julihocc-s22

May 5, 2024

1 Semana 22 | Regresiones regularizadas

Presenta Juliho David Castillo Colmenares

- Considere nuevamente la base de datos siguiente: FuelConsumptionCo2.xlsx

[ ]:
```
import pandas as pd

data = pd.read_csv('data.csv')
data.head()
```

[ ]:

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	\
0	2022	Acura	ILX	Compact	2.4	4	
1	2022	Acura	MDX SH-AWD	SUV: Small	3.5	6	
2	2022	Acura	RDX SH-AWD	SUV: Small	2.0	4	
3	2022	Acura	RDX SH-AWD A-SPEC	SUV: Small	2.0	4	
4	2022	Acura	TLX SH-AWD	Compact	2.0	4	

	TRANSMISSION	FUELTYPE	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY	\
0	AM8	Z	9.9	7.0	
1	AS10	Z	12.6	9.4	
2	AS10	Z	11.0	8.6	
3	AS10	Z	11.3	9.1	
4	AS10	Z	11.2	8.0	

	FUELCONSUMPTION_COMB	FUELCONSUMPTION_COMB_MPG	CO2EMISSIONS
0	8.6	33	200
1	11.2	25	263
2	9.9	29	232
3	10.3	27	242
4	9.8	29	230

- Elimine las columnas de tipo categórico de la base de datos y verifique que no existan datos nulos que deban ser eliminados de manera previa al análisis.

[ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 945 entries, 0 to 944
Data columns (total 13 columns):
```

```
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   MODELYEAR                 945 non-null    int64
 1   MAKE                      945 non-null    object
 2   MODEL                     945 non-null    object
 3   VEHICLECLASS              945 non-null    object
 4   ENGINESIZE                945 non-null    float64
 5   CYLINDERS                 945 non-null    int64
 6   TRANSMISSION              945 non-null    object
 7   FUELTYPE                  945 non-null    object
 8   FUELCONSUMPTION_CITY      945 non-null    float64
 9   FUELCONSUMPTION_HWY       945 non-null    float64
 10  FUELCONSUMPTION_COMB      945 non-null    float64
 11  FUELCONSUMPTION_COMB_MPG  945 non-null    int64
 12  CO2EMISSIONS              945 non-null    int64
dtypes: float64(4), int64(4), object(5)
memory usage: 96.1+ KB
```

[ ]:
```
import numpy as np

data = data.select_dtypes(include=np.number)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 945 entries, 0 to 944
Data columns (total 8 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   MODELYEAR                 945 non-null    int64
 1   ENGINESIZE                945 non-null    float64
 2   CYLINDERS                 945 non-null    int64
 3   FUELCONSUMPTION_CITY      945 non-null    float64
 4   FUELCONSUMPTION_HWY       945 non-null    float64
 5   FUELCONSUMPTION_COMB      945 non-null    float64
 6   FUELCONSUMPTION_COMB_MPG  945 non-null    int64
 7   CO2EMISSIONS              945 non-null    int64
dtypes: float64(4), int64(4)
memory usage: 59.2 KB
```

[ ]:
```
data.isna().sum()
```

[ ]:
```
MODELYEAR                 0
ENGINESIZE                0
CYLINDERS                 0
FUELCONSUMPTION_CITY      0
FUELCONSUMPTION_HWY       0
FUELCONSUMPTION_COMB      0
FUELCONSUMPTION_COMB_MPG  0
```

```
CO2EMISSIONS                0
dtype: int64
```

- Realice un análisis de regresión múltiple para pronosticar la variable "CO2 EMISSIONS" con las variables remanentes. Obtenga los indicadores de bondad de ajuste correspondientes (R cuadrada, Error medio absoluto, etc.). Utilice los coeficientes resultantes y pronostique la primera observación de la base de prueba. ¿Coincide su resultado con aquel obtenido con la instrucción "predict"? Explique.

[ ]:
```
target = 'CO2EMISSIONS'
X = data.drop(target, axis=1)
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 945 entries, 0 to 944
Data columns (total 7 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   MODELYEAR                 945 non-null    int64
 1   ENGINESIZE                945 non-null    float64
 2   CYLINDERS                 945 non-null    int64
 3   FUELCONSUMPTION_CITY      945 non-null    float64
 4   FUELCONSUMPTION_HWY       945 non-null    float64
 5   FUELCONSUMPTION_COMB      945 non-null    float64
 6   FUELCONSUMPTION_COMB_MPG  945 non-null    int64
dtypes: float64(4), int64(3)
memory usage: 51.8 KB
```

[ ]:
```
y = data[target]
y.info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 945 entries, 0 to 944
Series name: CO2EMISSIONS
Non-Null Count  Dtype
--------------  -----
945 non-null    int64
dtypes: int64(1)
memory usage: 7.5 KB
```

[ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[ ]:
```
# Linear Regression
from sklearn.linear_model import LinearRegression

model = LinearRegression()
```

```
model.fit(X_train, y_train)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
# Model report

from sklearn.metrics import mean_squared_error, r2_score

def model_report(model, X_test, y_test):
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f'Mean Squared Error: {mse}')
    print(f'R2 Score: {r2}')

model_report(model, X_test, y_test)
```

```
Mean Squared Error: 368.8672604772689
R2 Score: 0.9004924315633444
```

[ ]:
```
pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
```

[ ]:

	Coefficient
MODELYEAR	0.000000
ENGINESIZE	0.229055
CYLINDERS	4.064256
FUELCONSUMPTION_CITY	1.216424
FUELCONSUMPTION_HWY	3.522217
FUELCONSUMPTION_COMB	12.668709
FUELCONSUMPTION_COMB_MPG	-1.258234

[ ]:
```
# predict the test set with model.coef_ and model.intercept_
def linreg_predict(X_test, model):
    return model.coef_ @ X_test.T + model.intercept_

y_pred = linreg_predict(X_test, model)
y_pred.iloc[0]
```

[ ]:
```
242.9269403739375
```

[ ]:
```
model.predict(X_test)[0]
```

[ ]:
```
242.9269403739375
```

- Repita el ejercicio 3 pero aplicando un modelo de regresión Ridge mediante el valor de Alpha óptimo.

[ ]:
```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV

param_grid = {'alpha': [0.001, 0.01, 0.1, 1, 10, 100]}
grid_search = GridSearchCV(Ridge(), param_grid, cv=5)
grid_search.fit(X_train, y_train)

best_alpha = grid_search.best_params_['alpha']
print("Parámetro alfa óptimo:", best_alpha)
```

```
Parámetro alfa óptimo: 100
```

[ ]:
```
model_ridge = Ridge(alpha=best_alpha)
model_ridge.fit(X_train, y_train)
model_report(model_ridge, X_test, y_test)
```

```
Mean Squared Error: 356.3944320791084
R2 Score: 0.903857167224142
```

[ ]:
```
pd.DataFrame(model_ridge.coef_, X.columns, columns=['Coefficient'])
```

[ ]:

	Coefficient
MODELYEAR	0.000000
ENGINESIZE	0.928335
CYLINDERS	3.611584
FUELCONSUMPTION_CITY	5.153768
FUELCONSUMPTION_HWY	5.990494
FUELCONSUMPTION_COMB	5.614007
FUELCONSUMPTION_COMB_MPG	-1.404881

[ ]:
```
linreg_predict(X_test, model_ridge).iloc[0]
```

[ ]:
```
243.94905139169552
```

[ ]:
```
model_ridge.predict(X_test)[0]
```

[ ]:
```
243.94905139169552
```

- Repita el ejercicio 3 pero aplicando un modelo de regresión Lasso mediante el valor de Alpha óptimo.

[ ]:
```
import warnings
warnings.filterwarnings('ignore')

from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV

param_grid = {'alpha': [0.001, 0.01, 0.1, 1, 10]}
grid_search = GridSearchCV(Lasso(), param_grid, cv=5)  # 5-fold cross-validation
```

```
grid_search.fit(X, y)

best_alpha = grid_search.best_params_['alpha']
print("Best alpha:", best_alpha)
```

```
Best alpha: 0.1
```

[ ]:
```
model_lasso = Lasso(alpha=best_alpha)
model_lasso.fit(X_train, y_train)
model_report(model_lasso, X_test, y_test)
```

```
Mean Squared Error: 368.86948575942824
R2 Score: 0.9004918312595481
```

[ ]:
```
pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
```

[ ]:

	Coefficient
MODELYEAR	0.000000
ENGINESIZE	0.229055
CYLINDERS	4.064256
FUELCONSUMPTION_CITY	1.216424
FUELCONSUMPTION_HWY	3.522217
FUELCONSUMPTION_COMB	12.668709
FUELCONSUMPTION_COMB_MPG	-1.258234

[ ]:
```
linreg_predict(X_test, model_lasso).iloc[0]
```

[ ]:
```
243.67428466071166
```

[ ]:
```
model_lasso.predict(X_test)[0]
```

[ ]:
```
243.67428466071166
```

- ¿Cuál de los 3 modelos de regresión resultó ser el mejor? Explique a detalle.

[ ]:
```
r2_scores = pd.DataFrame({'Linear Regression': r2_score(y_test, model.
 ↪predict(X_test)),
                          'Ridge': r2_score(y_test, model_ridge.
 ↪predict(X_test)),
                          'Lasso': r2_score(y_test, model_lasso.
 ↪predict(X_test))}, index=['R2 Score']).T
r2_scores.sort_values(by='R2 Score', ascending=False)
```

[ ]:

	R2 Score
Ridge	0.903857
Linear Regression	0.900492
Lasso	0.900492

1.1 Conclusión

En base a los resultados del análisis de regresión realizado, se observa que los tres modelos evaluados (Ridge, Regresión Lineal y Lasso) presentan un buen ajuste a los datos, con un valor de R2 cercano a 0.9. Sin embargo, se destacan dos puntos importantes:

- **El modelo Ridge presenta el mayor valor de R2 (0.903857),** lo que indica que este modelo es el que mejor explica la variabilidad de la variable dependiente en función de las variables independientes.
- **Los modelos de Regresión Lineal y Lasso presentan un rendimiento muy similar, con valores de R2 de 0.900492 para ambos casos.** Esto sugiere que ambos modelos podrían ser alternativas viables al modelo Ridge, especialmente si se busca una mayor simplicidad o se tienen en cuenta consideraciones de penalización por complejidad.

En definitiva, la elección del modelo más adecuado dependerá de los objetivos específicos del análisis y de las características del conjunto de datos. Si se busca el mejor ajuste posible, el modelo Ridge se presenta como la mejor opción. Sin embargo, si se prioriza la simplicidad o se tienen en cuenta consideraciones de penalización por complejidad, los modelos de Regresión Lineal o Lasso podrían ser alternativas viables.

Se recomienda realizar un análisis más profundo de los tres modelos para evaluar su comportamiento en términos de sesgo, varianza y error de predicción. Además, se podrían considerar otros modelos de regresión no lineales si se sospecha que la relación entre las variables no es lineal.

Es importante tener en cuenta que la interpretación de los coeficientes de regresión solo es válida si se cumplen los supuestos del modelo de regresión lineal. Se recomienda realizar un análisis de los supuestos para garantizar la confiabilidad de los resultados.