A1.4 Selección de características

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 "A1.4 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 tácito 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 [ ]:
```
import pandas as pd
df = pd.read_csv("A1.4 Vino Tinto.csv")

print(df.shape)
print(df.head(5))
```

```
(1599, 12)
   acidezFija  acidezVolatil  acidoCitrico  azucarResidual  cloruros  \
0         7.4           0.70          0.00             1.9     0.076
1         7.8           0.88          0.00             2.6     0.098
2         7.8           0.76          0.04             2.3     0.092
3        11.2           0.28          0.56             1.9     0.075
4         7.4           0.70          0.00             1.9     0.076

   dioxidoAzufreLibre  dioxidoAzufreTotal  densidad    pH  sulfatos  alcohol  \
0                11.0                34.0    0.9978  3.51      0.56      9.4
1                25.0                67.0    0.9968  3.20      0.68      9.8
2                15.0                54.0    0.9970  3.26      0.65      9.8
3                17.0                60.0    0.9980  3.16      0.58      9.8
4                11.0                34.0    0.9978  3.51      0.56      9.4

   calidad
0        5
1        5
2        5
3        6
4        5
```

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]:
```
from sklearn.model_selection import train_test_split

train, test = train_test_split(df, train_size = 0.8)

print(train.shape)
print(test.shape)
```

```
(1279, 12)
(320, 12)
```

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:

a. estimator. Un modelo de regresión lineal. Te recomiendo usar la función LinearRegression de la librería sklearn.linear_model.

b. 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.

c. forward. Determina si se hace selección hacia adelante (True) o hacia atrás (False); en este caso queremos hacer selección hacia adelante.

d. scoring. La métrica que se usará para determinar si un modelo es mejor que otro, te recomiendo definirla como 'r2' para usar la R cuadrada.

e. cv. Si se desea realizar validación cruzada, y cuántas instancias de la misma. Te recomiendo definir este parámetro como 10.

In [3]:
```
from sklearn.linear_model import LinearRegression
from mlxtend.feature_selection import SequentialFeatureSelector as SFS

X_train = train.drop(columns=['calidad'])
y_train = train['calidad']

sfs = SFS(
    estimator=LinearRegression(),
    k_features=(2, 8),
    forward=True,
    scoring='r2',
    cv=10,
    n_jobs=-1
)

sfs.fit(X_train, y_train)

selected_features = list(sfs.k_feature_names_)
print("Características seleccionadas:")
print(selected_features)
```

```
Características seleccionadas:
['acidezVolatil', 'cloruros', 'dioxidoAzufreLibre', 'dioxidoAzufreTotal', 'pH', 'sul
fatos', 'alcohol']
```

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 cuadrada, 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 [ ]:
```
from sklearn.metrics import r2_score

# Entrenar el modelo solo con las variables seleccionadas
model = LinearRegression()
model.fit(X_train[selected_features], y_train)

# Predecir en el conjunto de prueba usando solo las variables seleccionadas
y_pred = model.predict(test[selected_features])

# Calcular y mostrar la R cuadrada
r2 = r2_score(test['calidad'], y_pred)
print(f"R cuadrada del modelo con selección hacia adelante: {r2:.4f}")
```

```
R cuadrada del modelo con selección hacia adelante: 0.3658
```

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 [ ]:
```
# Selección hacia atrás usando solo las variables seleccionadas previamente
sfs_backward = SFS(
    estimator=LinearRegression(),
    k_features=(2, 5),
    forward=False,
    scoring='r2',
    cv=10,
    n_jobs=-1
)

sfs_backward.fit(X_train[selected_features], y_train)

selected_features_backward = list(sfs_backward.k_feature_names_)
print("Características seleccionadas por selección hacia atrás:")
print(selected_features_backward)
```

```
Características seleccionadas por selección hacia atrás:
['acidezVolatil', 'cloruros', 'dioxidoAzufreTotal', 'sulfatos', 'alcohol']
```

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 cuadrada medida entre los modelos de los pasos 4 y 6, ¿cuál modelo consideras que es mejor? ¿Por qué?

In [8]:
```python
from sklearn.metrics import r2_score

# Entrenar el modelo solo con las variables seleccionadas por selección hacia atrás
model_backward = LinearRegression()
model_backward.fit(X_train[selected_features_backward], y_train)

# Predecir en el conjunto de prueba usando solo las variables seleccionadas por sel
y_pred_backward = model_backward.predict(test[selected_features_backward])

# Calcular y mostrar la R cuadrada
r2_backward = r2_score(test['calidad'], y_pred_backward)
print(f"R cuadrada del modelo con selección hacia atrás: {r2_backward:.4f}")

# Opinión sobre la diferencia en R cuadrada
print(
    "Comparando ambos modelos, el modelo con selección hacia adelante tiene una R c
    f"{r2:.4f}, mientras que el modelo con selección hacia atrás tiene una R cuadra
    f"{r2_backward:.4f}. El modelo con mayor R cuadrada es preferible, ya que expli
    "de la variable de salida. Sin embargo, si la diferencia es pequeña, el modelo
    "puede ser mejor por su interpretabilidad y menor riesgo de sobreajuste."
)
```

```
R cuadrada del modelo con selección hacia atrás: 0.3524
Comparando ambos modelos, el modelo con selección hacia adelante tiene una R cuadrad
a de
0.3658, mientras que el modelo con selección hacia atrás tiene una R cuadrada de
0.3524. El modelo con mayor R cuadrada es preferible, ya que explica mejor la variab
ilidad
de la variable de salida. Sin embargo, si la diferencia es pequeña, el modelo más si
mple (menos variables)
puede ser mejor por su interpretabilidad y menor riesgo de sobreajuste.
```