S4- REPLICA DE CASOS SKLEARN

Elaborado por	Bryan Stalin Pazmiño Salazar	
Dirigido a:	PhD. Marcela Cevallos Herrera	**Fecha de entrega**
		22-05-2023

Caso práctico

Replicar su tarea mediante la librería sklearn, es decir mediante el enfoque de ciencia de datos.

Instrucciones

PARTE 1.

- Importe la base de datos de la tarea de la semana 1 (Dummy Data.csv) en Jupyter Notebook.

```
In [52]: #PUNTO 1#
         Ruta="D:/Academico/UDLA/Análitica Predictiva/Tarea 1/"
         df = pd.read_csv(Ruta+'Dummy Data HSS(1).csv', sep=',')
         df
```

```
Out[52]: 0        6.566231
         1        9.237765
         2       15.886446
         3       30.020028
         4        8.437408
                   ...
         4567     4.472360
         4568    20.610685
         4569    19.800072
         4570    17.534640
         4571    15.966688
         Name: Radio, Length: 4572, dtype: float64
```

```
In [43]: df.isnull().sum()
         df
         print(df.isnull().sum())
         #Punto 2#
         #Variable Objetivo# #sales#
         #Variables Independientes# #Social Media y Radio#
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LinearRegression
         from sklearn.preprocessing import LabelEncoder
```

```
TV              10
Radio            4
Social Media     6
Influencer       0
Sales            6
dtype: int64
```

Recordemos que la base de datos de la semana 1 contenía valores *"null"* por lo que se debía efectuar una limpieza de datos.

- **Escoja su variable objetivo y las variables independientes considerando un enfoque de regresión lineal.**

Se escogió la variable objetivo “Sales” que en español son ventas, ya que al ser esta dependiente la única en realidad es la mas acertada para el análisis.

- **Realice un train/test split, separando un 90% de los datos para la submuestra de entrenamiento y 10% para la submuestra de prueba.**

```
#PUNTO 3#
X_train , X_test , y_train , y_test = train_test_split(X , y , test_size = 0.10,random_state =123)
```

```
print(X_train.shape,"",type(X_train))
print(y_train.shape,"\t ",type(y_train))
print(X_test.shape,"",type(X_test))
print(y_test.shape,"\t ",type(y_test))
```

```
(4091, 4)  <class 'pandas.core.frame.DataFrame'>
(4091,)          <class 'pandas.core.series.Series'>
(455, 4)  <class 'pandas.core.frame.DataFrame'>
(455,)    <class 'pandas.core.series.Series'>
```

- **Entrene al modelo de regresión lineal por sklearn.**

```
#PUNTO 4#
#El Modelo de Regresión Lineal por Sklearn#
```

```
modelo_regresion = LinearRegression()
modelo_regresion.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
predicciones_train = modelo_regresion.predict(X_train)
predicciones_test = modelo_regresion.predict(X_test)
```

- **Evalúe su modelo. ¿Es este aceptable?, para ello escoja las métricas correspondientes.**

```
#Punto 5#
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

```
  #MSE#
MSE_train = mean_squared_error(y_train, predicciones_train)
MSE_test = mean_squared_error(y_test, predicciones_test)
print(MSE_train)
print(MSE_test)
```

```
8.653104853739151
9.053409318682087
```

```
  #RMSE#
RMSE_train = np.sqrt(MSE_train)
RMSE_test = np.sqrt(MSE_test)
print(RMSE_train)
print(RMSE_test)
```

```
2.9416160275840135
3.008888385879757
```

```
  #MAE#
MAE_train = mean_absolute_error(y_train, predicciones_train)
MAE_test = mean_absolute_error(y_test, predicciones_test)
print(MAE_train)
print(MAE_test)
```

```
2.3584650397450866
2.420156808073673
```

```
  #R^2#
from sklearn.metrics import r2_score
```

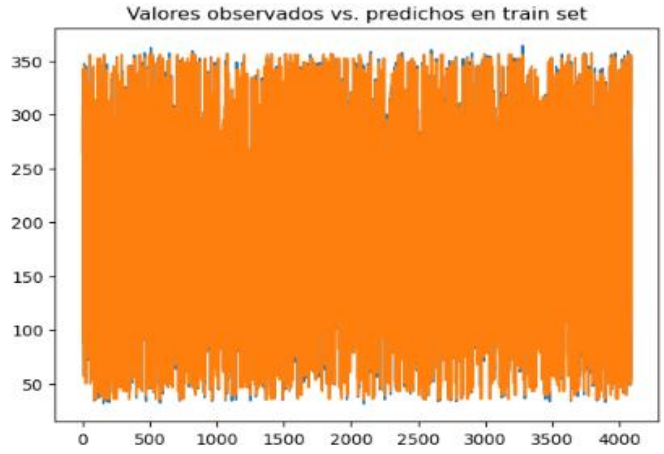
```
r_square_train = r2_score(y_train, predicciones_train)
r_square_test  = r2_score(y_test, predicciones_test)
print('El R^2 del subconjunto de entrenamiento es:' , r_square_train)
print('El R^2 del subconjunto de prueba es:' , r_square_test)
```

```
El R^2 del subconjunto de entrenamiento es: 0.9990026991367676
El R^2 del subconjunto de prueba es: 0.9989196736539484
```

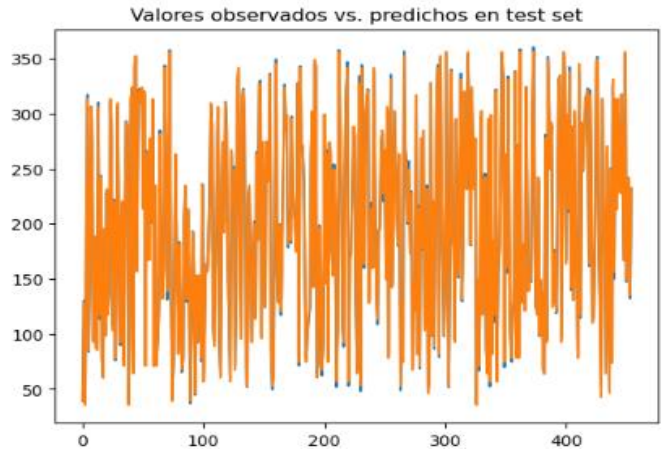
En base a las pruebas realizadas se puede concluir que el modelo es aceptable, en función de que los valores de las pruebas efectuadas indica que el modelo es capaz de ajustarse tanto a los datos de entrenamiento como a los datos de prueba (MSE), también indica el modelo tiene un buen ajuste a los datos (RMSE), por lo cual Un (MAE) bajo indica un mejor ajuste del modelo.

Para concluir sobre este tema el Coeficiente de determinación (R^2) cuanto más cercano a 1 sea el valor, mejor será el ajuste del modelo y como podemos observar es muy cercano a 1 lo que significa que el modelo será de mejor ajuste.

- **Compare sus predicciones con los datos reales mediante un gráfico.**




```
83]: fig, ax = plt.subplots()
     ax.plot(y_test.values)
     ax.plot(predicciones_test)
     plt.title("Valores observados vs. predichos en test set");
```



En el entrenamiento se puede ver en todos los datos que se ajustan al modelo en cambio en el gráfico de predicciones, como resultado, los valores que predice el modelo son bastante similares a los observados. Esto se refleja en las métricas de evaluación que arrojaron valores satisfactorios.

- **Identifique a las dos variables con mayor poder explicativo en el modelo ¿Cómo las identificó?**

```
# Print the Intercept:
print('intercepto:', modelo_regresion.intercept_)

# Print the Slope:
print('pendiente:', modelo_regresion.coef_)
```

```
intercepto: -0.07816727006067481
pendiente: [-2.48816612e-02 -3.64904345e-04 -8.46181382e-04  3.56151539e+00]
```

Las variables con mayor poder explicativo debido a su pendiente en el modelo son:

- Influencer.
- TV.

Esto en función de que la variable con la pendiente de mayor valor absoluto es la última (Influencer), con un valor aproximado de 3.5615. Esto indica que esta variable tiene una influencia significativa en la variable dependiente, al igual pasa con la pendiente de TV 2.48166.

PARTE 2

1. **Importe la base de datos de la tarea de la semana 2 (bank-additional-full)**

```
#Punto 1
Ruta2="D:/Academico/UDLA/Análitica Predictiva/Tarea 2/"
dfp2 = pd.read_csv(Ruta2+'bank-additional-full.csv', sep=';')
dfp2
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	e
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	1	999	0	nonexistent	
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	nonexistent	
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	...	2	999	0	nonexistent	
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	nonexistent	
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	3	999	1	failure	

2. **Escoja su variable objetivo y las variables independientes considerando un enfoque de regresión logística.**
   La Variable Objetivo seleccionada es "y", que son las personas que si eligieron hacer sus depositos o no.
   En este caso la base de datos de la semana 2 no contenía valores *"null"* por lo que no se debía efectuar una limpieza de datos.

3. **Realice un train/test split, separando un 90% de los datos para la submuestra de entrenamiento y 10% para la submuestra de prueba.**

```
#PUNTO 3#
X_train , X_test , y_train , y_test = train_test_split(X , y , test_size = 0.10,random_state =123)
print(X_train.shape,"",type(X_train))
print(y_train.shape,"\t ",type(y_train))S
print(X_test.shape,"",type(X_test))
print(y_test.shape,"\t ",type(y_test))
```

```
(37069, 20)  <class 'pandas.core.frame.DataFrame'>
(37069,)          <class 'pandas.core.series.Series'>
(4119, 20)  <class 'pandas.core.frame.DataFrame'>
(4119,)           <class 'pandas.core.series.Series'>
```

4. **Entrene al modelo de regresión logística por sklearn.**

```
#PUNTO 4#
#El Modelo de Regresión Lineal por Sklearn#
```

```
modelo_regresion = LinearRegression()
modelo_regresion.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
predicciones_train = modelo_regresion.predict(X_train)
predicciones_test = modelo_regresion.predict(X_test)
```

5. **Evalúe su modelo ¿Es este aceptable?, por ello escoja las métricas correspondientes.**

```
#Punto 5#
  #MSE#
MSE_train = mean_squared_error(y_train, predicciones_train)
MSE_test = mean_squared_error(y_test, predicciones_test)
print(MSE_train)
print(MSE_test)
```

```
0.06591981239851157
0.06339923938444496
```

```
  #RMSE#
RMSE_train = np.sqrt(MSE_train)
RMSE_test = np.sqrt(MSE_test)
print(RMSE_train)
print(RMSE_test)
```

```
0.25674853923345226
0.25179205584061815
```

```
#MAE#
MAE_train = mean_absolute_error(y_train, predicciones_train)
MAE_test = mean_absolute_error(y_test, predicciones_test)
print(MAE_train)
print(MAE_test)
```

```
0.15248598111856468
0.14845695509253587
```

```
#R^2#
r_square_train = r2_score(y_train, predicciones_train)
r_square_test  = r2_score(y_test, predicciones_test)
print('El R^2 del subconjunto de entrenamiento es:' , r_square_train)
print('El R^2 del subconjunto de prueba es:' , r_square_test)
```

```
El R^2 del subconjunto de entrenamiento es: 0.34275960528526794
El R^2 del subconjunto de prueba es: 0.34595860978784376
```

Los errores que podemos ver en las pruebas MSE, RMSE, MAE son bajos basándonos en esta métrica sería un modelo aceptable, pero R^2 está muy lejano de uno por ende el ajuste del modelo no será muy aceptable.

6. **Identifique a las dos variables con mayor poder explicativo en el modelo ¿Cómo las identifico?**

```
# Print the Intercept:
print('intercepto:', modelo_regresion.intercept_)

# Print the Slope:
print('pendiente:', modelo_regresion.coef_)
```

```
intercepto: -6.573540408383271
pendiente: [ 0.00047307  0.00148551  0.00521414  0.13494211 -0.07694089  0.00327161
 -0.01634645  0.00045785  0.00357876 -0.12088997  0.07554332  0.0002141
  0.00060054 -0.00056638  0.00856121 -0.01239298 -0.00114533 -0.00022807
  0.05784958  0.00451976]
```

- **Education**
- **day_of_week**

Esto en función de las pendientes calculadas