David Rodriguez Fragoso

A01748760

Momento de Retroalimentación: Módulo 2 Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución.

El propósito de esta entrega es realizar una implementación de un modelo de aprendizaje haciendo uso de frameworks tales como scikit learn. Para demostrar la efectividad del modelo creado, se generarán gráficas que nos ayuden a visualizar los resultados obtenidos en contraste con los resultados reales.

En esta ocasión haremos uso de un dataset que contiene información de vinos y trataremos de predecir la cantidad de alcohol que un vino tiene. Es importante mencionar que este dataset no necesita ser procesado y limpiado anteriormente.

El modelo implementado es el OLS (Ordinary Least Squares) y nos servirá para hacer predicciones con ayuda de una regresión lineal. En estadística, la regresión lineal o ajuste lineal es un modelo matemático usado para aproximar la relación de dependencia entre una variable dependiente Y, una serie de variables independientes X y una constante aleatoria.

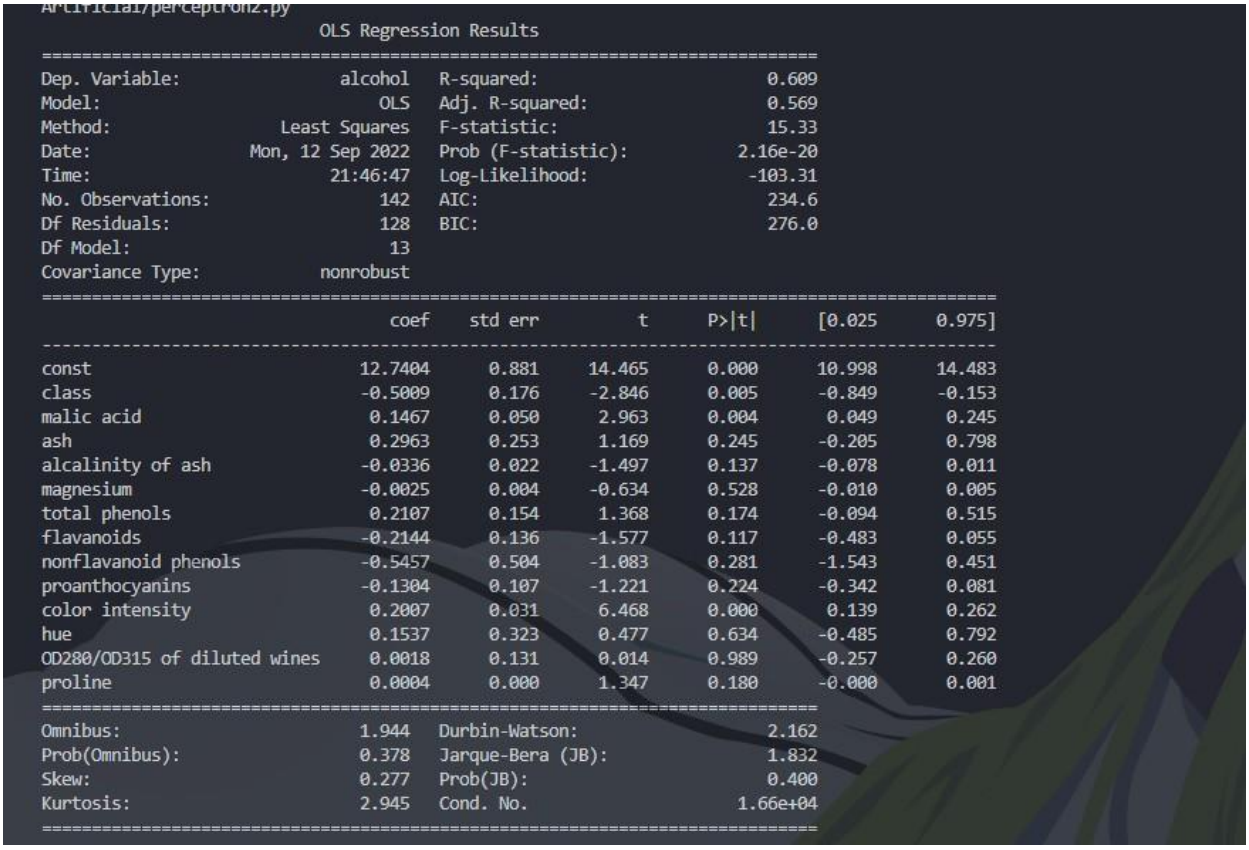
```
df = pd.read_csv('wine.csv')

dfX = df.drop('alcohol',axis=1)
dfY = df['alcohol']

#Dividimos los datos en una proporcion 80/20
xTrain, xTest, yTrain, yTest = train_test_split(dfX,dfY,test_size=0.2, random_state=1)

#Agregamos la columna constante a ambos datasets
xTrain = sm.add_constant(xTrain)
xTest = sm.add_constant(xTest)

model = sm.OLS(yTrain, xTrain).fit()
# Ejecutamos el modelo entranado usando los datos de prueba
prediction = model.predict(xTest)
print(model.summary())
```

Para configurar el modelo, primero hay que declarar el dataset que se usará y este se almacenará en un dataset de pandas. Seguido de esto debemos definir nuestras variables Y y X, en la primera almacenaremos la variable que nos interesa predecir y en la segunda todas las demás variables independientes que nos proporcionarán información.

En este caso separaremos el dataset en una proporción de 80/20 para datos de entrenamiento y datos de prueba. Más adelante haremos pruebas con otras proporciones.

A continuación agregaremos nuestra constante aleatoria a ambos datasets y prácticamente contamos con todo lo necesario para entrenar nuestro modelo. Finalmente haremos uso de la función OLS(y,x).fit() para entrenar nuestro modelo y lo probaremos ingresando los datos del dataset de pruebas.

Para obtener más información de nuestro modelo haremos uso del método summary() y de la librería matplotlib para obtener información más gráfica.

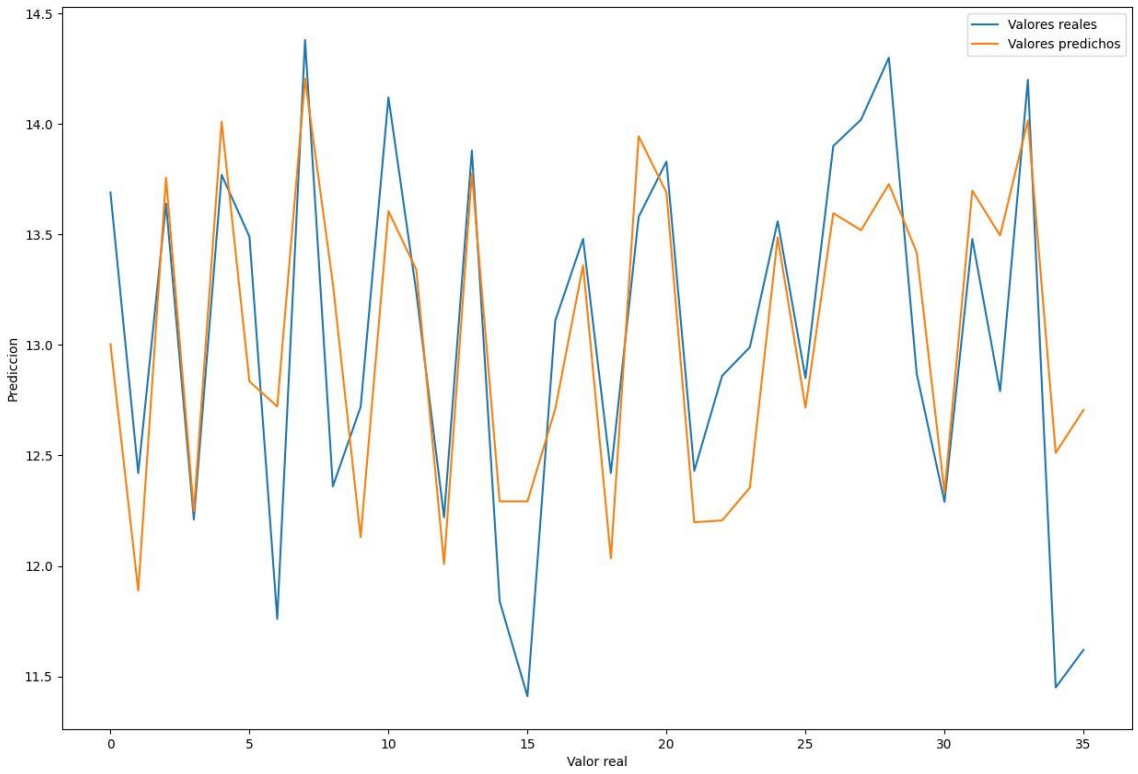


```
Artificial/perceptron2.py
                            OLS Regression Results
==============================================================================
Dep. Variable:                alcohol   R-squared:                       0.609
Model:                            OLS   Adj. R-squared:                  0.569
Method:                 Least Squares   F-statistic:                     15.33
Date:                Mon, 12 Sep 2022   Prob (F-statistic):           2.16e-20
Time:                        21:46:47   Log-Likelihood:                -103.31
No. Observations:                 142   AIC:                             234.6
Df Residuals:                     128   BIC:                             276.0
Df Model:                          13
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                           12.7404      0.881     14.465      0.000      10.998      14.483
class                           -0.5009      0.176     -2.846      0.005      -0.849      -0.153
malic acid                       0.1467      0.050      2.963      0.004       0.049       0.245
ash                              0.2963      0.253      1.169      0.245      -0.205       0.798
alcalinity of ash               -0.0336      0.022     -1.497      0.137      -0.078       0.011
magnesium                       -0.0025      0.004     -0.634      0.528      -0.010       0.005
total phenols                    0.2107      0.154      1.368      0.174      -0.094       0.515
flavanoids                      -0.2144      0.136     -1.577      0.117      -0.483       0.055
nonflavanoid phenols            -0.5457      0.504     -1.083      0.281      -1.543       0.451
proanthocyanins                 -0.1304      0.107     -1.221      0.224      -0.342       0.081
color intensity                  0.2007      0.031      6.468      0.000       0.139       0.262
hue                              0.1537      0.323      0.477      0.634      -0.485       0.792
OD280/OD315 of diluted wines     0.0018      0.131      0.014      0.989      -0.257       0.260
proline                          0.0004      0.000      1.347      0.180      -0.000       0.001
==============================================================================
Omnibus:                        1.944   Durbin-Watson:                   2.162
Prob(Omnibus):                  0.378   Jarque-Bera (JB):                1.832
Skew:                           0.277   Prob(JB):                        0.400
Kurtosis:                       2.945   Cond. No.                     1.66e+04
==============================================================================
```

```python
#Graficamos los resultados
plt.figure(figsize=(15, 15))
plt.plot(yTest.reset_index(drop=True), label='Valores reales')
plt.plot(prediction.reset_index(drop=True), label='Valores predichos')
plt.legend()
plt.xlabel('Valor real')
plt.ylabel('Prediccion')
plt.show()
```

Podemos observar que nuestro modelo logró una precisión de R cuadrada ajustada de 0.569, lo cual si bien no es un buen resultado final, sí nos puede servir como un buen punto de partida. Los coeficientes en la parte inferior nos indican qué tan fuertemente está correlacionada una

variable X con nuestra variable Y. Mientras mayor sea este coeficiente de correlación, más cambiará esta variable con respecto a la variable Y.



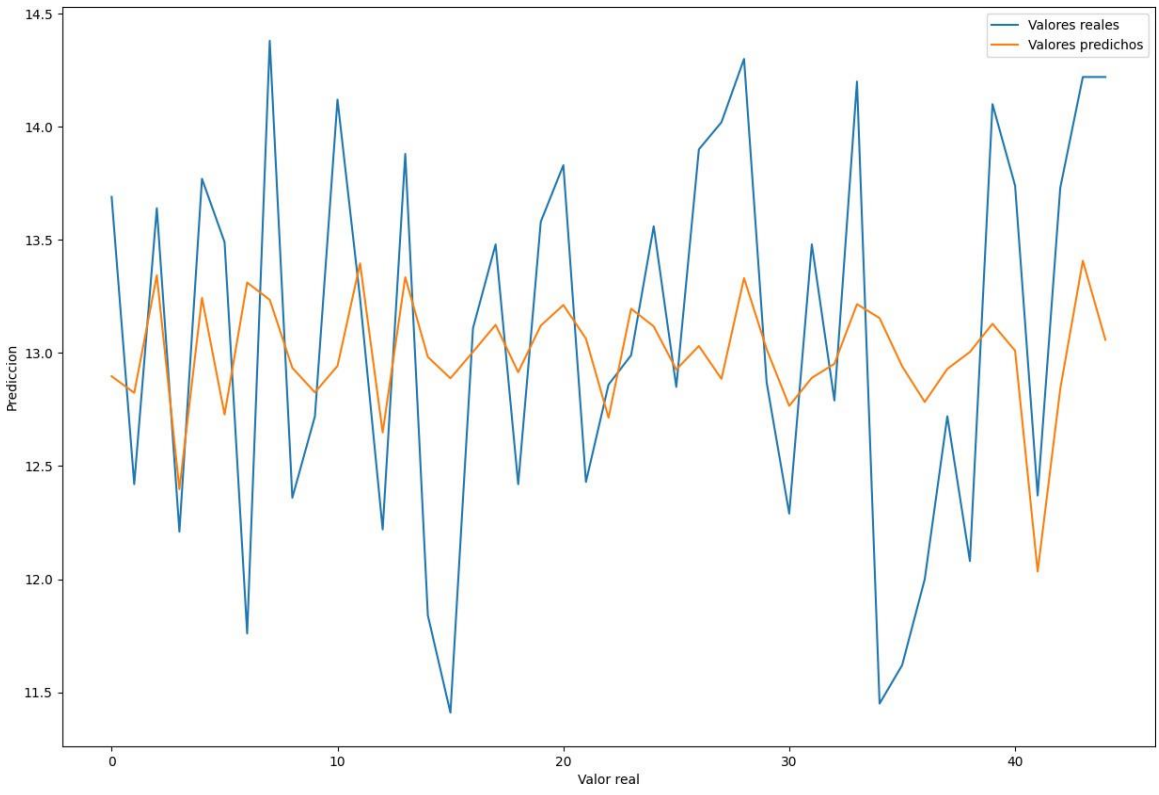
Esta es la gráfica que obtuvimos, la cual nos demuestra que hay valores en los que hemos tenido una buena aproximación, pero muchos otros en los que simplemente no. Además, con ayuda de la función mean_squared_error(y,x) podemos saber que con estas características de entrada y esta distribución, nuestro modelo tiene un MSE de 0.2766, el cual no está tan mal.

Finalmente haremos más pruebas variando las características de entrada y la distribución del dataset.

TEST 2:

```
dfX = df[['flavanoids', 'malic acid', 'ash']]
dfY = df['alcohol']
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                alcohol   R-squared:                       0.110
Model:                            OLS   Adj. R-squared:                  0.091
Method:                 Least Squares   F-statistic:                     5.709
Date:                Mon, 12 Sep 2022   Prob (F-statistic):            0.00103
Time:                        22:14:10   Log-Likelihood:                -161.67
No. Observations:                 142   AIC:                             331.3
Df Residuals:                     138   BIC:                             343.2
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         11.1948      0.556     20.124      0.000      10.095      12.295
flavanoids     0.2109      0.071      2.973      0.003       0.071       0.351
malic acid     0.1262      0.062      2.047      0.043       0.004       0.248
ash            0.4572      0.239      1.915      0.058      -0.015       0.929
==============================================================================
Omnibus:                        2.788   Durbin-Watson:                   1.897
Prob(Omnibus):                  0.248   Jarque-Bera (JB):                2.697
Skew:                          -0.335   Prob(JB):                        0.260
Kurtosis:                       2.919   Cond. No.                         38.1
==============================================================================
```



MSE: 0.6217

TEST 3:

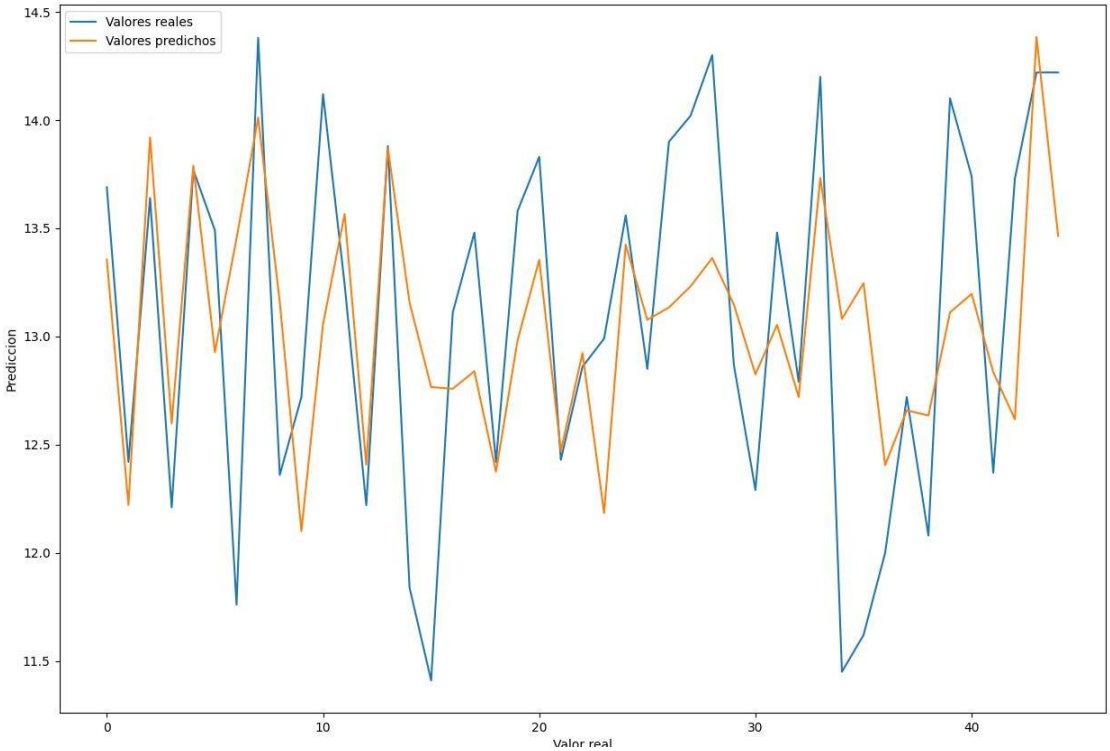
```
dfX = df[['flavanoids', 'malic acid', 'ash','magnesium','total phenols','alcalinity of ash']]
dfY = df['alcohol']
```

OLS Regression Results

Dep. Variable:	alcohol	R-squared:	0.310
Model:	OLS	Adj. R-squared:	0.277
Method:	Least Squares	F-statistic:	9.419
Date:	Mon, 12 Sep 2022	Prob (F-statistic):	1.55e-08
Time:	22:17:14	Log-Likelihood:	-133.07
No. Observations:	133	AIC:	280.1
Df Residuals:	126	BIC:	300.4
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	11.5617	0.664	17.422	0.000	10.248	12.875
flavanoids	-0.1551	0.125	-1.240	0.217	-0.403	0.092
malic acid	0.1382	0.057	2.434	0.016	0.026	0.251
ash	1.1761	0.280	4.205	0.000	0.623	1.730
magnesium	0.0035	0.005	0.731	0.466	-0.006	0.013
total phenols	0.3452	0.193	1.792	0.076	-0.036	0.727
alcalinity of ash	-0.1280	0.024	-5.407	0.000	-0.175	-0.081

Omnibus:	1.256	Durbin-Watson:	1.974
Prob(Omnibus):	0.534	Jarque-Bera (JB):	1.129
Skew:	-0.036	Prob(JB):	0.569
Kurtosis:	2.554	Cond. No.	1.16e+03



MSE: 0.51954

Después de analizar las pruebas realizadas, podemos concluir que un factor determinante en el entrenamiento de un modelo de machine learning son los datos ingresados, ya que como podemos ver, el MSE varió dependiendo de las columnas que el modelo recibe como entrada. No obstante, la proporción de datos de entrada en el entrenamiento y en las pruebas también juega un rol importante para modificar la precisión.