

A1.4 Selección de características

Luis Enrique Garcia Gallegos

Matricula: 649247

En la lectura interactiva de este módulo trabajamos con la base de datos de calidad de vino. Ahí, programamos **a mano**, métodos de selección de características. Pero, al final, te comenté que existen otras funciones que te permiten realizar este proceso de forma más sencilla y veloz. En esta actividad deberás generar un modelo de regresión lineal múltiple que contenga solamente las variables seleccionadas por un proceso de selección hacia adelante y eliminación hacia atrás. Utilizaremos el archivo de nombre `Vino_Tinto.csv`, donde podrás encontrar información para 1,599 observaciones distintas, con 11 mediciones para cada una de ellas, así como con una variable de salida, la calidad asignada a dicho vino. Los datos se descargaron del [UCI Machine Learning Repository](#), y originalmente se reportaron en una publicación científica para la revista [Decision Support Systems](#). La base de datos cuenta con la siguiente información:

- `acidezFija` . La acidez fija del vino, medida en **gramos de ácido tartárico por decímetro cúbico**.
- `acidezVolatil` . La acidez volátil del vino, medida en **gramos de ácido acético por decímetro cúbico**.
- `acidoCitrico` . **Gramos de ácido cítrico por decímetro cúbico**.
- `azucarResidual` . **Gramos de azúcar por decímetro cúbico**.
- `cloruros` . **Gramos de cloruro de sodio por decímetro cúbico**.
- `dioxidoAzufreLibre` . **Miligramos de dióxido de azufre libre por decímetro cúbico**.
- `dioxidoAzufreTotal` . **Miligramos de dióxido de azufre total por decímetro cúbico**.
- `densidad` . Medida en **gramos por centímetro cúbico**.
- `pH` . Valor del vino en la escala de **pH**.
- `sulfatos` . **Gramos de sulfato de potasio por decímetro cúbico**.
- `alcohol` . **Volúmen percentil** de alcohol en el vino.
- `calidad` . Mediana de la calidad otorgada por al menos tres catadores, en escala del **0 (muy malo) al 10 (excelente)**.


Desarrolla los siguientes puntos en una *Jupyter Notebook*, tratando, dentro de lo posible, que cada punto se trabaje en una celda distinta. Los comentarios en el código siempre son bienvenidos, de preferencia, aprovecha el *markdown* para generar cuadros de descripción que ayuden al lector a comprender el trabajo realizado.

1. Importa los datos del archivo `Vino_Tinto.csv` a tu ambiente de trabajo. Este archivo lo encontrarás en la misma página donde descargaste esta plantilla. Revisa las dimensiones del *data frame* e imprime en consola tanto dichas dimensiones como las primeras 5 filas de datos.

```
In [1]: import pandas as ps
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats as st
import mlxtend.feature_selection as mlx
import random
from math import sqrt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
datos=ps.read_csv("Vino_Tinto.csv")
datos.head(5)
```

```
Out[1]:
```

	acidezFija	acidezVolatil	acidoCitrico	azucarResidual	cloruros	dioxidoAzufreLibre	diox
0	7.4	0.70	0.00	1.9	0.076	11.0	
1	7.8	0.88	0.00	2.6	0.098	25.0	
2	7.8	0.76	0.04	2.3	0.092	15.0	
3	11.2	0.28	0.56	1.9	0.075	17.0	
4	7.4	0.70	0.00	1.9	0.076	11.0	



2. Separa el *data frame* en datos de entrenamiento y datos de prueba con una proporción **80/20**. Es decir, el **80%** de los datos se usarán para entrenar el modelo y el resto para validar sus resultados. Asegúrate que la partición sea aleatoria, no es una buena práctica simplemente tomar las primeras observaciones para entrenar y las últimas para probar. Imprime en pantalla las dimensiones de ambos conjuntos de datos. Revisa y asegúrate que la cantidad de observaciones de ambos conjuntos de datos sumen a la cantidad de datos original.

```
In [2]: random.seed(0)
caliz, prueba = train_test_split(datos, train_size=0.8)
nC=caliz.shape[0]
mC=caliz.shape[1]
nP=prueba.shape[0]
mP=prueba.shape[1]
print("Datos de entrenamiento: ", caliz.shape, "\tDatos de prueba: ",prueba.shape,
```

```
Datos de entrenamiento: (1279, 12)      Datos de prueba: (320, 12)
Total de datos: 1599
```

Se utilizo una ***semilla*** para poder trabajar para un caso específico porque de no ser así obtendríamos valores distintos cada vez que se ejecutara el código.

3. Genera la metodología de selección hacia adelante e imprime en consola los índices o los nombres de las características seleccionadas. Para realizar este proceso, te recomiendo que utilices la función `SequentialFeatureSelector` de la librería `mlxtend.feature_selection`, en este [enlace encontrarás más información sobre la misma](#).

Lo más probable es que cuando hayas descargado Anaconda, esta librería no se haya incluido en la distribución, por lo que deberás instalarla manualmente; al final de las instrucciones de la actividad te indico cómo hacerlo. Aquí te dejo una descripción de los parámetros que te recomiendo usar:

- `estimator`. Un modelo de regresión lineal. Te recomiendo usar la función `LinearRegression` de la librería `sklearn.linear_model`.
- `k_features`. Se puede seleccionar la cantidad de variables de salida que se desean, pero te recomiendo mejor usar un rango, y que el algoritmo determine el número adecuado. Por ejemplo, puedes definir el parámetro como **(2,8)**, si te interesa que el método seleccione entre **2** y **8** variables.
- `forward`. Determina si se hace selección hacia adelante (**True**) o hacia atrás (**False**); en este caso queremos hacer selección hacia adelante.
- `scoring`. La métrica que se usará para determinar si un modelo es mejor que otro, te recomiendo definirla como r^2 para usar la R^2 .
- `cv`. Si se desea realizar validación cruzada, y cuántas instancias de la misma. Te recomiendo definir este parámetro como **10**.

```
In [3]: XC=caliz.drop('calidad', axis=1)
        YC=caliz.calidad
        sfs=mlx.SequentialFeatureSelector(LinearRegression(), k_features=(2, 8), forward=True)
        sfs.fit(XC,YC)
        variables=sfs.k_feature_names_
        print("Características seleccionadas:\n\t", variables)
```

Características seleccionadas:

```
      ('acidezVolatil', 'cloruros', 'dioxidoAzufreLibre', 'dioxidoAzufreTotal',
'pH', 'sulfatos', 'alcohol')
```

Para poder decidir que variables son importantes para nuestro modelo usamos esta función la cual nos permitirá conocer que variables tienen mejor ajuste para nuestra predicción, para ello se empieza a hacer modelos a partir de una variable y de ahí se va agregando variables y se selecciona el que mejor ajuste tenga.

4. Entrenar un modelo que solamente contenga las variables seleccionadas, predecir la respuesta en las observaciones de prueba y medir la capacidad de predicción del modelo usando la R^2 , imprimiendo dicho valor en consola. Para el primer paso, simplemente necesitas usar la función `fit` en el modelo de regresión lineal creado

previamente, asegurándote de no introducir toda la información de **X**, sino solo de las variables seleccionadas. Para realizar las predicciones, puedes usar la función `predict` en los datos de prueba, pero recuerda para dichos datos también seleccionar solo las variables de interés. Para el último paso, te recomiendo usar la función `r2_score` de `sklearn.metrics`.

```
In [4]: xAjustadoC=XC
xAjustadoP=prueba.drop('calidad', axis=1)
for i in range(len(variables)):
    xAjustadoC=xAjustadoC.drop(variables[i], axis=1)
borrar=xAjustadoC.columns
xAjustadoC=XC
for i in range(len(borrar)):
    xAjustadoC=xAjustadoC.drop(borrar[i], axis=1)
    xAjustadoP=xAjustadoP.drop(borrar[i], axis=1)
modeloC=sm.OLS(YC, sm.add_constant(xAjustadoC))
resultadosC=modeloC.fit()
resultadosP=resultadosC.predict(sm.add_constant(xAjustadoP))
YP=prueba.calidad
rssP=sum((YP-resultadosP)**2)
tssP=sum((YP-np.mean(YP))**2)
rseP=sqrt(rssP/(nC-mC-1))
r2P=1-rssP/tssP
print(f"RSS de entrenamiento: {round(resultadosC.scale, 4)}\tRSS de prueba: {round(
print(f"R^2 de entrenamiento: {round(resultadosC.rsquared, 4)}\tR^2 de prueba: {rou
```

```
RSS de entrenamiento: 0.4173      RSS de prueba: 138.0307
R^2 de entrenamiento: 0.3625      R^2 de prueba: 0.3407
```

El `RSS` aumento significativo entre la prueba y el entrenamiento, no obstante el R^2 se mantuvo similar por lo que se puede decir que el ajuste es muy cercano a lo real.

5. Realizar un proceso de selección hacia atrás a partir de las variables seleccionadas por el método de selección hacia adelante e imprimir en consola los índices o nombres de las variables seleccionadas. Para realizar este proceso, te recomiendo usar la misma función del **paso 3**, pero definiendo ahora `forward=False`. También te recomiendo especificar una menor cantidad de variables posibles, por ejemplo: `k_features=(2,5)`.

```
In [5]: sfsInv=mlx.SequentialFeatureSelector(LinearRegression(), k_features=(2, 5), forward
sfsInv.fit(XC,YC)
variablesInv=sfsInv.k_feature_names_
print("Características seleccionadas:\n\t", variablesInv)
```

```
Características seleccionadas:
('acidezVolatil', 'cloruros', 'dioxidoAzufreTotal', 'sulfatos', 'alcohol')
```

Se repitió casi lo mismo que en el **paso 3** no obstante ahora partimos de todas las variables y vamos descartando las que no tengan tanto impacto en el modelo (esto lo hace la misma función). Si comparamos esto con el **paso 3** podremos observar que se quitaron variables.

6. Repetir el **paso 4**, pero para un modelo que contenga solamente las variables seleccionadas en el **paso 5**. Imprime en pantalla un breve texto que describa tu opinión sobre la diferencia en R^2 medida entre los modelos de los **pasos 4** y **6**, ¿cuál modelo consideras que es mejor? ¿Por qué?.

```
In [6]: xAjustadoCInv=XC
xAjustadoPInv=prueba.drop('calidad', axis=1)
for i in range(len(variablesInv)):
    xAjustadoCInv=xAjustadoCInv.drop(variablesInv[i], axis=1)
borrarInv=xAjustadoCInv.columns
xAjustadoCInv=XC
for i in range(len(borrarInv)):
    xAjustadoCInv=xAjustadoCInv.drop(borrarInv[i], axis=1)
    xAjustadoPInv=xAjustadoPInv.drop(borrarInv[i], axis=1)
modeloCInv=sm.OLS(YC, sm.add_constant(xAjustadoCInv))
resultadosCInv=modeloCInv.fit()
resultadosPInv=resultadosCInv.predict(sm.add_constant(xAjustadoPInv))
YPIInv=prueba.calidad
rssPInv=sum((YPIInv-resultadosP)**2)
tssPInv=sum((YPIInv-np.mean(YPIInv))**2)
rsePInv=sqrt(rssPInv/(nP-mP-1))
r2PInv=1-rssPInv/tssPInv
print(f"RSS de entrenamiento: {round(resultadosCInv.scale, 4)}\tRSS de prueba: {rou
print(f"R^2 de entrenamiento: {round(resultadosCInv.rsquared, 4)}\tR^2 de prueba: {
```

```
RSS de entrenamiento: 0.4203      RSS de prueba: 138.0307
R^2 de entrenamiento: 0.3569      R^2 de prueba: 0.3407
```

La diferencia en R^2 medida entre los modelos de los **pasos 4** y **6** es mínima, en este caso el R^2 es prácticamente el mismo. Esto puede ser debido a que el modelo que contiene solamente las variables seleccionadas en el **paso 5** es similar al que se generó en el **paso 3**, pero con menos variables, lo cual nos muestra que las variables que no se tomaron en cuenta no tienen un gran impacto con el modelo, por lo que se podría escoger el modelo con menos variables.

Firma de Honor: Doy mi palabra que he realizado esta actividad con integridad académica