Tecnológico de Monterrey

Análisis y Reporte sobre el desempeño del modelo

Diego Solis Higuera

A00827847

Inteligencia artificial avanzada para la ciencia de datos I
Grupo 101

Monterrey, Nuevo León

14 de septiembre de 2022

El objetivo es predecir el nivel de flavonoides que contiene una botella de vino. Para ello, se generó una matriz de correlación, donde se puede observar que el total de fenoles tiene una alta correlación ***(0.86)*** con los flavonoides (figura 1); por ello, se crea un modelo de regresión lineal que consta de 1 independiente – Total de fenoles – y una variable dependiente – Flavonoides.

Figura 1. Matriz de correlación de variables.

Separación del dataset

Antes de comenzar con el análisis de los modelos, es importante empezar por cómo se entrenaron, probaron y validaron. Al contar con un único dataset, se tuvo que dividir en dos partes; la primera parte fue utilizada para entrenar nuestro modelo y la segunda parte

para hacer pruebas con nuestro modelo. Además, tenemos una tercera parte que llamamos validación, que básicamente es cuando se ingresan datos reales a nuestro modelo y generamos una predicción.

Es importante recalcar la manera en la que está dividido nuestro dataset; la mayor parte de estos datos se utilizan para entrenar nuestro modelo, mientras que el resto se utilizan para hacer pruebas. Si hablamos estrictamente de números, el 75% de nuestros datos se utilizan para entrenar el modelo y el 25% de nuestros datos se utilizan para hacer pruebas. Se utilizó la función ***train_test_split*** de la librería scikit-learn para llevar a cabo este split. Esta misma regla (75% entrenamiento y 25% pruebas) aplica para ambos modelos analizados en este reporte.

Primer modelo

Utiliza el valor que mejor se ajuste al modelo de regresión generado. Al desplegar los parámetros de nuestro modelo, tenemos lo siguiente:

```
1 print('Parámteros de nuestro modelo: {}'.format(ml_mod.get_params()))
Parámteros de nuestro modelo: {'copy_X': True, 'fit_intercept': False, 'n_jobs': None, 'normalize': 'deprecated', 'positive': False}
```

Figura 2. Parámetros de nuestro modelo.

Inicialmente, como se menciona anteriormente, utilizamos un modelo de regresión lineal de primer orden. En este caso, es importante destacar que nuestro parámetro de “y_intercept tiene un valor de 0. Este parámetro representa la intercepción con el eje Y. Esto quiere decir que este dato no se cambia con la línea de tendencia que mejor se ajusta a nuestros datos. En otras palabras, nuestro modelo de primero orden solo contiene una theta, que en este caso es theta 1 (figura 3).

```
1 print('Y Intercept de modelo: {}'.format(ml_mod.intercept_))
Y Intercept de modelo: 0.0
```

Figura 3. Valor de Y_intercept de nuestro modelo.

Al realizar los pasos de entrenamiento, validación, pruebas, y ajustes de nuestros parámetros, se pudo concluir que el modelo tiene un score de ***64%*** de entrenamiento, ***70%*** de prueba y un error cuadrático medio de ***0.505705*** (Figura 4).

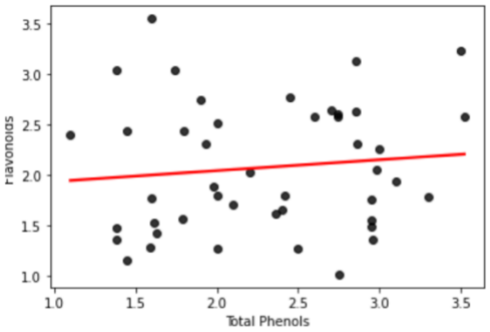
```
Score de datos de entrenamiento: 64.4173%
Score de datos de prueba: 70.3943%
Error cuadrático medio: 0.505705
```

Figura 4. Desempeño de modelo de regresión lineal.

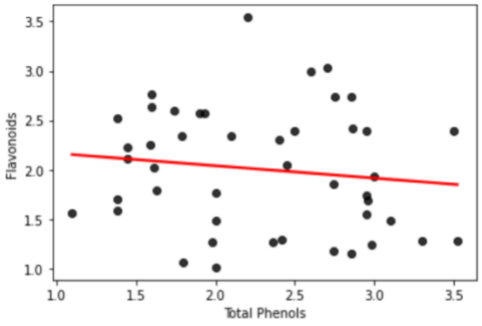
Análisis de varianza y sesgo

Varianza

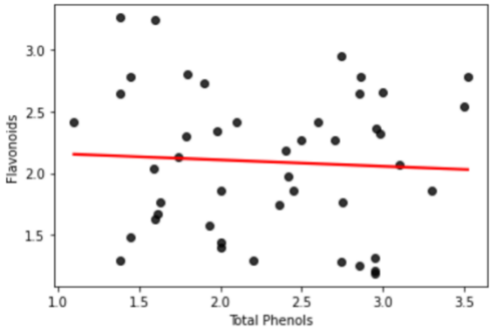
Para hace un análisis de varianza de nuestro modelo, tomamos 4 muestras aleatorias de nuestra muestra para el entrenamiento y predicción de resultados. A continuación, podemos observar las 4 gráficas, donde se muestra su desempeño y error cuadrático medio:



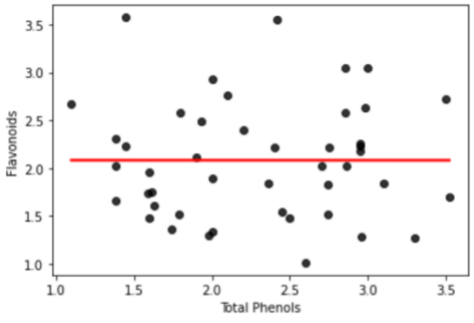

```
Score de datos de entrenamiento: 63.8246%
Score de datos de prueba: 70.7159%
Error cuadrático medio: 0.557452
```




```
Score de datos de entrenamiento: 63.0266%
Score de datos de prueba: 76.0968%
Error cuadrático medio: 0.438387
```




```
Score de datos de entrenamiento: 66.2809%
Score de datos de prueba: 63.4659%
Error cuadrático medio: 0.581673
```




```
Score de datos de entrenamiento: 67.6742%
Score de datos de prueba: 60.9345%
Error cuadrático medio: 0.666204
```

Figura 5. Comparación de varianza de modelo. El eje x representa el total de fenoles y el eje y representa la estimación de flavonoides.

Como se puede apreciar, tenemos resultados bastante similares en las 4 pruebas. Tenemos un score de entrenamiento que está entre 63% y 67%, y un score de pruebas que ronda entre 60% y 76%, lo que nos indica una baja varianza. Debido a esto, no es necesario hacer ajustes en la cantidad de datos que se introducen en la fase de pruebas.

Sesgo

El sesgo nos indica la diferencia entre un valor esperado y el valor obtenido. En la siguiente gráfica, podemos apreciar la estimación de los datos obtenidos con nuestro modelo, contra los datos reales. Como podemos observar, la gran mayoría de los datos del modelo no se ajustan a los datos reales, lo que indica sesgo alto (figura 6).

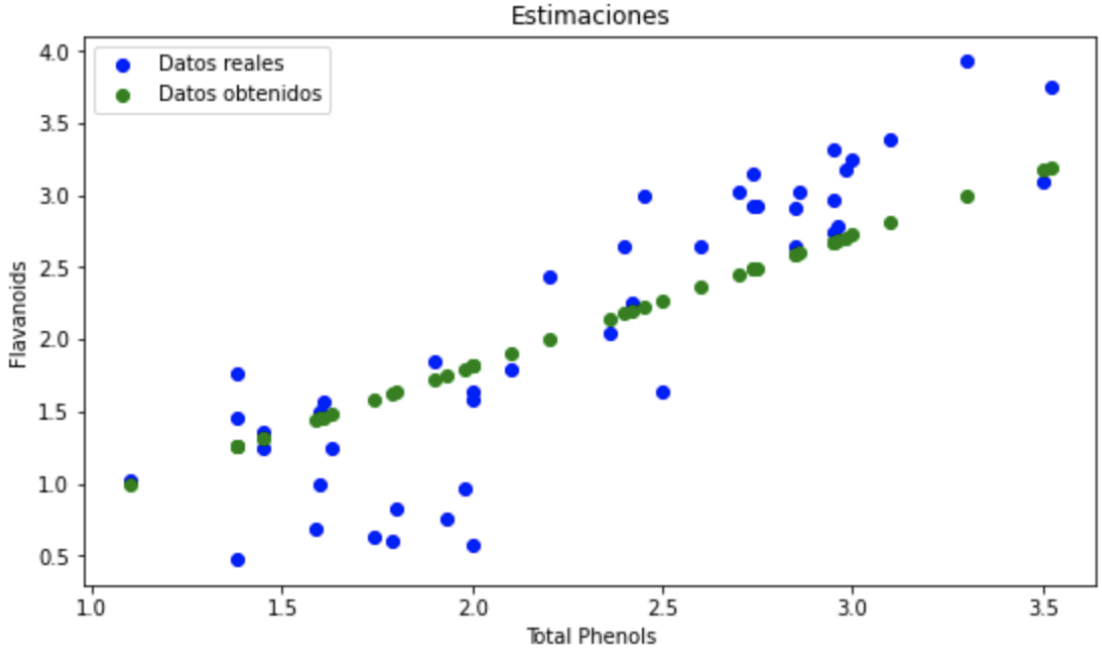


Figura 6. Gráfica de datos reales vs datos obtenidos por modelo.

Fitting

Al hacer este análisis, podemos determinar lo siguiente; como observamos, nuestro modelo tiene alto sesgo y baja varianza, así como un error cuadrático medio relativamente alto, lo que nos lleva a concluir que nuestro modelo sufre de *underfitting*.

A continuación, podemos ver una tabla comparativa en donde se despliegan los primeros diez datos de nuestra predicción contra los datos reales.

Flavanoids	
Real	Estimado
3.06	2.39
2.76	2.01
3.24	1.65
3.49	3.52
2.69	1.26
3.39	2.21
2.52	1.54
2.51	2.04
2.98	2.43
3.15	2.56

Figura 7. Estimaciones contra valores reales.

Segundo modelo

En primera instancia, es importante recordar que el propósito del primer modelo es predecir el nivel de Flavonoides que contiene el vino a partir de un indicador, en este caso la cantidad total de fenoles. Una vez terminamos de analizar nuestro primero modelo, el cual constaba de un modelo de regresión lineal de primer orden, detectamos algunos detalles que afectan el desempeño de predicción y genera datos erróneos. Por ello, en este segundo modelo, se aplicará de igual manera un modelo de regresión lineal pero se ajustarán parámetros y variables del primer modelo.

Al graficar los valores reales vs valores obtenidos, podemos analizar nuestro set de datos cuenta con múltiples variables, las cuales pueden ser indicadores clave para

determinar la cantidad de flavonoides que contiene el vino, por lo que resulta ineficiente utilizar una sola variable independiente, o feature. Por lo tanto, tomando en cuenta el coeficiente de kappa de cohen, que determina la concordancia observada, se utilizarán aquellas variables que contengan una correlación igual o superior a ***0.40*** (Figura 8).

Value of κ	**Strength of Agreement**
< 0.20	Poor
$0.21 - 0.40$	Fair
$0.41 - 0.60$	Moderate
$0.61 - 0.80$	Good
> 0.80	Very Good

Figura 8. Coeficiente kappa de Cohen

Tomando en cuenta lo anterior, podemos observar que las siguientes variables tienen una correlación decente con nuestra variable dependiente (flavonoides):

- Malic acid
- Total Phenols
- Nonflavanoid Phenols
- Proanthocyanins
- Hue
- OD280/OD315 of diluted wines
- Proline

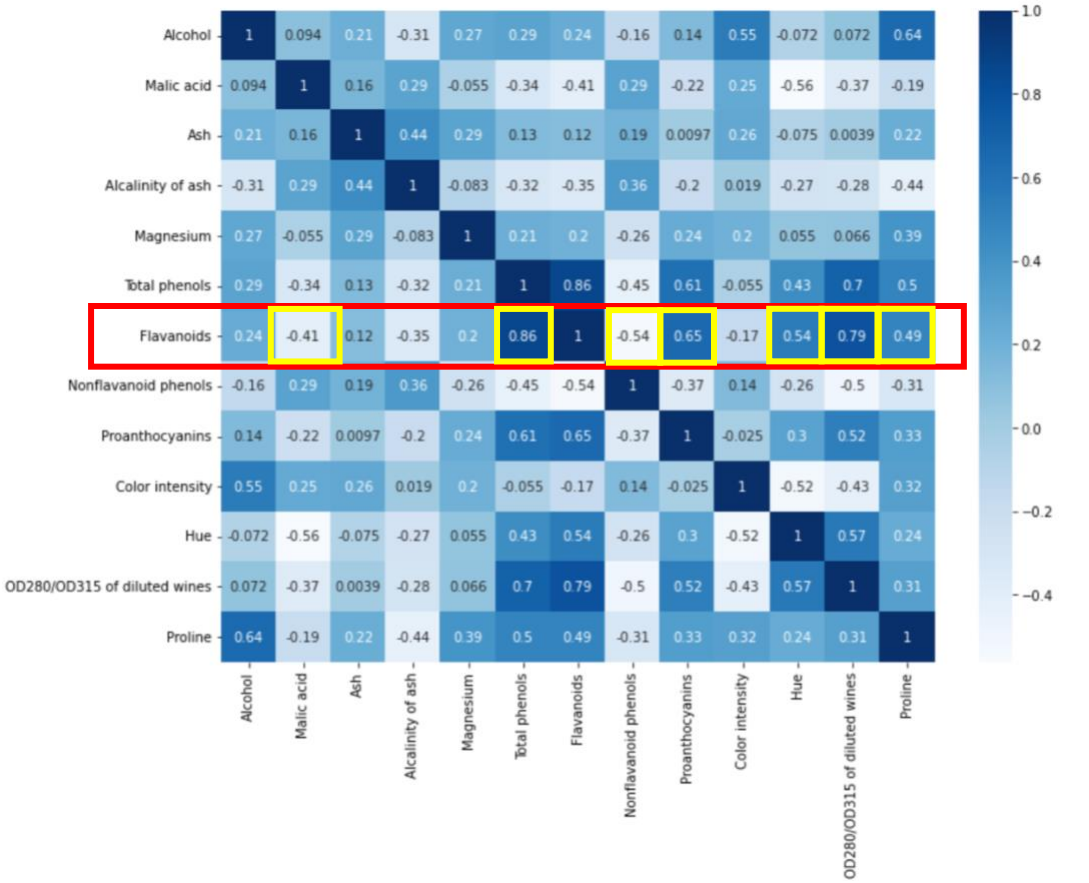


Figura 9. Variables con correlación considerable con flavanoides.

En la gráfica anterior, podemos analizar la correlación de las variables elegidas anteriormente, las cuales se muestran de color amarillo. Estas serán los features de nuestro modelo. La selección de más variables se hace con el objetivo de reducir nuestro sesgo, lo cual en cambio nos puede cambiar nuestro modelo a un estado de underfit a fit. Es importante tener cuidado con la selección de variables, ya que puede incrementar considerablemente la varianza de nuestro modelo.

Además, es importante incluir el valor de intersección de nuestro modelo con el eje Y, por lo que esta vez sí se incluirá este valor. A continuación podemos apreciar la modificación de los parámetros de nuestro modelo, lo que mejorará el desempeño:

```
Parámteros de nuestro modelo:
 {'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'normalize': 'deprecated', 'positive': False}
```

A su vez, podemos observar que esta vez sí se incluye nuestro y_intercept; a continuación podemos apreciar el valor que toma esta variable después de hacer el fit de nuestro modelo.

```
Y Intercept de modelo: [-1.20658017]
```

Figura 10. Y intercept de modelo.

Naturalmente, al incluir más variables, podemos analizar los coeficientes de nuestro modelo con el atributo *coef_*, el cual nos regresa el valor que toman los coeficientes una vez se haya entrenado nuestro modelo. Tenemos un total de 7 coeficientes, lo que corresponde a la cantidad de variables que utilizamos en nuestro modelo.

```
coeficientes de modelo:
[[-7.38445972e-02  8.19798976e-01 -4.39753504e-01  2.55149818e-01
   1.98506598e-01  3.66576594e-01  1.88921093e-04]]
```

Figura 11. Coeficientes de nuestro modelo.

Con el cambio de parámetros y la inclusión de más variables, logramos incrementar considerablemente el desempeño de nuestro modelo, donde ahora pasamos a un score de 85% para los datos de entrenamiento, 80% para los datos de prueba y un error cuadrático medio de 0.378596.

```
Score de datos de entrenamiento: 85.3979%
Score de datos de prueba: 80.47%
Error cuadrático medio: 0.378596
```

Figura 12. Análisis de desempeño de modelo de séptimo orden.

Análisis de varianza y sesgo

Varianza

Para determinar la varianza de nuestro modelo, al igual que el primer modelo, realizamos pruebas con 4 muestras aleatorias, Aunque es imposible graficar los datos debido a que

trata de un modelo de séptimo orden, podemos determinar la varianza si observamos el score de entrenamiento y pruebas en las 4 pruebas.

```
Score de datos de entrenamiento: 83.8659%
Score de datos de prueba: 87.1943%
Error cuadrático medio: 0.344642

Score de datos de entrenamiento: 83.4731%
Score de datos de prueba: 87.7665%
Error cuadrático medio: 0.334811

Score de datos de entrenamiento: 85.6072%
Score de datos de prueba: 79.3123%
Error cuadrático medio: 0.430204

Score de datos de entrenamiento: 84.0388%
Score de datos de prueba: 84.6983%
Error cuadrático medio: 0.405228
```

Figura 13. Scores de 4 pruebas aleatorias

Aquí podemos observar cómo nuestra varianza es baja, tal y como en el primer modelo. Incluso, lo varianza en este modelo, lógicamente es todavía menor a la anterior. Nuestro score de datos de entrenamiento oscila entre el 83% y 85%, mientras que el score de datos de prueba ronda entre 79% y 87%. Con esto concluimos que tiene una varianza baja.

Sesgo

Como mencioné anteriormente, es difícil hacer un análisis visual con gráficas que representan nuestro modelo, por lo que es un poco más complicado determinar el nivel de sesgo en este modelo. Para ello, regresamos a las 4 pruebas con muestras aleatorias que se tomaron de nuestro set de datos.

```
Score de datos de entrenamiento: 83.8659%
Score de datos de prueba: 87.1943%
Error cuadrático medio: 0.344642

Score de datos de entrenamiento: 83.4731%
Score de datos de prueba: 87.7665%
Error cuadrático medio: 0.334811

Score de datos de entrenamiento: 85.6072%
Score de datos de prueba: 79.3123%
Error cuadrático medio: 0.430204

Score de datos de entrenamiento: 84.0388%
Score de datos de prueba: 84.6983%
Error cuadrático medio: 0.405228
```

Figura 14. Scores y error cuadrático medio de 4 pruebas.

En este caso, nos enfocamos en el error cuadrático medio. El error cuadrático medio es la suma de la varianza de una estimación más el cuadrado del sesgo. Como podemos observar, en términos generales, nuestro error cuadrático medio también disminuyó. En promedio, este valor es de 0.38, cuando en nuestro primer modelo era de 0.56. Podemos determinar que el cambio de nuestra varianza no fue significativo, y con esto, podemos determinar que tenemos un sesgo medio.

Fitting

Esto, nos puede llevar a concluir que tanto la varianza como el sesgo disminuyeron, pues anteriormente comprobamos que el primer elemento disminuyó, y por esto, podemos determinar que el segundo elemento siguió el mismo comportamiento. Por lo tanto, el ajuste de nuestro modelo mejoró, y podemos determinar que este tiene un fit medio.

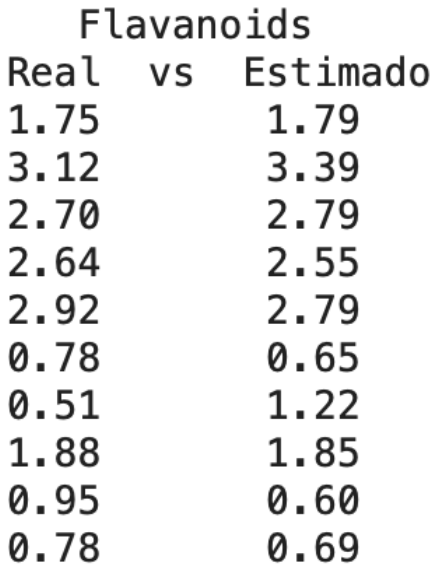


Figura 15. Estimaciones contra valores reales.