UAX Universidad Alfonso X el Sabio

Técnicas de IA: Regresiones
Linear Regression Lab 1

Data and Lab Description

En este laboratorio construirás tus primeros modelos de regresión lineal utilizando Python. En particular, tu tarea será predecir el precio medio de viviendas en diferentes suburbios de Boston y comprender qué factores afectan a los precios de las viviendas y de qué manera.

El archivo **Boston.xlsx** contiene información sobre el precio medio de las viviendas (en miles de dólares, almacenado en la variable **medv**) en 506 suburbios de Boston, junto con el siguiente conjunto de variables explicativas:

- **crim**: tasa de criminalidad per cápita por ciudad.
- **zn**: proporción de terreno residencial zonificado para lotes de más de 25,000 pies cuadrados.
- **indus**: proporción de terreno destinado a negocios no minoristas por ciudad.
- **chas**: variable ficticia del río Charles (= 1 si el tramo limita con el río; 0 en caso contrario).
- **nox**: concentración de óxidos de nitrógeno (partes por cada 10 millones).
- **rm**: número promedio de habitaciones por vivienda.
- **age**: proporción de unidades ocupadas por sus propietarios construidas antes de 1940.
- **dis**: media ponderada de las distancias a cinco centros de empleo en Boston.
- **rad**: índice de accesibilidad a autopistas radiales.
- **tax**: tasa de impuesto a la propiedad por cada $10,000 de valor.
- **ptratio**: ratio de alumnos por maestro en la ciudad.
- **lstat**: porcentaje de población de bajo estatus.

Preparación y Exploración Inicial

1. **Abre un nuevo proyecto en tu entorno de desarrollo (por ejemplo, Jupyter Notebook o VS Code)** y carga el conjunto de datos de Boston. Puedes utilizar pandas para cargar el archivo. Asegúrate de que los tipos de datos se infieran automáticamente y convierte manualmente la variable chas a tipo binario si no lo ha hecho correctamente:

```python
import pandas as pd

# Cargar el dataset
df = pd.read_excel("Boston.xlsx")

# Convertir 'chas' a tipo entero (binario)
df['chas'] = df['chas'].astype(int)
```

2. Realiza un análisis univariado**: inspecciona la distribución de todas las variables. Puedes utilizar describe() para estadísticas generales y hist() o seaborn.displot() para visualizar distribuciones:**

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Estadísticas descriptivas
print(df.describe())

# Histogramas para cada variable
df.hist(bins=30, figsize=(15, 10))
plt.tight_layout()
plt.show()
```

3. **Realiza un análisis bivariado**: establece medv como variable dependiente y todas las demás como explicativas. Crea una matrix de correlación entre las viarables, utiliza diagramas de dispersión (scatterplot) y diagramas de caja (boxplot) para explorar las relaciones, por ejemplo:

```python
# Diagramas de dispersión
for column in df.columns:
    if column != 'medv':
        sns.scatterplot(x=df[column], y=df['medv'])
        plt.title(f'medv vs {column}')
        plt.xlabel(column)
        plt.ylabel('medv')
        plt.show()

# Diagramas de caja agrupando por intervalos
for column in df.columns:
    if column != 'medv':
        df['binned'] = pd.qcut(df[column], q=10, duplicates='drop')
        sns.boxplot(x='binned', y='medv', data=df)
        plt.xticks(rotation=45)
        plt.title(f'Boxplot de medv por deciles de {column}')
        plt.show()
        df.drop('binned', axis=1, inplace=True)
```

Simple Linear Regression

4. **¡**Ahora estamos listos para estimar nuestro primer modelo de aprendizaje supervisado**!** Intentaremos predecir el precio medio de la vivienda (medv) usando solo la variable lstat mediante una regresión lineal simple. Tendréis que usar la librería scikit-learn en Python.

5. Bien hecho! Has creado tu primer modelo de regresión lineal. Ahora es momento de **investigar y evaluar el modelo** que acabas de construir. Ahora analiza los siguientes aspectos:

    - Coeficientes de regresión: El coeficiente indica cuánto cambia el valor medio de medv por cada unidad de cambio en lstat.
    - Grafico de dispersión - Valor real vs predicho: **¿Qué debes observar?** Si los puntos están cercanos a la línea diagonal, el modelo está prediciendo bien; si los puntos se dispersan mucho, el modelo tiene errores grandes; si los errores se concentran más en ciertas zonas del gráfico (por ejemplo, en precios bajos o altos), hay **sesgo o heterocedasticidad**.
    - Metricas del modelo como el coeficiente de determinación (R-squared) y MSE.
    - Distribución de errores (residuos): Analiza si La distribución es **simétrica y centrada en cero**, si hay simetrías o colas largas que puede indicar que el modelo no esté capturando bien la relación.

Multiple Linear Regression

6. Ahora entrena un modelo de regresión lineal utilizando **todas las variables disponibles** como predictores. Para la variable “chas”:

    - **¿Cómo interpretamos la variable** chas**?**
      Como es una variable binaria (0 o 1), su coeficiente representa la diferencia esperada en el precio medio de la vivienda entre las zonas que **limitan con el río Charles** y las que no. Si el coeficiente es positivo, sugiere que vivir cerca del río incrementa el valor.

7. Vamos a construir un modelo que use lstat, age y su **interacción** como predictores.

```python
# Crear columna de interacción
df['lstat_age'] = df['lstat'] * df['age']

# Modelo con interacción
X_interact = df[['lstat', 'age', 'lstat_age']]
y = df['medv']

model_interact = LinearRegression()
model_interact.fit(X_interact, y)

# Mostrar coeficientes
coef_names = X_interact.columns
for name, coef in zip(coef_names, model_interact.coef_):
    print(f"{name}: {coef:.4f}")
```

8. Transformaciones de Variables: lstat² y log(rm), con **validación cruzada**. Vamos ahora a transformar dos variables:

    - Agregar lstat² para capturar una posible relación cuadrática.
    - Agregar log(rm) para suavizar posibles efectos exponenciales del número de habitaciones.

    Hint: importa “cross_val_score” desde sklearn.model_selection

```python
from sklearn.model_selection import cross_val_score

model_trans = LinearRegression()
scores = cross_val_score(model_trans, X_transformed, y, cv=5, scoring='r2')
```

Sobreajuste y como controlarlo

¿Qué es el overfitting?: Overfitting ocurre cuando un modelo funciona muy bien con los datos de entrenamiento pero falla al predecir sobre datos nuevos (de prueba o validación)

9. Analizar el Overfitting con regresión lineal

    - Usen el conjunto de datos completo (Boston).

    - Divídanlo aleatoriamente en:

    - **10% para entrenamiento**
    - **90% para prueba**

    - Entrenen un modelo de **regresión lineal** con todas las variables.
    - Calculen el **R^2** sobre el conjunto de entrenamiento.
    - Usen ese mismo modelo para predecir sobre el conjunto de prueba.
    - Calculen el **R^2** del conjunto de prueba.
    - Comparen ambos valores: ¿el R^2 del entrenamiento es mucho mayor que el del test? Si es así, el modelo está sobreajustado.

10. Investiga sobre la Lasso y Ridge Regression y como aplicarla para luchar el sobreajuste.

Hint: Ridge y Lasso son una variante de la regresión lineal que sirven para luchar el dobreajuste y minimizar el efecto de aquellas variables menos significativas en el modelo. La diferencia principal entre Ridge y Lasso es que **Ridge reduce los coeficientes sin llevarlos a cero, mientras que Lasso puede reducir algunos coeficientes exactamente a cero**, realizando así selección de variables.

```
from sklearn.linear_model import Ridge

ridge5 = Ridge(alpha=5)
ridge5.fit(X_train, y_train)

# Métricas
r2_train_ridge5 = r2_score(y_train, ridge5.predict(X_train))
r2_test_ridge5 = r2_score(y_test, ridge5.predict(X_test))
```