Tecnológico de Monterrey

Módulo 2 Análisis y Reporte sobre el desempeño del modelo

Inteligencia artificial avanzada para la ciencia de datos I (Gpo 101)

Aylin Camacho Reyes - A01379272 (The cool other)

Ivan Mauricio Amaya Contreras

08 de Septiembre del 2022, Monterrey N.L.

Durante cinco semanas hemos visto como funcionan los modelos de predicción. En este caso se utilizo un conjunto de datos que contenia información sobre los componentes del vino, para el analisis de los datos se usaron frameworks de python que facilitaran el proceso.

La primera fase consiste explorar los datos para conocer la información que nos estan brindando, saber si tiene variables culitativas o cuantitativas, como se encuentran distribuidos los datos por medio de la estadistica descriptiva y si existen inconsistencias como valores nulos.

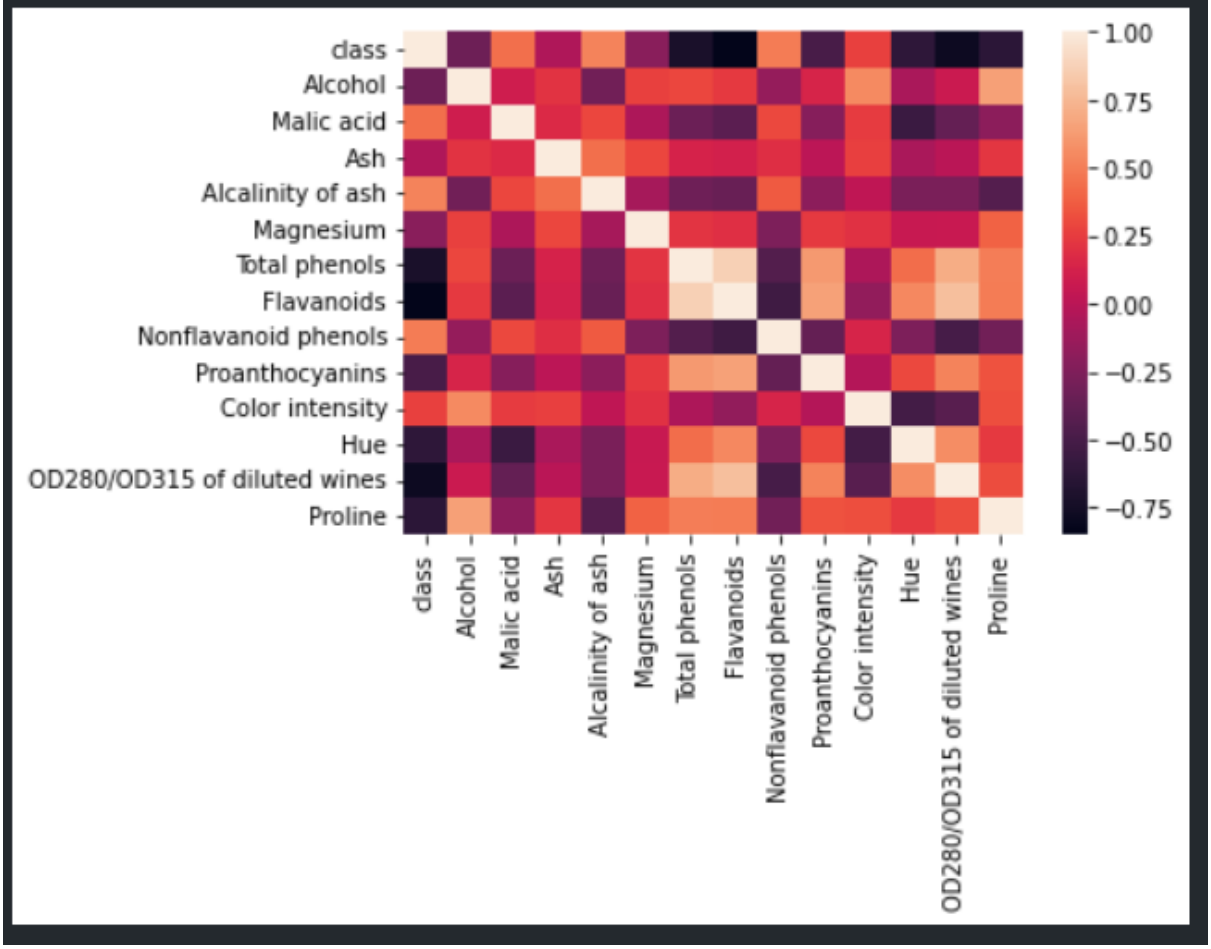
```
#exploring the data

#data types
print(df.info())

#statistical summary
print(df.describe())

#inconsistencies
print(df.isnull().sum())
print(df.isna().sum())
```

Tambien para explorar los datos se realizon un heat map para observar la relación entre las variables con el coeficiente de pearson, en este caso no consideramos que las dos variables que mostraban mayor correlación fueran a ser las unicas que determinaran el valor de la variables dependiente.

```
#Correlacion de variables
corr = df.corr()  # Coeficiente de Pearson
sb.heatmap(corr)
```

✓ 0.6s

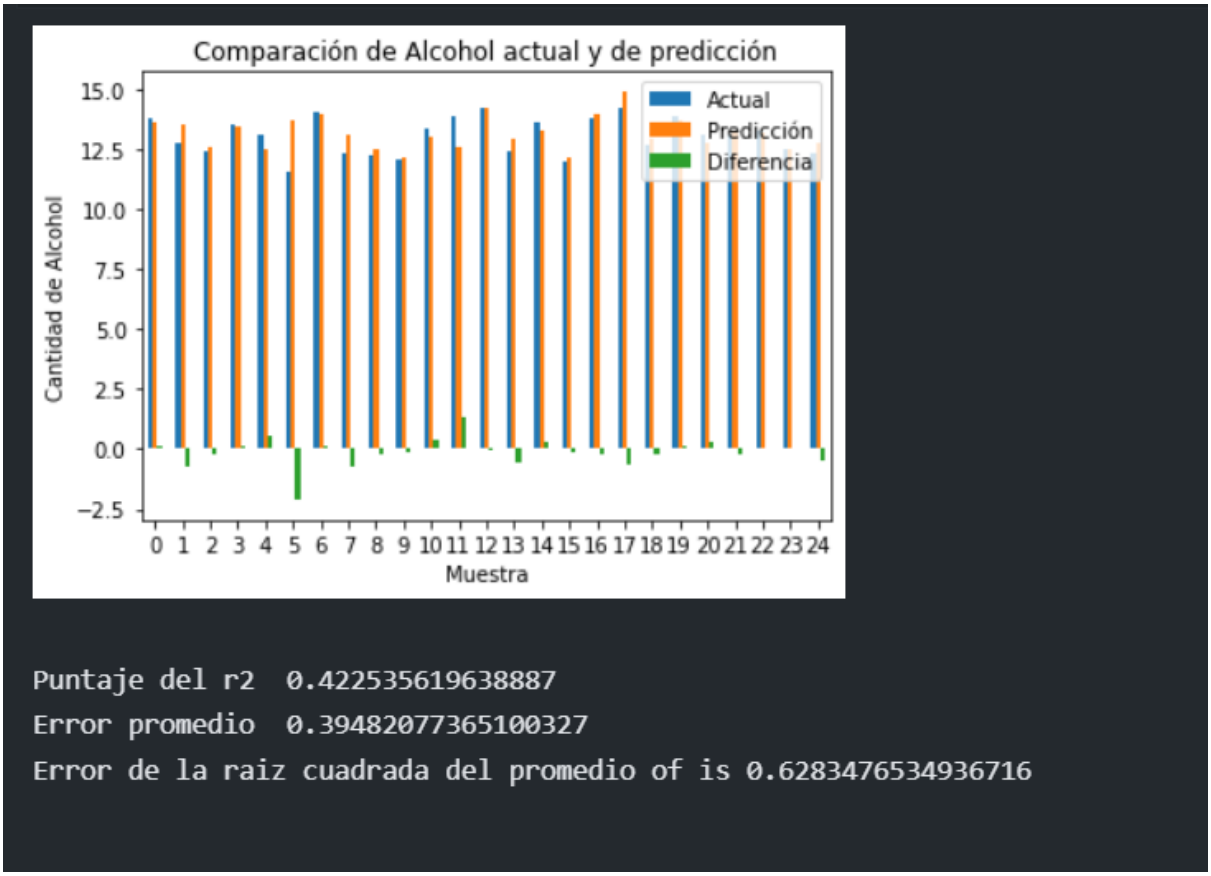
```
<AxesSubplot:>
```



Una vez entendidos los datos se pason a la parte del modelado, donde escogimos como variable dependiente el nivel de alchol del vino y como variables independientes el resto de columnas a excepción de “class” ya que era una variable de tipo cualitativa.

```
#modeling

x = df[["Malic acid", "Ash", "Alcalinity of ash", "Magnesium", "Total phenols",  "Flavanoids",
"Nonflavanoid phenols", "Proanthocyanins", "Color intensity", "Hue", "OD280/OD315 of diluted wines", "Proline"]].values
y = df[['Alcohol']].values
✓ 0.4s

X_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

modelo_regresion = LinearRegression()  # modelo de regresión
```

Para inicar el modelo se declaran las variables independientes como “x” y las independientes como “y”, para separar los datos en los que seran usados para prueba (20%) y los de validación (8%0) con la ayuda de las librerias de sklearns.




```
Puntaje del r2  0.422535619638887
Error promedio  0.39482077365100327
Error de la raiz cuadrada del promedio of is 0.6283476534936716
```

Posteriormente calculamos con la ayuda de la librerias de sklearn los grados de sesgo y varianza. Como se muestra a continuación podemos observar que el sesgo entre lo estimado en base a la muestra a lo real de la población es cercano a 0 pero sigue siendo mayor por lo que podemos asumir que hay valores más separados de la media a la derecha. Mientras la varianza se encuentra cercana a cero por lo que nos indica en que nivel los datos de la muestra son diferentes de otra muestra.

```python
#calculate the bias and variance of the model
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

y_train_pred = cross_val_predict(modelo_regresion, x, y)
print('Bias: %.3f' % (np.mean(y_train_pred - y)))
print('Variance: %.3f' % (np.mean((y_train_pred - y)**2)))
print('R2: %.3f' % (r2_score(y, y_train_pred)))
```

✓ 0.1s

```
Bias: 0.020
Variance: 0.382
R2: 0.418
```