BREVE GUÍA PARA LA ELABORACIÓN DE LA TAREA DE MINERÍA DE DATOS Y MODELIZACIÓN PREDICTIVA

En esta guía se muestra un pequeño ejemplo sobre cómo elaborar el informe correspondiente a la tarea para incluir justificación y/o introducción de lo que se va a realizar en cada apartado, códigos y salidas junto con los comentarios de las mismas en las 25 páginas, máximo, destinadas para ello.

Se describirán solo algunos apartados y de forma similar se debe proceder para el resto. Si dentro de cada apartado, según los resultados que se vayan obteniendo, es necesario realizar algún paso más, éstos se deben indicar, detallando y justificando estos pasos, así como el código correspondiente, salida del mismo y comentario de la salida. Si el código que se utilizaría ya se ha detallado en otro apartado, basta con hacer referencia al código descrito en otro apartado anteriormente.

Las justificaciones, introducciones y los comentarios a las salidas aquí presentadas son meramente orientativas. Según vuestros resultados las justificaciones y los comentarios a las salidas serán diferentes.

Descripción de los apartados:

A) DEPURACIÓN DE DATOS

- **Importación del conjunto de datos y asignación correcta de los tipos de variables.**

Para importar los datos se utiliza el siguiente código:

```
datos = pd.read_csv('fichero.csv')
```

Para observar el tipo de variable que ha sido asignada al importar los datos se utiliza el código datos.dtypes. Al ejecutar el código se observa que las variables compra, CalifProductor, Region no han sido asignadas correctamente, tal y como se muestra a continuación (se presenta solo la parte de la salida que presenta las variables que están mal asignadas, no es necesario ponerlas todas):

```
Compra          int64
CalifProductor  int64
Region          float64
```

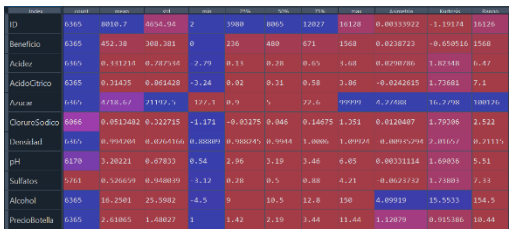
Estas variables son categóricas, por lo que se cambia su asignación con el siguiente código: numericasAcategoricas = ['Compra', 'CalifProductor', 'Region']

```
for var in numericasAcategoricas:
  datos[var] = datos[var].astype(str)
```

- **Análisis descriptivo del conjunto de datos. Número de observaciones, número y naturaleza de variables, datos erróneos etc.**

A continuación, se muestra un análisis descriptivo de las variables explicativas.

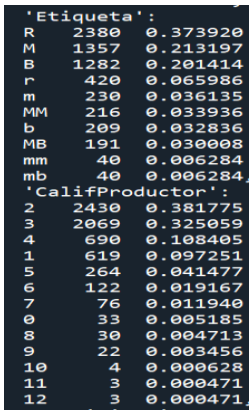
```
descriptivos_num = datos.describe().T
for num in numericas:
    descriptivos_num.loc[num, "Asimetria"] = datos[num].skew()
    descriptivos_num.loc[num, "Kurtosis"] = datos[num].kurtosis()
    descriptivos_num.loc[num, "Rango"] = np.ptp(datos[num].dropna().values)
```



	count	mean	std	min	25%	50%	75%	max	Asimetria	Kurtosis	Rango
ID	6365	8010.7	4654.94	2	3988	8065	12027	16128	0.00333922	-1.19174	16126
Beneficio	6365	452.38	308.381	0	236	480	671	1568	0.0238723	-0.650516	1568
Acidez	6365	6.331214	0.787568	2.79	6.13	6.28	6.65	3.68	0.0298786	1.82348	6.47
AcidoCitrico	6365	0.31435	0.801428	-3.24	0.02	0.31	0.58	3.86	-0.0242615	1.73681	7.1
Azucar	6465	6738.67	21192.5	127.1	0.9	5	22.6	99999	4.27488	16.2798	100126
CloruroSodico	6066	0.0513482	0.322715	-1.171	-0.03275	0.046	0.14675	1.351	0.0120407	1.79306	2.522
Densidad	6465	0.994704	0.0266166	0.888009	0.988245	0.9944	1.0006	1.09924	0.0895294	2.01657	0.21115
pH	6170	3.20221	0.67833	0.54	2.96	3.19	3.46	6.05	0.00331114	1.69036	5.51
Sulfatos	5761	0.526659	0.948039	-1.12	0.28	0.5	0.88	4.21	0.0623732	1.73803	7.33
Alcohol	6365	16.2501	25.5982	-4.5	9	10.5	12.8	150	4.09919	15.5533	154.5
PrecioBotella	6365	2.61065	1.48027	1	1.42	2.19	3.44	11.44	1.12079	0.915386	10.44

Se puede observar que la variable azúcar tiene valores perdidos no declarados que han sido codificados con el valor 99999. De igual forma indicar los errores observados en la salida para el resto de las variables.

Para detectar errores en las variables categóricas se muestran las frecuencias de cada una de sus categorías: analizar_variables_categoricas(datos)



```
'Etiqueta':
R     2380   0.373920
M     1357   0.213197
B     1282   0.201414
r      420   0.065986
m      230   0.036135
MM     216   0.033936
b      209   0.032836
MB     191   0.030008
mm      40   0.006284
mb      40   0.006284
'CalifProductor':
2     2430   0.381775
3     2069   0.325059
4      690   0.108405
1      619   0.097251
5      264   0.041477
6      122   0.019167
7       76   0.011940
0       33   0.005185
8       30   0.004713
9       22   0.003456
10       4   0.000628
11       3   0.000471
12       3   0.000471
```

Se puede observar que la variable Etiqueta presenta errores de escritura (una misma categoría está escrita en mayúscula y minúscula y las considera distintas categorías). La variable CalifProductor presenta categorías que están poco representadas.

De igual forma se procede con el resto de variable que tengan algún error. Las variables que no presenten errores no es necesario mostrar ninguna salida.

- **Corrección de los errores detectados.**

Para todos los errores detectados en el apartado anterior se indica como se van a corregir y el código que hace la corrección. Siguiendo con el ejemplo anterior se indicaría como se van a corregir los errores detectados.

Los valores 99999 de la variable azúcar van a ser reemplazados por valores perdidos:

```
datos['Azucar'] = datos['Azucar'].replace(99999, np.nan)
```

La variable Etiqueta presenta errores de escritura, por lo que se van a unir en una misma categoría aquellas que sean las mismas pero estén escritas en mayúscula y minúscula, concretamente las minúsculas se reescriben como mayúsculas.

```
datos['Etiqueta'] = datos['Etiqueta'].replace({'b': 'B', 'm': 'M', 'mb': 'MB', 'mm': 'MM', 'r': 'R'})
```

Se van a reagrupar las categorías de la variable CalifProductor de modo que todas estén representadas. Concretamente, se unen las categorías 0 y 1 por un lado, y las categorías de la 5 a la 12 por otro.

```
datos['CalifProductor'] = datos['CalifProductor'].replace({'0': '0-1', '1': '0-1', '2': '2', '3': '3', '4': '4', '5': '5-12', '6': '5-12',  '7': '5-12', '8': '5-12', '9': '5-12', '10': '5-12', '11': '5-12', '12': '5-12'})
```

En este caso no es necesario mostrar la salida que muestre el cambio realizado en los datos.

De igual forma se procede para el resto de las variables para las que se haya detectado errores. Además, faltaría analizar más errores que se puedan presentar en los datos como valores perdidos, número de valores distintos en una variable continua para ver si es mejor pasarla a categórica, etc.

Para el resto de los apartados solicitados en la tarea se procede de igual forma, indicar que se va a hacer (corresponde a la introducción o justificación), cómo se va a hacer indicando el código y las salidas para determinar las conclusiones.

B) MODELOS DE REGRESIÓN

Para los bloques de los modelos de regresión tanto lineal como logística, la forma de proceder es la misma que la descrita anteriormente.

A continuación, se muestran algunos apartados como ejemplos.:

- **Construcción del modelo de regresión lineal. Selección de variables clásica**

Hay que destacar que no es necesario mostrar todas las salidas de los modelos de regresión con selección clásica que se prueben, basta con mostrar una tabla que recoja los resultados más importantes.

Vamos a ver un ejemplo de cómo debe explicarse la construcción de un modelo de regresión lineal mediante selección clásica

En primer lugar, se debe explicar brevemente en qué consisten los métodos de selección clásica y como se construyen. A continuación, se muestra el código que construye estos modelos:

```
x_train, x_test, y_train, y_test = train_test_split(todo, varObjCont, test_size = 0.2, random_state = 1234567)

modeloStepAIC = lm_stepwise(y_train, x_train, var_cont_sin_transf, var_categ, [], 'AIC')

modeloStepBIC = lm_stepwise(y_train, x_train, var_cont_sin_transf, var_categ, [], 'BIC')

modeloBackAIC = lm_backward(y_train, x_train, var_cont_sin_transf, var_categ, [], 'AIC')
```

```
modeloBackBIC = lm_backward(y_train, x_train, var_cont_sin_transf, var_categ, [], 'BIC')

modeloForwAIC = lm_forward(y_train, x_train, var_cont_sin_transf, var_categ, [], 'AIC')

modeloForwBIC = lm_forward(y_train, x_train, var_cont_sin_transf, var_categ, [], 'BIC')
```

Al aplicar los distintos métodos de selección, los resultados más importantes se recogen en la siguiente tabla, de modo que nos permita determinar cuál es el mejor modelo:

Método	Métrica	R^2 Train	R^2 Test	Nº Parámetros
Backward	AIC			
Backward	BIC			
Forward	AIC			
Forward	BIC			
Stepwise	AIC			
Stepwise	BIC			

Comentar los resultados mostrados en la tabla, de modo que se justifique cuáles podrían ser los modelos candidatos a ser óptimos.

- **Construcción del modelo de regresión lineal. Selección de variables aleatoria**

Para la selección aleatoria se procede de igual forma que para la selección de variables clásica anteriormente descrita. Se explica en que consiste la selección de variables aleatoria. A continuación, se muestra el código que realiza la selección de variables aleatoria.

```
variables_seleccionadas = {
    'Formula': [],
    'Variables': []
}
# Realizar 30 iteraciones de selección aleatoria.
for x in range(30):
    print('---------------------------- iter: ' + str(x))
    # Dividir los datos de entrenamiento en conjuntos de entrenamiento y prueba.
    x_train2, x_test2, y_train2, y_test2 = train_test_split(x_train, y_train,
                                                test_size = 0.3, random_state = 1234567 + x)
    # Realizar la selección stepwise utilizando el criterio BIC en la submuestra.
    modelo = lm_stepwise(y_train2.astype(int), x_train2, var_cont, var_categ, interacciones_unicas, 'BIC')
        # Almacenar las variables seleccionadas y la fórmula correspondiente.
    variables_seleccionadas['Variables'].append(modelo['Variables'])
    variables_seleccionadas['Formula'].append(sorted(modelo['Modelo'].model.exog_names))
# Unir las variables en las fórmulas seleccionadas en una sola cadena.
variables_seleccionadas['Formula'] = list(map(lambda x: '+'.join(x), variables_seleccionadas['Formula']))
```

```
    # Calcular la frecuencia de cada fórmula y ordenarlas por frecuencia.
frecuencias = Counter(variables_seleccionadas['Formula'])
frec_ordenada = pd.DataFrame(list(frecuencias.items()), columns = ['Formula', 'Frecuencia'])
frec_ordenada = frec_ordenada.sort_values('Frecuencia', ascending = False).reset_index()
# Identificar las dos modelos más frecuentes y las variables correspondientes.
var_1 = variables_seleccionadas['Variables'][variables_seleccionadas['Formula'].index(
    frec_ordenada['Formula'][0])]
var_2 = variables_seleccionadas['Variables'][variables_seleccionadas['Formula'].index(
    frec_ordenada['Formula'][1])]
```

A continuación, se muestran las salidas de los modelos que más se repiten, que serán los candidatos a ser óptimos.

Var_1: {'cont': [], 'categ': ['Clasificacion', 'CalifProductor'], 'inter': [('Etiqueta', 'Clasificacion'), ('Densidad', 'Etiqueta')]}

Var_2: {'cont': [], 'categ': ['Clasificacion', 'Etiqueta'], 'inter': [('Etiqueta', 'Clasificacion'), ('pH', 'CalifProductor')]}

- **Construcción del modelo de regresión lineal. Selección del modelo ganador**

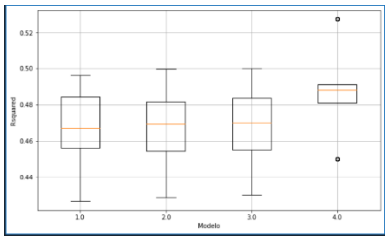
Explicar cómo se determina el modelo ganador. Que metodología se utiliza para obtener el modelo ganador y que resultados se presentan al aplicar la metodología.

A continuación, se presenta el código para determinar el modelo ganador.

```
results = pd.DataFrame({
    'Rsquared': [],
    'Resample': [],
    'Modelo': []
})
# Realiza el siguiente proceso 20 veces (representado por el bucle `for rep in range(20)`)
for rep in range(20):
    # Realiza validación cruzada en cuatro modelos diferentes y almacena sus R-squared en listas separadas
    modelo1VC = validacion_cruzada_lm(5, x_train, y_train, var_cont1, var_categ1)
    modelo2VC = validacion_cruzada_lm(5, x_train, y_train, var_cont2, var_categ2)
    modelo3VC = validacion_cruzada_lm(5, x_train, y_train, var_cont3, var_categ3)
    modelo4VC = validacion_cruzada_lm(5, x_train, y_train, var_cont4, var_categ4, var_interac4)
        # Crea un DataFrame con los resultados de validación cruzada para esta repetición
    results_rep = pd.DataFrame({
        'Rsquared': modelo1VC + modelo2VC + modelo3VC + modelo4VC,
        'Resample': ['Rep' + str((rep + 1))] * 5 * 4,  # Etiqueta de repetición
```

```
    'Modelo': [1] * 5 + [2] * 5 + [3] * 5 + [4] * 5  # Etiqueta de modelo (1, 2, 3 o 4)
})
# Concatena los resultados de esta repetición al DataFrame principal 'results'
results = pd.concat([results, results_rep], axis=0)
```

Se muestran los resultados obtenidos:



Se comentan los resultados, mostrados en el gráfico, según la salida obtenida.

Por tanto, para determinar el modelo ganador se debe explicar en qué consiste la metodología que se va a utilizar, poner el código, salidas y comentarios a las salidas y determinar las conclusiones.

De igual modo se procede para el resto de los apartados. Aquí solo se ha mostrado algunos ejemplos para algunos apartados.

Para el **MODELO DE REGRESIÓN LOGÍSTICA** deben explicarse y justificarse todos los pasos a realizar para responder a todos los apartados solicitados en la tarea. Aunque los pasos y códigos a utilizar son muy similares en regresión lineal y logística, debe indicarse también lo que es igual, haciendo referencia al apartado correspondiente en regresión lineal, sin necesidad de volver a detallarlo de nuevo. Los códigos que son distintos en regresión lineal y logística, aunque tienen bastantes similitudes, se deben indicar las diferencias y se puede destacar solo la parte del código que es diferente respecto al código detallado anteriormente en regresión lineal, haciendo siempre referencia a dicho código.