

DATOS PERSONALES		FIRMA
Nombre:	DNI:	
Apellidos:		
ESTUDIO	ASIGNATURA	CONVOCATORIA
MÁSTER UNIVERSITARIO EN INGENIERÍA MATEMÁTICA Y COMPUTACIÓN (PLAN 2016)	4391020006.- TÉCNICAS MULTIVARIANTES	Extraordinaria
FECHA	MODELO	CIUDAD DEL EXAMEN
10-12/09/2021	Modelo - C	
Etiqueta identificativa		

## INSTRUCCIONES GENERALES

1. Ten disponible tu documentación oficial para identificarte, en el caso de que se te solicite.
2. Rellena tus datos personales en todos los espacios fijados para ello y lee atentamente todas las preguntas antes de empezar.
3. Las preguntas se contestarán en la lengua vehicular de esta asignatura.
4. Si tu examen consta de una parte tipo test, indica las respuestas en la plantilla según las características de este.
5. Debes contestar en el documento adjunto, respetando en todo momento el espaciado indicado para cada pregunta. Si este es en formato digital, los márgenes, el interlineado, fuente y tamaño de letra vienen dados por defecto y no deben modificarse. En cualquier caso, asegúrate de que la presentación es suficientemente clara y legible.
6. Entrega toda la documentación relativa al examen, revisando con detenimiento que los archivos o documentos son los correctos. El envío de archivos erróneos o un envío incompleto supondrá una calificación de “no presentado”.
7. Durante el examen y en la corrección por parte del docente, se aplicará el Reglamento de Evaluación Académica de UNIR que regula las consecuencias derivadas de las posibles irregularidades y prácticas académicas incorrectas con relación al plagio y uso inadecuado de materiales y recursos.

# Puntuación

## Preguntas

- Opción personalizada 10.00 puntos

El examen consta de cuatro preguntas.

Las puntuaciones son:

Pregunta 1: 1.5 puntos.

Pregunta 2: 2 puntos.

Pregunta 3: 1.5 puntos.

Pregunta 4: 5 puntos.

Responde a las preguntas en el espacio indicado entre las páginas 3 y 15.

Encontrarás las preguntas del examen a partir de la página 16.

¡Suerte!

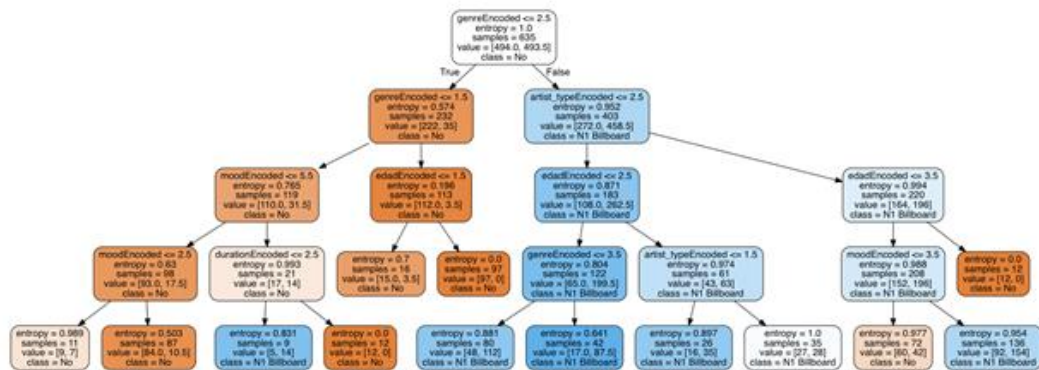
1. Pregunta 1 (Responder en 1 caras)

2. Pregunta 2 (Responder en 2 caras)

3. Pregunta 3 (Responder en 1 caras)

4. Pregunta 4 (Responder en 5 caras)

(1) (1.5 puntos) Explica la siguiente figura.



- (2) (1.5 puntos) La siguiente salida de regresión recoge los resultados de una especificación para estimar la posición en el ranking de los jugadores de una liga de básquet:

Dependent Variable: *RANKING*

Method: Least Squares

Sample: 1 195

Included observations: 195

Variable	Coefficient	Std. Error	t-Statistic	Prob.
%_TIROS_DE_2	0.980772	0.260425	3.766038	0.0002
%_TIROS_DE_3	3.888446	0.356306	10.913220	0.0000
%_TIROS_LIBRES	0.641341	0.278114	2.306035	0.0222
REBOTES_DEFENSIVOS	1.067425	0.277576	3.845530	0.0002
FALTAS_COMETIDAS	2.478362	0.325485	7.614369	0.0000
BALONES_RECUPERADOS	2.329961	0.516692	4.509377	0.0000
BALONES_PERDIDOS	2.092656	0.441097	4.744204	0.0000
C	11.143260	3.748271	2.972908	0.0033
R-squared		Mean dependent var	141.91790	
Adjusted R-squared		S.D. dependent var	79.32274	
S.E. of regression	23.015440	Akaike info criterion		
Sum squared resid	99055.870000	Schwarz criterion		
Log likelihood	-884.160900	F-statistic		
Durbin-Watson stat	1.969352	Prob(F-statistic)		

Calcula el coeficiente de determinación y el coeficiente de determinación ajustado. ¿Qué se puede decir de la bondad del ajuste?

- (3) (2 puntos) Describe y explica con detalle el siguiente código de python. Comenta los resultados.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.metrics import r2_score
4 np.random.seed(42)
5 n_samples, n_features = 50, 100
6 X = np.random.randn(n_samples, n_features)
7 idx = np.arange(n_features)
8 coef = (-1) ** idx * np.exp(-idx / 10)
9 coef[10:] = 0
10 y = np.dot(X, coef)
11 y += 0.01 * np.random.normal(size=n_samples)
12 n_samples = X.shape[0]
13 X_train, y_train = X[:n_samples // 2], y[:n_samples // 2]
14 X_test, y_test = X[n_samples // 2:], y[n_samples // 2:]
15 from sklearn.linear_model import ElasticNet
16 alpha = 0.1
17 enet = ElasticNet(alpha=alpha, l1_ratio=0.198)
18 y_pred_enet = enet.fit(X_train, y_train).predict(X_test)
19 r2_score_enet = r2_score(y_test, y_pred_enet)
20 print(enet)
21 print("r^2 on test data : %f" % r2_score_enet)
```

ElasticNet(alpha=0.1, copy\_X=True, fit\_intercept=True, l1\_ratio=0.198, max\_iter=1000, normalize=False, positive=False, precompute=False, random\_state=None, selection='cyclic', tol=0.0001, warm\_start=False)  
r<sup>2</sup> on test data : 0.704555

- (4) (5 puntos) El conjunto de datos 'Carseats' contiene información sobre la venta de sillas infantiles en 400 tiendas distintas. Para cada una de las 400 tiendas se han registrado 11 variables. Se pretende generar un modelo de clasificación que permita predecir si una tienda tiene ventas altas (Sales  $\geq 8$ ) o bajas (Sales  $< 8$ ) en función de todas las variables disponibles.

Las primeras instrucciones que puedes usar son:

```
1 import numpy as np
2 import pandas as pd
3 import statsmodels.api as sm
4 import matplotlib.pyplot as plt
5 from sklearn.model_selection import train_test_split
6 from sklearn.tree import DecisionTreeClassifier
7 from sklearn.tree import plot_tree
8 from sklearn.tree import export_graphviz
9 from sklearn.tree import export_text
10 from sklearn.model_selection import GridSearchCV
11 from sklearn.compose import ColumnTransformer
12 from sklearn.preprocessing import OneHotEncoder
13 from sklearn.metrics import accuracy_score
14 from sklearn.metrics import confusion_matrix
15 import warnings
16 warnings.filterwarnings('once')
17 carseats = sm.datasets.get_rdataset("Carseats", "ISLR")
```

- Elabora un estudio descriptivo del conjunto de datos.
- Crea una nueva variable dicotómica (0, 1) llamada `ventas_altas`, que recoja las ventas altas o bajas.
- Ajusta un árbol de clasificación empleando como variable respuesta `ventas_altas` y como predictores todas las variables disponibles. Usa los hiperparámetros `max_depth=5` y `criterion='gini'`, el resto déjalos por defecto. Divide los datos en dos grupos, uno de entrenamiento y otro de test.

En los datos hay variables categóricas, por lo que, antes de entrenar el modelo, es necesario aplicar one-hot-encoding. Las instrucciones que puedes usar son:

```
1 cat_cols = X_train.select_dtypes(include=['object', 'category']).columns.
   to_list()
2 numeric_cols = X_train.select_dtypes(include=['float64', 'int']).columns.
   to_list()
3 preprocessor = ColumnTransformer(
4     [ ('onehot', OneHotEncoder(handle_unknown='ignore'), cat_cols)],
5     remainder='passthrough'
6 )
```

- El resultado devuelto por `ColumnTransformer` es un numpy array, por lo que se pierden los nombres de las columnas. Convierte el output del `ColumnTransformer` en dataframe y añade el nombre de las columnas.
- Una vez entrenado el árbol, represéntalo mediante la función `plot_tree()`. Analiza los resultados.
- Evalúa la capacidad predictiva del árbol calculando el accuracy en el conjunto de test.
- Estudia la importancia de cada predictor.

## Pregunta 1:

La figura mostrada es un árbol de decisión y es utilizada para problemas de clasificación. Los árboles de decisión son binarios, es decir, tiene solamente 2 ramificaciones por cada nodo. El árbol mostrado en la figura es desarrollado a través de herramientas de SKlearn mediante el objeto DecisionTreeClassifier.

El árbol de decisión, tiene varias partes a lo que se logra visualizar (la imagen tiene baja resolución), se cuenta con `genreEncoded`, `artist_typeEncoded`, `edadEncoded`, `moodEncoded` y `durationEncoded`.

En el primer nivel, tenemos 635 valores de `genreEncoded` con entropía 1, que se dividen en 232 muestras de `genreEncoded` menor a 1.5 con entropía 0.574, y 403 muestras de `artist_typeEncoded` menor a 2.5 con entropía 0.952. Si sumamos ambas, da el valor de las muestras del nivel superior ( $403+232=635$ ). Si bajamos nivel a nivel, vemos que se va ramificando el árbol de decisión, considerando que en el último nivel (en este caso son 4 niveles a partir del inicial, 5 en total), se obtienen todos los valores de las muestras en diferentes categorías. Sumando  $11+87+9+12+80+42+26+35+72+136$  del último nivel, y sumando  $16+97+12$  del nivel anterior, donde finaliza esa categoría, da un total de 635.

Resumiendo, el árbol de decisión, lo que genera son niveles de categorías diferentes, donde un lado es el lado verdadero donde la condición anterior se cumple, y el otro lado es falso, donde la condición anterior no se cumple (normalmente a la izquierda es verdadero, y a la derecha es falso). En los árboles de decisión no es necesario escalar las variables predictoras.

## Pregunta 2:

## Pregunta 3:

Las 3 primeras líneas del código son importaciones de `numpy` (librería de uso numérico), `matplotlib` (gráficos matemáticos y estadísticos), y `Sklearn` (uso para estadística y machine learning).

La línea número 4, genera una semilla para iniciar números aleatorios siempre con la misma semilla, y así, al correr el código en diferentes situaciones, siempre generará los mismos valores aleatorios.


La siguiente línea, genera 2 variables, una de `samples` (muestras), y una de `features` (características), ingresando en `samples` un valor de 50, y en `features` un valor de 100, generando en `X` los valores aleatorios correspondientes. Luego hasta la línea 14, es desarrollo de código de coeficientes, y de valores de entrenamiento y test para variables `X` y `Y`, obteniendo un arreglo `X` y un arreglo `Y`, donde el arreglo `X` es una distribución uniforme, mientras el arreglo `Y` es una distribución normal de valores aleatorios configurados con las variables de `samples` y `features` generadas con anterioridad.

A partir de este punto, se importa de `SKLearn` la librería que hace referencia a `ElasticNet`. Una red elástica funciona con regresiones `Lasso` y `Ridge` al mismo tiempo, en la cual actúan las penalizaciones de ambas regresiones a la vez.

En este caso, es ajustado el hiperparámetro en 0.1 y el argumento `l1_ratio` con valor de 0.198, se entrena el modelo de `ElasticNet` por medio de “fit” con los datos de entrenamiento, se hace un test por medio de `predict` con los datos de test, y se obtiene el coeficiente de determinación  $R^2$ , el cual indica si se tiene asociación entre la variable predictora y la variable de respuesta. En este caso el  $R^2$  score muestra un valor de 0.70455, lo cual significa que la variable es una buena predictora, tiene buena asociación entre la variable de entrenamiento y la variable de prueba.

## Pregunta 4:

La pregunta 4 esta descrita en el notebook adjunto como .ipynb y como pdf.

 `carseats.data.head()`

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	9.50	138	73	11	276	120	Bad	42	17	Yes	Yes
1	11.22	111	48	16	260	83	Good	65	10	Yes	Yes
2	10.06	113	35	10	269	80	Medium	59	12	Yes	Yes
3	7.40	117	100	4	466	97	Medium	55	14	Yes	Yes
4	4.15	141	64	3	340	128	Bad	38	13	Yes	No

[+ Code](#) [+ Text](#)

### Estudio descriptivo

[11] `carseats_data = carseats.data`

[13] `carseats_data.describe()`

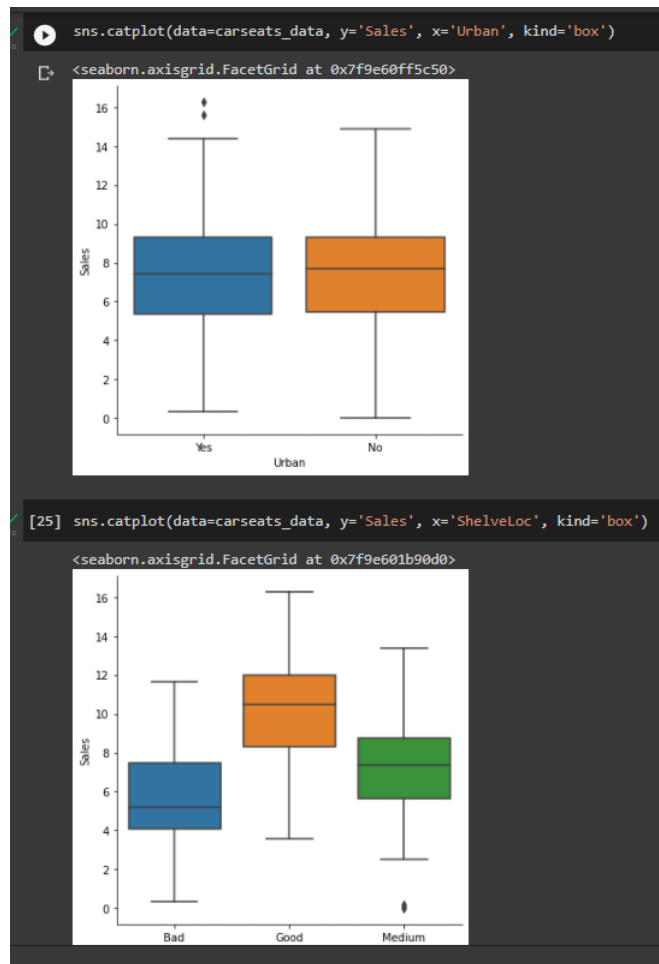
	Sales	CompPrice	Income	Advertising	Population	Price	Age	Education
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	7.496325	124.975000	68.657500	6.635000	264.840000	115.795000	53.322500	13.900000
std	2.824115	15.334512	27.986037	6.650364	147.376436	23.676664	16.200297	2.620528
min	0.000000	77.000000	21.000000	0.000000	10.000000	24.000000	25.000000	10.000000
25%	5.390000	115.000000	42.750000	0.000000	139.000000	100.000000	39.750000	12.000000
50%	7.490000	125.000000	69.000000	5.000000	272.000000	117.000000	54.500000	14.000000
75%	9.320000	135.000000	91.000000	12.000000	398.500000	131.000000	66.000000	16.000000
max	16.270000	175.000000	120.000000	29.000000	509.000000	191.000000	80.000000	18.000000



## Preguntas



## Preguntas



## Preguntas

### variable dicotomica ventas\_altas

```
[30] carseats_data["ventas_altas"] = np.where(carseats_data.Sales > 8,1,0);
```

### Arbol de clasificacion

```
[31] #dividiendo el dataset
x = carseats_data[['CompPrice', 'Income', 'Advertising', 'Population', 'Price', 'Age', 'Education', 'Urban', 'US']]
y = carseats_data['ventas_altas']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

```
[35] #tamaño de dataset de entrenamiento
x_train.shape
```

```
(320, 9)
```

```
[38] #tamaño de dataset de pruebas
x_test.shape
```

```
(80, 9)
```

```
[43] ##transformando las variables categoricas
cat_cols = x_train.select_dtypes(include=['object', 'category']).columns.to_list()
numeric_cols = x_train.select_dtypes(include=['float64', 'int']).columns.to_list()
```

```
[45] print(f"Variables categoricas: {cat_cols}")
print(f"Variables numericas: {numeric_cols}")
```

```
Variables categoricas: ['Urban', 'US']
Variables numericas: ['CompPrice', 'Income', 'Advertising', 'Population', 'Price', 'Age', 'Education']
```

```
[64] preprocessor = ColumnTransformer(
    [ ('onehot', OneHotEncoder(handle_unknown='ignore'), cat_cols)], remainder = 'passthrough'
)
```

## Preguntas

```
▼ D
```

```
[68] x_train_raw = preprocessor.fit_transform(x_train)
```

```
[69] x_train.head()
```

	CompPrice	Income	Advertising	Population	Price	Age	Education	Urban	US
3	117	100	4	466	97	55	14	Yes	Yes
18	110	110	0	408	68	46	17	No	Yes
202	121	78	4	413	130	46	10	No	Yes
250	137	105	10	435	156	72	14	Yes	Yes
274	135	93	2	67	119	34	11	Yes	Yes

```
[70] columns_names = ["urban_no", "urban_yes", "US_no", "US_yes", 'CompPrice', 'Income', 'Advertising', 'Population', 'Price', 'Age', 'Education']
```

```
[72] x_train_data = pd.DataFrame(x_train_raw, columns=columns_names)
```

```
x_train_data.head()
```

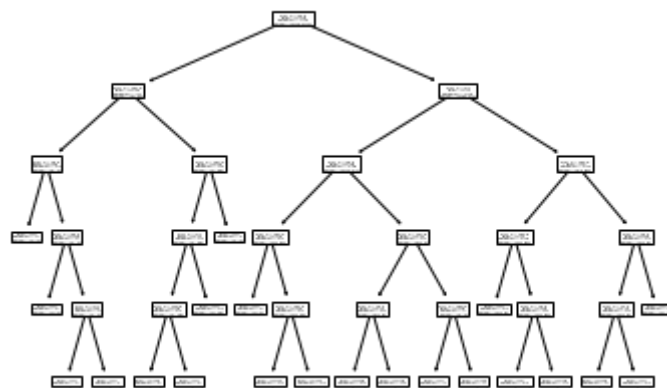
	urban_no	urban_yes	US_no	US_yes	CompPrice	Income	Advertising	Population	Price	Age	Education
0	0.0	1.0	0.0	1.0	117.0	100.0	4.0	466.0	97.0	55.0	14.0
1	1.0	0.0	0.0	1.0	110.0	110.0	0.0	408.0	68.0	46.0	17.0
2	1.0	0.0	0.0	1.0	121.0	78.0	4.0	413.0	130.0	46.0	10.0
3	0.0	1.0	0.0	1.0	137.0	105.0	10.0	435.0	156.0	72.0	14.0
4	0.0	1.0	0.0	1.0	135.0	93.0	2.0	67.0	119.0	34.0	11.0

### Entrenando el modelo

```
✓ 0s ▶ clf = DecisionTreeClassifier(  
    max_depth = 5,  
    criterion='gini'  
)  
  
    clf = clf.fit(x_train_data, y_train)
```

```
✓ 0s [75] predictions = clf.predict(x_train_data)
```

```
✓ 1s [76] plot_tree(clf)
```



## Preguntas

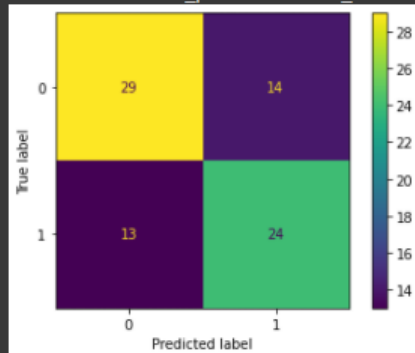
```
[85] from sklearn.metrics import plot_confusion_matrix
```

```
confusion_matrix(y_test, predictions)
```

```
array([[29, 14],  
       [13, 24]])
```

```
[87] plot_confusion_matrix(clf, x_test_data, y_test)
```

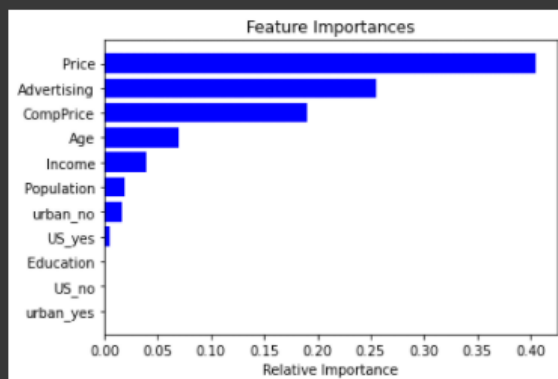
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f9e56ece290>
```



G)

```
[89] importances = clf.feature_importances_  
indices = np.argsort(importances)
```

```
plt.title('Feature Importances')  
plt.barh(range(len(indices)), importances[indices], color='b', align='center')  
plt.yticks(range(len(indices)), [columns_names[i] for i in indices])  
plt.xlabel('Relative Importance')  
plt.show()
```



segun el grafico anterior, lo mas influyente es el precio.