UNIVERSIDAD DE LAS AMÉRICAS

FACULTAD DE POSGRADOS

MAESTRÍA EN ANALÍTICA PREDICTIVA

S4- REPLICA DE CASOS SKLEARN

MsC. Andrea Escobar

Ana Cristina Prado

Gladys Margarita Yambay

18 de marzo de 2024

Contenido

Introducción

La ciencia de datos es el estudio de datos con el fin de extraer información significativa combinando principios y prácticas del campo de las matemáticas, la estadística, la inteligencia artificial o machine learning para analizar grandes cantidades de datos y lograr descubrir patrones o insights y utilizarlos como guía para la toma de decisiones y la planificación estratégica (IBM, 2024).

La analítica predictiva utiliza datos históricos para realizar previsiones precisas sobre los patrones de datos que pueden producirse. Se caracteriza por el uso de técnicas como el machine learning, previsión, patrones o modelo predictivo, mediante el entrenamiento a las computadoras para localizar conexiones de causalidad (AWS, 2024).

Para el modelo y análisis predictivo, se puede utilizar en Python la librería llamada: scikit-learn, también llamada sklearn, la cual contiene un conjunto de rutinas para hacer el análisis predictivo, que incluyen clasificadores, algoritmos de clusterización, etc. Está basada en NumPy, SciPy y matplotlib, de forma que es fácil reaprovechar el código que use estas librerías.

Objetivo

Realizar el siguiente análisis:

- Replicar la tarea mediante la librería sklearn, es decir mediante el enfoque de Machine Learning.

Desarrollo

Parte 1

1. **Importe la base de datos de la tarea de la semana 1 (DummyDataHSS.csv) en Jupyter Notebook.**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")

df=pd.read_csv("data/DummyDataHSS.csv")
df
```

	TV	Radio	Social Media	Influencer	Sales
0	16.0	6.566231	2.907983	Mega	54.732757
1	13.0	9.237765	2.409567	Mega	46.677897
2	41.0	15.886446	2.913410	Mega	150.177829
3	83.0	30.020028	6.922304	Mega	298.246340
4	15.0	8.437408	1.405998	Micro	56.594181
...	...	...	...	...	...
4567	26.0	4.472360	0.717090	Micro	94.685866
4568	71.0	20.610685	6.545573	Nano	249.101915
4569	44.0	19.800072	5.096192	Micro	163.631457
4570	71.0	17.534640	1.940873	Macro	253.610411
4571	42.0	15.966688	5.046548	Micro	148.202414

4572 rows × 5 columns

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

   Se revisa a continuación la información descriptiva, valores nulos, perdidos, atípicos en las variables.

```
df.describe()
```

	TV	Radio	Social Media	Sales
count	4562.000000	4568.000000	4566.000000	4566.000000
mean	54.066857	18.160356	3.323956	192.466602
std	26.125054	9.676958	2.212670	93.133092
min	10.000000	0.000684	0.000031	31.199409
25%	32.000000	10.525957	1.527849	112.322882
50%	53.000000	17.859513	3.055565	189.231172
75%	77.000000	25.649730	4.807558	272.507922
max	100.000000	48.871161	13.981662	364.079751

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4572 entries, 0 to 4571
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   TV            4562 non-null   float64
 1   Radio         4568 non-null   float64
 2   Social Media  4566 non-null   float64
 3   Influencer    4572 non-null   object 
 4   Sales         4566 non-null   float64
dtypes: float64(4), object(1)
memory usage: 178.7+ KB
```

Revisamos datos perdidos.

```
df.isna().sum()
```

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

```
df[(df['TV'].isna()==True)].TV
```

```
13    NaN
26    NaN
46    NaN
75    NaN
99    NaN
119   NaN
141   NaN
163   NaN
183   NaN
210   NaN
Name: TV, dtype: float64
```

Los tipos de variables identificadas son:

- Variables numéricas: TV, Radio, Social Media, Sales.
- Variables categóricas: Influencer.

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

```
from sklearn.linear_model import LinearRegression
```

```
var_cuantitativas = df.select_dtypes('number').columns
var_cualitativas  =df.select_dtypes('object').columns
```

```
df.Influencer
```

```
0        Mega
1        Mega
2        Mega
3        Mega
4       Micro
        ...
4567    Micro
4568     Nano
4569    Micro
4570    Macro
4571    Micro
Name: Influencer, Length: 4572, dtype: object
```

```
from sklearn.preprocessing import LabelEncoder
```

```
# creating instance of labelencoder
labelencoder = LabelEncoder()
```

```
df.Influencer = labelencoder.fit_transform(df.Influencer)
```

Imputamos valores perdidos (de Nan a 0)

```
df.loc[df["TV"].isna() == True] = 0.00
df.loc[df["Social Media"].isna() == True] = 0.00
df.loc[df["Radio"].isna() == True] = 0.00
df.loc[df["Sales"].isna() == True] = 0.00
```

```
X = df[df.columns.difference(['Sales'])]
y = df.Sales
```

```
from sklearn.model_selection import train_test_split
```

```
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))
```

```
(4114, 4)  <class 'pandas.core.frame.DataFrame'>
(4114,)          <class 'pandas.core.series.Series'>
(458, 4)  <class 'pandas.core.frame.DataFrame'>
(458,)    <class 'pandas.core.series.Series'>
```

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

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

```
▾ LinearRegression
LinearRegression()
```

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

Se revisa las siguientes métricas:

MSE	Error cuadrático
RMSE	Raíz cuadrada del error cuadrático
MAE	Error absoluto medio
R^2	R cuadrado

Métricas de evaluación

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

```
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)
```

```
8.578573047377612
9.26235302768186
```

```
#RMSE

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

```
2.928920116250631
3.043411412819808
```

```
#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)
```

```
2.342956316000361
2.4322670081156677
```

```
#R2

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: 0.9990221329746033
El R^2 del subconjunto de prueba es: 0.9989833030139602
```

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

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

```
intercepto: -0.12760467026194533
pendiente: [ 0.0040457  -0.0137957   0.02028841  3.56458367]
```

Se observa por los datos obtenidos de las métricas lo siguiente:

R2 > 0.50 (0.99) lo que nos indica que las variables independientes aportan efectivamente en la explicación de la variable dependiente.

Este porcentaje de 0.99 también nos indica que el modelo es probable que este sobre ajustado, es decir la data puede estar sesgada.

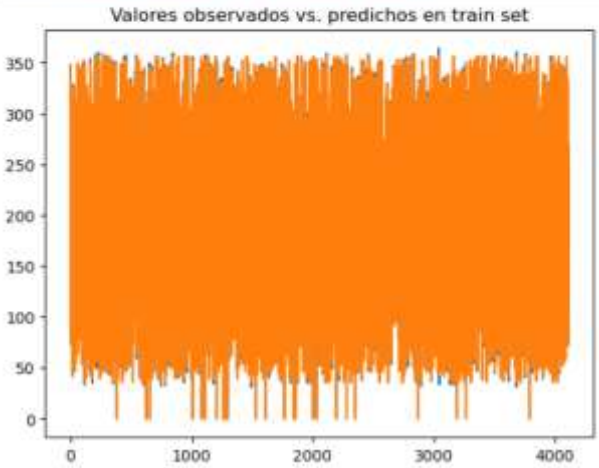
6. Compare sus predicciones con los datos reales mediante un gráfico.

6. Compare sus predicciones con los datos reales mediante un gráfico.

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

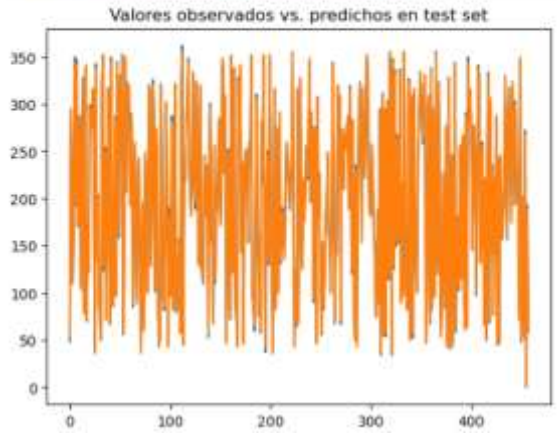
Predicciones es el subgrupo de datos de entrenamiento

```
fig, ax = plt.subplots()
ax.plot(y_train.values)
ax.plot(predicciones_train)
plt.title("Valores observados vs. predichos en train set");
```



Predicciones es el subgrupo de datos de prueba

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



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

Para calcular la importancia de cada variable independiente en el modelo ajustamos el modelo de regresión lineal y extraemos los coeficientes de los predictores, los cuales nos indicaran la importancia de los mismos en el modelo, para este análisis las variables explicativas se las va a normalización el usando de la función StandardScaler del module de sklearn.

```
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
```

```
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)
```

```
modelo_regresion_std = LinearRegression()
modelo_regresion_std.fit(X_train_std, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Coeficientes

```
importancia = modelo_regresion_std.coef_
```

```
importancia
```

```
array([ 4.50198384e-03, -1.34139077e-01,  4.50957267e-02,  9.37102035e+01])
```

```
# Resumen
for i,v in enumerate(importancia):
    print('Variable explicativa No. %0d, Score: %.5f' % (i,v))
```

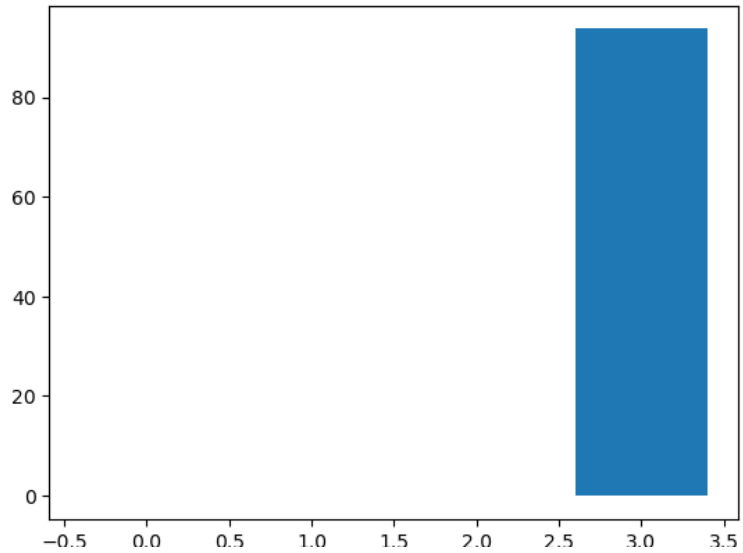
```
Variable explicativa No. 0, Score: 0.00450
Variable explicativa No. 1, Score: -0.13414
Variable explicativa No. 2, Score: 0.04510
Variable explicativa No. 3, Score: 93.71020
```

```
nom_variables = ['Influencer','Social Media', 'Radio','TV']
```

```
for nombre1, importancia1 in zip(nom_variables, importancia):
    print('Variable explicativa: %s Score: %.5f' % (nombre1, importancia1))
```

```
Variable explicativa: Influencer Score: 0.00450
Variable explicativa: Social Media Score: -0.13414
Variable explicativa: Radio Score: 0.04510
Variable explicativa: TV Score: 93.71020
```

```
# Graficar la importancia o "feature importance"
plt.bar([x for x in range(len(importancia))], importancia)
plt.show()
```



La variable explicativa más significativa según la gráfica es:

Variable explicativa No. 3, Score: 93.71 (TV).

8. **Realice un informe con sus respuestas y adjunte a la plataforma en formato PDF**

- Se revisa inicialmente los descriptivos de todas las variables, así como: valores nulos, perdidos, atípicos.
- Identificamos los tipos de variables:
    - Variables numéricas: TV, Radio, Social Media, Sales.
    - Variables categóricas: Influencer.
- Se dumifica la variable categórica de Influencer e imputamos con el valor de 0 a variables numéricas que contienen valores perdidos.
- Realizamos un análisis preliminar de los datos separando 90% para el entrenamiento y 10% para prueba
- Se entrena el modelo de regresión lineal con el enfoque de Machine Learning
- Evaluamos en el modelo con las métricas: MSE, RMSE, MAE y R^2.En el cual se observa por los datos obtenidos de las métricas lo siguiente:
    - R2 > 0.50 (0.99) lo que nos indica que las variables independientes aportan efectivamente en la explicación de la variable dependiente.
    - Este porcentaje de 0.99 también nos indica que el modelo es probable que este sobre ajustado, es decir la data puede estar sesgada.
- Al comparar las predicciones con los datos reales, mediante gráfica se observa que están muy cercanos, por lo que se puede aceptar el modelo.
- Para calcular la importancia de cada variable independiente en el modelo ajustamos el modelo de regresión lineal y extraemos los coeficientes de los predictores, los cuales nos indicaran la importancia de los mismos en el modelo, para este análisis las variables explicativas se las va a normalización el usando de la función StandardScaler del module de sklearn.
    - Encontrando que la variable explicativa más significativa según la gráfica es: Variable explicativa No. 3, Score: 93.71 (TV).

Parte 2

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

```
dfg=pd.read_csv("data/bank-additional-full.csv",delimiter=";")
dfg
```

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

41188 rows × 21 columns

2. **Escoja su variable objetivo y las variables independientes considerando un enfoque de regresión logística.**

```
dfg.describe()
```

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	93.575664	-40.502600	3.621291	5167.035911
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	0.578840	4.628198	1.734447	72.251528
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	-26.900000	5.045000	5228.100000

```
dfg.isna().sum()
```

```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

```
varg_cuantitativas = dfg.select_dtypes('number').columns
varg_cualitativas = dfg.select_dtypes('object').columns
dfg1= dfg[varg_cuantitativas]
```

```
# Creating instance of labelencoder
labelencoder = LabelEncoder()
dfg[varg_cualitativas]=dfg[varg_cualitativas].apply(LabelEncoder().fit_transform)
dfg
```

41188 rows × 21 columns

```
dfg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  int32
 2   marital         41188 non-null  int32
 3   education       41188 non-null  int32
 4   default         41188 non-null  int32
 5   housing         41188 non-null  int32
 6   loan            41188 non-null  int32
 7   contact         41188 non-null  int32
 8   month           41188 non-null  int32
 9   day_of_week     41188 non-null  int32
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  int32
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  int32
dtypes: float64(5), int32(11), int64(5)
memory usage: 4.9 MB
```

Se observa completitud de los datos 41188 registros (no hay datos perdidos) Se escoge como variable objetivo '**y**', la cual indica si cliente efectivamente va a realizar o no un depósito a plazo y las variables independientes: **age, job, marital, education, default, housing, loan, contact, month,day_of_week,duration,campaign,pdays,previous,poutcome,emp.var.rate, cons.price.idx, cons.conf.idx,nr.employed**.

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

```
X_logit = dfg[dfg.columns.difference(['y'])]
y_logit = dfg.y
```

```
X_train_logit , X_test_logit , y_train_logit , y_test_logit = train_test_split(X_logit , y_logit , test_size = 0.10,random_state =123)
```

```
print(X_train_logit.shape,"",type(X_train_logit))
print(X_train_logit.shape,"\t ",type(y_train_logit))
print(X_test_logit.shape,"",type(X_test_logit))
print(X_test_logit.shape,"\t ",type(y_test_logit))
```

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

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

```
from sklearn.linear_model import LogisticRegression
```

```
modelo_logistico = LogisticRegression()
modelo_logistico.fit(X_train_logit, y_train_logit)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
predicciones_train_logit = modelo_logistico.predict(X_train_logit)
predicciones_test_logit  = modelo_logistico.predict(X_test_logit)
```

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

Matriz de confusión

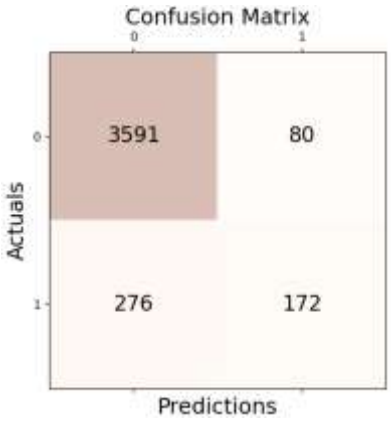
```
from sklearn import metrics
from sklearn.metrics import precision_score, recall_score, f1_score
```

```
cm = metrics.confusion_matrix(y_test_logit, predicciones_test_logit)
print(cm)
```

```
[[3591   80]
 [ 276  172]]
```

```
fig, ax = plt.subplots(figsize=(5, 5))
ax.matshow(cm, cmap=plt.cm.Oranges, alpha=0.3)
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(x=j, y=i,s=cm[i, j], va='center', ha='center', size='xx-large')

plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix', fontsize=18)
plt.show()
```



Precisión

```
print('Precision: %.3f' % precision_score(y_test_logit, predicciones_test_logit))
```

```
Precision: 0.683
```

El número de elementos identificados correctamente como positivos sobre el total de positivos verdaderos es mayor a 0.50, por lo que la precisión es aceptable.

Exhaustividad

```
print('Recall: %.3f' % recall_score(y_test_logit, predicciones_test_logit))
```

```
Recall: 0.384
```

F1 score

```
print('F1 Score: %.3f' % f1_score(y_test_logit, predicciones_test_logit))
```

```
F1 Score: 0.491
```

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

Se realiza la identificación con el análisis de: coeficientes y la característica de importancia.

```
sc_logit=StandardScaler()
```

```
X_train_std_logit = sc_logit.fit_transform(X_train_logit)
X_test_std_logit = sc_logit.transform(X_test_logit)
```

```
modelo_logistico_std = LogisticRegression()
modelo_logistico_std.fit(X_train_std_logit, y_train_logit)
```

```
▾ LogisticRegression
LogisticRegression()
```

Coeficientes

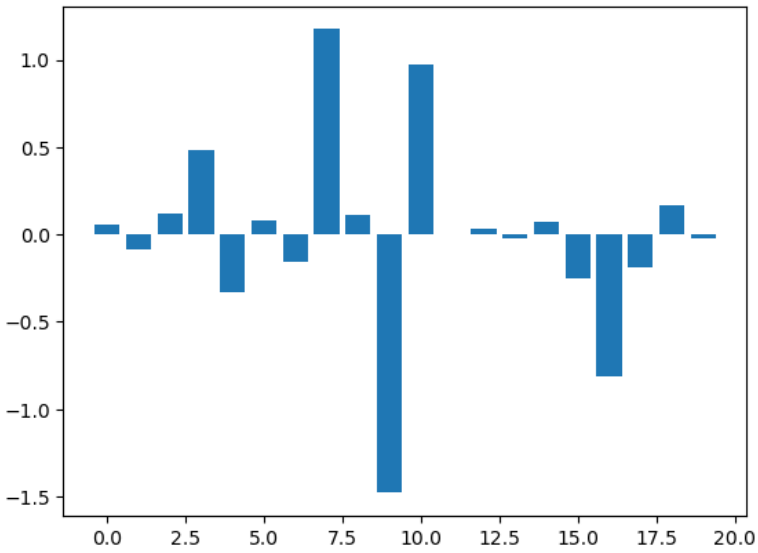
labelenc 9/9

```
importancia_logit = modelo_logistico_std.coef_[0]
```

```
# Resumen
for i,v in enumerate(importancia_logit):
    print('Variable explicativa No. %0d Score: %.5f' % (i,v))
```

```
Variable explicativa No. 0 Score: 0.05797
Variable explicativa No. 1 Score: -0.08466
Variable explicativa No. 2 Score: 0.11919
Variable explicativa No. 3 Score: 0.48147
Variable explicativa No. 4 Score: -0.32747
Variable explicativa No. 5 Score: 0.07930
Variable explicativa No. 6 Score: -0.15473
Variable explicativa No. 7 Score: 1.17450
Variable explicativa No. 8 Score: 0.11087
Variable explicativa No. 9 Score: -1.48018
Variable explicativa No. 10 Score: 0.97317
Variable explicativa No. 11 Score: -0.00121
Variable explicativa No. 12 Score: 0.03639
Variable explicativa No. 13 Score: -0.02025
Variable explicativa No. 14 Score: 0.07601
Variable explicativa No. 15 Score: -0.24897
Variable explicativa No. 16 Score: -0.81194
Variable explicativa No. 17 Score: -0.18870
Variable explicativa No. 18 Score: 0.16530
Variable explicativa No. 19 Score: -0.02531
```

```
# Graficar la importancia o "feature importance"
plt.bar([x for x in range(len(importancia_logit))], importancia_logit)
plt.show()
```



Según el feature importance, se validan las principales variables explicativas más significativas, las cuales tienen el valor de coeficiente más alto:

Variable explicativa No. 7 Score: 1.17450

Variable explicativa No. 10 Score: 0.97317

7. **Realice un informe de la parte 1 y 2, con sus respuestas y adjunte a la plataforma en formato PDF.**

- Se revisa inicialmente la descripción de todas las variables, así como: valores nulos, perdidos, atípicos.
  Se observa completitud de los datos 41188 registros (no hay datos perdidos).
  Se escoge como variable objetivo '**y**', la cual indica si cliente efectivamente va a realizar o no un depósito a plazo y las variables independientes: age, job, marital, education, default,housing,loan,contact,month,day_of_week,duration,campaign,pdays,previous, poutcome,emp.var.rate, cons.price.idx, cons.conf.idx,nr.employed.
- Realizamos un análisis preliminar de los datos separando 90% para el entrenamiento y 10% para prueba
- Se entrena el modelo de regresión logística con el enfoque de Machine Learning
- Evaluamos en el modelo con las métricas: Matriz de confusión, Precisión, Exhaustividad, F1 Score
  La Matriz de confusión es aceptable ya que de 3591 sólo 276 no concuerdan con el modelo y de 172, 80, representan más del 50% aceptable.
  Con el dato obtenido de la precisión (0.68), se identifica que número de elementos identificados correctamente como positivos sobre el total de positivos verdaderos es mayor a 0.50, por lo que la precisión es aceptable.
- Con el análisis del feature importance, se obtienen las principales variables explicativas más significativas, las cuales tienen el valor de coeficiente más alto, que son:

  Variable explicativa No. 7 Score: 1.17450

  Variable explicativa No. 10 Score: 0.97317

Conclusiones

- La libreria Scikit-Learn una vez que se entiende el uso básico y su sintaxis para un tipo de modelo, cambiar a un nuevo modelo o algoritmo es muy sencillo.

- La librería Scikit-Learn ofrece un gran número de algoritmos con campos de aplicación en muchos sectores como la industria, los seguros, la comprensión de datos de clientes, etc.

En resumen, Scikit-Learn o Sklearn es una librería de machine learning de Python. Cuenta con algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad, la cual apoyará en la ejecución de modelos de regresión lineal o logística, simplificando la validación de supuestos como es en el enfoque econométrico, en su lugar, valida directamente la revisión de las métricas de errores o para el tratamiento de valores típicos o nulos.

Referencias

AWS. (17 de 03 de 2024). *AWS*. Obtenido de AWS: https://aws.amazon.com/es/what-is/data-science/#:~:text=La%20ciencia%20de%20datos%20es,analizar%20grandes%20cantidades%20de%20datos.

IBM. (17 de 03 de 2024). *IBM*. Obtenido de IBM: https://www.ibm.com/mx-es/topics/data-science

UDLA. (18 de 03 de 2024). Datos de Panel.