MEMORIA PRÁCTICA 1

APC

SUSANA SÁNCHEZ ROPERO, 1494978
RONGRONG ZHANG, 1495351
MIKHAIL PESKOV, 1534227

Índice

INTRODUCCIÓN

En esta primera práctica de la asignatura, nos centraremos en aplicar modelos de regresión. Para ello, tenemos que analizar y entender la base de datos que se nos ha asignado: *Precios de la vivienda en Boston*.

En los siguientes apartados profundizaremos en la base de datos y responderemos las preguntas que se nos han formulado.

El GitHub es el siguiente: https://github.com/1495351/GPA203_0930.git

APARTADO C: ANALIZANDO DATOS

Este apartado nos servirá para conocer y familiarizarnos con nuestra base de datos.

Analizaremos y entenderemos todos y cada uno de los atributos que la componen.

Los atributos que componen nuestra base de datos sobre los precios de la vivienda de Boston son los siguientes:

- 'CRIM': tasa de criminalidad per cápita por ciudad.
  - Es un atributo de tipo float.
  - El rango de los datos varía entre 0.009 y 88.976.
- 'ZN': proporción de terreno residencial zonificado para lotes de más de 25,000 pies cuadrados.
  - Es un atributo de tipo float.
  - El rango de los datos varía entre 0 y 100.000.
- 'INDUS': proporción de acres comerciales no minoristas por ciudad.
  - Es un atributo de tipo float.
  - El rango de los datos varía entre 0.460 y 27.740.
- 'CHAS': variable ficticia del río Charles (= 1 si el tramo limita con el río; 0 de lo contrario)
  - Es un atributo de tipo int.
  - El rango de los datos varía entre 0 y 1.
    - Así que podemos decir que es un atributo binario.
- 'NOX': concentración de óxidos nítricos.
  - Es un atributo de tipo float.
  - El rango de los datos varía entre 0.385 y 0.871.
- 'RM': número medio de habitaciones por vivienda.
  - Es un atributo de tipo float.
  - El rango de los datos varía entre 3.561 y 8.780.
- 'AGE': proporción de unidades ocupadas por el propietario construidas antes de 1940.
  - Es un atributo de tipo float.
  - El rango de los datos varía entre 2.900 y 100.00.
- 'DIS': distancias ponderadas a cinco centros de empleo de Boston.
  - Es un atributo de tipo float.
  - El rango de los datos varía entre 1.130 y 12.127.
- 'RAD': índice de accesibilidad a carreteras radiales.
  - Es un atributo de tipo int.
  - El rango de los datos varía entre 1 y 24.000.
- 'TAX': tasa de impuesto a la propiedad de valor completo por $10,000.
  - Es un atributo de tipo float.
  - El rango de los datos varía entre 187.000 y 711.000.
- 'PTRATIO': ratio alumno-profesor por ciudad.
  - Es un atributo de tipo float.
  - El rango de los datos varía entre 12.600 y 22.000.
- 'R': 000(Bk−0.63)2 donde Bk es la proporción de personas de color por ciudad.
  - Es un atributo de tipo float.
  - El rango de los datos varía entre 0.320 y 396.900.

- 'LSTAT': %menor estatus de la población.
  - Es un atributo de tipo float.
  - El rango de los datos varía entre 1.730 y 37.970.
- 'MEDV': Valor medio de las casas ocupadas por el propietario en $1000s.
  - Es un atributo de tipo float.
  - El rango de los datos varía entre 5.000 y 50.000.

Podemos ver, que el atributo CHAS es una variable categórica, que tiene dos valores posibles, 0 y 1, que indican si cruza el río Charles o no (es una variable inventada). Y el atributo RAD también es una variable categórica ya que nos dice el índice de accesibilidad a carreteras radiales.

Para datos categóricos, una simple tabulación de la frecuencia de cada categoría es la mejor exploración no gráfica para el análisis de datos. Sin embargo, esos atributos no son útiles a la hora de hacer la regresión, y a la hora de predecir el MEDV.

Tratamiento de datos

Primero de todo, hemos averiguado si en nuestra base de datos había elementos null.

Para esto, hemos usado *'dataset.isnull().sum()'.* El cual hace un recuento por cada atributo de los elementos null. En nuestro caso, por suerte, no hemos encontrado ninguno.

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

Ilustración 1: nulls

Estadísticas

Para ayudarnos a conocer los datos, miraremos las estadísticas.

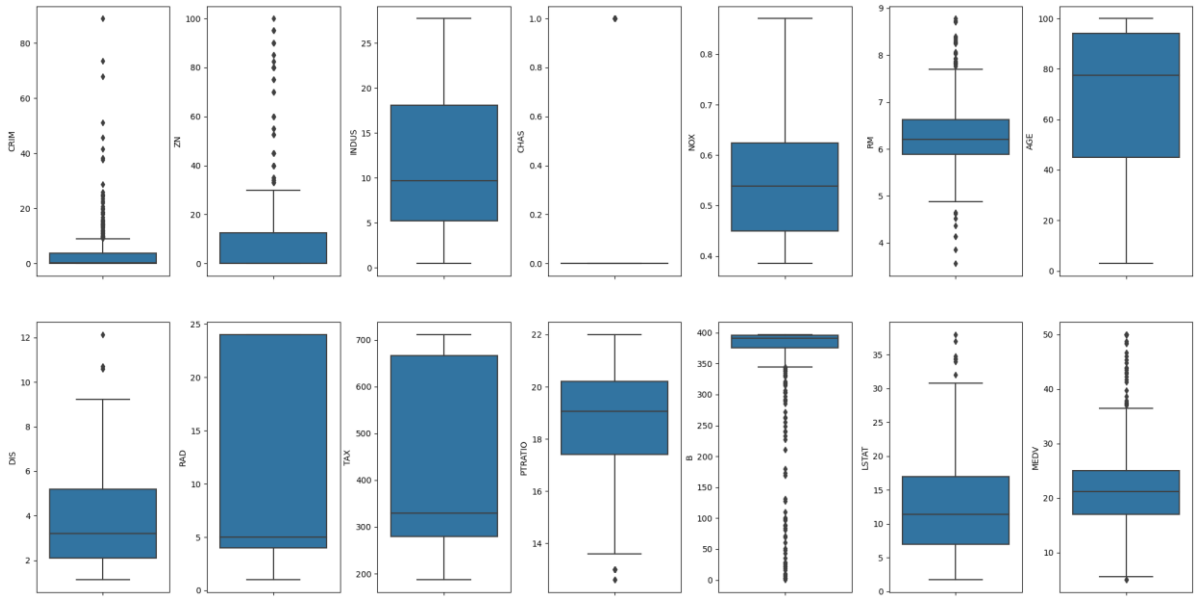


Ilustración 2: boxplot

La boxplot de RM es las que más se acercan a una distribución normal. Pero también se asemeja a la forma bimodal, ya que tiene la mediana desplazada. Lo comprobaremos con los histogramas.

Si observamos cada uno de los atributos en las boxplot, vemos que CRIM, ZN, RM, DIS, PTRATIO, B, LSTAT y MEDV, tienen bastantes outliers (valores atípicos).

Veamos cual es el porcentaje de outliers en cada uno de estos atributos:

```
Columna CRIM -> porcentaje de valores atípicos = 13.10%
Columna ZN -> porcentaje de valores atípicos = 13.49%
Columna INDUS -> porcentaje de valores atípicos = 0.00%
Columna CHAS -> porcentaje de valores atípicos = 100.00%
Columna NOX -> porcentaje de valores atípicos = 0.00%
Columna RM -> porcentaje de valores atípicos = 5.95%
Columna AGE -> porcentaje de valores atípicos = 0.00%
Columna DIS -> porcentaje de valores atípicos = 0.99%
Columna RAD -> porcentaje de valores atípicos = 0.00%
Columna TAX -> porcentaje de valores atípicos = 0.00%
Columna PTRATIO -> porcentaje de valores atípicos = 2.98%
Columna B -> porcentaje de valores atípicos = 15.08%
Columna LSTAT -> porcentaje de valores atípicos = 1.19%
Columna MEDV -> porcentaje de valores atípicos = 7.34%
```

Ilustración 3: outliers

Estos, desgraciadamente, no se pueden tratar, ya que entran dentro de los rangos de datos y si los tratamos perderemos información. A la hora de normalizar los datos se nos resolverá este problema.

También, vemos que los atributos INDUS, AGE, DIS, RAD y TAX tienen la mediana desplazada, no está en el centro, así que podemos afirmar que la distribución de los datos no es simétrica.

Una vez identificados y analizados los atributos, debemos responder a las siguientes preguntas:

1. ¿Qué atributos tienen una distribución gaussiana?

Para estudiar la distribución gaussiana de los atributos, miraremos los histogramas de cada atributo.

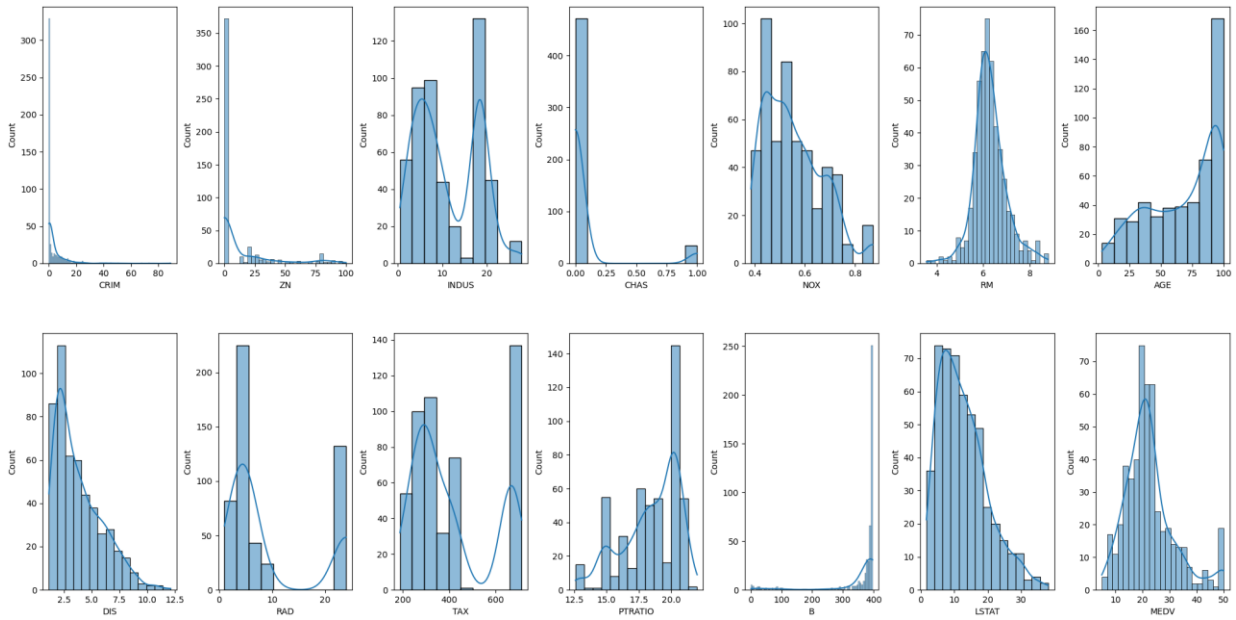


Ilustración 4: histogramas

Observando los histogramas, podemos ver que el atributo RM tiene, lo que se puede parecer, a una distribución gaussiana. Y podríamos decir que MEDV también.

Para comprobar poder comprobar las hipótesis anteriores, usamos *normaltest()* de *cipy.stats.* Que al ejecutarse, nos indicará si ese atributo tiene una distribución normal.

```
0 The null hypothesis can be rejected
1 The null hypothesis can be rejected
2 The null hypothesis can be rejected
3 The null hypothesis can be rejected
4 The null hypothesis can be rejected
5 The null hypothesis can be rejected
6 The null hypothesis can be rejected
7 The null hypothesis can be rejected
8 The null hypothesis can be rejected
9 The null hypothesis can be rejected
10 The null hypothesis can be rejected
11 The null hypothesis can be rejected
12 The null hypothesis can be rejected
13 The null hypothesis can be rejected
```

Ilustración 5: normaltest()

Desgraciadamente, *normaltest(()* indica que ninguno de nuestros atributos tiene distribución normal. Por lo tanto, el atributo RM que creíamos que era gaussiano, no lo es.

2. ¿Cuál es el atributo objetivo?

Después de analizar la base de datos, hemos identificado el atributo MEDV como atributo objetivo. Este es el dato, de todo el conjunto, que más se acerca al precio de la vivienda, ya que nos indica el valor medio de las casas ocupadas de la zona. Este es el atributo más importante, ya que en definitiva, es el atributo de interés.

El conjunto de todos los demás atributos hace que el precio de la vivienda de Boston varíe, por lo tanto, estamos seguros de que será nuestro atributo objetivo.

Hemos visto que el rango de valores que componen este atributo fluctúa entre los 5.000 y 50.000. Por lo tanto, podemos afirmar que es el atributo correcto para ser el objetivo.

APARTADO B: PRIMERAS REGRESIONES

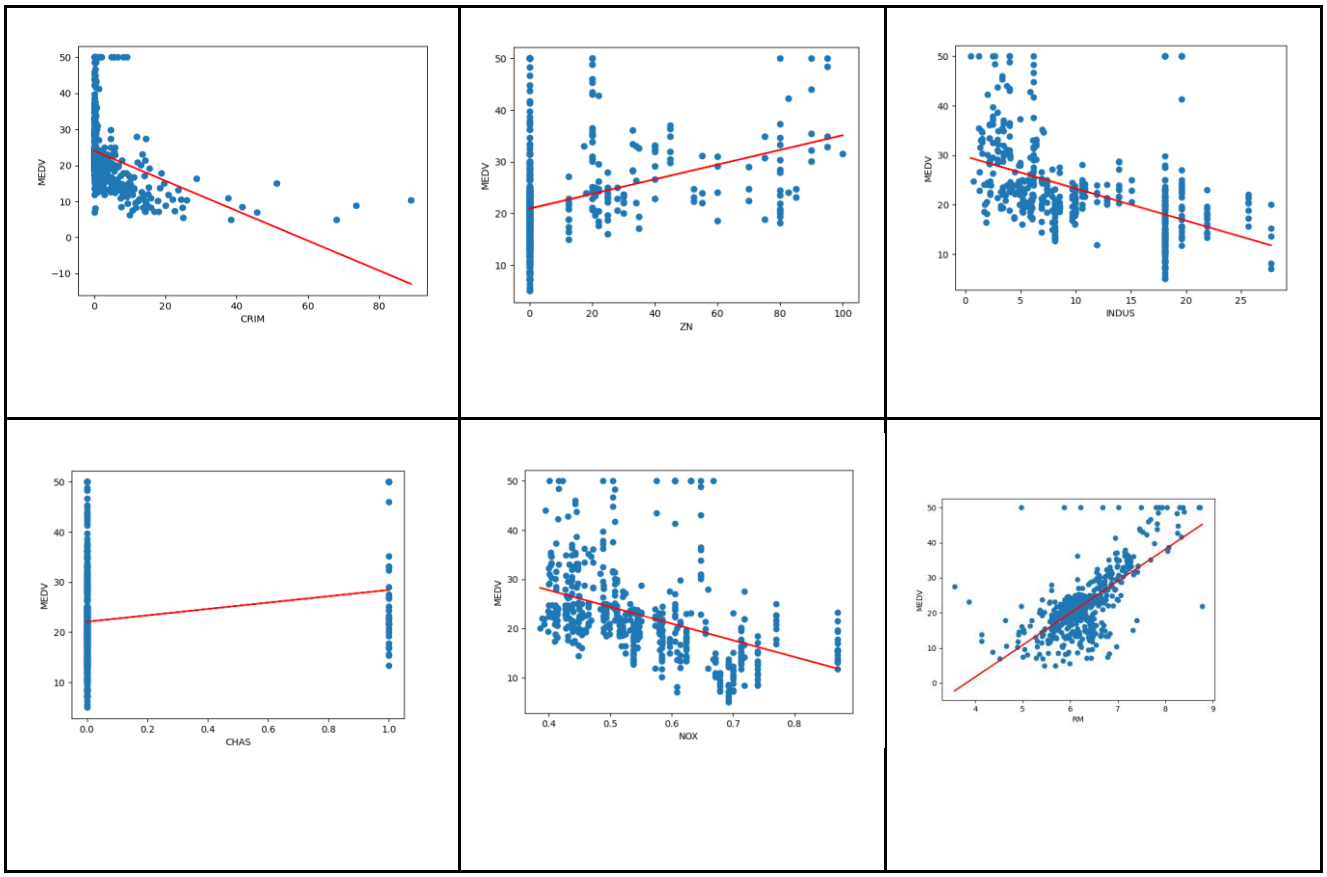
Antes de entrar a este segundo apartado, vamos a entrenar los datos con un modelo de regresión linear sin estandarizado de datos a cada uno de los atributos, para luego más tarde, poder hacer la comparación con un modelo de regresión linear con estandarizado de datos.

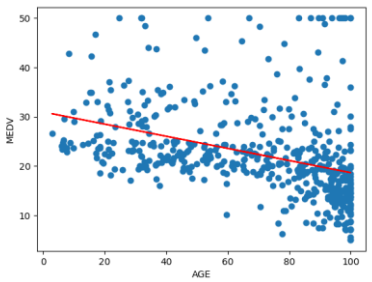
Determinando el atributo por lo cual tenga un error cuadrático medio más bajo.

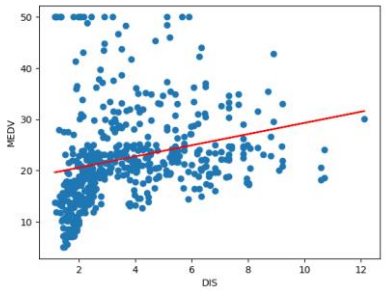
Utilizaremos la librería numpy para facilitar los cálculos y también usaremos la librería sklearn para entrenar con el modelo regresión linear.

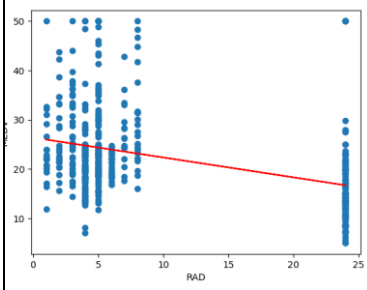
Modelo regresión lineal sin estandarizado de datos

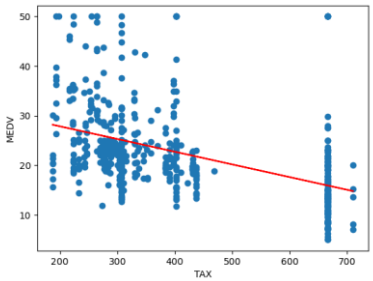
A continuación se mostrarán las gráficas correspondientes a cada atributo vs el atributo objetivo MEDV entrenados con un modelo de regresión lineal sin estandarizado de datos. Junto con los valores respectivos de MSE y R2.

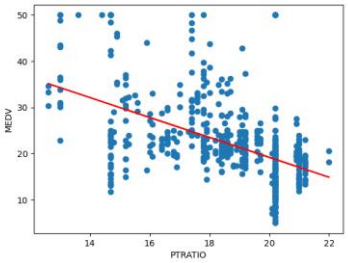


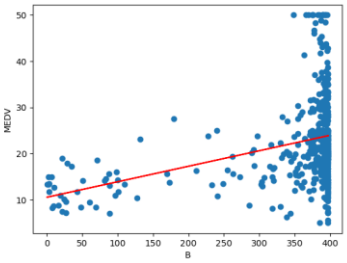

MEDV
AGE

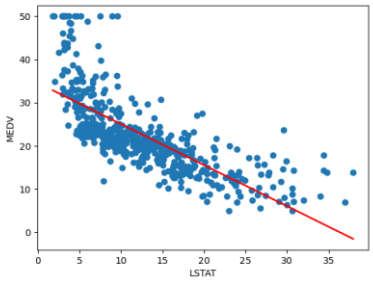

MEDV
DIS


RAD


MEDV
TAX

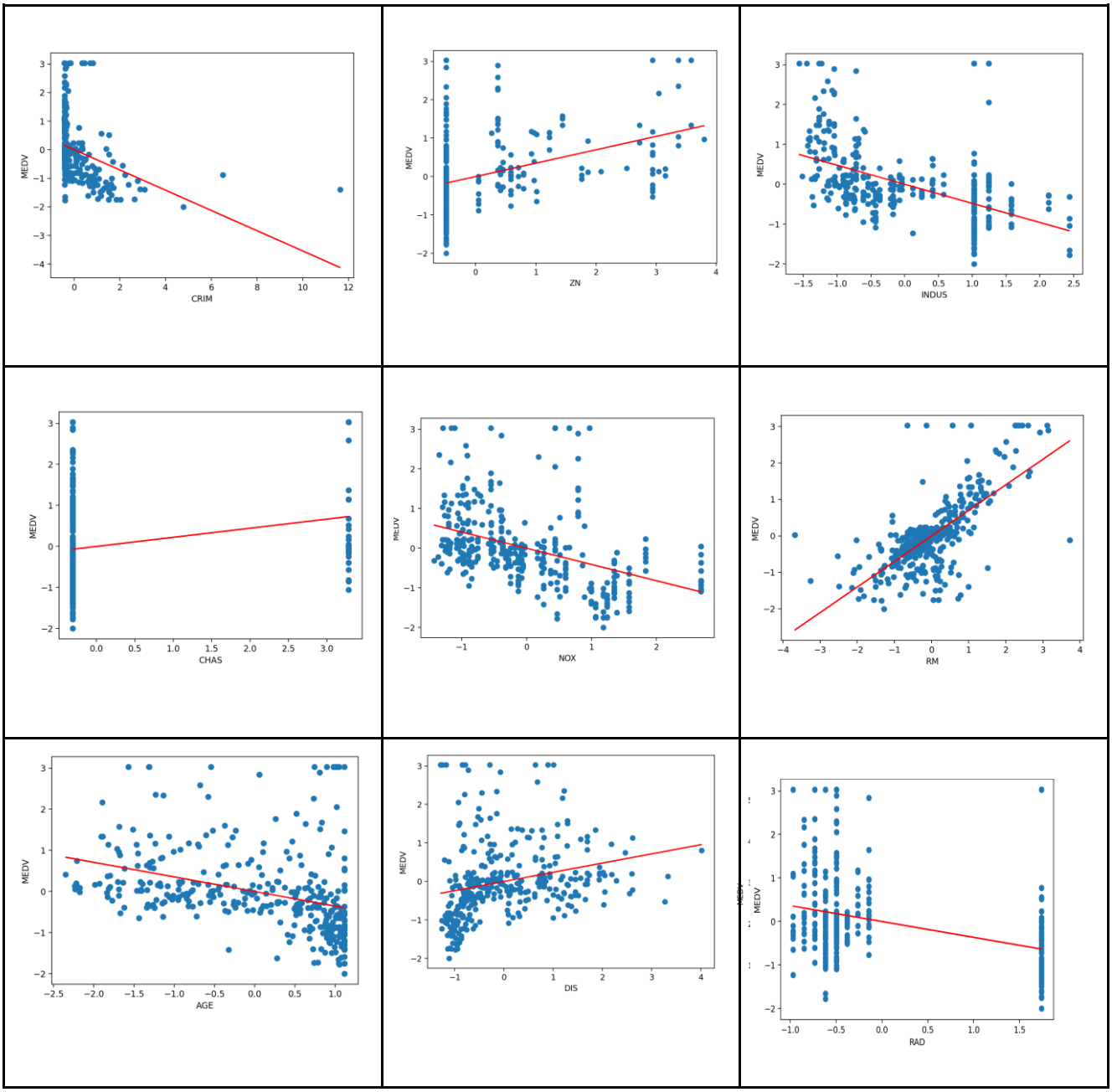

MEDV
PTRATIO

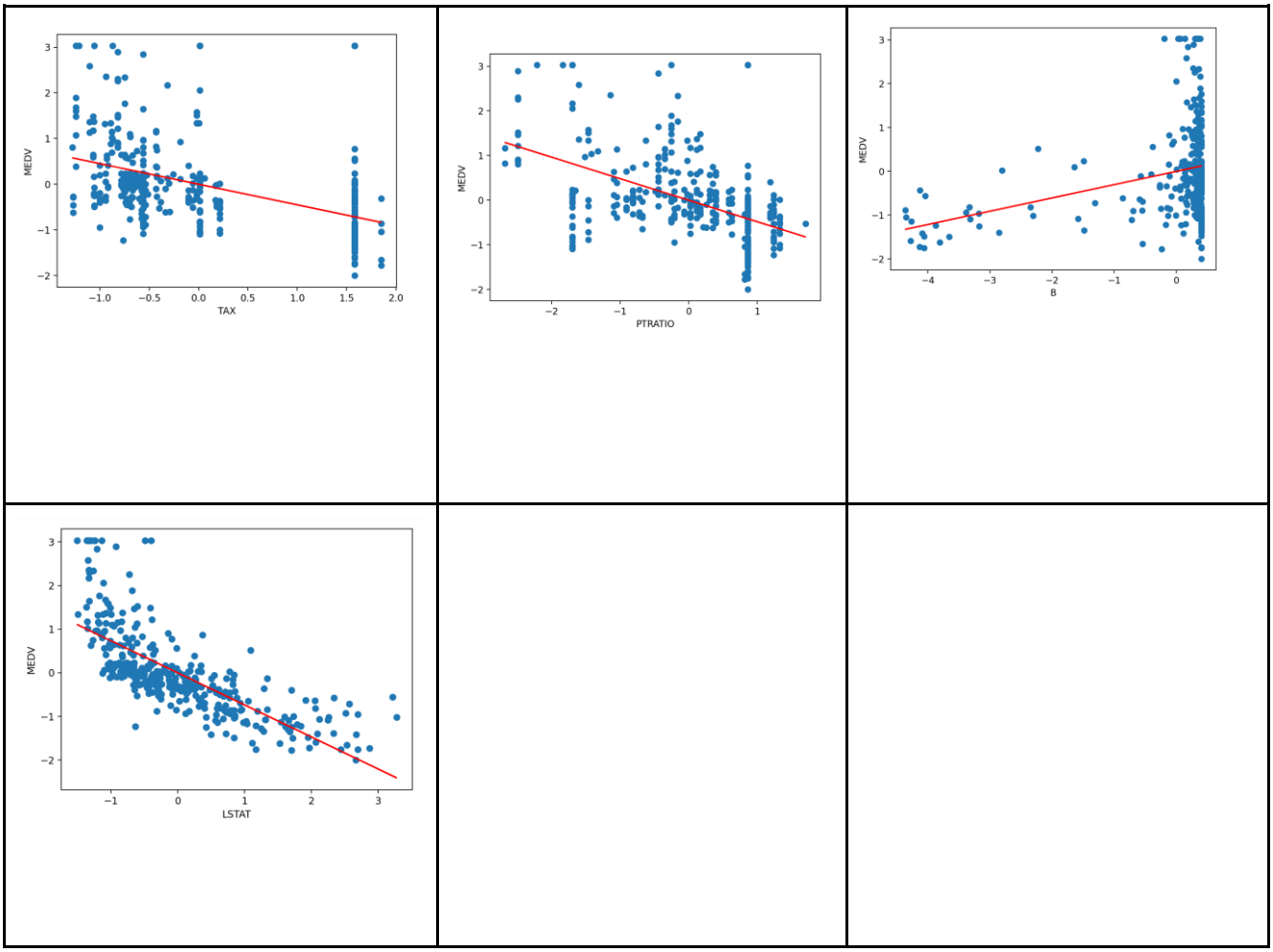

MEDV
B


MEDV
LSTAT

Modelo regresión lineal con estandarizado de datos

A continuación se mostrarán las gráficas correspondientes a cada atributo vs el atributo objetivo MEDV entrenados con un modelo de regresión lineal con estandarizado de datos.





En los gráficos se ve que obtenemos, ligeramente y casi imperceptible, mejores resultados (la línea de la regresión más ajustada) con los atributos que hemos considerado los más importantes (los que tienen mayor correlación con el objetivo). Pero no hay un cambio que se pueda considerar importante.

El cambio notorio que hay al estandarizar los datos, es en el rango de datos del MSE.

Los MSE y R2 de estos atributos serán mostrados y explicados en los siguientes apartados.

1. ¿Cuáles son los atributos más importantes para hacer una buena predicción?

Para poder responder esta primera pregunta, debemos hacer dos cosas:

1. Mirar cuales son los atributos que están más correlacionados con la variable objetivo.
2. Una vez hecho el primer punto, mirar el MSE qué, indicará si sirven, o no, estos atributos para ser utilizados en la predicción de la variable objetivo.

En la pregunta tres de este mismo apartado, analizamos la matriz de correlación de nuestra base de datos con estos estandarizados. Y concluimos que, para nuestra variable objetivo MEDV, es decir el valor medio de las viviendas de Boston, tenemos dos atributos que están altamente correlacionados con él. Siendo estos el porcentaje de residentes en el barrio de bajo nivel de vida (LSTAT) y el número medio de habitaciones de las cuales dispone la vivienda (RM).

Si calculamos el error cuadrático medio de todos los atributos. Nos fijamos en el/los que tengan menor error, y si estos coinciden con los dos anteriores, serán los candidatos a ser los más importantes para hacer una buena predicción.

El error cuadrático medio es el MSE. Antes de poder calcularlo, tenemos que separar el dataset entre test y train, y normalizarlos.

A continuación, obtendremos un modelo de la regresión lineal el cual usaremos para entrenar nuestro conjunto de train.

Dicho modelo, será usado para obtener predicciones sobre el conjunto de *test_x* normalizado.

Una vez hecho, procederemos a calcular el MSE y el coeficiente de determinación con la función *mean_squared_error* y *r2_score()*, respectivamente, de la librería *sklearn.* Pasándole por parámetro el conjunto *test_y* normalizado y las predicciones calculadas anteriormente.

Estos han sido los resultados tras calcular el MSE de todos los atributos normalizados:

MSE - STANDARIZED												
CRIM	**ZN**	**INDUS**	**CHAS**	**NOX**	**RM**	**AGE**	**DIS**	**RAD**	**TAX**	**PTRATIO**	**B**	**LSTAT**
0.74	0.74	0.77	0.89	0.68	0.44	0.69	0.80	0.71	0.61	0.65	0.76	0.38

R2 - STANDARIZED												
CRIM	**ZN**	**INDUS**	**CHAS**	**NOX**	**RM**	**AGE**	**DIS**	**RAD**	**TAX**	**PTRATIO**	**B**	**LSTAT**
0.14	0.14	0.34	-0.03	0.20	0.49	0.19	0.07	0.18	0.29	0.25	0.12	0.55

Si observamos las tablas anteriores, vemos que el atributo CHAS es el que tiene mayor MSE, y LSTAT el que menos. Por lo tanto, el más importante es LSTAT, ya que es el que tiene menos error. Este atributo corresponde al porcentaje de la población de menor estatus.

A este atributo le sigue RM, que es el número medio de habitaciones por vivienda.

Podemos afirmar que tenemos dos atributos importantes. De los cuales el más significativo es el porcentaje de estatus de la población y el siguiente es el número medio de habitaciones en las casas.

Si ahora nos fijamos en los resultados de R2, que es una media estadística de qué tan cerca están los datos de la línea de regresión. Tiene un rango entre 0 y 1, siendo el 1 el que mejor se ajusta al modelo de datos.

Sabiendo esto, podemos ver en la gráfica anterior que, los atributos que tienen R2 más cercano a 1 son, LSTAT y RM.

Por lo tanto, nuestra teoría era cierta, ya que estos dos atributos, LSTAT y RM son los más importantes a la hora de hacer una buena predicción. Y el MSE nos lo ha confirmado.

2. ¿Con qué atributo se consigue un MSE menor?

Como hemos indicado en la pregunta anterior, el atributo LSTAT es el que menos MSE tiene de todos.

El MSE da mucho más peso a los errores más grandes, por lo tanto, LSTAT al tener el MSE menor a todos, quiere decir que no contiene prácticamente errores en comparación al resto del dataset.

3. ¿Qué correlación hay entre los atributos de vuestra base de datos?

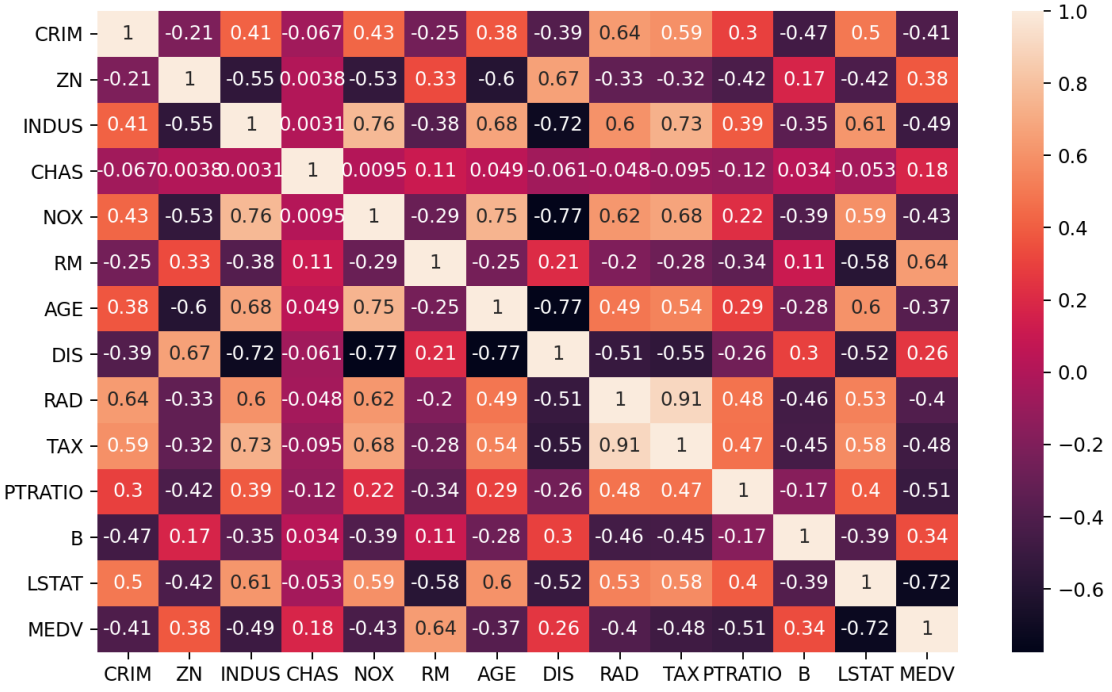


Ilustración 6: matriz correlación

Si observamos la matriz de correlación, y concretamente nuestro atributo objetivo MEDV, podemos ver que está fuertemente relacionado con dos atributos. Recordemos que MEDV es el valor medio de las casas ocupadas por el propietario.

La correlación más fuerte que observamos es del atributo LSTAT, que representa el porcentaje de bajo estatus de la población.

Y el segundo atributo con mayor correlación es RM, el número medio de habitaciones por vivienda.

Vistos estos resultados, podemos concluir, que para nuestra variable objetivo, el valor medio de las viviendas de Boston, debemos centrarnos en los atributos que afectan de una manera considerable, siendo estos el nivel de vida de los residentes del barrio y en número de habitaciones de las cuales dispone la vivienda.

Los demás atributos, no tienen tanta influencia a la hora de decidir si una vivienda es buena candidata para invertir en ella.

Si echamos un vistazo al apartado uno, podemos ver que los atributos que tienen mayor importancia coinciden con los que tienen una correlación más fuerte.

4. ¿Cómo influye la normalización en la regresión?

La normalización de la base de datos tiene múltiples beneficios, entre los cuales destacan:

- Facilitar el acceso e interpretación de los datos.
- Reducir el tiempo y complejidad de revisión de las bases de datos.
- Optimizar el espacio de almacenamiento.
- Prevenir borrados indeseados.
- Ausencia de dependencias incoherentes

En general, tiene como objetivo optimizar los datos y ofrecer integridad.

Para normalizar nuestra base de datos hemos utilizado la función *standarize()* proporcionada en el notebook de la práctica.

En las gráficas del principio del apartado, se muestra la regresión lineal de los atributos de entrada con el atributo objetivo MEDV. Observamos que la regresión lineal de atributo CHAS y RAD no se parece nada al resultado esperado porque son atributos categóricos como ya hemos mencionado anteriormente.

De todas estas gráficas, vemos que el atributo RM y LSTAT tiene una distribución ligeramente más concentrada.

Al normalizar los datos, los MSE se encuentran en un rango entre 0 y 1 en comparación a los datos no normalizados, que se encuentran en un rango mucho mayor. La diferencia se ve muy clara en las gráficas que se muestran más abajo. Son señales de que el modelo hace el fit mejor.

En las siguientes tablas se muestran los resultados de los MSE y R2 de los datos no normalizados y normalizados:

MSE - NO STANDARIZED												
CRIM	**ZN**	**INDUS**	**CHAS**	**NOX**	**RM**	**AGE**	**DIS**	**RAD**	**TAX**	**PTRATIO**	**B**	**LSTAT**
70.74	73.74	60.38	60.38	65.80	42.53	71.07	76.86	64.70	57.95	54.07	68.26	35.81

R2 - NO STANDARIZED												
CRIM	**ZN**	**INDUS**	**CHAS**	**NOX**	**RM**	**AGE**	**DIS**	**RAD**	**TAX**	**PTRATIO**	**B**	**LSTAT**
0.084	0.100	0.218	0.034	0.148	0.449	0.080	0.005	0.162	0.250	0.300	0.116	0.536

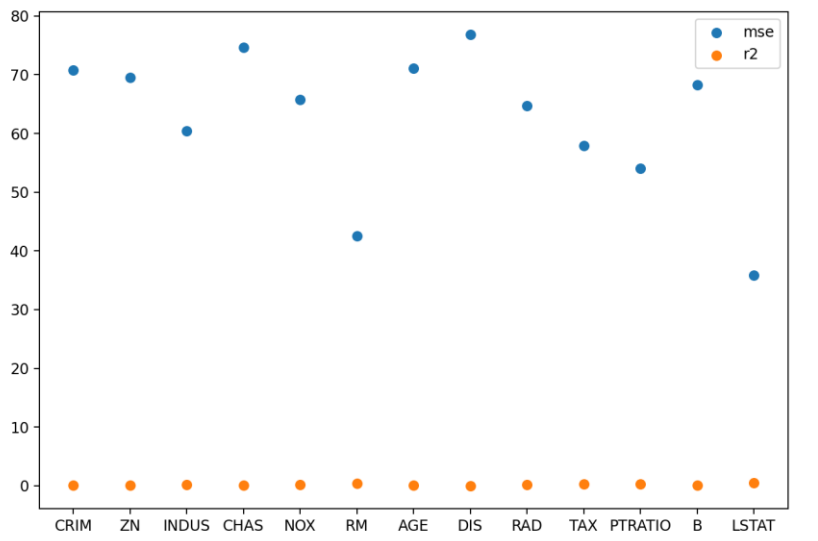


Ilustración 7:datos no estandarizados

MSE - STANDARIZED												
CRIM	**ZN**	**INDUS**	**CHAS**	**NOX**	**RM**	**AGE**	**DIS**	**RAD**	**TAX**	**PTRATIO**	**B**	**LSTAT**
0.74	0.74	0.77	0.89	0.68	0.44	0.69	0.80	0.71	0.61	0.65	0.76	0.38

R2 - STANDARIZED												
CRIM	**ZN**	**INDUS**	**CHAS**	**NOX**	**RM**	**AGE**	**DIS**	**RAD**	**TAX**	**PTRATIO**	**B**	**LSTAT**
0.14	0.14	0.34	-0.03	0.20	0.49	0.19	0.07	0.18	0.29	0.25	0.12	0.55

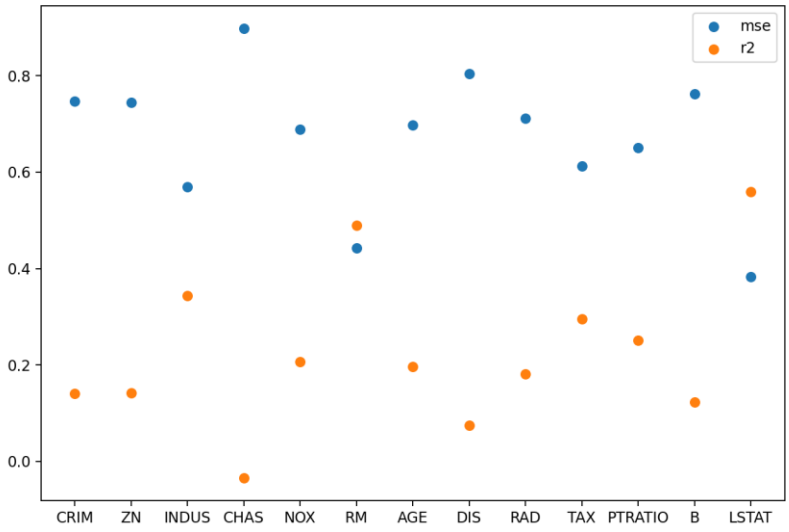


Ilustración 8: Datos estandarizados

Comprobar la predicción de los modelos de RM y LSTAT

Hemos entrenado dos modelos, uno que predice MEDV a partir del atributo RM y otro con LSTAT.

Estos modelos nos dan los siguientes resultados.

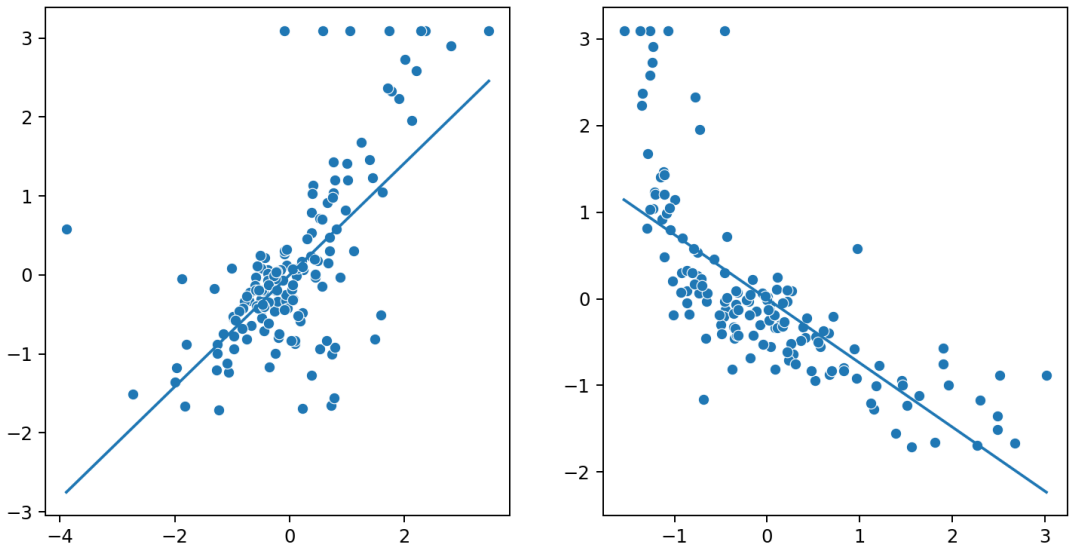


Ilustración 9: predicciones RM y LSTAT

La gráfica de la izquierda corresponde con el modelo del atributo RM y el de la derecha el de LSTAT.

Aquí vemos el error del valor predicho con el valor real. Vemos que la predicción es bastante buena.

Para acabar de comprobarlo haremos lo siguiente:

Calculamos el score con el módulo *sklearn.metrics* y nos da un 0.635. Por lo tanto, la clasificación es buena, como hemos dicho.

En la gráfica siguiente se muestra la malla en la que se ven las predicciones de RM (idx1) y LSTAT(idx2).

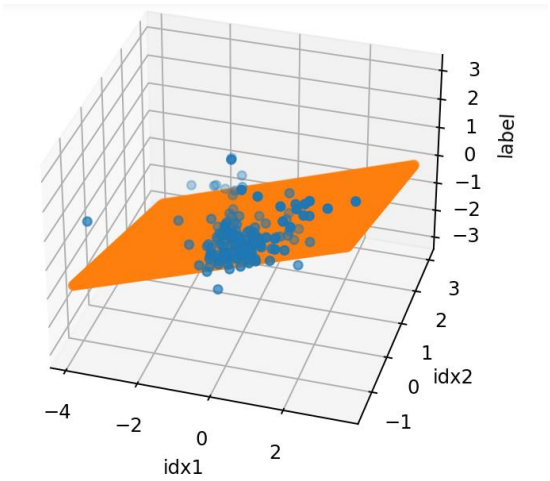


Ilustración 10: malla predicciones

Evaluar el modelo con un conjunto de validación no visto

Para comprobar que nuestro modelo se ajusta a datos no vistos anteriormente, vamos a entrenarlo con un conjunto de validación. Como no tenemos, lo hemos generado de esta forma:

```
X = dataset.drop(['CRIM', 'ZN','INDUS','CHAS','NOX', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'MEDV'], axis = 1)
y = dataset["MEDV"]

x_train, y_train, x_val, y_val = split_data(X.values, y.values)

train_x_norm, mean, std = standarize(x_train)
test_x_norm, _, _ = standarize(x_val, mean, std)

train_y_norm, mean, std = standarize(y_train[:, None])
test_y_norm, _, _ = standarize(y_val[:, None], mean, std)
```

Y predecimos a partir de los atributos más importantes establecidos anteriormente, RM y LSTAT.

Obtenemos los siguientes MSEs y R2s:

- Error en atribut RM: 0.655404
- R2 score en atribut RM: -1.363865
- Error en atribut LSTAT: 0.375597
- R2 score en atribut LSTAT: 0.651872

Esto indica que, aun haciendo la predicción con datos desconocidos, el modelo predice de manera correcta.

5. ¿Cómo mejora la regresión cuando se filtran esos atributos de las muestras que no contienen información?

En este caso, hemos escogido dos atributos, RM y LSTAT que son los que hemos considerado más importantes para hacer el análisis y los que tienen el menor error cuadrático medio.

En el apartado anterior "Evaluar el modelo con un conjunto de validación no visto", hemos usado como conjunto de train, únicamente los atributos RM y LSTAT.

Como vemos en los datos anteriores, aplicamos la regresión lineal multivariante y observamos que la media del error cuadrático se ha reducido respecto a la regresión lineal univariante. El coeficiente de determinación ha aumentado.

En conclusión, en este caso en concreto, una regresión lineal multivariante es mejor que una regresión lineal univariante. Ya que el MSE se ve reducido.

6. Si se aplica un PCA, a cuántos componentes se reduciría el espacio? ¿Por qué?

Para calcular el PCA (Análisis de componentes principales), usamos *sklearn.descomposition*. Con el conjunto de train_x normalizado, hacemos un *pca.fit_transform()* y con el test_x, también normalizado, un *pca_transorm()* simplemente.

Después de esto, aplicamos los pasos que hemos seguido hasta ahora, es decir, con un modelo de regresión linear entrenado, hacemos la predicción y sacamos los valores de MSE y R2.

Una vez hecho esto, obtenemos los resultados siguientes:

MSE							
PCA_1	**PCA_2**	**PCA_3**	**PCA_4**	**PCA_5**	**PCA_6**	**PCA_7**	**PCA_8**
0.523	0.465	0.341	0.327	0.262	0.269	0.275	0.278

R2							
PCA_1	**PCA_2**	**PCA_3**	**PCA_4**	**PCA_5**	**PCA_6**	**PCA_7**	**PCA_8**
0.388	0.455	0.601	0.618	0.694	0.685	0.678	0.675

Si lo mostramos gráficamente tendría esta forma:

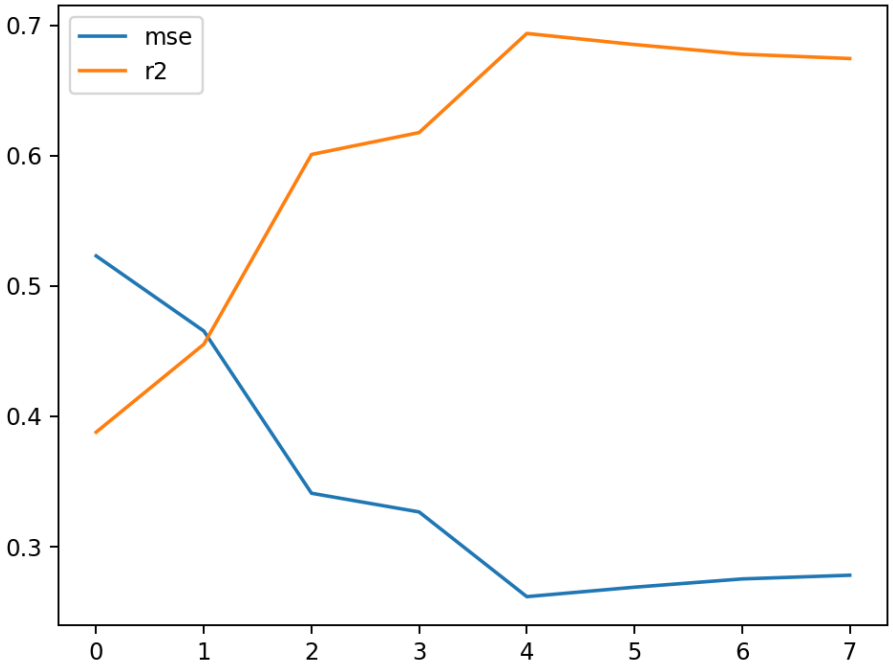


Ilustración 11: PCA

APARTADO A: EL DESCENSO DE GRADIENTE

Para poder calcular el descenso de gradiente, primero deberemos implementar los siguientes pasos:

- Definir la función de coste y de gradiente
- Estudiar cómo el uso de regularizadores afecta al resultado: overfitting, underfitting, etc.
- Visualización de los datos a analizar i explicación paso a paso del procedimiento
- Visualización del proceso de descenso de gradiente
- Modificar el learning rate y el número de iteraciones

Definición función coste y gradiente

```
class Regressor(object):
    def __init__(self):
        self.costs = []

    def predict(self, x):
        return np.matmul(x, self.w)

    def cost(self, prediction, y):
        cost = (1/(2*len(y))) * (sum((prediction - y) ** 2))
        cost += self.reg * sum(self.w ** 2)
        return cost

    def update(self, x, hy, y):
        error = (np.matmul(x, self.w)) - y
        self.w = self.w - ((self.alpha/len(y)) * np.matmul(x.transpose(), error)
                           -(self.reg/len(y) * self.w))

    def train(self, x, y, max_iter, epsilon, alpha, reg):
        self.max_iter = max_iter
        self.epsilon = epsilon
        self.alpha = alpha
        self.reg = reg
        X = x.copy()
        X.insert(0, 'bias', np.ones(len(x)))
        self.w = np.random.rand(len(x.columns))
        prediction = self.predict(x)
        self.costs.append(self.cost(prediction, y))
        self.update(x, prediction, y)
        iter = 1
        while (iter <= 1000):
            prediction = self.predict(x)
            self.costs.append(self.cost(prediction, y))
            if((self.costs[iter] - self.costs[iter - 1]) > epsilon):
                break
            self.update(x, prediction, y)
            iter += 1
```

Ilustración 12: nuestro regresor

- ¿Cómo influyen todos los parámetros en el proceso de descenso? ¿Qué valores de learning rate convergen más rápido a la solución óptima? ¿Cómo influye la inicialización del modelo en el resultado final?

Para responder a estas preguntas, hemos entrenado diferentes modelos usando nuestra implementación del regresor con diferentes valores de learning rate y lambda. Como métrica principal hemos utilizado MSE. También hemos comparado los pasos necesarios para la convergencia de los modelos.

Para learning rate hemos probado valores [0.01, 0.1, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2, 0.3, 0.35]. Y para lambdas los valores [0,0.001,0.01, 0.1, 0.15, 0.2, 0.3, 0.5].

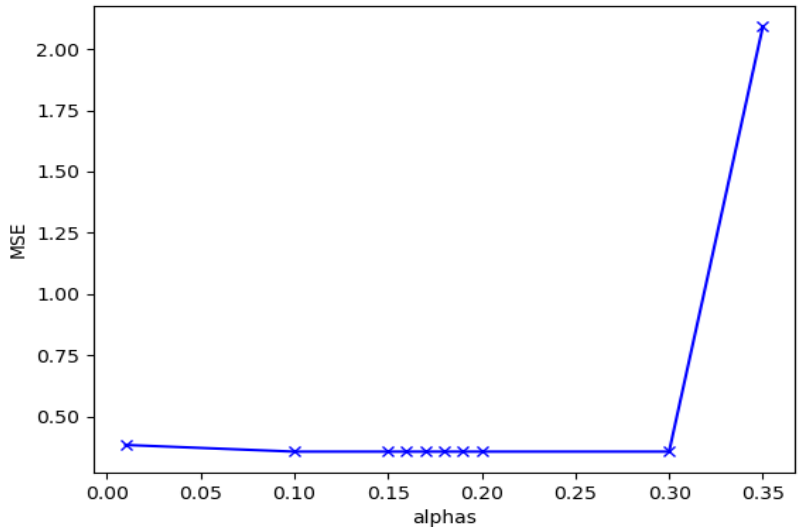


Ilustración 13

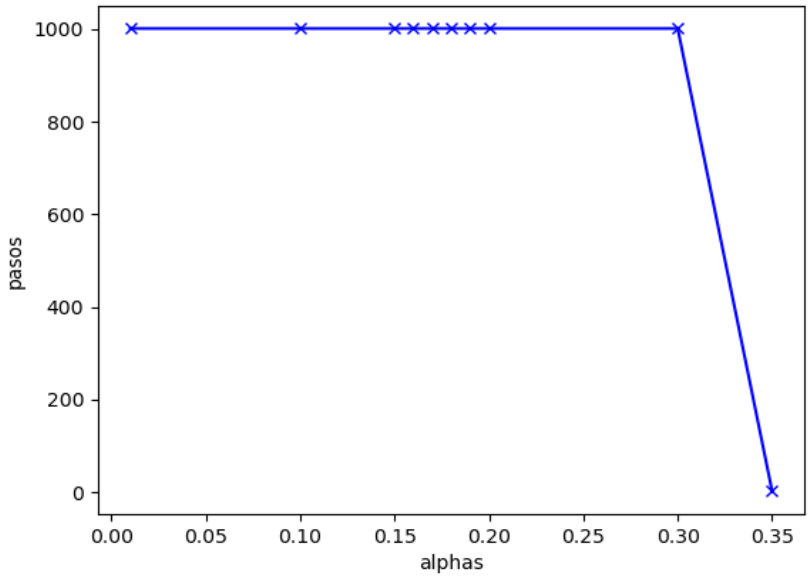


Ilustración 14

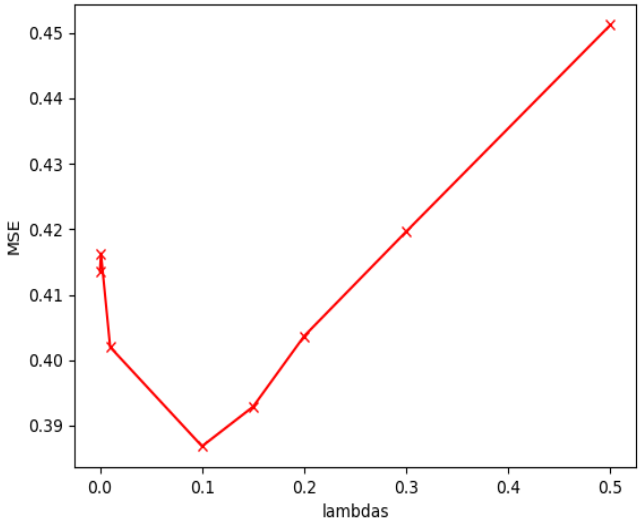


Ilustración 15

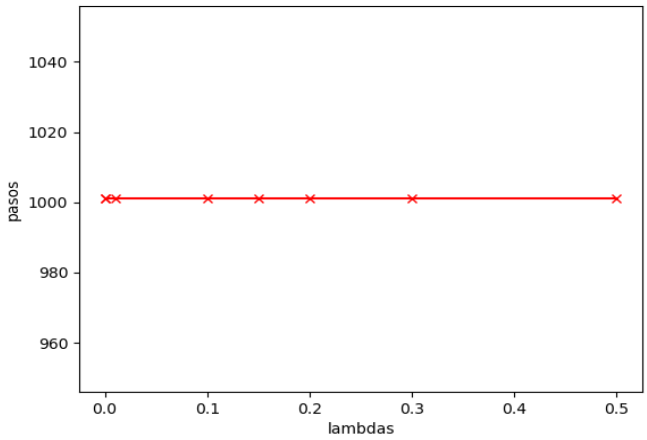


Ilustración 16

Con los resultados obtenidos podemos hacer la conclusión: el valor de learning rate afecta la calidad de la predicción y la rapidez de la convergencia, pero la rapidez nos lleva a soluciones no-óptimas. El valor del regularizador (lambda) también tiene un papel muy importante para la calidad de predicciones, pero no afecta nada a la rapidez de convergencia.

- ¿Qué funciones polinomiales (de diferente grado, de diferentes combinaciones de atributos, ...) habéis escogido para ser aprendidas con vuestro descenso del gradiente? ¿cuál ha dado el mejor resultado (en error y rapidez en convergencia)?

Para hacer las pruebas necesarias, vamos a estudiar las métricas finales de los siguientes modelos:

Modelo 1: Con todo el conjunto de atributos de X

Modelo 2: Con el atributo LSTAT que tiene la mayor correlación con el objetivo, LSTAT al cuadrado, LSTAT al cubo.

Modelo 3: Con 2 atributos más importantes: LSTAT y RM.

Modelo 4: LSTAT^2 + LSTAT*RM + RM^2.

El modelo 1 da el error de 0.357 con alpha = 0.15 y lambda = 0.1. Convergencia en 1001 iteraciones.

El modelo 2 con los mismos parámetros cae en diferentes mínimos locales y da el error entre 34 y 148. Esto es una señal del coeficiente de aprendizaje demasiado alto. Después de cambiarlo por 0.1 el error es 0.399. Los cambios en la lambda, sorprendentemente, no cambiaron el resultado. Convergencia en 33 iteraciones.

El modelo 3 con los parámetros alpha = 0.15 y lambda = 0.1 hace una predicción con error 0.469 y claramente no está afectado por cambios pequeños en los parámetros de entrada. Convergencia en 1000 iteraciones

El modelo 4 da diferentes resultados y depende del coeficiente de regularización (lambda). El mejor resultado era con lambda = 0.5 y error 0.789. Convergencia en 79 iteraciones.

Entre los 4 modelos entrenados, el primero da el mejor resultado predictivo y el segundo es el más rápido, siendo similar al primero en el error.

- ## Utilice el regularizador en la fórmula de función de coste y descenso del gradiente y provee polinomios de diferente grado. ¿Cómo afecta el valor del regularizador?

Como ya hemos comentado en la respuesta anterior, el regularizador sirve para reducir el error en el conjunto de validación en el caso de que tengamos alta probabilidad de overfitting, ya que penaliza el uso de demasiados atributos. Pero tiene que estar bien ajustado, porque la penalización demasiado alta resulta en bias alto (underfitting) (que se ve en el gráfico).

En el caso de nuestra base de datos y con los polinomios que hemos utilizado, el regularizador no tenía mucho significado, pero lo tendría en el caso de utilizar más atributos o menor cantidad de datos.

- ## ¿Qué diferencia (cuantitativa y cualitativa) hay entre vuestro regresor y el de la librería?

Para dar nuestra respuesta a esta pregunta simplemente hemos calculado el MSE después de aplicar el descenso del gradiente.

```
x_train, x_test, y_train, y_test = train_test_split(x, y,  random_state=23)
regr = regression(x_train, y_train)
predictions = regr.predict(x_test)
print("Resultado con la librería: ", mse(predictions, y_test))

regr = Regressor()
regr.train(x_train, y_train, max_iter = 10000, epsilon = 0.001, alpha = 0.15, reg = 0.1)
predictions = regr.predict(x_test)
print("Resultado con nuestra función:", mse(predictions, y_test))
```

```
Resultado con la librería:  0.3573407107095565
Resultado con nuestra función: 0.357051561874362
```

Ilustración 17

Según la métrica MSE las funciones dan los resultados similares. Por una causa inexplicable, nuestro regresor tiene el MSE un poco menor, puede ser porque con nuestra función podemos ajustar algunos parámetros, como alpha, lambda y epsilon.

- ¿Tiene sentido el modelo (polinomial) encontrado cuando se visualiza sobre los datos?

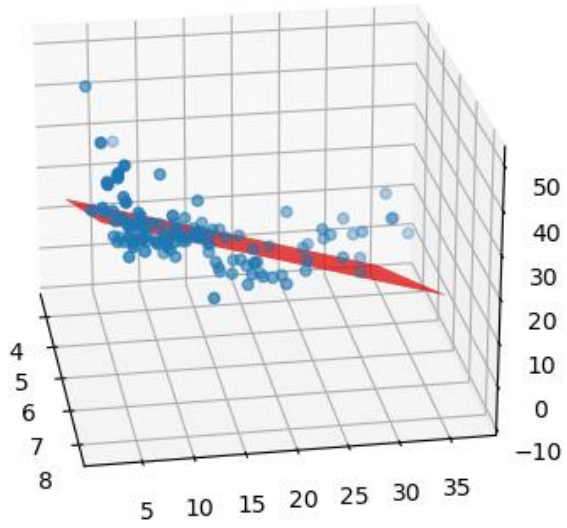


Ilustración 18

Construyendo el gráfico podemos ver si nuestro regresor se ajusta bien a los datos. Esto sirve para modelos lineales y polinomiales.

- ¿Ayuda la visualización a identificar aquellas muestras para las que el regresor obtiene los peores resultados de predicción?

Sí, por ejemplo, en el caso de nuestro gráfico de arriba se ve el plano rojo, que es nuestro modelo. Los puntos azules que están más lejos del plano son las muestras, que el regresor predice peor.