de esta entrega es analizar el desempeño de una de las dos implementaciones anteriores. En este caso haremos uso de la segunda implementación, la cual hace uso de frameworks para simplificar el proceso.

Como recordatorio, haremos uso del dataset wine.csv disponible en el repositorio de GitHub de este documento.

Para asegurar un análisis completo de esta, se abordarán los siguientes puntos:

- Separación de datos (Training/Test)
- Diagnóstico y explicación del grado de bias
- Diagnóstico y explicación del grado de varianza
- Diagnóstico y explicación del nivel de ajuste del modelo

Nos apoyaremos de información obtenida a partir de gráficas y cálculos obtenidos de las librerías de python matplotlib y sklearn.

Separación de datos (Training/Test)

Para esta parte nos apoyaremos de la funcióntrain_test_split() de sklearn.

```
xTrain, xTest, yTrain, yTest = train_test_split(dfX,dfY,test_size=0.25
, random_state=1)
```

La separación de los datos es esencial en un modelo de entrenamiento, ya que es gracias a esta que podemos reducir en gran escala el nivel de overfitting. Esto se debe a que, si entrenamos a nuestro modelo con todos los datos disponibles, es muy probable que este en vez de aprender, se memorice los datos y al momento de realizar una predicción con datos nuevos, la precisión de esta sea muy diferente a la obtenida en la fase de entrenamiento.

En nuestro modelo usaremos una distribución de datos de 75% para entrenamiento y 25% para pruebas, esto es un porcentaje arbitrario, pero por lo general se recomienda que el porcentaje de datos de entrenamiento sea cercano a 80%.

Diagnóstico y explicación del grado de bias/ Diagnóstico y explicación del grado de varianza

```
mse, bias, var = bias_variance_decomp
(model, xTrain.values, yTrain.values, xTest.values, yTest.values, loss
='mse', num_rounds=200, random_seed=1)
```

Para calcular los porcentajes de bias y de varianza usaremos la función bias_variance_decomp() de sklearn.

Los resultados obtenidos son los siguientes:

```
MSE: 0.275
Bias: 0.237
Variance: 0.038
```

El MSE nos indica el promedio de pérdida que tendremos en cada predicción y es el resultado de la suma del bias y de la varianza.

El bias nos indica el nivel de sesgo que tiene nuestro modelo, por lo general se busca que este sea bajo, pero tampoco muy cercano a cero, ya que esto último podría indicar que nuestro modelo está haciendo overfitting. En resumen, un bias muy alto podría indicar que nuestro modelo está haciendo underfitting y que está generalizando los resultados, mientras que un bias muy bajo justo lo contrario.

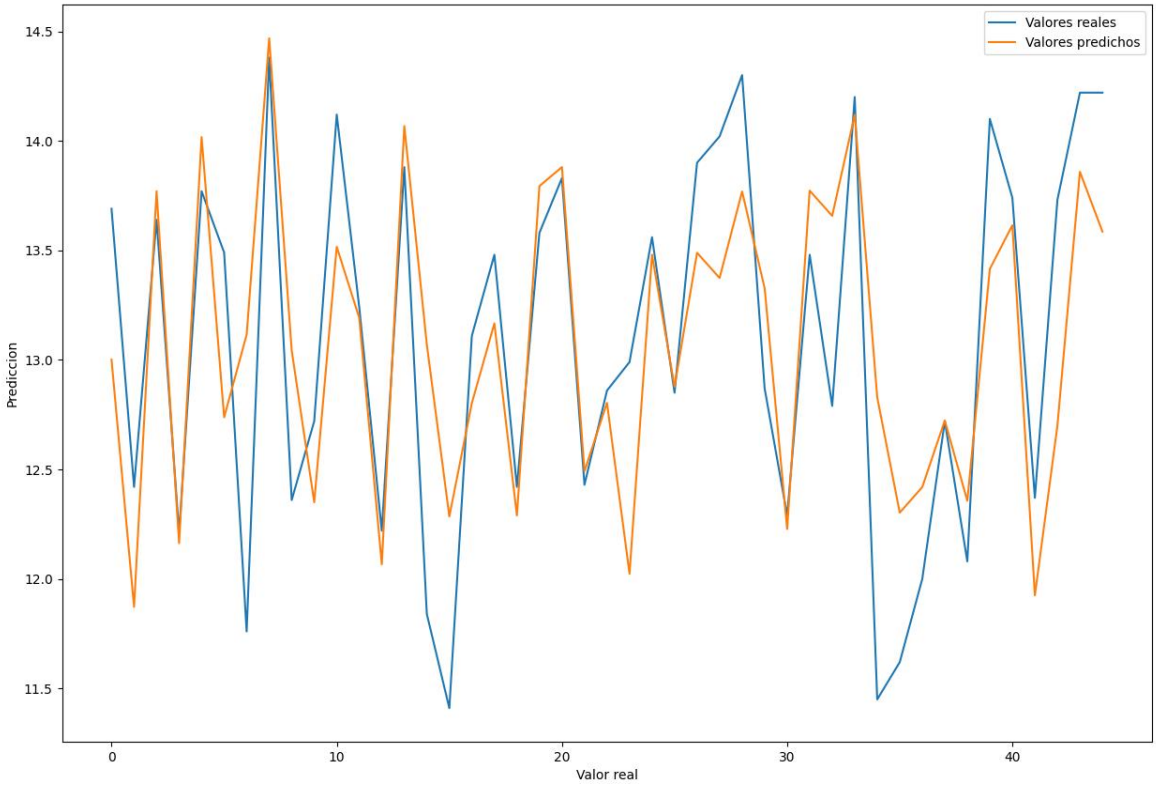
La varianza nos indica algo similar al bias, ya que esta nos dice qué tanto varían nuestros datos para hacer las predicciones. Una varianza alta podría indicar que nuestro modelo está haciendo overfitting, ya que sus resultados variarían mucho con tal de predecir correctamente el resultado; podemos notar esto fácilmente si hay un outlyer muy lejano a los resultados y el modelo logra predecirlo correctamente.

En realidad, los resultados obtenidos son bastante buenos, ya que estos no solo tienen valores bajos, sino que también están balanceados (no muy alejados uno del otro).

Diagnóstico y explicación del nivel de ajuste del modelo

El modelo ha demostrado ser bueno, ya que su porcentaje de MSE, de bias y de varianza son bajos y nivelados, por lo que la probabilidad de hacer overfitting es baja y la de hacer overfitting también.

Con ayuda de la librería matplotlib generaremos gráficas comparando los resultados de las predicciones generadas con el dataset de test.



Conclusión

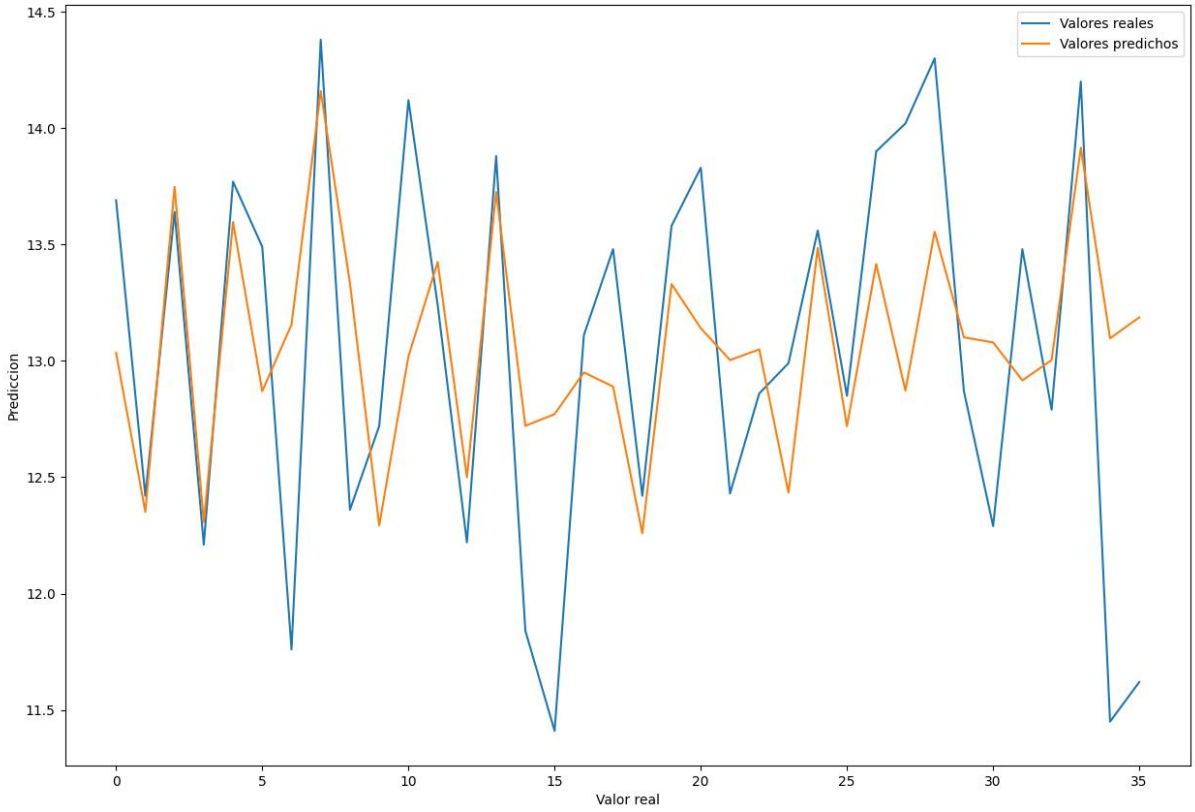
Es evidente que, pese a que nuestro modelo es bueno, tampoco es perfecto y tiene posibles puntos de mejora. Para mejorar el modelo podríamos modificar la proporción de datos de entrenamiento o las columnas usadas para el entrenamiento del mismo.

Recordemos que actualmente estamos usando 10 columnas (['flavanoids', 'malic acid', 'ash','magnesium','total phenols','alcalinity of ash','OD280/OD315 of diluted wines','proline','hue','color intensity']), pero no necesariamente son las más óptimas para el entrenamiento. Tendríamos que probar diferentes combinaciones de estas y de los porcentajes de entrenamiento y pruebas. Incluso si nos alejamos un poco de esta implementación, puede que otros modelos de aprendizaje nos dieran mejores resultados.

Finalmente haremos un par de pruebas variando los parámetros anteriormente mencionados:

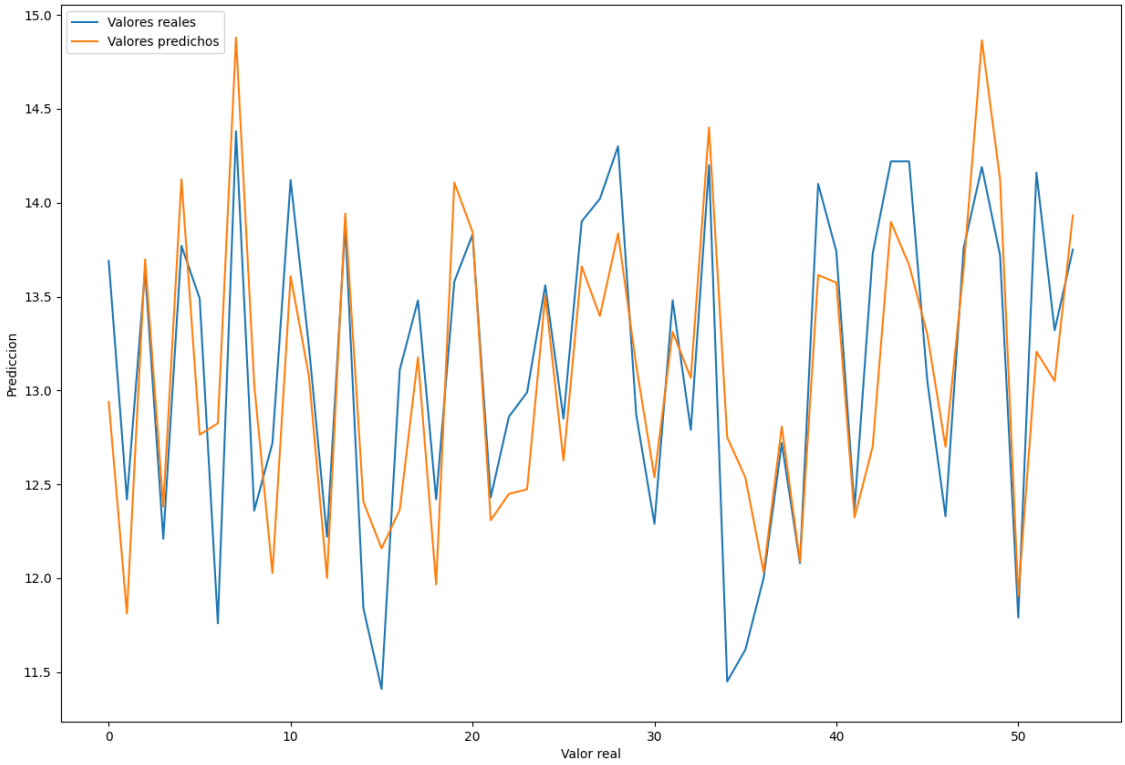
- Columnas: ['flavanoids', 'malic acid', 'ash','magnesium','total phenols','alcalinity of ash','OD280/OD315 of diluted wines']
- Porcentajes: training 80%, test 20%

```
MSE: 0.559
Bias: 0.526
Variance: 0.033
```



- Columnas: ['flavanoids', 'malic acid', 'ash','magnesium','total phenols','alcalinity of ash','OD280/OD315 of diluted wines','proline','hue','color intensity']
- Porcentajes: training 70%, test 30%

```
MSE: 0.275
Bias: 0.237
Variance: 0.038
```



Podemos observar que la primera prueba no nos dio mejores resultados, pero la segunda sí. Podemos entonces sugerir que este modelo puede ser refinado modificando los parámetros mencionados.