Selección de Características

August 24, 2025

1 Selección de Características Sergio Alejandro Zamora Dávila

Importo los datos del archivo “Vino Tinto.csv” al ambiente de trabajo. Reviso las dimensiones del data frame y las imprimo en consola

[11]:
```
import pandas as pd
df = pd.read_csv("VinoTinto.csv")
print(df.shape)
print(df.columns)
print(df.head(5))
```

```
(1599, 12)
Index(['acidezFija', 'acidezVolatil', 'acidoCitrico', 'azucarResidual',
       'cloruros', 'dioxidoAzufreLibre', 'dioxidoAzufreTotal', 'densidad',
       'pH', 'sulfatos', 'alcohol', 'calidad'],
      dtype='object')
   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
```

Se realiza un split de los datos en conjuntos de entrenamiento (80%) y prueba (20%). Se imprimen las dimensiones de ambos conjuntos y se verifica que la suma de observaciones coincida con el total

del dataset original.

[12]:
```
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, train_size = 0.8)

print(train.shape)
print(test.shape)
print("Total observaciones:", len(df), "->", len(train) + len(test))

X = train.drop('calidad', axis = 1)
Y = train.calidad
```

```
(1279, 12)
(320, 12)
Total observaciones: 1599 -> 1599
```

Se utiliza la función SequentialFeatureSelector de la librería mlxtend para identificar las variables más relevantes para predecir la calidad del vino.

a. estimator. Un modelo de regresión lineal.

b. k_features. Se puede seleccionar la cantidad de variables de salida que se desean, aunque lo recomendable es mejor usar un rango, y que el algoritmo determine el número adecuado.

c. forward. Determina si se hace selección hacia adelante (True) o hacia atrás (False), que ahora se usará la selección hacia adelante.

d. scoring. La métrica que se usará para determinar si un modelo es mejor que otro, si se define con 'r2' se usará la R cuadrada.

e. cv. Si se desea realizar validación cruzada, y cuántas instancias de la misma.

[24]:
```
from sklearn.linear_model import LinearRegression
from mlxtend.feature_selection import SequentialFeatureSelector as SFS

estimator = LinearRegression()

sfs_forward = SFS(
    estimator,
    k_features=(2, 8),      # rango sugerido
    forward=True,           # hacia adelante
    floating=False,
    scoring='r2',           # métrica R^2
    cv=10,
    n_jobs=-1,
)

sfs_forward = sfs_forward.fit(X, Y)

idx_forward = list(sfs_forward.k_feature_idx_)
vars_forward = list(sfs_forward.k_feature_names_)
```

```
print("Índices seleccionados (forward):", idx_forward)
print("Variables seleccionadas (forward):", vars_forward)
```

```
Índices seleccionados (forward): [1, 4, 5, 6, 8, 9, 10]
Variables seleccionadas (forward): ['acidezVolatil', 'cloruros',
'dioxidoAzufreLibre', 'dioxidoAzufreTotal', 'pH', 'sulfatos', 'alcohol']
```

Se entrena un modelo de regresión lineal usando únicamente las variables seleccionadas en el paso anterior. Luego se predice la calidad en el conjunto de prueba y se calcula el R^2 para evaluar la capacidad predictiva del modelo.

[18]:
```
from sklearn.metrics import r2_score

vars_fs = vars_forward

xTrain = train[vars_fs]
yTrain = train['calidad']

xTest = test[vars_fs]
yTest = test['calidad']

model_fs = LinearRegression()
model_fs.fit(xTrain, yTrain)

y_pred_fs = model_fs.predict(xTest)

r2_fs = r2_score(yTest, y_pred_fs)

print("R^2 en prueba (modelo con selección hacia adelante):", round(r2_fs, 4))
```

```
R^2 en prueba (modelo con selección hacia adelante): 0.2717
```

A partir del dataset completo, se realiza un proceso de selección hacia atrás usando las mismas funciones de mlxtend, pero ahora con forward=False y un rango de variables más pequeño (k_features=(2,5)). Se imprimen los índices y nombres de las variables seleccionadas.

[15]:
```
backwardSel = SFS(
    estimator,
    k_features=(2, 5),    # rango más pequeño
    forward=False,        # hacia atrás
    floating=False,
    scoring='r2',
    cv=10,
    n_jobs=-1,
)

X = df.drop('calidad', axis=1)
Y = df['calidad']
```

```
backwardSel = backwardSel.fit(X, Y)

idx_backward = list(backwardSel.k_feature_idx_)
vars_backward = list(backwardSel.k_feature_names_)

print("Índices seleccionados (backward):", idx_backward)
print("Variables seleccionadas (backward):", vars_backward)
```

```
Índices seleccionados (backward): [1, 4, 6, 9, 10]
Variables seleccionadas (backward): ['acidezVolatil', 'cloruros',
'dioxidoAzufreTotal', 'sulfatos', 'alcohol']
```

Se repite el procedimiento del paso 4, pero usando únicamente las variables seleccionadas en la selección hacia atrás. Se compara el R^2 de este modelo con el del modelo de selección hacia adelante y se reflexiona sobre cuál de los dos modelos tiene mejor desempeño y por qué.

[23]:
```
xTrain = train[vars_backward]
yTrain = train['calidad']

xTest = test[vars_backward]
yTest = test['calidad']

model_be = LinearRegression()
model_be.fit(xTrain, yTrain)

y_pred_be = model_be.predict(xTest)
r2_be = r2_score(yTest, y_pred_be)

print("R^2 en prueba (modelo con selección hacia atrás):", round(r2_be, 4))
print("R^2 en prueba (modelo con selección hacia adelante):", round(r2_fs, 4))

if r2_be > r2_fs:
    print("Conclusión: el modelo con selección hacia atrás tiene mejor␣
 ↪desempeño en prueba.")
elif r2_be < r2_fs:
    print("Conclusión: el modelo con selección hacia adelante tiene mejor␣
 ↪desempeño en prueba.")
else:
    print("Conclusión: ambos modelos tienen el mismo desempeño en prueba según␣
 ↪R^2.")
```

```
R^2 en prueba (modelo con selección hacia atrás): 0.2653
R^2 en prueba (modelo con selección hacia adelante): 0.2717
Conclusión: el modelo con selección hacia adelante tiene mejor desempeño en
prueba.
```

Aunque el modelo de selección hacia adelante presenta un valor ligeramente mayor de ^2 (0.2717), esto se logra utilizando 7 variables, mientras que el modelo de selección hacia atrás alcanza un valor

similar (0.2653) con solo 5 variables. El modelo backward tiene una menor complejidad con un desempeño muy cercano al otro modelo. La elección final entre ambos no debe basarse únicamente en $\hat{}2$, sino también en un criterio de penalización por número de variables utilizadas, como el AIC/BIC o con técnicas de regularización como Ridge/Lasso ( ), que ayudan a equilibrar ajuste y simplicidad.